ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΕΞΕΙΔΙΚΕΥΣΗΣ

Επικοινωνίες επιπέδου χρήστη για δίκτυα Fast Ethernet σε περιβάλλον Linux

Ιωάννης Σταγάκης

Επιβλέπων Καθηγητής: Βασίλειος Δημακόπουλος

Ιωάννινα, Ιούνιος 2004

Περιεχόμενα

Κεφάλαιο 1
Εισαγωγή

1.1 Λίγη ιστορία

Η εξέλιξη των υπολογιστικών συστημάτων έφερε μια τεράστια αλλαγή στην ζωή του ανθρώπου. Ο υπολογιστής σήμερα κυριαρχεί σαν εργαλείο στην εργασία, στην παραγωγή αγαθών αλλά και στην ψυχαγωγία του ανθρώπου. Η παλαιότερη και ογκώδης εμφάνιση των υπολογιστικών συστημάτων έχει συμπιεστεί δραματικά στις μέρες μας, επωφελούμενη από την εξέλιξη των ολοκληρωμένων κυκλωμάτων που χρησιμοποιούνται σε αυτά. Ψηφιακές συσκευές ελάχιστου μεγέθους έχουν γίνει πλέον αναπόσπαστο κομμάτι της καθημερινότητας του ανθρώπου. Η επικοινωνία και η ανταλλαγή πληροφοριών έχει αποκτήσει τεράστια σημασία και έχει αναχθεί σε ένα πολύ σημαντικό τομέα της σύγχρονης ζωής. Τα υπολογιστικά συστήματα, συνδεμένα πλέον μεταξύ τους μέσω του παγκόσμιου διαδικτύου, ανταλλάσσουν πληθώρα πληροφοριών πολλαπλών μορφών ταχύτατα και αξιόπιστα. Πλέον η διασύνδεση και επικοινωνία των υπολογιστικών συστημάτων θεωρείται αυτονόητο αγαθό ακόμα και για τον απλούστερο χρήστη υπολογιστών. Η ταχύτατη αποδοχή και επέκταση του Διαδικτύου εμφανίζεται πλέον σαν ένα αναπάντεχο, παρόλα αυτά καλοδεχούμενο φαινόμενο, στην ιστορική διαδρομή της επιστήμης των υπολογιστών.

Η διασύνδεση απομακρυσμένων υπολογιστικών συστημάτων παρόλα αυτά, δεν είναι μια ιδέα που προέκυψε τα τελευταία χρόνια. Από τα τέλη της δεκαετίας του '60, σχεδόν ταυτόχρονα με την μαζική εξάπλωση των υπολογιστών σε επιστημονικά ιδρύματα, γεννήθηκε και η ιδέα της διασύνδεσης των υπολογιστών μεταξύ τους με στόχο την ανταλλαγή πληροφοριών. Ο προάγγελος του Διαδικτύου ήταν ένα δίκτυο ευρείας περιοχής (Arpanet), που συνέδεε μεταξύ τους υπολογιστικά συστήματα σε ερευνητικά κέντρα, στρατιωτικές βάσεις και κυβερνητικές υπηρεσίες των Ηνωμένων Πολιτειών Αμερικής, χρηματοδοτούμενο από το Υπουργείο Άμυνας των ΗΠΑ. Η χρησιμότητα της ανταλλαγής αρχείων και ηλεκτρονικών μηνυμάτων αποδεικτικέ μεγάλη, οπότε και στη συνέχεια νέα πανεπιστήμια και ερευνητικά ιδρύματα συνδέθηκαν στο δίκτυο αυτό. Η διασύνδεση των υπολογιστών γινόταν πάνω από νοικιασμένες γραμμές των 56Kbps. Στα μέσα της δεκαετίας του '70 το δίκτυο εξαπλώθηκε πέρα από τα σύνορα των ΗΠΑ προς την δυτική Ευρώπη. Στην επόμενη δεκαετία, η χρησιμότητα της διασύνδεσης ώθησε διάφορα ευρείας περιοχής δίκτυα

που είχαν σχηματιστεί σε Ευρώπη, Ασία και Αμερική, να διασυνδεθούν, ώστε να καταλήξουμε τελικά σε αυτό που σήμερα ονομάζουμε Διαδίκτυο (Internet).

1.2 Τα πρωτόκολλα

Τα πρωτόκολλα που κυριάρχησαν για την επικοινωνία των απομακρυσμένων υπολογιστικών συστημάτων πάνω από τον φυσικό ιστό του Διαδικτύου είναι τα TCP/IP και UDP/IP [5]. Το TCP και UDP αποτελούν πρωτόκολλα επιπέδου μεταφοράς (επίπεδο 4 στο OSI) ενώ το IP αποτελεί πρωτόκολλο επιπέδου δρομολόγησης (επίπεδο 3 στο OSI).

Το TCP αποτελεί πρωτόκολλο επιπέδου μεταφοράς με σύνδεση (connection-oriented) μεταξύ των υπολογιστικών συστημάτων που ανταλλάσσουν δεδομένα. Τα υπολογιστικά συστήματα αποτελούν τα άκρα του καναλιού απ' οπού θα ξεκινήσουν και θα καταλήξουν τα δεδομένα. Για τη δημιουργία αυτού του εικονικού καναλιού πάνω από το φυσικό ιστό του Διαδικτύου γίνεται μια τριπλή χειραψία με ανταλλαγή δεδομένων αρχικοποίησης της επικοινωνίας, μεταξύ των άκρων του καναλιού. Στη συνέχεια η επικοινωνία είναι ελεγχόμενη σε επίπεδο ανταλλαγής δεδομένων μεταξύ των κόμβων - άκρων του δικτύου. Αν για κάποιο λόγο χαθούν κάποια δεδομένα κατά τη μεταφορά, ο αποστολέας ενημερώνεται από στον παραλήπτη και εκκινεί διαδικασία επαναποστολής των δεδομένων.

Το UDP αποτελεί πρωτόκολλο επιπέδου μεταφοράς χωρίς σύνδεση (connectionless). Το υπολογιστικό σύστημα σε αυτή την περίπτωση στέλνει δεδομένα προς ένα απομακρυσμένο σταθμό του δικτύου, χωρίς πριν να τον έχει ενημερώσει για την πρόθεσή του αυτή. Σε αυτή την περίπτωση η απώλεια των δεδομένων, πάνω από μία διαδρομή του δικτύου, δεν γνωστοποιείται στον αποστολέα.

Και τα δύο πρωτόκολλα επιπέδου μεταφοράς έχουν τη χρησιμότητα τους, η οποία και δεικνύεται άμεσα από την φύση της επικοινωνίας που απαιτείται μεταξύ των υπολογιστικών συστημάτων. Η επικοινωνία μετά από σύνδεση, που χαρακτηρίζει το TCP πρωτόκολλο, απαιτεί μεγαλύτερη επεξεργασία των πακέτων προς ανταλλαγή από τους κόμβους που συμμετέχουν σε αυτή. Σαν αντιστάθμισμα για την αυξημένη επεξεργασία των πακέτων που απαιτεί το πρωτόκολλο, εξασφαλίζεται ασφάλεια στην μετάδοση των δεδομένων στον απομακρυσμένο κόμβο του δικτύου.

Σε αντίθεση με το TCP, το πρωτόκολλο UDP δεν παρέχει αυτή την ασφάλεια στη μετάδοση των δεδομένων. Τα δεδομένα στέλνονται χωρίς σύνδεση στον απομακρυσμένο υπολογιστή οπότε και αν χαθεί κάποιο μέρος τους δεν παρέχεται από το πρωτόκολλο τρόπος να ειδοποιηθεί ο αποστολέας. Αντιστάθμισμα αυτής της

ανασφαλής συμπεριφοράς του πρωτοκόλλου είναι η ελάχιστη επεξεργασία που απαιτούν τα πακέτα κατά τη μεταφορά.

Αν και η σύλληψη και υλοποίηση των πρωτοκόλλων μεταφοράς έγινε αρκετά παλιά, πριν την μεγάλη εξάπλωση του Διαδικτύου, τα πρωτόκολλα εμφάνισαν μια εξαιρετικά σταθερή συμπεριφορά κατά την χρήση τους, ακόμα και σε καταστάσεις αυξημένης κίνησης και αριθμού κόμβων στο Διαδίκτυο. Συνέπεια αυτού είναι η χρήση τους πλέον ως εξορισμού πρωτοκόλλων για την μεταφορά δεδομένων μεταξύ απομακρυσμένων υπολογιστικών συστημάτων. Παρόλα αυτά κάποιες φορές και κάτω από ειδικές συνθήκες οπού η επικοινωνία έχει συγκεκριμένα χαρακτηριστικά και στόχους, τα πρωτόκολλα αυτά δεν κρίνονται σαν πλέον κατάλληλα για τη χρήση τους κατά την μεταφορά των δεδομένων.

1.3 Παράλληλη επεξεργασία και δίκτυα σταθμών εργασίας

Η επίλυση συγκεκριμένων προβλημάτων με τη χρήση υπολογιστικών συστημάτων έχει αποδειχθεί ότι επιτελείται ταχύτερα με την τμηματοποίηση του προβλήματος σε κομμάτια, τα οποία δεν εξαρτώνται άμεσα το ένα από το άλλο, και την ταυτόχρονη εξαγωγή αποτελεσμάτων μετά από υπολογισμό των κομματιών αυτών και την συγκέντρωση των αποτελεσμάτων για την εύρεση του τελικού ζητούμενου. Η παρατήρηση αυτή οδήγησε στην κατασκευή υπολογιστικών συστημάτων με πολλαπλούς επεξεργαστές, στους οποίους θα διανέμονται κάθε χρονική στιγμή ανεξάρτητα κομμάτια του προβλήματος προς υπολογισμό ώστε να ευρεθεί τελικά το ζητούμενο. Οι παράλληλες μηχανές αυτές έχουν σχεδιαστεί με τέτοιο τρόπο ώστε η επικοινωνία μεταξύ των επεξεργαστών να είναι ταχύτατη με στόχο την γρηγορότερη συγκομιδή ενδιάμεσων αποτελεσμάτων και την εύρεση του τελικού ζητούμενου. Παρόλα αυτά το κόστος κατασκευής ενός τέτοιου παράλληλου υπολογιστικού συστήματος είναι συχνά τεράστιο λόγω των αυξημένων δυνατοτήτων που παρουσιάζει ένα τέτοιο σύστημα σε σχέση με τους κλασσικούς σταθμούς εργασίας ενός επεξεργαστή, αλλά και λόγω των μη συμβατικών διασυνδέσεων που χρησιμοποιούν.

Στις μέρες μας οι κλασσικοί σταθμοί εργασίας παρουσιάζουν αλματώδη διεύρυνση των δυνατοτήτων τους με πολλαπλά αυξημένη ταχύτητα επεξεργασίας. Επιπλέον τα τοπικά δίκτυα διασύνδεσης των σταθμών εργασίας εμφανίζουν υψηλές ταχύτητες μεταφοράς δεδομένων (της τάξεως Gpbs), οι οποίες είναι άμεσα συγκρίσιμες με αυτές του εσωτερικού διαύλου του υπολογιστή. Και όλα αυτά σε κόστος πολύ μικρότερο σε σχέση με το κόστος των παραλλήλων συστημάτων επεξεργασίας.

Έτσι γίνεται ιδιαίτερα ελκυστική η ιδέα της αξιοποίησης διασυνδεόμενων σταθμών εργασίας μέσω ενός ταχύτατου τοπικού δικτύου ώστε να κατασκευάσουμε μια ιδεατή υπολογιστική μηχανή προς αξιοποίηση σε παράλληλο υπολογισμό προβλημάτων. Η ιδεατή αυτή παράλληλη μηχανή πρέπει να καταφέρει να επιδείξει υπολογιστικές δυνατότητες παραπλήσίες ενός παράλληλου υπολογιστικού συστήματος. Κύριος στόχος σε αυτή την προσπάθεια είναι ελαχιστοποίηση του χρόνου επικοινωνίας μεταξύ των απομακρυσμένων επεξεργαστών των σταθμών επεξεργασίας, ώστε να επιδεικνύεται μια ταχύτατη ανταλλαγή μηνυμάτων κατά τον υπολογισμό. Μια τέτοια λύση για παράλληλο υπολογισμό συχνά αναφέρεται σαν Δίκτυο Σταθμών Εργασίας (Network Of Workstations – NOW).

Για την επίτευξη μιας ταχύτατης επικοινωνίας, κοντά στα φυσικό όριο που προσφέρει το τοπικό δίκτυο, είναι απαραίτητη η χρήση κατάλληλου πρωτοκόλλου επικοινωνίας με ελάχιστη επεξεργασία από την πλευρά των απομακρυσμένων κόμβων. Σε αυτή την περίπτωση η χρήση της εξορισμού στοίβας πρωτοκόλλων επικοινωνίας TCP/IP και UDP/IP επιβάλλει μεγάλο φόρτο στην επεξεργασία των δεδομένων προς ανταλλαγή. Επίσης λόγω των ειδικών συνθηκών επικοινωνίας, η οποία προορίζει τα μηνύματα προς ανταλλαγή μόνο μέσα σε τοπικό δίκτυο, η χρήση ιδιωτικά υλοποιημένων πρωτοκόλλων συνήθως εμφανίζονται ως η καλύτερη επιλογή.

1.4 Αντικείμενο εργασίας

Στην εργασία μας διερευνήσαμε τη δυνατότητα χρήσης κοινών καρτών δικτύου για προσωπικούς υπολογιστές με στόχο την υλοποίηση επικοινωνίας σε επίπεδο χρήστη, δηλαδή χωρίς την παρεμβολή του πυρήνα του λειτουργικού συστήματος. Η υλοποίηση μας δεν επικεντρώνεται μόνο στην ανταλλαγή μηνυμάτων μέσω συγκεκριμένου πρωτοκόλλου σε επίπεδο χρήστη, αλλά παρέχει στην διεργασία χρήστη μια πλατφόρμα για απευθείας χρήση της κάρτας δικτύου χωρίς την παρέμβαση του πυρήνα κατά την αποστολή και λήψη μηνυμάτων. Χρησιμοποιώντας την πλατφόρμα αυτή μια διεργασία χρήστη μπορεί να αναπτύξει πλέον ιδιωτικά πρωτόκολλα που καλύπτουν σε κάθε περίπτωση τις ιδιαίτερες ανάγκες και στόχους της επικοινωνίας που θέλει να επιτύχει.

Η υλοποίηση περιλαμβάνει ένα άρθρωμα (module) που εισάγεται στον πυρήνα και αποτελεί την διασύνδεση των διεργασιών χρήστη με το υλικό της κάρτας δικτύου. Πρόσθετα υλοποιήθηκε μια βιβλιοθήκη συναρτήσεων που επιτρέπουν σε διεργασίες χρήστη την αποστολή και λήψη μηνυμάτων, αποκρύπτοντας από τις διεργασίες την λεπτομέρειες λειτουργίας της κάρτας δικτύου. Η βιβλιοθήκη περιλαμβάνει μια ελαφρότερη υλοποίηση της UDP/IP στοίβα πρωτοκόλλων για

ανταλλαγή μηνυμάτων με στόχο την συμβατότητα επικοινωνίας με υπάρχοντες εφαρμογές και την διερεύνηση της επίδοσης της επικοινωνίας επιπέδου χρήστη σε σχέση με την κλασσική στοίβα πρωτοκόλλων UDP/IP που υλοποιείται στον πυρήνα του λειτουργικού συστήματος.

Τα αποτελέσματα των μετρήσεων μας δείχνουν μια βελτίωση της τάξης των 30usec κατά την ανταλλαγή μηνυμάτων με χρήση της βιβλιοθήκης για επικοινωνία επιπέδου χρήστη.

1.5 Διάρθρωση εργασίας

Η εργασία είναι οργανωμένη ως εξής:

- Στο κεφάλαιο 2 θα αναπτύξουμε τις προσπάθειες που έχουν γίνει μέχρι στιγμής για την αντιμετώπιση του προβλήματος της ταχύτατης ανταλλαγής μηνυμάτων και της δημιουργίας ιδεατών παράλληλων μηχανών σε Δίκτυο από Σταθμούς Εργασίας, καθώς και σε ποιά σημεία διαφοροποιείται η δική μας.
- Στο κεφάλαιο 3 παρουσιάζουμε τις δυνατότητες και τον τρόπο επικοινωνίας με τον ελεγκτή Intel 82558 για κάρτες δικτύου Fast Ethernet.
- Στο κεφάλαιο 4 παρουσιάζουμε την τρόπο προγραμματισμού ενός οδηγού συσκευής με την χρήση αρθρωμάτων (Modules) καθώς και την υλοποίηση του οδηγού συσκευής για την κάρτα δικτύου Intel e100 Fast Ethernet.
- Στο κεφάλαιο 5 παρουσιάζουμε την πολιτικές λειτουργίας που διέπουν τον οδηγό συσκευής και τις δομές δεδομένων που αυτός χρησιμοποιεί.
- Στο κεφάλαιο 6 παρουσιάζουμε αποτελέσματα μετρήσεων που επιτυγχάνει ο οδηγός συσκευής κατά την ανταλλαγή μηνυμάτων.
- Τέλος στο κεφάλαιο 7 συνοψίζουμε την εργασία και συζητάμε μελλοντικές επεκτάσεις / κατευθύνσεις.

Κεφάλαιο 2
Σχετικές Εργασίες

2.1 Παράλληλες μηχανές

Η κρισιμότητα της καθυστέρησης στην ανταλλαγή μηνυμάτων μεταξύ επεξεργαστών που εκτελούν ένα παράλληλο πρόγραμμα είχε μελετηθεί αρχικά σε υπολογιστικά συστήματα με πολλαπλούς επεξεργαστές. Είναι σημαντικό τα αποτελέσματα που υπολογίζει ένας επεξεργαστής σε ένα τέτοιο σύστημα να είναι όσο το δυνατόν πιο άμεσα διαθέσιμα στους υπόλοιπους επεξεργαστές ώστε να συνεχίζεται χωρίς καθυστέρηση ο υπολογισμός. Η ανταλλαγή των αποτελεσμάτων μεταξύ των επεξεργαστών που συμμετέχουν σε έναν υπολογισμό γίνεται μέσω μηνυμάτων, οπότε η χαμηλή καθυστέρηση κατά την μεταφορά ενός μηνύματος εμφανίζεται σαν κρίσιμος παράγοντας για τον υπολογισμό που επιτελείται. Βασική αιτία καθυστέρησης σε τέτοια συστήματα είναι η ενδιάμεση αποθήκευση των μεταφερόμενων δεδομένων κατά την παραλαβή του μηνύματος από τον επεξεργαστή και πριν αυτά γίνουν διαθέσιμα στη διεργασία που απευθύνονταν.

Η πρώτη σημαντική προσπάθεια για αποφυγή αυτού του κόστους ήταν τα Active Messages [9] σε παράλληλες μηχανές nCUBE/2 και CM-5 και χρησιμοποιώντας σαν γλώσσα προγραμματισμού των διεργασιών την Split-C. Τα Active Messages κατάφερναν να ελαχιστοποιήσουν το κόστος της επικοινωνίας, χάρη στην εξάλειψη της ανάγκης για ενδιάμεση αποθήκευση των λαμβανομένων μηνυμάτων. Η στρατηγική που προτάθηκε ήταν η άμεση κλήση μιας διαδικασίας παραλαβής από την πλευρά της διεργασίας στην οποία απευθύνονταν τα δεδομένα κατά την λήψη του μηνύματος. Με τον τρόπο αυτό, τα δεδομένα του μηνύματος μεταφερόταν κατευθείαν στο χώρο της διεργασίας στην οποία και απευθύνονταν οπότε και ήταν άμεσα διαθέσιμα σε αυτή για συνέχιση των υπολογισμών.

2.2 Δίκτυα από σταθμούς εργασίας

Τα τελευταία χρόνια παρατηρείται μια αλματώδης αύξηση της υπολογιστικής δύναμης των επεξεργαστών με ταυτόχρονη μείωση του κόστους παραγωγής τους. Ως αποτέλεσμα αυτού, οι σύγχρονοι προσωπικοί υπολογιστές είναι πλέον άμεσα

συγκρίσιμοι σε υπολογιστική δύναμη με τους εξειδικευμένους σταθμούς εργασίας που χρησιμοποιούνται για απαιτητικούς υπολογισμούς. Το γεγονός αυτό οδήγησε στην χρήση υλικού προσωπικών υπολογιστών, το οποίο παρέχεται μαζικά στην αγορά, σε συνδυασμό με κατάλληλο λογισμικό συστήματος, για τη δημιουργία νέων σταθμών εργασίας υψηλών επιδόσεων και μικρού κόστους απόκτησης. Ταυτόχρονα το υλικό δικτύωσης σταθμών εργασίας, ακολουθώντας την πορεία του υλικού υπολογιστικών συστημάτων, προσφέρει υψηλές ταχύτητες μεταφοράς δεδομένων σε χαμηλή τιμή.

Έχοντας στη διάθεση μας τοπικά δίκτυα υψηλών επιδόσεων που συνδέουν σταθμούς εργασίας, φυσικό επακόλουθο είναι η χρήση των συνδεδεμένων σταθμών για την επίλυση παραλλήλων προβλημάτων. Οι εξειδικευμένοι παράλληλοι υπολογιστές πλέον εμφανίζονται όλο και σπανιότερα και τη θέση τους παίρνουν ιδεατές παράλληλες μηχανές οι οποίες βασίζονται σε Δίκτυα από Σταθμούς Εργασίας (Network Of Workstations).

Η χαμηλή καθυστέρηση για την επικοινωνία σε τέτοια δίκτυα, αποτελεί μια έντονη πρόκληση κατά την χρήση ιδεατών παράλληλων μηχανών. Ο κλασσικός τρόπος επικοινωνίας με την χρήση των ευρύτατα αποδεκτών πρωτοκόλλων TCP/IP και UDP/IP αποδεικνύεται ανεπαρκής λόγω των ειδικών συνθηκών και απαιτήσεων που παρουσιάζονται σε τέτοιες επικοινωνίες. Αυτό που παρουσιάζεται σαν άμεση λύση είναι η χρήση ιδιωτικά επινοημένων πρωτοκόλλων μικρών υπολογιστικά απαιτήσεων, ώστε να αντιμετωπισθούν οι καθυστερήσεις που εμφανίζουν οι κλασσικές στοίβες πρωτοκόλλων [4].

2.2.1 Παράγοντες καθυστέρησης

Οι κλασσικές στοίβες πρωτοκόλλων σχεδιάστηκαν για επικοινωνία σε ένα ετερόκλητο δίκτυο ευρείας περιοχής. Για την εξασφάλιση της μεταφοράς των δεδομένων δίχως σφάλματα σε προορισμούς ενδιαμέσως των οποίων παρεμβάλλονται πιθανώς διαφορετικού τύπου δίκτυα, ο σχεδιασμός της στοίβας πρωτοκόλλων του Διαδικτύου περιλαμβάνει κώδικες εντοπισμού λαθών καθώς και πληροφορίες για την δρομολόγηση των δεδομένων έως τον τελικό αποδέκτη. Στην ειδική περίπτωση που απευθυνόμαστε αποκλειστικά σε γειτονικούς μας κόμβους του τοπικού δικτύου η προσθήκη δεδομένων για τη δρομολόγηση κρίνεται περιττή. Επιπλέον η όλη διαδικασία παραγωγής και ελέγχου τιμών για την εξασφάλιση της ακεραιότητας των δεδομένων, προσθέτει καθυστέρηση κατά την αποστολή και λήψη δεδομένων, η οποία εμφανίζεται σημαντική στην περίπτωση μεταφοράς μεγάλου όγκου δεδομένων.

Σε περιπτώσεις μεταφοράς δεδομένων μικρού όγκου, σημαντική καθυστέρηση υπεισέρχεται λόγω του σχεδιασμού των πρωτοκόλλων σαν κομμάτι του πυρήνα του

λειτουργικού συστήματος [12]. Η επιλογή αυτή αφενός επιτρέπει ασφάλεια στην διαδικασία επεξεργασίας των δεδομένων των πρωτοκόλλων, η οποία γίνεται χωρίς την παρέμβαση του χρήστη, αφετέρου γεννάει επιπλέον ανάγκη για αντιγραφές δεδομένων στη μνήμη του συστήματος λόγω του σαφή διαχωρισμού του χώρου του πυρήνα από τον χώρο των διεργασιών χρήστη. Οι διεργασίες χρήστη κατά την κλήση υπηρεσιών που προσφέρει ο πυρήνας επιφέρουν μια μεταγωγή περιβάλλοντος από περιοχή χρήστη σε περιοχή πυρήνα. Αυτή η μεταγωγή περιβάλλοντος επιφέρει καθυστέρηση όταν εκτελείται κάτι που συμβαίνει π.χ κατά την αποστολή και λήψη δεδομένων. Οι καθυστερήσεις αυτές σαφώς και υπάρχουν κατά την μεταφορά μεγάλου όγκου δεδομένων, απλά υποβαθμίζονται από την καθυστέρηση που γεννιέται από τη πολλαπλή επεξεργασία των δεδομένων τόσο κατά τις αντιγραφές τους από περιοχές μνήμης όσο και από την ανάγνωσή τους για την δημιουργία του κώδικα ελέγχου λαθών.

2.2.2 Εργασίες πάνω σε εξειδικευμένα δίκτυα

Κατά την αξιοποίηση των Δικτύων από Σταθμούς Εργασίας σαν μια ιδεατή παράλληλη μηχανή, οι προσπάθειες επικεντρώθηκαν στην υπερνίκηση των παραγόντων καθυστέρησης που υπεισέρχονται κατά την χρήση των κλασσικών πρωτοκόλλων μεταφοράς δεδομένων.

Για το λόγο αυτό σχεδιάστηκαν δίκτυα εξειδικευμένα που προσφέρουν αρκετές δυνατότητες από την πλευρά των συσκευών δικτύωσης. Βασική προϋπόθεση πάντα η υψηλή ταχύτητα μεταφοράς δεδομένων που προσφέρει το φυσικό μέσο στο οποίο βασίζεται το δίκτυο. Δημοφιλές δίκτυο σε αυτή την περίπτωση εμφανίζεται το Myricom Myrinet [15] ταχύτητα μεταφοράς δεδομένων 1,28 Gbps. Στην κάρτα δικτύου του προσφέρει μνήμη SRAM 1 MByte καθώς και προγραμματιζόμενο επεξεργαστή για την υλοποίηση λειτουργιών του δικτύου απευθείας από το υλικό. Εργασίες που βασίζονται σε δίκτυο Myrinet είναι τα Illinois Fast Messages [16], Fast Sockets [19], BIP [18].

Κύριο χαρακτηριστικό τους είναι η χρήση αποκλειστικά ενός ιδιωτικά σχεδιασμένου πρωτοκόλλου για την μεταφορά των δεδομένων καθώς και η απευθείας χρήση των δυνατοτήτων της κάρτας δικτύου από την περιοχή της διεργασίας χρήστη. Με τον τρόπο αυτό αποφεύγεται το κλασσικό μονοπάτι μέσα από τον πυρήνα και εξαλείφεται όποια καθυστέρηση παρουσιάζεται από πολλαπλές αποθηκεύσεις και χρόνους μεταγωγής από περιοχή χρήστη σε πυρήνα λειτουργικού συστήματος.

Όλες οι υλοποιήσεις καταφέρνουν χαμηλούς χρόνους κατά την μεταφορά μηνύματος υποβοηθούμενες άμεσα από τον προγραμματισμό που επιδέχεται η υψηλού κόστους κάρτα δικτύου της Myricom. Επιπλέον, χρησιμοποιώντας τη

δυνατότητα προγραμματισμού του προσαρμογέα δικτύου στα δίκτυα Myrinet, εμφανίζονται κάποιες υλοποιήσεις που προσπαθούν να προσομοιώσουν τις προδιαγραφές που θέτει το πρότυπο Virtual Interface Architecture (VIA) [7],[10] για το υλικού μιας κάρτας δικτύου. Το πρότυπο VIA προτάθηκε από τις εταιρίες Compaq, Intel και Microsoft ώστε να δώσει προδιαγραφές λειτουργιών υλικού για προσαρμογείς δικτύων υψηλών επιδόσεων. Τέλος αξιοπρόσεχτες υλοποιήσεις εμφανίζονται και για άλλου τύπου δίκτυα όπως τα HPAM [14], υλοποίηση των Active Messages για Δίκτυα FDDI και U-Net [8] για Δίκτυα ATM.

2.2.3 Εργασίες πάνω σε δίκτυα Ethernet

Το Ethernet αποτελεί για πολλά χρόνια τον πιο διαδεδομένο τρόπο δικτύωσης υπολογιστών σε τοπικό επίπεδο. Η εξάπλωση του αυτή ώθησε αρκετές εργασίες να διερευνήσουν τις δυνατότητες που αυτό προσφέρει σαν βάση ενός Δικτύου από Σταθμούς Εργασίας. Αν και οι μέχρι πρότινος ταχύτητες των 100 Mbps για το Fast Ethernet, δεν πρόσφεραν έδαφος για υψηλές επιδόσεις, η εξέλιξη του σε ταχύτητες στη τάξης των Gbps σίγουρα το θέτει άμεσα συγκρίσιμο με προηγούμενες υλοποιήσεις σε περισσότερο εξειδικευμένα δίκτυα.

Η βασική διάφορα του Ethernet σε κάθε περίπτωση από τα υπόλοιπα δίκτυα είναι οι περιορισμένες δυνατότητες που προσφέρονται από το υλικό της κάρτας δικτύου. Λόγω του χαμηλού κόστους των καρτών δικτύου Ethernet, οι οποίες αποτελούν την πλέον συνήθη επιλογή για τοπική δικτύωση, δεν προσφέρεται από αυτές κανένα είδος προγραμματισμού τους από το χρήστη για εκτέλεση εξειδικευμένων λειτουργιών. Η συνολική λειτουργία τους βασίζεται σε προαποφασισμένες λειτουργίες που παρέχει η κάρτα προς το λογισμικό που καλείται να την διαχειρισθεί. Αυτό επιβάλει μια διαφορετική προσέγγιση στον τρόπο υλοποίησης ενός συστήματος βασισμένο σε υλικό δικτύου χαμηλού κόστους και το οποίο βασίζεται άμεσα στις δυνατότητες που προσφέρονται από την επικοινωνία μεταξύ περιφερειακών συσκευών και λειτουργικού συστήματος.

Χαρακτηριστικό των περισσοτέρων κοινών περιφερειακών συσκευών είναι η άμεση εξάρτησή τους από τον πυρήνα του λειτουργικού συστήματος. Η αλληλεπίδρασή τους με το λειτουργικό σύστημα επιτελείται μέσω αιτήσεων διακοπής οι οποίες και δηλώνουν την εμφάνιση κάποιου αξιοπρόσεχτου συμβάντος στην συσκευή. Ο πυρήνας του λειτουργικού συστήματος αφού λάβει γνώσει, μέσω της αίτησης διακοπής, για την ανάγκη διαχείρισης της συσκευής, λόγω αλλαγής στην λειτουργική κατάστασης της, καλεί μια δηλωμένη συνάρτηση που θα διερευνήσει και θα διαχειριστεί το αίτιο που προκαλέσει την αλλαγή αυτή. Η συνάρτηση αυτή ονομάζεται χειριστής αιτήσεων διακοπής και αποτελεί συνήθως κομμάτι του οδηγού

συσκευής της περιφερειακής συσκευής. Ο οδηγός συσκευής από την πλευρά του αποτελεί λειτουργικό κομμάτι του πυρήνα του λειτουργικού συστήματος, με ρόλο την σύνδεση του πυρήνα με την περιφερειακή συσκευή.

Κατά τον κλασσικό σχεδιασμό ενός λειτουργικού συστήματος, οι διεργασίες χρήστη όταν επιθυμούν να επικαλεστούν την λειτουργικότητα που προσφέρουν οι περιφερειακές συσκευές (αποστολή και λήψη πακέτων στην περίπτωση των καρτών δικτύου), χρειάζονται την διαμεσολάβηση του πυρήνα του λειτουργικού συστήματος. Το μέσο για την επικοινωνία διεργασιών χρήστη με τον πυρήνα αποτελούν οι κλήσεις συστήματος. Κατά την χρήση μιας κλήσης συστήματος, η εκτέλεση της διεργασίας σε επίπεδο χρήστη προσωρινά σταματάει και η εκτέλεση συνεχίζεται στον πυρήνα του λειτουργικού συστήματος πάνω στην υλοποίηση της κλήσης συστήματος. Μετά την ολοκλήρωση της κλήσης συστήματος η εκτέλεση συνεχίζεται στο χώρο της διεργασία χρήστη καθώς της παρέχονται τα αποτελέσματα της κλήσης συστήματος. Η διαμεταγωγή περιβάλλοντος που επιτελείται, από την περιοχή χρήστη σε περιοχή πυρήνα και στη συνέχεια σε περιοχή χρήστη, αποτελεί φόρτο κατά την χρήση κλήσεων συστήματος.

Στόχος των υλοποιήσεων για γρήγορη επικοινωνία σε δίκτυα Ethernet, είναι η αντιμετώπιση όλων των παραγόντων που επιφέρουν καθυστέρηση στην επικοινωνία, όπως αναφέρθηκαν νωρίτερα, προσαρμοσμένη στις ειδικές απαιτήσεις που επιβάλει η χρήση κοινών περιφερειακών καρτών δικτύου.

Υλοποιήσεις σε Fast Ethernet

Η πρώτη αξιόλογη προσπάθεια χρήσης ενός τοπικού δικτύου Fast Ethernet για ανταλλαγή μηνυμάτων με χαμηλή καθυστέρηση είναι το U-Net/FE [23], το οποίο και αποτελεί τη μεταφορά του U-Net από δίκτυο ATM σε Fast Ethernet.

Κύρια χαρακτηριστικά της υλοποίησης αποτελεί η αντιστοίχηση μνήμης από περιοχή χρήστη σε περιοχή πυρήνα ώστε να μειωθούν οι αντιγραφές των δεδομένων μεταξύ διαφορετικών περιοχών μνήμης. Με την αντιστοίχηση μνήμης η διεργασία χρήστη μπορεί άμεσα να προσπελάσει μέσω δεικτών περιοχές μνήμης του πυρήνα στις οποίες εναποτίθονται τα δεδομένα. Επιπλέον το πρωτόκολλο μεταφοράς των δεδομένων δεν είναι κάποιο από τα πρωτόκολλα του Διαδικτύου ή κάποιο άλλο γνωστό πρωτόκολλο, οπότε και δεν έχει νόημα η παράδοση τους στον πυρήνα για περαιτέρω επεξεργασία.

Το πρωτόκολλο που χρησιμοποιείται είναι ένα ιδιωτικό πρωτόκολλο υλοποιημένο από το U-Net/FE, το οποίο και διαχειρίζεται ένα άρθρωμα (module) στον πυρήνα του λειτουργικού συστήματος. Κατά την παραλαβή των δεδομένων ο οδηγός συσκευής, που υλοποιεί το άρθρωμα, φροντίζει τα δεδομένα να μεταφέρονται, μετά από ελέγχους, στην κατάλληλη περιοχή του πυρήνα ώστε να είναι πλέον άμεσα διαθέσιμα στην διεργασία στην οποία απευθύνονται. Για την μείωση του κόστους

κατά την μεταγωγή περιβάλλοντος από περιοχή διεργασίας χρήστη σε περιοχή πυρήνα, γίνεται χρήση ελαφρών κλήσεων συστήματος που προσφέρονται από το σύνολο εντολών της x86 αρχιτεκτονικής της Intel. Η υλοποίηση πετυχαίνει χρόνο 57 usec round trip κατά την μεταφορά μηνύματος 40 Byte μεταξύ δυο κόμβων.

Ακολουθώντας το μοντέλο του U-Net/FE παρουσιάστηκαν μεταγενέστερα νέες υλοποιήσεις στο ευρέως διαδεδομένο Fast Ethernet. Χρησιμοποιώντας όλες τις τεχνικές υλοποίησης, που επέδειξε το U-Net/FE, το οποίο βασίζεται στην αντιστοίχηση μνήμης, τη χρήση ιδιωτικών πρωτοκόλλων για μεταφορά μηνύματος και κλήσεις συστήματος χαμηλού κόστους, οι νεότερες υλοποιήσεις εμφανίζουν μια ελαφρά βελτίωση στην μέτρηση καθυστέρησης κατά την μεταφορά μηνύματος. Το κύριο κομμάτι που διαφοροποιούνται είναι ο τρόπος διαχείρισης της αντιστοιχισμένης περιοχής χρήστη στην οποία δημιουργούνται τα προς αποστολή μηνύματα και εναποτίθονται τα παραλαμβανόμενα μηνύματα. Τέτοιες υλοποιήσεις αποτελούν τα Directed Point [13], GAMMA [3] και PUPA [22]. Το Directed Point καταφέρνει χρόνο κοντά στα 55usec round trip για μήνυμα μεγέθους 40 Byte ενώ το GAMMA 61,4 (cold cache) και 36,8 (hot cache). Το PUPA αποτελεί υλοποίηση με έλεγχο ασφαλούς μεταφοράς των δεδομένων οπότε οι καθυστερήσεις που παρουσιάζονται είναι αρκετά υψηλότερες, της τάξης των 208 usec για μεταφορά προς μια κατεύθυνση μηνύματος 32 Byte.

Διαφορετική αντιμετώπιση του προβλήματος παρουσιάζουν οι υλοποιήσεις των MESH [1] και Bobnet [6]. Η υλοποίηση του MESH βασίζεται στην επικοινωνία μιας μόνο διεργασίας (η οποία υλοποιεί διάφορα threads) με την κάρτα δικτύου και βασίζεται σε βελτιστοποιήσεις του μεταφραστή και του χρονοδρομολογητή διεργασιών του συστήματος για την επιτυχίας χαμηλής καθυστέρησης. Η υλοποίηση του Bobnet ακολουθεί το προτεινόμενο πρότυπο του VIA για την υλοποίηση των συναρτήσεων αρχικοποίησης και ανταλλαγής δεδομένων και παρουσιάζει βελτιώσεις στην υλοποίηση του οδηγού συσκευής για την επίτευξη χαμηλής καθυστέρησης.

Υλοποιήσεις σε Gigabit Ethernet

Η εμφάνιση του Gigabit Ethernet αποτελεί τη φυσική εξέλιξη του Fast Ethernet σε ταχύτητες της τάξης του Gbps. Δεδομένης της συμβατότητας στην επικοινωνία των δυο δικτύων, το Gigabit Ethernet παρουσιάζεται σαν το επικρατέστερο μέσο δικτύωσης σε τοπικό επίπεδο για δικτυώσεις υψηλής ταχύτητας.

Αναγνωρίζοντας τις μεγάλες δυνατότητες που εμφανίζει το Gigabit Ethernet κατά την υιοθέτησή του ως μέσο δικτύωσης, υπάρχουσες υλοποιήσεις σε Fast Ethernet μεταφέρθηκαν σε προσαρμογείς δικτύου Gigabit Ethernet ώστε να διερευνήσουν τις επιδόσεις του νέου δικτύου. Οι υλοποιήσεις MESH και Bobnet εμφανίζουν μετρήσεις και σε Gigabit Ethernet εκτός από Fast Ethernet. Νέες υλοποιήσεις σε Gigabit Ethernet κάνουν χρήση προγραμματιζόμενων προσαρμογέων

δικτύου που παρέχει η εταιρία Alteon. Οι υλοποιήσεις αυτές μοιάζουν αρκετά με τις υλοποιήσεις σε Myrinet, δεδομένου ότι ο προσαρμογέας δικτύου εκτελεί ένα επαυξημένο σύνολο λειτουργιών για την επίτευξη της χαμηλής καθυστέρησης κατά την επικοινωνία. Τέτοιες υλοποιήσεις είναι οι Arsenic [17] και EMP [21].

2.2.4 Η δική μας υλοποίηση

Η υλοποίηση μας βασίζεται στις ίδιες επιδιώξεις με τις υπόλοιπες υλοποιήσεις πάνω σε Fast Ethernet. Χρησιμοποιώντας μια οικονομική κάρτα διασύνδεσης Fast Ethernet σε ευρέως χρησιμοποιούμενο υλικό συστήματος, επιδιώκουμε την επίτευξη χαμηλής καθυστέρησης κατά την μεταγωγή μηνύματος μεταξύ απομακρυσμένων κόμβων του δικτύου. Για την επίτευξη του στόχου μας λαμβάνουμε υπόψη τους παράγοντες που συντελούν στην καθυστέρηση κατά την διαμεταγωγή του μηνύματος και επιδιώκουμε την εξάλειψη τους ή τουλάχιστον την ελαχιστοποίηση τους.

Παρόλα αυτά, αν και κοινός ο στόχος με τις υπόλοιπες υλοποιήσεις που παρουσιάστηκαν νωρίτερα, το σύστημά μας παρουσιάζει χαρακτηριστικά που το διαφοροποιούν αρκετά από τις υπόλοιπες υλοποιήσεις. Βασικός άξονας της υλοποίησης μας είναι η διερεύνηση της απόδοσης που μπορεί να επιτευχθεί κατά την υλοποίηση ενός επιπέδου μεταφοράς μηνυμάτων γενικότερης χρηστικότητας για επικοινωνία με απομακρυσμένους κόμβου του δικτύου, χωρίς αποκλειστικό στόχο τις παράλληλες εφαρμογές.

Κύρια χαρακτηριστικά της υλοποίησης μας, πάνω στο στόχο που θέσαμε, είναι τα εξής:

1. *Συμβατότητα με την στοίβα πρωτοκόλλων UDP/IP*

   Ο σχεδιασμός του πρωτοκόλλου UDP, στο επίπεδο μεταφοράς δεδομένων του προτύπου OSI, επιδιώκει την ταχεία μεταφορά μηνύματος χωρίς σύνδεση σε απομακρυσμένους κόμβους του δικτύου. Η σχεδίαση του πρωτοκόλλου αποφεύγει την περίπλοκή διαχείριση των ανταλλασώμενων μηνυμάτων, με αντίτιμο την μη εξασφάλιση της παραλαβής τους. Παρά τα ελκυστικά χαρακτηριστικά στη σχεδίαση του πρωτοκόλλου, εμφανίζεται καθυστέρηση επιπλέον αυτής που εμφανίζεται λόγω των παραγόντων που αναφέραμε νωρίτερα. Παράγοντας καθυστέρησης αποτελεί η επιπλέον προσπέλαση των δεδομένων προς μεταφορά για την παραγωγή κώδικα εύρεσης λαθών μεταφοράς. Αυτό το χαρακτηριστικό εξασφάλιζε την ακεραιότητα των δεδομένων προς μεταφορά κατά την διάσχιση διαφορετικών δικτύων, αλλά είναι περιττό κατά την ανταλλαγή δεδομένων σε τοπικό δίκτυο. Ο λόγος είναι η υλοποίηση στο υλικό της κάρτας δικτύου ταχύτατου κώδικα εύρεσης λαθών οπότε, στην σπανιότατη περίπτωση λάθους, τα δεδομένα απορρίπτονται

έγκαιρα από χαμηλότερο επίπεδο χωρίς να φτάνουν λανθασμένα στο επίπεδο μεταφοράς.

Στην υλοποίηση μας ο χρήστης μπορεί να επιλέξει την χρήση UDP/IP σαν πρωτόκολλο επικοινωνίας. Σε αυτή την περίπτωση γίνεται χρήση ενός ελαφρού πρωτοκόλλου UDP/IP, το οποίο υλοποιείται σε βιβλιοθήκη επιπέδου χρήστη χωρίς την παρέμβαση του πυρήνα. Με την χρήση των πρωτοκόλλων UDP/IP, δίνεται η δυνατότητα στο χρήστη για επικοινωνία μεταξύ κόμβων γειτονικών τοπικών δικτύων αλλά και γενικότερα με οποιοδήποτε κόμβο του Διαδικτύου.

2. *Χρήση ιδιωτικά σχεδιασμένων πρωτοκόλλων σε επίπεδο χρήστη*

   Αναγνωρίζοντας την ανάγκη που παρουσιάζεται σε συγκεκριμένες εφαρμογές για την χρήση πρωτοκόλλων ελαφρού κόστους, το σύστημα μας επιτρέπει την επιλογή του Χρήστη για την μεταφορά μηνυμάτων βασιζόμενα σε ιδιωτικά πρωτόκολλα που υλοποιούνται σε επίπεδο διεργασίας χρήστη. Η διεργασία αρκεί να δηλώσει ένα μοναδικό αριθμό στο σύστημα, ο οποίος στη συνέχεια θα δεικνύει την χρήση ιδιωτικού πρωτοκόλλου κατά την μεταφορά μηνυμάτων. Με τον τρόπο αυτό αποφεύγεται κάθε επεξεργασία του μηνύματος από το σύστημα μεταφοράς του μηνύματος και η επεξεργασία του αναφύεται αποκλειστικά στην διεργασία χρήστη.

3. *Αποφυγή κλήσεων συστήματος*

   Η χρήση κλήσεων συστήματος στο κρίσιμο μονοπάτι αποστολής – λήψης δεδομένων, επιφέρει καθυστέρηση λόγω της απαραίτητης μεταγωγής περιβάλλοντος από επίπεδο χρήστη σε επίπεδο πυρήνα συστήματος. Στη υλοποίηση μας η αποστολή και λήψη δεδομένων γίνεται απευθείας στην συσκευή δικτύου από την περιοχή χρήστη μέσω της αντιστοίχησης μνήμης που έχει επιτελεστεί κατά την αρχικοποίηση της διεργασίας, σε αντίθεση με όλες τις προηγούμενες υλοποιήσεις για Fast Ethernet οι οποίες και χρησιμοποιούν ελαφρές κλήσεις συστήματος. Κλήσεις συστήματος στην υλοποίηση μας χρησιμοποιούνται μόνο σε μη κρίσιμα τμήματα της διεργασίας όπως π.χ κατά την αρχικοποίηση της επικοινωνίας διεργασίας με το επίπεδο μεταφοράς μηνυμάτων.

4. *Ταυτόχρονη χρήση προσαρμογέα μεταξύ πολλαπλών διεργασιών*

   Το σύστημα μεταφοράς μηνυμάτων υλοποιεί πολύπλεξη - απόπλεξη δεδομένων κατά την μεταφορά μηνυμάτων, επιτρέποντας την ταυτόχρονη χρήση της κάρτας δικτύου μεταξύ πολλαπλών διεργασιών. Για την αποφυγή συνθηκών ανταγωνισμού μεταξύ των διεργασιών κατά την αποστολή

μηνυμάτων, υλοποιήθηκε πρότυπος αλγόριθμος αμοιβαίου αποκλεισμού πολλαπλών διεργασιών αποκλειστικά με χρήση εντολών επιπέδου χρήστη, χωρίς κλήσεις συστήματος. Επιπλέον για την ενίσχυση της γενικότητας της χρήσης του συστήματος, επιτρέπεται η ανάμιξη χρήσης ιδιωτικών πρωτοκόλλων επιπέδου χρήστη και στοίβας πρωτοκόλλων UDP/IP.

5. *Υλοποίηση πραγματικής μηδενικής αντιγραφής μηνύματος*

   Με αντιστοίχηση μνήμης διεργασίας και πυρήνα λειτουργικού συστήματος και με χρήση δεικτών κατά την αποστολή μηνύματος επιτυγχάνεται πραγματική μηδενική αντιγραφή (zero-copy) των δεδομένων χρήστη κατά την αποστολή, σε αντίθεση με την πλειονότητα των υλοποιήσεων σε Fast Ethernet που υλοποιούν μοναδική αντιγραφή (single–copy) κατά την αποστολή (εκτός U-net/FE). Επιπλέον μηδενική αντιγραφή δεδομένων επιτυγχάνεται και κατά την λήψη δεδομένων μετά από επιλογή της διεργασίας παραλήπτη. Σε αυτή την περίπτωση διακινδυνεύεται η ακεραιότητα των λαμβανομένων δεδομένων σε περίπτωση πλήρωσης του χώρο υποδοχής. Διαφορετικά η διεργασία μπορεί να επιλέξει να υλοποιείται μοναδική αντιγραφή δεδομένων στο ιδιωτικό χώρο της, εξασφαλίζοντας με τον τρόπο αυτό τη διαφύλαξης των λαμβανομένων δεδομένων. Όλες οι υλοποιήσεις για Fast Ethernet υλοποιούν πάντα μοναδική αντιγραφή κατά την λήψη των δεδομένων, μη δίνοντας δυνατότητα επιλογής μηδενικής αντιγραφής στην διεργασία παραλήπτη.

Κεφάλαιο 3
Intel 82558 Ethernet Controller

3.1 Εισαγωγή

Ο ελεγκτής που χρησιμοποιήσαμε για την υλοποίηση της εργασίας είναι ο Intel 82558 που ανήκει στη οικογένεια Intel 8255x Ethernet Controllers [11]. Η κάρτα στην οποία βρισκόταν ο ελεγκτής αυτός είναι μια Intel Ether Pro100, PCI κάρτα δικτύου Ethernet 10/100 Mbps. Η συγκεκριμένη κάρτα είναι μια κλασσική, οικονομική και ευρέως χρησιμοποιούμενη λύση που προσφέρεται από την Intel για σύνδεση ενός υπολογιστή σε δίκτυο Ethernet 10/100 Mbps.

Η εργασία μας βασίζεται στη διερεύνηση μιας διαφορετικής προσέγγισης στη μεταγωγή μηνυμάτων πάνω από τοπικό δίκτυο Ethernet με στόχο την ελαχιστοποίηση της καθυστέρησης στη μεταγωγή μηνύματος μεταξύ των κόμβων του δικτύου. Μη θέλοντας να διαφοροποιηθούμε από την μεγάλη πλειοψηφία των συστημάτων και υλικού που χρησιμοποιούνται για την επίτευξη της συνδεσιμότητας σε τοπικό δίκτυο, χρησιμοποιήσαμε τη συγκεκριμένη κάρτα δικτύου. Κομμάτι της εργασίας είναι και η διερεύνηση της ελευθερίας που προσφέρει μια κοινή κάρτα δικτύου στον πειραματισμό για καλύτερη απόδοση και αξιοποίηση του δικτύου.

3.2 Απαιτήσεις Μνήμης

Για την εκμετάλλευση της λειτουργικότητας της κάρτας δικτύου και την αποστολή και λήψη δεδομένων στο δίκτυο, απαιτούνται κάποιες περιοχές μνήμης του συστήματος τις οποίες θα διαχειρίζονται από κοινού ο οδηγός συσκευής της κάρτας δικτύου και η κάρτα δικτύου καθαυτή. Οι περιοχές που απαιτούνται είναι οι εξής:

1. *Περιοχή αντιστοίχησης καταχωρητών κάρτας δικτύου (Control/Status Registers – CSR).*

   Για να προσπελάσει ο οδηγός συσκευής τους καταχωρητές αυτούς, επιτελείται μια αντιστοίχηση τους σε διευθύνσεις του χώρου διευθύνσεων του πυρήνα. Οι καταχωρητές αυτοί ελέγχουν τα τμήματα της κάρτας που είναι υπεύθυνα για

εκτέλεση εντολών ενεργειών (Control Unit – CU) και διαχείριση λαμβανομένων δεδομένων από το τοπικό δίκτυο (Receive Unit – RU).

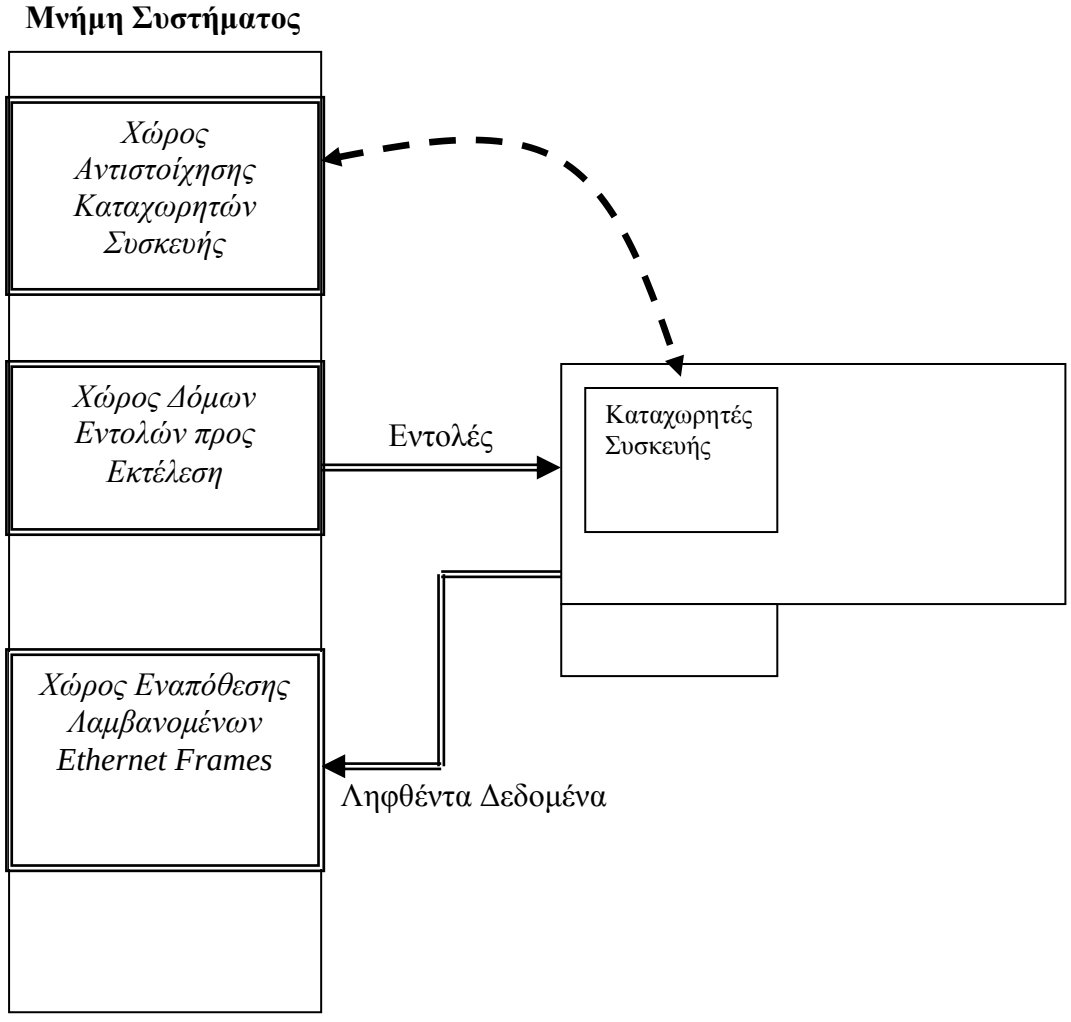
2. *Περιοχή αποθήκευσης εντολών προς εκτέλεση (Command Block List – CBL)*

   Στην περιοχή αυτή της μνήμης, που δεσμεύεται από τον οδηγό συσκευής της κάρτας δικτύου, διαμορφώνονται οι δομές με εντολές προς εκτέλεση από το Command Unit της κάρτας δικτύου (π.χ εντολή αποστολής Ethernet Frame).

3. *Περιοχή εναπόθεσης λαμβανομένων Ethernet Frames (Receive Frame Area – RFA)*

   Στην περιοχή αυτή της μνήμης, που δεσμεύεται από τον οδηγό συσκευής της κάρτας δικτύου, εναποτίθονται από Receive Unit της κάρτας δικτύου, σε κατάλληλα διαμορφωμένες δομές, τα Ethernet Frames που λαμβάνονται από το δίκτυο.

Σχηματικά οι περιοχές μνήμης που απαιτεί για την λειτουργία της η κάρτα δικτύου εμφανίζονται στο Σχήμα 3-1.



Σχήμα 3-1: Απαιτήσεις Μνήμης Ελεγκτή Intel 82558

3.3 Καταχωρητές Ελέγχου

Για να επιτευχθεί επικοινωνία μεταξύ φυσικής συσκευής και λειτουργικού συστήματος, γίνεται χρήση καταχωρητών που βρίσκονται πάνω στη φυσική συσκευή της κάρτας δικτύου. Γράφοντας και διαβάζοντας από τους καταχωρητές αυτούς μπορούμε να δώσουμε εντολές στην κάρτα και να πάρουμε δείγματα της κατάστασης της κάρτας αντίστοιχα. Το λειτουργικό σύστημα και οδηγοί συσκευών πρέπει να μπορούν να προσπελάσουν τις μνήμες αυτές που βρίσκονται στις περιφερειακές συσκευές του υπολογιστικού συστήματος. Δυο τρόποι προσφέρονται από τον πυρήνα του Linux για να γίνει εφικτό αυτό:

1. *Χρήση εντολών I/O*

   Για να χρησιμοποιηθούν εντολές τέτοιου τύπου θα πρέπει να αντιστοιχισθούν από το χώρο διευθύνσεων I/O, συγκεκριμένες διευθύνσεις στους καταχωρητές της περιφερειακής συσκευής. Τις διευθύνσεις αυτές τις έχει διαμοιράσει στις περιφερειακές συσκευές του συστήματος του πρότυπο διασύνδεσης PCI. Στη συνέχεια με χρήση εντολών που δέχονται διευθύνσεις I/O προσπελαύνονται οι καταχωρητές της περιφερειακής συσκευής.

2. *Χρήση εντολών προσπέλασης μνήμης*

   Ομοίως με τις διευθύνσεις I/O, το πρότυπο διασύνδεσης PCI έχει διαμοιράσει στις περιφερειακές συσκευές διευθύνσεις μνήμης που δεν έχουν φυσικό αντίκρισμα στην μνήμη του συστήματος. Μετά από καταχώρηση στον πυρήνα της αντιστοίχησης των διευθύνσεων αυτών, οι καταχωρητές της περιφερειακής συσκευής προσπελάζονται με εντολές προσπέλασης μνήμης.

Οι δυο τρόποι αυτοί μπορούν να χρησιμοποιηθούν κατ' εναλλαγή, αλλά συνήθως προτιμότερος είναι ο δεύτερος τρόπος. Για τον ελεγκτή 82558 οι καταχωρητές εμφανίζουν την εξής μορφή στον χώρο διευθύνσεων:

Upper Word 31 16	**Lower Word** 15 0	**Offset**
SCB Command Word	SCB Status Word	0h
SCB General Pointer		4h
PORT		8h
EEPROM Control Register	Reserved	Ch
MDI Control Register		10h
RX DMA Byte Count		14h

Σχήμα 3-2: Control/Status Registers – CSR

Εκτός των παραπάνω υπάρχουν και επιπλέον πεδία στην CSR περιοχή, τα οποία όμως δεν είναι άμεσα χρήσιμα στον οδηγό συσκευής μας. Ιδιαίτερα μας ενδιαφέρουν οι διευθύνσεις στο offset 0h και 4h. Στο offset 0h βρίσκεται το SCB (Status Control Block). Προσπελαύνοντας περιοχές μέσα στα 4 αυτά Bytes, ελέγχουμε τη λειτουργίες που προσφέρει ο ελεγκτής της κάρτας δικτύου. Διαβάζοντας τα πρώτα 16 bits του SCB μπούμε να πάρουμε πληροφορίες για την κατάσταση του ελεγκτή της κάρτας μας (SCB Status Word). Με τα επόμενα 16 bits δίνουμε εντολές ελέγχου στον ελεγκτή της κάρτας (SCB Command Word). Στο offset 4h (SCB General Pointer) καταχωρούμε φυσικές διεύθυνσης του συστήματος, σε περίπτωση που απαιτούνται από εντολές που γράφουμε στο SCB Command Word.

Τη συνολική λειτουργία της κάρτας μπορούμε να διαχωρίσουμε σε δύο τμήματα:

1. Το τμήμα που είναι υπεύθυνο για τη λήψη των Ethernet Frames από το δίκτυο και την τοποθέτηση τους στη μνήμη του συστήματος (Receive Unit – RU).
2. Το τμήμα που είναι υπεύθυνο για την εκτέλεση των εντολών του χρήστη που παρέχονται στην κάρτα (Command Unit – CU).

Γράφοντας στο SCB Command Word, ο οδηγός συσκευής δίνει εντολές ελέγχου στα δυο τμήματα (CU και RU)του ελεγκτή. Διαβάζοντας από το SCB Status Word, ο οδηγός συσκευής παίρνει πληροφορίες για την κατάσταση των τμημάτων αυτών. Επιπλέον από το SCB Status Word, ο οδηγός συσκευής λαμβάνει γνώση για το αίτιο που οδήγησε την κάρτα δικτύου να ενεργοποιήσει την IRQ Line, οπότε και να στείλει στον πυρήνα μια αίτηση διακοπής. Για την επιβεβαίωση της λήψης του IRQ ο οδηγός συσκευής γράφει πάνω στο bit που παρουσιάζουν την αιτία του IRQ, ώστε να είναι δυνατή η αποστολή νέας αίτησης διακοπής από την κάρτα δικτύου.

Με περισσότερη λεπτομέρεια το SCB 32bit Dword εμφανίζεται ως εξής:

15 8	7 6	5 2	1	0
Status/Acknowledge IRQ	CU Status	RU Status	0	0

Σχήμα 3-3: SCB Status Word

31 26	25	24	23 20	19	18 16
Specific Interrupt Mask Bits	SI	M	CU Command	0	RU Command

Σχήμα 3-4: SCB Command Word

3.4 Οργάνωση περιοχών μνήμης

3.4.1 Περιοχή εντολών

Εκτός από τις εντολές που θέτουμε στο Status Control Block (SCB) του Control Status Register (CSR) για τον έλεγχο των μηχανών του Command Unit (CU) και Receive Unit (RU) η κάρτα δικτύου υποστηρίζει εντολές για υλοποίηση βασικών αναγκών, όπως η αποστολή δεδομένων στο δίκτυο με τη μορφή Ethernet Frames, καθώς και ανάγκες ελέγχου και αρχικοποίησης της. Οι εντολές αυτής της κατηγορίας τοποθετούνται στη Λίστα Εντολών (Command Block List – CBL) που βρίσκεται σε μια περιοχή μνήμης του πυρήνα του υπολογιστικού συστήματος και η οποία δεσμεύεται από τον οδηγό συσκευής της κάρτας δικτύου. Η κάθε εντολή που δημιουργείται προς εκτέλεσης περιλαμβάνεται στη λίστα σαν μια δομή από πεδία παραμέτρων που αφορούν την εκτέλεση της εντολής και ονομάζεται Δομή Εντολής (Command Block - CB).

Οι εντολές που αναγνωρίζει η CU μηχανή του ελεγκτή της κάρτας δικτύου υλοποιούνται σαν μια ακολουθία από πεδία με καθορισμένη σημασία για κάθε εντολή. Βασικό πεδίο αποτελεί ο κωδικός εντολής (CMD), ο οποίος και δεικνύει τη σημασία των υπολοίπων πεδίον της δομής βάση του τύπου της εντολής. Οι εντολές τοποθετούνται στο CBL και διαβάζονται από τη CU μηχανή του ελεγκτή κάρτας δικτύου, η οποία αφού αναγνωρίσει τον τύπο της εντολής μέσο του πεδίου CMD, αναλαμβάνει να την εκτελέσει βάση των παραμέτρων που έχουν τεθεί για την εντολή στη δομή. Με το πέρας της εντολής επιπλέον ενέργειες καθορίζονται από τιμές που έχουν τοποθετηθεί στα πεδία της δομής της εντολής. Τα πεδία των εντολών, κοινά για όλες τις εντολές που υποστηρίζονται από τον ελεγκτή της κάρτας δικτύου είναι τα εξής:

Offset	Command Word Bits 31:16					Status Word Bits 15:0			
00h	EL	S	I	0000000000	CMD	C	X	OK	XXXXXXXXXXXX
04h	Link Offset								
08h	Optional Address and Data Fields								

Σχήμα 3-5: Επικεφαλίδα Δομή Εντολής

Στη συνέχεια ακολουθούν πεδία που ορίζουν ειδικές παραμέτρους και δεδομένα που δέχεται η κάθε εντολή.

Έναρξη εκτέλεσης εντολών

Μια γενική ακολουθία βημάτων για την εκτέλεση κάθε εντολής στο CBL, από τη μηχανή CU του ελεγκτή της κάρτας δικτύου, είναι τα εξής:

1. Ο ελεγκτής κάρτας δικτύου διαβάζει τα 16 πρώτα Bytes στο CB της εντολής προς εκτέλεση.
2. Στη συνέχεια γίνεται ανάλυση των bits στο πεδίο CMD στη δομή της εντολής, ώστε να βρεθεί ο τύπος της εντολής προς εκτέλεση.
3. Αποθηκεύεται το Link Offset, ώστε να είναι γνωστή η διεύθυνση της επόμενης εντολής στο CBL. Όταν υπάρχει επόμενη εντολή μεταφέρονται πριν τελειώσει η εκτέλεση της παρούσας εντολής και αποθηκεύονται τα πεδία: EL, S, I και CMD. Με τον τρόπο αυτό είναι άμεση η γνώση για τυχόν ύπαρξη επόμενης εντολής με το πέρας της εκτέλεσης της παρούσας εντολής.
4. Εκτελείται η παρούσα εντολή, βάση των παραμέτρων που έχουν τεθεί στα πεδία που αναγνωρίζει η εντολή μέσα στο Command Block (CB).

Πέρας εκτέλεσης εντολής ενεργειών

Το πέρας εκτέλεσης μιας εντολής που έχει τεθεί στο CBL είναι ασύγχρονο σε σχέση με το χρόνο έναρξης της. Ο λόγος είναι ότι ο χρόνος εκτέλεσης μιας εντολής είναι άμεσα εξαρτώμενος από διάφορους παράγοντες όπως ο τύπος της εντολής, ένταση δραστηριότητας του Receive Unit (RU), καθυστέρηση του διαύλου επικοινωνίας PCI. Για όλο το χρόνο εκτέλεσης της εντολής το Command Unit (CU) βρίσκεται σε κατάσταση ενεργή (ACTIVE). Η ακόλουθη διαδικασία εκτελείται και προσδιορίζει την κατάσταση του CU μετά το πέρας της παρούσης εντολής:

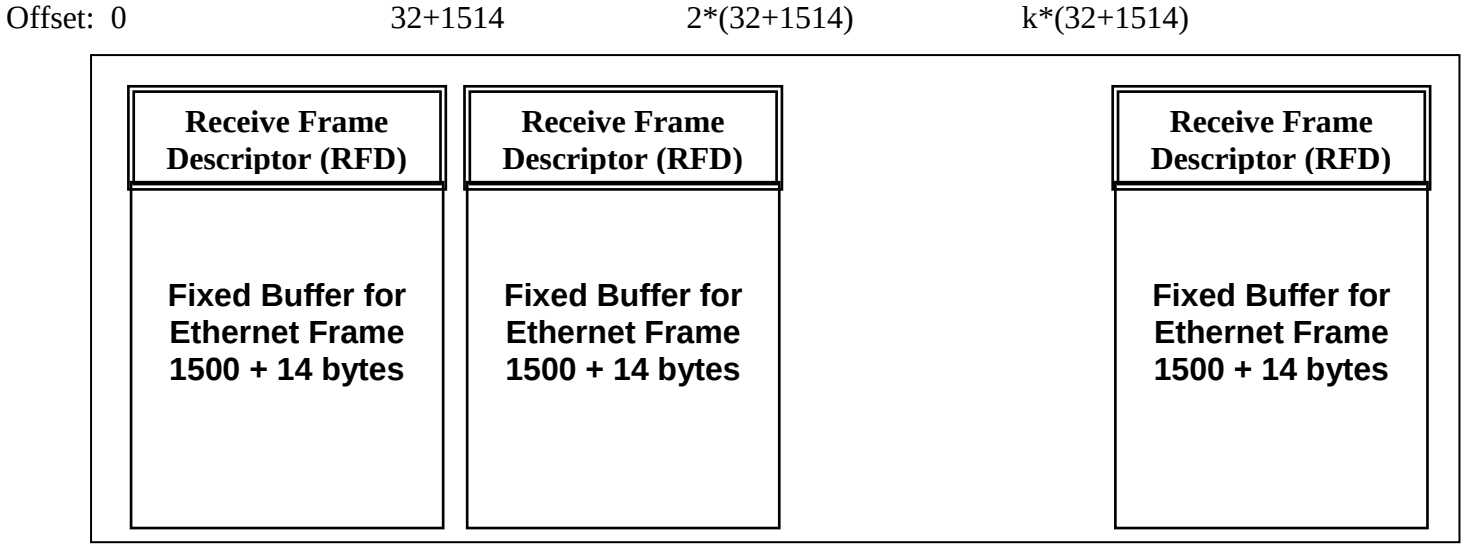
1. Γράφονται τα bits C και OK στην επικεφαλίδα της δομής της εντολής στο CB δηλώνοντας πέρας της εντολής και επιτυχής εκτέλεση της αντίστοιχα.
2. Αν έχει τεθεί το I bit στην επικεφαλίδα της δομής της εντολής τότε στέλνεται μια αίτηση διακοπής στον πυρήνα του λειτουργικού συστήματος (CX IRQ).
3. Αν έχει τεθεί το S bit στην επικεφαλίδα της δομής της εντολής, το CU μεταβαίνει από ενεργή (ACTIVE) κατάσταση σε κατάσταση διαθεσιμότητας (SUSPENDED). Αν η συσκευή έχει αρχικοποιηθεί κατάλληλα ο ελεγκτής στέλνεται μια αίτηση διακοπής στον πυρήνα του λειτουργικού συστήματος (CNA IRQ) κατά την μετάβαση αυτή .
4. Αν έχει τεθεί το EL bit στην επικεφαλίδα της δομής της εντολής δηλώνει το συγκεκριμένη δομή εντολής CB σαν τελευταία στη λίστα εντολών CBL προς εκτέλεση. Σε αυτή την περίπτωση στέλνεται μια αίτηση διακοπής στον πυρήνα του λειτουργικού συστήματος (CNA IRQ) δηλώνοντας τη μετάβαση του CU από ενεργή κατάσταση (ACTIVE) σε ανενεργή κατάσταση (IDLE).

5. Η συσκευή ανανεώνει τα CU Status bits στο SCB του ελεγκτή, υποδηλώνοντας αλλαγή κατάστασης του CU.

3.4.2 Δομή περιοχής λήψης Ethernet Frames

Σε μια περιοχή μνήμης (Receive Frame Area - RFA) του πυρήνα του λειτουργικού συστήματος, η οποία δεσμεύεται από τον οδηγό συσκευής της κάρτας δικτύου, διαμορφώνονται κατάλληλοι χώροι αποθήκευσης για συγκράτηση των δεδομένων που προέρχονται από το τοπικό δίκτυο με τη μορφή Ethernet Frames. Οι χώροι διαμορφώνονται πριν ενεργοποιηθεί η RU μηχανή της κάρτας δικτύου και η φυσική διεύθυνση μνήμης του χώρου αυτού παρέχεται μέσω εντολής στο SCB Command Word στον ελεγκτή της κάρτας.

Για τον κάθε χώρο αποθήκευσης που διαμορφώνεται, ενεργοποιούνται κάποια πεδία που δεικνύουν δυνατότητες του συγκεκριμένου χώρου αποθήκευσης καθώς και ένα πεδίο 4 Bytes που δεικνύει την αρχή του αμέσως επομένου χώρου αποθηκεύσεις. Τα πεδία αυτά, τα οποία βρίσκονται στα πρώτα bytes του χώρου αποθηκεύσεις, αποτελούν τον περιγραφέα πλαισίου (Receive Frame Descriptor – RFD). Αμέσως μετά τον περιγραφέα πλαισίου ακολουθεί κενός χώρος μέσα στον οποίο αποθηκεύεται αυτούσιο το λαμβανόμενο Ethernet Frame. Με τον τρόπο αυτό σχηματίζεται μια λίστα από διαμορφωμένους χώρους αποθήκευσης, στους οποίους η RU μηχανή αποθηκεύει τα λαμβανόμενα δεδομένα. Η μορφή του χώρου αποθήκευσης δεδομένων για κάθε Ethernet Frame είναι η εξής:



Σχήμα 3-6: Receive Frame Area

Η μορφή του περιγραφέα πλαισίου (Receive Frame Descriptor – RFD) για κάθε διαμορφωμένο χώρο αποθήκευσης είναι η εξής:

Offset	Command Word Bits 31:16						Status Word Bits 15:0			
00h	EL	S	000000000	H	SF	000	C	0	OK	Status Bits
04h	Link Address									
08h	Optional Address and Data Fields									
0Ch	0	0	Size				EOF	F	Actual Count	

Σχήμα 3-7: Περιγραφέας Πλαισίου Χώρου Λήψης

Στην υλοποίηση μας για βελτιστοποίηση της πρόσβασης μνήμης από την κάρτα δικτύου, οι περιοχές εναπόθεσης λαμβανομένων δεδομένων, για κάθε ένα από τους περιγραφείς πλαισίου, είναι στοιχισμένη στα 32 bytes, όσο είναι δηλαδή το μέγεθος της γραμμής κρυφής μνήμης του συστήματος μας.

3.5 Βασικές εντολές ελεγκτή Intel 82558

3.5.1 Εντολή αρχικοποίησης MAC διεύθυνσης

Η κάρτα δικτύου με το που ξεκινάει τη λειτουργία της μετά από μια επανεκκίνηση του υπολογιστικού συστήματος (η μια επανεκκίνηση της κάρτας μέσω κατάλληλής εντολής σε αυτή), αποκτά σαν Πηγαία Ethernet διεύθυνση (Ethernet Source Address) την διεύθυνση πολλαπλής αποστολής (Broadcast Address) για το δίκτυο Ethernet. Παρόλα αυτά πρέπει στην κάρτα να τεθεί μοναδική διεύθυνση για να επιτευχθεί επικοινωνία μεταξύ ζευγαριού κόμβων στο τοπικό δίκτυο. Η διεύθυνση που έχει ανατεθεί μοναδικά στη κάρτα από τον κατασκευαστή της είναι αποθηκευμένη σε μια EEPROM που βρίσκεται πάνω στη κάρτα. Όταν διαβάσουμε τα 6 πρώτα bytes αυτής της μνήμης έχουμε την διεύθυνση που έχει ανατεθεί στη κάρτα. Επόμενο βήμα είναι η ενημέρωση του ελεγκτή κάρτας για τη μοναδική διεύθυνση που θα χρησιμοποιεί για την λήψη και αποστολή Ethernet Frames. Αυτό επιτυγχάνεται με τη χρήση της εντολής Individual Address Setup (CMD: 001) που υποστηρίζει ο ελεγκτής της Κάρτας δικτύου για αυτό το σκοπό. Η μορφή της εντολής είναι η ακόλουθη:

Offset	Command Word Bits 31:16					Status Word Bits 15:0			
00h	EL	S	I	0000000000	001	C	X	OK	XXXXXXXXXXXXX
04h	Link Address								
08h	IA 4th Byte, IA 3rd Byte					IA 2nd Byte, IA 1st Byte			
						IA 6th Byte, IA 5th Byte			

Σχήμα 3-8: Δομή Εντολής Αρχικοποίησης MAC Διεύθυνσης

Individual Address: Η παράμετρος που πρέπει να ορισθεί για την εντολή αυτή είναι τα 6 bytes που θα περιέχουν τη MAC διεύθυνση της κάρτας δικτύου. Τα bytes αυτά αριθμούνται με τη σειρά μου θα εμφανιστεί η MAC Source Address στο δίκτυο κατά την αποστολή Ethernet Frames. Αν π.χ η διεύθυνση που πρέπει να καταχωρηθεί στον ελεγκτή είναι 00 AA 00 01 02 03h τότε θα πρέπει να τεθούν οι παράμετροι της εντολής ως εξής:
IA Byte 1 00h
IA Byte 2 AAh
IA Byte 3 00h
IA Byte 4 01h
IA Byte 5 02h
IA Byte 6 03h

3.5.2 Εντολή αρχικοποίησης παραμέτρων ελεγκτή κάρτας δικτύου

Χρησιμοποιώντας την εντολή αρχικοποίησης, ο οδηγός συσκευής ορίζει τις παραμέτρους λειτουργίας της κάρτας δικτύου. Η εντολή αρχικοποίησης δέχεται σαν παραμέτρους 22 Bytes στα οποία και ορίζονται πεδία με τις παραμέτρους της συσκευής. Η μορφή της εντολής είναι η ακόλουθη:

Offset	Command Word Bits 31:16					Status Word Bits 15:0			
00h	EL	S	I	0000000000	010	C	X	OK	XXXXXXXXXXXXX
04h	Link Address								
08h	Byte 3			Byte 2		Byte 1			Byte 0
0Ch	Byte7			Byte 6		Byte 5			Byte 4
10h	Byte 11			Byte 10		Byte 9			Byte 8
14h	Byte 15			Byte 14		Byte 13			Byte 12
18h	Byte 19			Byte 18		Byte 17			Byte 16
1Ch	00 00 00 00			00 00 00 00		Byte 21			Byte 20

Σχήμα 3-9: Δομή Εντολής Αρχικοποίησης Παραμέτρων Ελεγκτή

Για περισσότερες πληροφορίες για τις παραμέτρους της εντολής στο [11] καθώς και στο παράρτημα Α.

3.5.3 Εντολή αποστολής δεδομένων (Transmit Command)

Για την αποστολή δεδομένων σε απομακρυσμένους κόμβους του δικτύου ο χρήστης πρέπει να δημιουργήσει μια δομή εντολής αποστολής δεδομένων και να την εντάξει στην λίστα εντολών προς εκτέλεση. Η δομή εντολής που πρέπει να δημιουργήσει ο χρήστης αρχικά περιλαμβάνει την επικεφαλίδα της εντολής, η οποία και έχει την ακόλουθη μορφή:

Offset	Command Word Bits 31:16								Status Word Bits 15:0				
00h	EL	S	I	CID	000	NC	SF	100	C	X	OK	U	XXXXXXXXXXX
04h	Link Address												
08h	Transmit Buffer Descriptor Array Address												
0Ch	TBD Number				Transmit Threshold				EOF	0	Transmit Block Byte Count		

Σχήμα 3-10: Δομή Εντολής Απλουστευμένης Αποστολής Δεδομένων

Ακολούθως της επικεφαλίδας ο χρήστης καταχωρεί τα δεδομένα που θα σταλούν στο δίκτυο με την μορφή ενός Ethernet Frame. Τα διάφορα πεδία που περιέχονται στην επικεφαλίδα της εντολής αποστολής έχουν την εξής σημασία:

EL: Ενεργοποιώντας το bit αυτό δεικνύεται η δομή της παρούσας εντολής σαν η τελευταία στο CBL. Η Command Unit (CU) μηχανή θα μεταβεί από ενεργή (ACTIVE) σε ανενεργή κατάσταση (IDLE) μετά την εκτέλεση της εντολής. Η μετάβαση αυτή δημιουργεί μια αίτηση διακοπής (CNA IRQ) προς το λειτουργικό σύστημα.

S: Ενεργοποιώντας το bit η CU μηχανή θα περάσει με το πέρας της τρέχουσας εντολής από την ενεργή κατάσταση (ACTIVE) σε κατάσταση αναστολής (SUSPENDED). Αίτηση διακοπής (CNA IRQ) προς το λειτουργικό σύστημα θα σταλεί μόνο στην περίπτωση που έχει ορισθεί η παράμετρος αυτή στον ελεγκτή της κάρτας δικτύου.

I: Ενεργοποιώντας το bit αυτό στέλνεται μια αίτηση διακοπής (CX IRQ) στον πυρήνα του λειτουργικού συστήματος μετά το πέρας της τρέχουσας εντολής.

CID: Το πεδίο αυτό δεικνύει το χρόνο καθυστέρησης για την αποστολή του CNA IRQ στον πυρήνα του λειτουργικού συστήματος.

NC: Αν τεθεί 0 η MAC διεύθυνση και το CRC του Ethernet Frame τοποθετούνται αυτόματα από τον ελεγκτή της κάρτας δικτύου.

Αν τεθεί 1 δεν προσθέτονται MAC διεύθυνση και CRC στο Ethernet Frame, υποθέτοντας ότι αυτά περιέχονται ήδη στο Ethernet Frame που έχει τοποθετηθεί στη δομή με την εντολή αποστολής.

SF: Αν τεθεί το bit αυτό 0 υλοποιείται ο τρόπος «Απλουστευμένης Αποστολής» δεδομένων. Αν τεθεί το bit αυτό 1 υλοποιείται ο τρόπος «Ευέλικτης Αποστολής» δεδομένων.

C: Το bit αυτό ενεργοποιείται όταν όλα τα δεδομένα που ΄χουν τοποθετηθεί στη δομή, έχουν προσπελαστεί από την CU μηχανή.

OK: Το bit αυτό ενεργοποιείται αν η αποστολή των δεδομένων που περιέχονται στη δομή, αποστάλθηκαν επιτυχώς. Διαφορετικά αν το C bit είναι ενεργοποιημένο και το OK bit ανενεργό δεικνύει λάθος κατά την αποστολή των δεδομένων στο δίκτυο.

U: Το bit αυτό δεικνύει ότι εμφανίστηκε underrun κατά την αποστολή αυτού η προηγούμενων Ethernet Frames.

Transmit Threshold: Ο αριθμός αυτός πολλαπλασιαζόμενος με το 8 δεικνύει τον αριθμό των bytes που πρέπει να βρίσκονται στη FIFO περιοχή του ελεγκτή κάρτας δικτύου, πριν αρχίσει η αποστολή τους στο δίκτυο. Η αριθμοί αυτοί κυμαίνονται από 1 έως 0E0h.

EOF: Το bit αυτό δεικνύει ότι όλα τα δεδομένα προς αποστολή βρίσκονται στη δομή που έχει οριστεί με αυτή την επικεφαλίδα. Το bit αυτό ενεργοποιείται από το χρήστη με κριτήριο τη συνέπεια, αν και δεν ελέγχεται από τον ελεγκτή της κάρτας δικτύου.

TCB Byte Count: Το πεδίο αυτό μεγέθους 14 bit δεικνύει το μέγεθος των δεδομένων που έχουν τοποθετηθεί αμέσως μετά την επικεφαλίδα της εντολής αποστολής δεδομένων.

Όπως επισημάναμε με τον τρόπο αυτό ο ελεγκτής κάρτας δικτύου αναγνωρίζει από το πεδίο CMD=100 (16-18 bits) ότι πρόκειται για εντολή αποστολής δεδομένων με τη μορφή Ethernet Frame στο δίκτυο και αποστέλλει δεδομένα μεγέθους TCB Byte Count που έχουν τοποθετηθεί από το χρήστη ακολούθως της επικεφαλίδας της εντολής (offset 10h). Για κάθε Ethernet Frame προκειμένου να αποσταλεί στο δίκτυο πρέπει να δημιουργηθεί μια εντολή αποστολής δεδομένων στο CBL. Αυτός ο τρόπος αποστολής δεδομένων αποτελεί την «***Απλουστευμένη Αποστολή***» (Simplified).

Ο δεύτερος εναλλακτικός τρόπος για την αποστολή δεδομένων ονομάζεται «***Ευέλικτη Αποστολή***» (flexible) και ενεργοποιείται θέτοντας το πεδίο SF της επικεφαλίδας εντολής αποστολής ίσο με 1. Αυτή η πολιτική αποστολής δεδομένων δεν υποχρεώνει την τοποθέτηση των δεδομένων μέσα στην δομή της εντολής αποστολής δεδομένων στη CBL. Τα δεδομένα μπορεί να βρίσκονται σε οποιαδήποτε λογική διεύθυνση του πυρήνα του λειτουργικού συστήματος. Αρκεί η τοποθέτηση της διεύθυνσης μνήμης, στην οποία βρίσκονται τα δεδομένα, καθώς και το μέγεθος αυτών, σε ένα πίνακα και τοποθέτησης της διεύθυνσης του πίνακα αυτού μέσα στην επικεφαλίδα της εντολής αποστολής δεδομένων, στο πεδίο TBD Array Address. Ο

πίνακας που έχει σχηματιστεί από το χρήστη και περιέχει τις διευθύνσεις και τα μεγέθη των δεδομένων προς αποστολή επιτρέπεται να περιέχει πολλές καταχωρήσεις, οπότε και το τελικό Ethernet Frame θα δημιουργηθεί από διαδοχική εναπόθεση δεδομένων με τη σειρά που οι διευθύνσεις τους εμφανίζονται στον πίνακα. Ο πίνακας αυτός έχει τη μορφή:

Odd Word (Bits 31:16)			**Even Word (Bits 15:0)**
Transmit Buffer #0 Address			
0000000000000000	EL	0	Size (Actual Count)
Transmit Buffer #1 Address			
0000000000000000	EL	0	Size (Actual Count)
Transmit Buffer #2 Address			
0000000000000000	EL	0	Size (Actual Count)

Σχήμα 3-11: Δομή TBD Array

Ανά δυο γραμμές του πίνακα αποτελούν ένα TBD (Table Buffer Descriptor). Τα πεδία που αποτελούν το κάθε TBD είναι τα εξής:

Transmit Buffer #N: Η διεύθυνση από την οποία θα αρχίσει η ανάγνωση των δεδομένων προς αποστολή. Αποτελεί μια 32bit φυσική διεύθυνση του υπολογιστικού συστήματος.

EL: Ενεργοποιώντας το bit αυτό επισημαίνεται το τελευταίο TBD για τον πίνακα περιγραφέων που συνολικά δημιουργούν το Ethernet Frame προς αποστολή.

Size: Το μέγεθος των δεδομένων που θα ανακληθούν από τη περιοχή μνήμης αρχίζοντας από την διεύθυνση που αναγράφεται στο TBD.

Ο αριθμός των TBDs μέσα στον πίνακα αυτό, σημειώνεται στο πεδίο TBD Number της επικεφαλίδας εντολής αποστολής.

Τέλος μπορούμε να χρησιμοποιήσουμε ένα τρόπο αποστολής δεδομένων που χρησιμοποιεί τις δυνατότητες της «Ευέλικτης Αποστολής», αλλά παρέχει άμεση πρόσβαση στα δυο πρώτα TBD του πίνακα διευθύνσεων των δεδομένων προς αποστολή. Ο τρόπος αυτός ονομάζεται «Επαυξημένη Ευέλικτη Αποστολή» και ενεργοποιείται μέσω της εντολής Configure του ελεγκτή της κάρτας δικτύου κατά την αρχικοποίηση της λειτουργίας του. Η «Επαυξημένη Ευέλικτη Αποστολή» επαυξάνει την κεφαλίδα εντολής αποστολής κατά 16 bytes, στα οποία και υλοποιείται ένας TBD πίνακας διευθύνσεων με 2 μόνο θέσεις. Σε αυτή την περίπτωση η κεφαλίδα της εντολής αποστολής έχει την εξής μορφή:

Offset	Command Word Bits 31:16								Status Word Bits 15:0				
00h	EL	S	I	CID	000	NC	SF	100	C	X	OK	U	XXXXXXXXXXXX
04h	Link Address												
08h	Transmit Buffer Descriptor Array Address												
0Ch	TBD Number				Transmit Threshold				EOF	0	Transmit Block Byte Count		
10h	Transmit Buffer #0 Address												
14h	0000000000000000								EL	0	Size (Actual Count)		
18h	Transmit Buffer #1 Address												
1Ch	0000000000000000								EL	0	Size (Actual Count)		

Σχήμα 3-12: Δομή Εντολής Επαυξημένης Αποστολής Δεδομένων

Με τον τρόπο αυτό δεν χρειάζεται ξεχωριστή διεύθυνση μνήμης για υλοποίηση του TBD πίνακα αν αυτός περιέχει μόνο 2 TBDs. Αν παρόλα αυτά χρειάζονται περισσότερα TBDs η αρχική διεύθυνση των επιπλέον TBDs τοποθετείται, όπως περιγράφηκε νωρίτερα, στο TBD Array Address της επικεφαλίδας της εντολής αποστολής δεδομένων. Τη δυνατότητα της «Επαυξημένης Ευέλικτης Αποστολής» χρησιμοποιούμε σαν βασικό τρόπο αποστολής στην υλοποίηση του οδηγού συσκευής μας.

3.6 Λοιπά χαρακτηριστικά

Ο ελεγκτής Intel 82558 υποστηρίζει επιπλέον χαρακτηριστικά και εντολές προς εκτέλεση που δεν χρησιμοποιήθηκαν στην υλοποίηση μας ή χρησιμοποιήθηκαν για λόγους αποσφαλμάτωσης. Για την αποσφαλμάτωση του οδηγού συσκευής χρησιμοποιήθηκε η εντολή Dump, η οποία αντιγράφει σε χώρο μνήμης πυρήνα το περιεχόμενο των καταχωρητών του ελεγκτή. επιπλέον υποστηρίζονται από τον ελεγκτή ένα σύνολο από εντολές οι οποίες ενεργοποιούνται γράφοντας τον κατάλληλο κωδικό τους στο πεδίο Port του CSR. Χρήσιμη εντολή της κατηγορίας αυτής είναι η Selective Reset, η οποία καλείται κατά την δυναμική αφαίρεση του αρθρώματος από τον πυρήνα. Περισσότερες λεπτομέρειες μπορεί κανείς να βρει στο εγχειρίδιο του ελεγκτή [11].

Κεφάλαιο 4
Αρθρώματα και Οδηγός Συσκευής

4.1 Εισαγωγή στα αρθρώματα

Τα αρθρώματα (modules) [2], [20] αποτελούν ένα ευέλικτο τρόπο για την δυναμική φόρτωση λειτουργιών στον πυρήνα του λειτουργικού συστήματος, με συνέπεια την επαύξηση της λειτουργικότητας που αυτός προσφέρει στους χρήστες. Οι λειτουργίες αυτές κυμαίνονται από μια απλή διάσχιση δομών του λειτουργικού συστήματος για συλλογή στοιχείων, έως τη φόρτωση ενός οδηγού συσκευής ώστε να υποστηριχτεί η λειτουργικότητα μιας συσκευής του υπολογιστικού συστήματος. Η δυνατότητα να υλοποιούμε οδηγούς συσκευών με τη χρήση αρθρώματα μας δίνει την ευχέρεια να τους ενεργοποιούμε μόνο όταν συντρέχουν συγκεκριμένες συνθήκες στο λειτουργικό σύστημα. Με τον τρόπο αυτό δεν επιβαρύνεται η αρχική ενεργοποίηση του πυρήνα με πληθώρα λειτουργιών που στο τέλος πιθανό να αποδειχθούν περιττές. Εκτός αυτού η χρήση modular οδηγών συσκευών έχει αρκετά πλεονεκτήματα σε σχέση με τον κλασσικό τρόπο φόρτωσης οδηγών συσκευών:

1. Με την χρήση modular οδηγών συσκευών γίνεται εξοικονόμηση των αριθμών αιτήσεων διακοπής (Interrupt Request), ένας πεπερασμένος πόρος σε πολλά συστήματα υπολογισμού (15 για αρχιτεκτονική προσωπικών υπολογιστών). Με τη δυναμική φόρτωση modular οδηγών συσκευών, μια συσκευή δεσμεύει ένα αριθμό αίτησης διακοπής για όσο χρόνο θα διαρκέσει η χρήση της. Κατά την αφαίρεση του οδηγού συσκευής ο αριθμός αίτησης διακοπής αποδεσμεύεται και ένας νέος οδηγός συσκευής είναι ελεύθερος να τον χρησιμοποιήσει. Το σύστημα μπορεί με αυτό τον τρόπο να έχει πολλές περισσότερες συσκευές που θα χρησιμοποιεί μη ταυτόχρονα.
2. Ο προγραμματισμός και η αποσφαλμάτωση modular οδηγών συσκευής είναι ασύγκριτα ευκολότερος από τον παραδοσιακό τρόπο. Η φόρτωση και η αφαίρεση τους είναι δυναμική κατά τον χρόνο λειτουργίας του συστήματος χωρίς καν να απαιτείται επανεκκίνηση του. Ο κλασσικός τρόπος αντίθετα απαιτεί τη ενημέρωση αρχείων για τον οδηγό συσκευής που θέλουμε να εισάγουμε στο σύστημα και επαναμετάφραση του πυρήνα του συστήματος για συμπερίληψη του. Μετά από επανεκκίνηση του συστήματος ο οδηγός

συσκευής θα εισαχθεί στο σύστημα κάτι ιδιαιτέρα χρονοβόρο ειδικά αν ο οδηγός συσκευής περιέχει κάποιο σφάλμα.

3. Ο πυρήνας του λειτουργικού συστήματος κατά την εγκατάσταση του σε ένα νέο υπολογιστικό σύστημα πρέπει να είναι σε θέση να υποστηρίξει μια μεγάλη γκάμα από περιφερειακές συσκευές που πιθανώς θα συναντήσει στο σύστημα. Αν όλοι οι δυνατοί οδηγοί συσκευής έχουν περιληφθεί στο πυρήνα προς εγκατάσταση καταλήγουμε σε ένα τεράστιο πυρήνα με μεγάλες απαιτήσεις μνήμης για την φόρτωση του. Εναλλακτικά με τη χρήση modular οδηγών συσκευής ο πυρήνας προς εγκατάσταση περιέχει ένα ελάχιστο αριθμό βασικών οδηγών συσκευών και ανάλογα με το υλικό που θα συναντήσει δυναμικά φορτώνονται οι οδηγοί συσκευής για την υποστήριξη του.

4.1.1 Δυναμική φόρτωση και αφαίρεση αρθρωμάτων

Στη γενική περίπτωση για να επιτύχουμε επαύξηση των λειτουργιών που προσφέρει ο πυρήνας του λειτουργικού συστήματος, διάφορες διαδικασίες ενεργοποιούνται και καταγράφουν στον πυρήνα τη λειτουργικότητα που παρέχουν. Οι διαδικασίες αυτές στη συνέχεια παραμένουν ανενεργές, ως ότου οι υπηρεσίες τους να κληθούν από τον πυρήνα του λειτουργικού συστήματος. Τα αρθρώματα προσφέρουν δυναμική φόρτωση και επαύξηση των λειτουργιών του πυρήνα ανά πάσα στιγμή κατά το χρονικό διάστημα λειτουργίας του πυρήνα. Ομοίως δίδεται η δυνατότητα να αφαιρεθούν οποιαδήποτε στιγμή είναι ανενεργά και δεν προσφέρουν υπηρεσίες σε κάποιο κομμάτι του υπολογιστικού συστήματος. Για την υποστήριξη της δυναμικής φύσης τους τα αρθρώματα χρησιμοποιούν δύο βασικές συναρτήσεις:

int init_module(void)
Η συνάρτηση αυτή καλείται κατά την φόρτωση και ενσωμάτωση του module στον πυρήνα του λειτουργικού συστήματος.

void cleanup_module (void)
Η συνάρτηση αυτή καλείται κατά την αφαίρεση του module από τον πυρήνα του λειτουργικού συστήματος.

Η διαδικασία *cleanup_module()* δεν επιβάλλεται να υπάρχει σε ένα module. Παρόλα αυτά είναι συνήθως χρήσιμη για την αφαίρεση της λειτουργικότητας που εγγράφει στον πυρήνα το module κατά την φόρτωση του. Αντίθετα η διαδικασία *init_module()* είναι απαραίτητη και εκεί συνήθως επιτελείται έλεγχος αν ικανοποιούνται όλες οι συνθήκες που είναι απαραίτητες για το module ώστε να προσφέρει επιτυχώς τις υπηρεσίες του.

Γενικά ένα module δεν πρέπει να το φανταστούμε σαν μια διεργασία. Η διεργασία κατά την φόρτωση της εκτελείται ως ότου να φτάσει τελικά στο τέλος της. Αντίθετα το module κατά την φόρτωση του εκτελεί κάποιες αρχικές λειτουργίες και καταγράφει στον πυρήνα (μέσω της *init_module()*) τις υπηρεσίες που μπορεί να προσφέρει. Στη συνέχεια παραμένει ανενεργό και καλείται μόνο όταν (και αν) οι υπηρεσίες του απαιτηθούν από τον πυρήνα. Αν στο μέλλον κριθεί σκόπιμο το module μπορεί να αφαιρεθεί, οπότε και αφαιρείται η λειτουργικότητα που παρέχει στον πυρήνα. Ολόκληρο το χρονικό διάστημα που βρίσκεται στον πυρήνα, ο κώδικας του θεωρείται ότι αποτελεί μέρος του κώδικα του πυρήνα του λειτουργικού συστήματος.

4.2 Εισαγωγή στους οδηγούς συσκευής

Χρησιμοποιώντας το εν γένει χαρακτηριστικό των αρθρωμάτων, να ενσωματώνονται στον πυρήνα του λειτουργικού συστήματος και να αποτελούν κομμάτι του κώδικα του, μπορούν να χρησιμοποιηθούν σαν τρόπος δυναμικής υλοποίησης οδηγών συσκευής. Αυτή την ευελιξία που προσφέρουν τα αρθρώματα χρησιμοποιήσαμε στην υλοποίηση ενός οδηγού συσκευής για κάρτα τοπικού δικτύου Ethernet, βασισμένη στην διαχείριση των λειτουργιών της από ελεγκτή Intel 82558.

Ο οδηγός συσκευής αποτελεί το συνδετικό κομμάτι ανάμεσα στο υλικό μιας περιφερειακής συσκευής και του πυρήνα του λειτουργικού συστήματος. Είναι αυτός που ενώνει το χάσμα ανάμεσα στον διαφορετικό τρόπο υλοποίησης των λειτουργιών μιας περιφερειακής συσκευής και των σαφώς ορισμένων δυνατοτήτων που επιτρέπει ο πυρήνας του λειτουργικού συστήματος να χρησιμοποιούν οι διεργασίες για την προσπέλαση και χρήση των περιφερειακών συσκευών.

Λόγω της ευελιξίας που χαρακτηρίζει τα σύγχρονα υπολογιστικά συστήματα, το υλικό μιας περιφερειακής συσκευής μπορεί να παρέχεται από διαφορετικούς κατασκευαστές. Το ολοκληρωμένο κύκλωμα που βρίσκεται πάνω στη περιφερειακή συσκευή και φροντίζει για την επικοινωνία της με το υπόλοιπο υπολογιστικό σύστημα είναι ο ελεγκτής συσκευής, ο οποίος και καθορίζει τις λειτουργίες και δυνατότητες που μπορεί να επιτύχει το υλικό. Βάση των προσφερόμενων λειτουργιών του ελεγκτή και του τρόπου υλοποίησης αυτών, ένας προσαρμοσμένος σε αυτές τις δυνατότητες οδηγός συσκευής είναι απαραίτητος ώστε να γίνουν χρήσιμες οι λειτουργίες της περιφερειακής συσκευής στο υπόλοιπο υπολογιστικό σύστημα.

Από την άλλη πλευρά ο πυρήνας του λειτουργικού συστήματος προκείμενου να προσπελάσει τις λειτουργίες των οδηγών συσκευής και των ιδίων των περιφερειακών συσκευών, παρέχει ένα συγκεκριμένο πρότυπο συναρτήσεων διασύνδεσης που οι οδηγοί συσκευών καλούνται να υλοποιήσουν εσωτερικά τους. Αναγνωρίζοντας την πληθώρα των κατηγοριών περιφερειακών συσκευών που

μπορούν να δεχθούν τα σύγχρονα υπολογιστικά συστήματα και των διαφορετικών απαιτήσεων αυτών προς την παροχή λειτουργικότητας στους χρήστες, ο πυρήνας του λειτουργικού συστήματος παρέχει διαφόρων τύπου διασυνδέσεις ανάλογα με την κατηγορία της περιφερειακής συσκευής. Καταλήγουμε έτσι να έχουμε διαφόρων τύπου οδηγούς συσκευής που υλοποιούν το πρότυπο διασύνδεσης των περιφερειακών συσκευών με τον πυρήνα του λειτουργικού συστήματος.

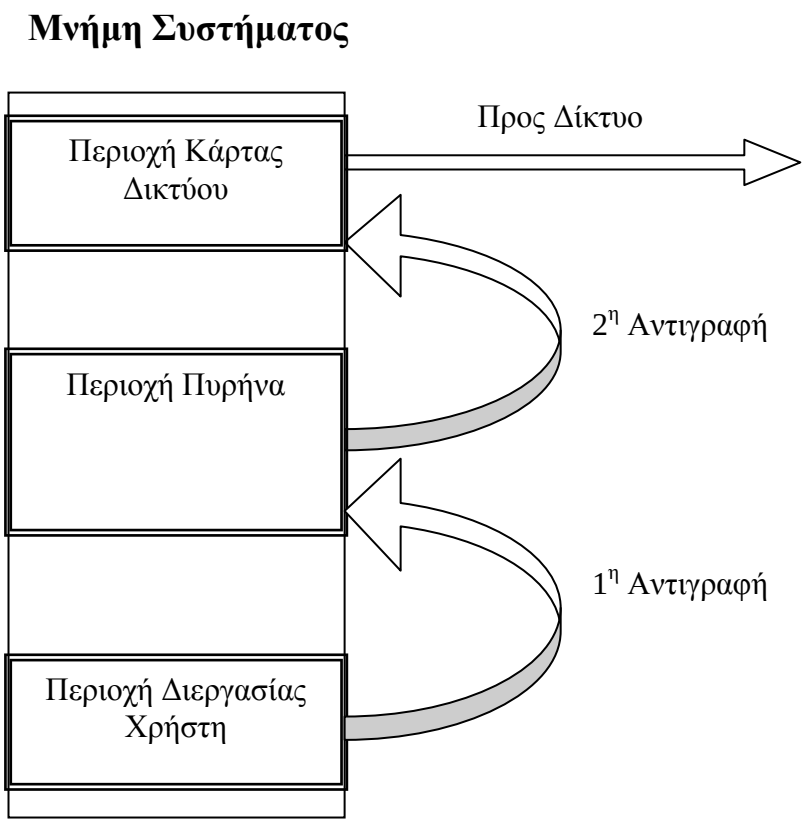
Για τα συστήματα της οικογενείας Unix, στα οποία ανήκει και το Linux πάνω στο οποίο έγινε η υλοποίηση της εργασίας, πολύ διαδεδομένοι είναι οι οδηγοί συσκευής για συσκευές χαρακτήρα (Char Device Drivers). Άλλες κατηγορίες οδηγών συσκευής είναι οι οδηγοί συσκευής δικτύου (Network Device Drivers) και οι οδηγοί συσκευής μαζικών δεδομένων (Block Device Drivers). Στην κατηγορία συσκευής χαρακτήρα ανήκει η σειριακή θύρα του υπολογιστικού συστήματος, στην κατηγορία συσκευής δικτύου οι κάρτες δικτύου και στην κατηγορία μαζικών δεδομένων ο σκληρός δίσκος.

4.2.1 Οδηγός συσκευής για γρήγορη επικοινωνία

Λογικά οι κάρτες δικτύου σαν περιφερειακές συσκευές διασυνδέονται με τον πυρήνα του λειτουργικού συστήματος με οδηγούς συσκευής δικτύου. Οι οδηγοί συσκευής αυτής της κατηγόριας υλοποιούν εσωτερικά τους τις συναρτήσεις που ορίζει το πρότυπο του πυρήνα που θα ενσωματωθούν. Οι συναρτήσεις αυτές είναι σχεδιασμένες να παρέχουν την κλασική λειτουργικότητα που απαιτείται από τις κάρτες δικτύου, όπως αποστολή δεδομένων και λήψη δεδομένων. Τα δεδομένα διατρέχουν διάφορα μονοπάτια μέσα στον πυρήνα για την επεξεργασία αυτών από τα πρωτόκολλα επικοινωνίας, έως ότου καταλήξουν να αποσταλούν στο δίκτυο ή να μεταφερθούν στο χρήστη κατά την αποστολή και λήψη αντίστοιχα. Αυτή η κλασική αντιμετώπιση των δεδομένων έγκειται στον τρόπο που υλοποιούνται τα πρωτόκολλα επικοινωνίας στην πλειοψηφία των λειτουργικών συστημάτων σαν εσωτερικά στον πυρήνα του λειτουργικού συστήματος.

Σίγουρα αυτή η κατεύθυνση προσφέρει ασφάλεια στην επικοινωνία απομακρυσμένων υπολογιστικών συστημάτων καθώς και δεν απαιτεί από τον χρήστη να γνωρίζει τεχνικά χαρακτηριστικά του πρωτοκόλλου για την κατάλληλη διαμόρφωση των επικεφαλίδων των δεδομένων προς αποστολή. Επιβάλει όμως πολλές αντιγραφές δεδομένων μέχρι αυτά να καταλήξουν στην ιδεατή περιοχή μνήμης από την οποία και θα τα προσπελάσει ο χρήστης. Οι αντιγραφές αυτές, όπως αναφέραμε και νωρίτερα είναι σημαντικά επιβαρυντικές στην καθυστέρηση της αποστολής και λήψης των δεδομένων μεταξύ χρηστών απομακρυσμένων συστημάτων. Στην καλύτερη περίπτωση απαιτούνται 2 αντιγραφές των δεδομένων

για την αποστολή και λήψη τους, βάση του κλασικού τρόπου αντιμετώπισης τους από οδηγούς συσκευής και πυρήνα λειτουργικού συστήματος.



Σχήμα 4-1: Κλασσικό Μονοπάτι Αποστολής Δεδομένων

Θέλοντας να επιτύχουμε μικρότερες καθυστερήσεις στην μεταφορά των δεδομένων μεταξύ απομακρυσμένων συστημάτων, αλλά και την υποστήριξη πρωτοκόλλων επιπέδου χρήστη δεν θα ενστερνιστούμε την κλασική διαδρομή των δεδομένων μέσα από τον πυρήνα. Αντίθετα θα προσπαθήσουμε να μειώσουμε τις αντιγραφές των δεδομένων μεταξύ περιοχών μνήμης του συστήματος. Βάση των απαιτήσεων μας αυτών, δεν εγείρεται καμιά σκοπιμότητα για χρήση του παραδοσιακού προτύπου συναρτήσεων που προσφέρονται προς υλοποίηση σε ένα οδηγό συσκευής δικτύου και διαλέγουμε ο οδηγός συσκευής μας να υλοποιεί το πρότυπο του οδηγού συσκευής χαρακτήρα.

Η επιλογή αυτού του τύπου οδηγού συσκευής δεν είναι τυχαία, αντιθέτως είναι επιτακτική λόγο της δυνατότητας αντιστοίχησης μνήμης (memory map) που προφέρει σε συσκευές τέτοιου τύπου ο πυρήνας του λειτουργικού συστήματος. Εκτός αυτής της κλήσης συστήματος, ελάχιστες άλλες κλήσεις συστήματος του προτύπου, για τους οδηγούς συσκευής αυτού του τύπου, είναι άμεσα χρήσιμες στον οδηγό συσκευής μας, δεδομένου της προσπάθειας να υλοποιήσουμε επικοινωνία από επίπεδο χρήστη, χωρίς την παρέμβαση του πυρήνα.

Όσον αφορά την αποστολή και λήψη δεδομένων, αυτή θα υλοποιηθεί σε επίπεδο χρήστη μέσω συναρτήσεων βιβλιοθήκης, κερδίζοντας με αυτό τον τρόπο χρόνο από τη μεταγωγή περιβάλλοντος (context switch) που απαιτείται κατά την κλασική αποστολή και λήψη δεδομένων.

4.2.2 Πρότυπο διασύνδεσης οδηγού συσκευής χαρακτήρα

Το πρότυπο οδηγού συσκευής χαρακτήρα παρέχει τις συναρτήσεις που πρέπει να υλοποιήσει ο οδηγός συσκευής, σαν μέσο διασύνδεσης του με τον πυρήνα και τελικούς αποδέκτες τους χρήστες του υπολογιστικού συστήματος. Οι συναρτήσεις που προσφέρει το πρότυπο αυτό είναι οι εξής:

```
struct file operations {
 struct module * owner;
 loff_t (*llseek) (struct file *,loff_t,int);
 ssize_t (*read) (struct file *,char *,size_t,loff_t *);
 ssize_t (*write) (struct file *,char *,size_t,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);
 int (*mmap) (struct file *,struct vm_area_struct *);
 int (*open) (struct inode *,struct file *);
 int (*flush) (struct file *);
 int (*release) (struct inode *,struct file *);
 int (*fsync) (struct file *,struct dentry *,int datasync);
 int (*fasync) (int,struct file *,int);
 int (*lock) (struct file *,int,struct file_lock *);
 ssize_t (*readv) (struct file *,struct iovec *,unsigned long,loff_t *);
 ssize_t (*writev) (struct file *,struct iovec *,unsigned long,loff_t *);
 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);
};
```

Η υλοποίηση ενός οδηγού συσκευής δεν επιβάλει και την υλοποίηση όλων των συναρτήσεων του πρότύπου. Οι συναρτήσεις που δεν υλοποιούνται εσωτερικά στον οδηγό συσκευής, αν καλεστούν από το χρήστη δεν επιφέρουν κανένα αποτέλεσμα. Οι κλήσεις συστήματος που υλοποιούνται στο οδηγό συσκευής μας είναι:

open: καλείται κατά το άνοιγμα της συσκευής από την διεργασία χρηστή. Επιστρέφει ένα περιγραφέα αρχείου που η διεργασία χρήστη θα χρησιμοποιήσει στις υπόλοιπες κλήσεις συστήματος που υλοποιεί ο οδηγός συσκευής.

release: καλείται κατά τον τερματισμό της χρήσης του οδηγού συσκευής από την διεργασία χρήστη. Αναλαμβάνει να ελευθερώσει περιοχές μνήμης και δεδομένα που αφορούν τη διεργασία που τερμάτισε.

ioctl: στην πραγματικότητα η ioctl κλήση συστήματος παρέχει τον τρόπο υλοποίησης πολλαπλών κλήσεων συστήματος στον οδηγό συσκευής οι οποίες δεν παρέχονται άμεσα από το πρότυπο του οδηγού συσκευής. Η διαφοροποίηση γίνεται

μέσω ενός ορίσματος της ioctl που δεικνύει και την κλήση συστήματος που θα κληθεί.
mmap: καλείται για την χυλοποίηση αντιστοίχησης μνήμης μεταξύ διεργασίας χρήστη και πυρήνα λειτουργικού συστήματος. Λόγο της σπουδαιότητας της παρουσιάζεται λεπτομερώς στην επόμενη παράγραφο.

4.3 Η αντιστοίχηση μνήμης

Όπως αναφέραμε σημαντικός παράγοντας καθυστέρησης στην μεταφορά δεδομένων αποτελούν οι πολλαπλές αντιγραφές δεδομένων στη μνήμη του συστήματος. Ο τρόπος να αντιμετωπιστεί ο παράγοντας αυτός καθυστέρησης είναι η αντιστοίχηση περιοχής μνήμης πυρήνα σε διευθύνσεις μνήμης περιοχής χρήστη. Αυτή η δυνατότητα προσφέρεται ήδη από τον πυρήνα λειτουργικών συστημάτων τύπου Unix σε οδηγούς συσκευής χαρακτήρα και καλείται από τις διεργασίες χρήστη με την κλήση συστήματος *mmap()*. Αντίθετα λόγω της διαφορετικής φιλοσοφίας που ακολουθείται στη μεταφορά δεδομένων από οδηγούς συσκευής δικτύου, η κλήση συστήματος *mmap()* δεν προσφέρεται στο πρότυπο κλήσεων συστήματος για συσκευές καρτών δικτύου. Αυτός ο παράγοντας ώθησε την υλοποίηση μας να δηλώνει τον οδηγό συσκευής στον πυρήνα σαν συσκευή χαρακτήρα.
Η κλήση συστήματος *mmap* για διεργασίες χρήστη έχει την εξής μορφή:

```
void* mmap(void *start,size_t length,int prot,int flags,int fd,off_t offset)
```

Οι παράμετροι της κλήσης συστήματος είναι οι εξής:
`start`: Η διεύθυνση περιοχής χρήστη από την οποία θα ξεκινήσει η αντιστοίχηση της νέας περιοχής μνήμης. Αν τεθεί 0, το σύστημα αναλαμβάνει να βρει μια κατάλληλη διεύθυνση από την οποία θα ξεκινήσει η αντιστοίχηση.
`length`: Το μέγεθος της περιοχής προς αντιστοίχηση.
prot: Παράμετροι οι οποίες υποδεικνύουν τον τρόπο προστασίας πρόσβασης που θα τεθεί στην αντιστοιχιζόμενη περιοχή. Πιθανές παράμετροι είναι:

```
PROT_EXEC
PROT_READ
PROT_WRITE
PROT_NONE
```

καθώς και η χρήση πολλαπλών παραμέτρων μεσώ του τελεστή OR.
`flags`: Προσδιορίζεται αν είναι δυνατή η κοινή χρήση της μνήμης προς αντιστοίχηση από πολλαπλές διεργασίες.

Παράμετροι που μπορούν να τεθούν είναι στο όρισμα:

```
MAP_FIXED
MAP_SHARED
MAP_PRIVATE
```

`fd`: Η αντιστοίχηση μνήμης επιτελείται είτε προς ένα κοινό αρχείο του συστήματος, είτε προς ένα αρχείο του καταλόγου /dev το οποίο και υποδεικνύει στο σύστημα συγκεκριμένη συσκευή. Στη μεν πρώτη περίπτωση ο χώρος μνήμης που αντιστοιχίζεται είναι ο χώρος που καταλαμβάνει το αρχείο στη συσκευή αποθήκευσης. Στη δεύτερη περίπτωση ο οδηγός συσκευής πρέπει να έχει υλοποιήσει τον προσαρμοσμένο στις ανάγκες του τρόπο για διαχείριση και αντιστοίχηση της μνήμης μετά από την κλήση συστήματος από διεργασία χρήστη. Σε κάθε περίπτωση ο χρήστης παρέχει σαν όρισμα τον περιγραφέα αρχείου που του επιστράφηκε από την κλήση συστήματος *open* για το αρχείο.

`offset`: Το όρισμα αυτό δέχεται αριθμό πολλαπλάσιο του μεγέθους σελίδας του συστήματος, βάση του οποίου μπορούμε να υπερπηδήσουμε σελίδες που βρίσκονται στην αρχή ενός αρχείου και να αντιστοιχίσουμε σελίδες βαθύτερα στο αρχείο.

Η κλήση συστήματος επιστρέφει ένα δείκτη ο οποίος δείχνει σε ιδεατή διεύθυνση χρήστη και αποτελεί την αρχή της περιοχής μνήμης που αντιστοιχίστηκε από τον οδηγό συσκευής. Το μέγεθος της περιοχής αυτής είναι όσο δηλώθηκε κατά την κλήση συστήματος της mmap. Από αυτό το σημείο και έπειτα, κάθε προσπέλαση δεδομένων από μια ιδεατή διεύθυνση περιοχής χρήστη μέσα στην αντιστοιχιζόμενη περιοχή, θα ανακατευθύνεται από τον πυρήνα στη αντιστοιχιζόμενη διεύθυνση μνήμης που οργάνωσε ο οδηγός συσκευής.

4.4 Υλοποίηση του αρθρώματος

Το άρθρωμα που υλοποιήθηκε παρέχει το συνδετικό επίπεδο μεταξύ των διεργασιών επιπέδου χρήστη και της συσκευής της κάρτας δικτύου. Το άρθρωμα που εισάγεται στον πυρήνα του λειτουργικού συστήματος επιτελεί διάφορες λειτουργίες που μπορούμε να τις κατηγοριοποιήσουμε ως εξής:

Λειτουργίες ελεγκτή κάρτας δικτύου

Ο σχεδιασμός των κοινών περιφερειακών συσκευών επιβάλει την υλοποίηση στον πυρήνα του συστήματος διαδικασιών που έχουν ως σκοπό την διαχείριση των καταστάσεων στην οποία βρίσκεται σε διάφορες περιόδους η λειτουργία του ελεγκτή της κάρτας δικτύου. Τυπικό παράδειγμα είναι χείριστης διακοπών που καλείται κατά την άφιξη στον πυρήνα αίτησης διακοπής από την κάρτα δικτύου. Επιπλέον λειτουργίες μη χρήσιμες στις διεργασίες χρήστη, αλλά σημαντικές γα την ορθή

λειτουργία του ελεγκτή της κάρτας δικτύου επιτελούνται στο εσωτερικό του αρθρώματος, όπως π.χ η αρχικοποίηση της κάρτας δικτύου.

Λειτουργίες του προτύπου οδηγού συσκευής χαρακτήρα

Κάθε άρθρωμα που εισάγεται στον πυρήνα, για την επικοινωνία του με τις διεργασίες χρήστη, υλοποιεί συναρτήσεις που αναμένεται να χρησιμοποιήσει ο χρηστής για την εκμετάλλευση της λειτουργικότητας που αυτό παρέχει. Οι συναρτήσεις καταχωρούνται στον πυρήνα κατά την αρχικοποίηση του αρθρώματος σύμφωνα με το πρότυπο του οδηγού συσκευής που το άρθρωμα επιλέγει να υλοποιήσει. Οι συναρτήσεις αυτές μπορεί να μην έχουν άμεση εξάρτηση με το συγκεκριμένο υλικό συσκευής που διαχειρίζεται το άρθρωμα, αλλά να χαρακτηρίζουν γενικότερα τη λειτουργία του.

Λειτουργίες διαχείρισης διεργασιών χρήστη

Τέλος υλοποιούνται λειτουργίες που εξυπηρετούν αποκλειστικά τη σωστή λειτουργία του αρθρώματος σαν μέσο διασύνδεσης τόσο προς την πλευρά της φυσικής συσκευής, όσο και προς την πλευρά των διεργασιών χρήστη. Οι σημαντικότερες λειτουργίες αυτής της κατηγορίας αφορούν την διαχείριση της πλευράς των διεργασιών χρήστη και έχουν να κάνουν με την ανανέωση πληροφοριών, σε δομές δεδομένων του αρθρώματος, που αφορούν τις διεργασίες χρήστη οι οποίες έχουν δηλώσει ενδιαφέρον για την χρήση της λειτουργικότητας που αυτό παρέχει.

4.4.1 Αρχικοποίηση αρθρώματος

Με την δυναμική φόρτωση του αρθρώματος καλείται η συνάρτηση *init_module()* με κύριο μέλημα της την απαραίτητη εύρεση και αρχικοποίηση της κάρτας δικτύου, την δέσμευση μνήμης πυρήνα για αρχικοποίηση των δομών που θα χρησιμοποιεί ο οδηγός συσκευής για την λειτουργία του καθώς και την αρχικοποίηση αυτών. Επιπλέον γίνεται καταχώρηση στον πυρήνα του τύπου του οδηγού συσκευής που θα υλοποιηθεί εντός του module, που στην περίπτωση μας είναι οδηγός συσκευής χαρακτήρα.

Εύρεση κάρτας δικτύου

Για να επιτευχθεί η αρχικοποίηση της κάρτας δικτύου, πρέπει να ευρεθεί η κάρτα δικτύου στο σύνολο των περιφερειακών συσκευών του υπολογιστικού συστήματος. Λόγω της φυσικής διασύνδεσης της περιφερειακής συσκευής κάρτας δικτύου, οπού

ακολουθεί το πρότυπο διασύνδεσης PCI, η αναζήτηση επικεντρώνεται σε δομές του πυρήνα του λειτουργικού συστήματος που αφορούν την διασύνδεση PCI. Οι δομές αυτές έχουν δημιουργηθεί κατά την εκκίνηση του λειτουργικού συστήματος, βάση των στοιχείων που αποθηκεύει το πρότυπο PCI σε ιδιωτικούς του καταχωρητές. Στη λίστα των δομών περιλαμβάνονται όλες οι συσκευές PCI του υπολογιστικού συστήματος που αναγνωρίστηκαν από το πρότυπο PCI, με στοιχεία που αφορούν την ακριβή ταυτότητα της περιφερειακής συσκευής, τις περιοχές μνήμης που έχει κατοχυρώσει (αλλά όχι ακόμα καταχωρήσει) το πρότυπο PCI στη συγκεκριμένη συσκευή καθώς και τον αριθμό αίτησης διακοπής με τον οποίο ο οδηγός συσκευής θα ειδοποιείται για ασύγχρονα συμβάντα που λαμβάνουν χώρα στην συσκευή.

Η περιοχή μνήμης που έχει κατοχυρώσει το πρότυπο PCI στην περιφερειακή συσκευή εξαρτάται άμεσα από την λειτουργικότητα της περιφερειακής συσκευής. Αντικατοπτρίζει καταχωρητές μνήμης που βρίσκονται πάνω στην περιφερειακή συσκευή και γράφοντάς ή διαβάζοντας από αυτούς, δίνονται εντολές ή λαμβάνονται αποτελέσματα αντίστοιχα από την περιφερειακή συσκευή. Για να γίνει πραγματική προσπέλαση σε αυτούς τους καταχωρητές από τον οδηγό συσκευής πρέπει να τους αποδοθούν διευθύνσεις μνήμης, μέσα από τον χώρο διευθύνσεων μνήμης και οι οποίες δεν έχουν φυσικό αντίκρισμα στη μνήμη του υπολογιστικού συστήματος. Με τον τρόπο αυτό προσπελάζοντας τις διευθύνσεις αυτές στην πραγματικότητα προσπελάζουμε καταχωρητές περιφερειακών συσκευών και όχι φυσική μνήμη συστήματος.

Έχοντας ευρεθεί λοιπόν η κατάλληλη δομή στη λίστα των συσκευών που αφορούν περιφερειακές συσκευές PCI και έχοντας συγκρατήσει τις περιοχές μνήμης που πρέπει να ανατεθούν στους καταχωρητές της συσκευής, το επόμενο βήμα είναι να γίνει η πραγματική αντιστοίχηση αυτών από τον οδηγό συσκευής, οπότε και να λάβει γνώση για την αντιστοίχηση αυτή ο πυρήνα του λειτουργικού συστήματος. Η αντιστοίχηση αυτή επιτελείται από την συνάρτηση

```
MemBase = ioremap(MemAreaStart,4096)
```

αφού έχουν προηγηθεί έλεγχοι ότι η συγκεκριμένη περιοχή δεν έχει αντιστοιχισθεί σε άλλη συσκευή. Η συνάρτηση ioremap επιστρέφει ένα δείκτη με τον οποίο πλέον θα χρησιμοποιεί ο οδηγός συσκευής για την προσπέλαση της περιοχής αυτής.

Στη συσκευή επιπλέον έχει ανατεθεί ένας αριθμός αίτησης διακοπής, ώστε να ειδοποιεί τον οδηγό συσκευής για ασύγχρονα γεγονότα που λαμβάνουν χώρα στην συσκευή. Ο αριθμός διακοπής αντιστοιχεί σε μια φυσική γραμμή διασύνδεσης της περιφερειακής συσκευής με τον επεξεργαστή του υπολογιστικού συστήματος. Όταν η περιφερειακή συσκευή θέλει να ενημερώσει τον επεξεργαστή για κάποιο ασύγχρονο γεγονός, ενεργοποιεί την φυσική διασύνδεση τους. Μέλημα του επεξεργαστή πλέον είναι να ενημερώσει τον οδηγό συσκευής για την αίτηση αυτή και αυτό επιτυγχάνεται

με το κάλεσμα μιας συνάρτησης, που παρέχει ο οδηγός συσκευής και ονομάζεται χειριστής διακοπών (Interrupt Handler).

Η συνάρτηση αυτή έχει υλοποιηθεί εντός του αρθρώματος του οδηγού συσκευής, οπότε και πρέπει να δηλωθεί στον πυρήνα σαν αντιστοίχηση του αριθμού αίτηση διακοπής που έχει ανατεθεί στην συσκευή της κάρτας δικτύου. Η αντιστοίχηση επιτελείται με την συνάρτηση

```
request_irq(dev->irq,ethfHandler,SA_SHIRQ,"ethf",dev)
```

που παρέχει ο πυρήνας του λειτουργικού συστήματος, και επιτελείται κατά την εκτέλεση της *init_module()*.

Αρχικοποίηση κάρτας δικτύου

Τελειώνοντας με τις απαραίτητες ενέργειές για την εύρεση της κάρτας δικτύου και την ενημέρωση του πυρήνα για τον τρόπο επικοινωνίας του οδηγού συσκευής με αυτή, απαραίτητες ενέργειες πρέπει να επιτελεστούν και στην κάρτα δικτύου ώστε να καταστεί δυνατή η χρήση της για αποστολή και λήψη δεδομένων.

Πλέον ο οδηγός συσκευής έχει αντιστοιχίσει διευθύνσεις φυσικής μνήμης στους καταχωρητές ελέγχου της κάρτας οπότε και μπορεί να δώσει εντολές στην κάρτα γράφοντας στους καταχωρητές αυτούς. Για τον ελεγκτή Intel 82558 της κάρτας δικτύου που χρησιμοποιούμε, οι καταχωρητες που αντιστοιχίζονται και με τους οποίους ελέγχεται η κάρτα ονομάζονται Control and Status Registers (CSR). Όπως αναφέραμε και στην περιγραφή του ελεγκτή της κάρτας, οι καταχωρητες CSR ελέγχουν τις δυο σημαντικές μηχανές της κάρτας:

1. Το Control Unit (CU), επιφορτισμένο με την εκτέλεση εντολών ενεργειών της κάρτας όπως η αποστολή πακέτων.
2. Το Receive Unit (RU), επιφορτισμένο με την λήψη Ethernet Frames από το τοπικό δίκτυο και την εναπόθεση τους στη μνήμη του συστήματος.

Εκτός από την αντιστοίχηση των καταχωρητών CSR στη φυσική μνήμη, η κάρτα δικτύου επιβάλει και την ανάθεση σε αυτή δυο περιοχών φυσικής μνήμης από τις οποίες:

- Το CU θα λαμβάνει από μια λίστα ειδικά διαμορφωμένων δομών τις εντολές προς εκτέλεση. Η περιοχή μνήμης αυτή ονομάζεται Command Block List (CBL).
- Το RU θα τοποθετεί σε αυτή τα λαμβανόμενα Ethernet Frames από το τοπικό δίκτυο. Η περιοχή μνήμης αυτή ονομάζεται Receive Frame Area (RFA).

Τις δυο αυτές περιοχές μνήμης ο οδηγός συσκευής τις δεσμεύει με την εντολή *kmalloc()* από τη φυσική μνήμη του συστήματος και εν συνεχεία τις διαμορφώνει με τρόπο συμβατό με την μορφή που αναγνωρίζουν τα CU και RU. Η δέσμευση και

διαμόρφωση των περιοχών αυτών γίνεται μέσα στην συνάρτηση *init_module()* του οδηγού συσκευής. Το μέγεθος των περιοχών αυτών έγκειται στην ευχέρεια του συστήματος να παραχωρήσει μνήμη στον οδηγό συσκευής. Ειδικά για την περιοχή λήψης Ethernet Frames που χρησιμοποιεί το RU ένα σεβαστό χώρος μνήμης είναι απαραίτητος ώστε να μην εμφανίζονται συχνά φαινόμενα κορεσμού της περιοχής από ταχύτατη εισροή Ethernet Frames από το δίκτυο. Επιπλέον τα μεγέθη για τις δυο αυτές περιοχές μνήμης μπορούν να δοθούν δυναμικά κατά την φόρτωση του αρθρώματος με τον οδηγό συσκευής στον πυρήνα.

Δεσμεύοντας ο οδηγός συσκευής τις περιοχές μνήμης, πρέπει να ενημερωθεί ο ελεγκτής της κάρτας δικτύου για τις διευθύνσεις φυσικής μνήμης των περιοχών αυτών. Αυτό επιτυγχάνεται με την τοποθέτηση στο CSR των εντολών LOAD ADDRESS τόσο στο πεδίο εντολών του CU όσο και στο πεδίο εντολών του RU. Σε κάθε περίπτωση η φυσική διεύθυνση μνήμης τοποθετείται στο πεδίο General_Pointer του CSR.

Ο ελεγκτής κάρτας δικτύου είναι πλέον έτοιμος να εκτελέσει εντολές ενεργειών που θα τοποθετήσει ο οδηγός συσκευής στο CBL, καθώς και ενεργοποιηθεί το RU που θα λαμβάνει εισερχόμενα Ethernet Frame και θα τα τοποθετεί στο RFA. Πριν όμως αρχίσει η λήψη των Ethernet Frames ο οδηγός συσκευής οφείλει να διαμορφώσει κάποιες παραμέτρους που ελέγχουν την λήψη και αποστολή δεδομένων από την κάρτα δικτύου. Αυτό επιτυγχάνεται με την εντολή ενέργειας Configure.

Διαμορφώνεται λοιπόν το πρώτο Block από το CBL να περιέχει την εντολή Configure με τις κατάλληλες παραμέτρους. Γράφοντας στο CSR στο πεδίο CU command τον κωδικό εντολής Active και θέτοντας το πεδίο General_Pointer 0, ενεργοποιείται το CU να εκτελέσει εντολές αρχίζοντας από το offset 0 του CBL, οπότε και εκτελείται η εντολή Configure.

Επιπλέον γράφεται στο πεδίο RU command του CSR ο κωδικός Ready και η τιμή 0 στο πεδίο General_Pointer του CSR, οπότε και ενεργοποιείται το RU για λήψη Ethernet Frames και τοποθέτησης τους στο RFA, αρχίζοντας από το offset 0 αυτού.

Η κάρτα δικτύου είναι πλέον έτοιμη για πλήρη χρήση της από τον οδηγό συσκευής και τους χρήστες που θα χρησιμοποιήσουν τη λειτουργικότητα που προσφέρει.

Αρχικοποίηση των δομών διαχείρισης διεργασιών χρήστη

Σε μνήμη που δεσμεύεται από την περιοχή μνήμης του πυρήνα του λειτουργικού συστήματος, διαμορφώνονται δομές που διατηρούν δεδομένα για τις διεργασίες χρήστη που χρησιμοποιούν το άρθρωμα. Κατά την αρχικοποίηση όλες οι δομές περιέχονται σε μια λίστα ελευθέρων δομών, από την οποία κάθε φορά εξάγουμε μια

δομή και αφού την γεμίσουμε δεδομένα την εισάγουμε στη λίστα χρησιμοποιούμενων δομών. Η λειτουργία της λίστας αυτής περιγράφεται λεπτομερέστερα στο επόμενο κεφάλαιο.

Καταχώρηση του οδηγού συσκευής στον πυρήνα

Κατά την κλήση της συνάρτησης αρχικοποίησης του αρθρώματος, ο οδηγός συσκευής καταχωρείται στον πυρήνα σαν οδηγός συσκευής χαρακτήρα. Κατά την καταχώρηση αυτή παρέχεται στον πυρήνα μια δομή με τις συναρτήσεις του προτύπου του οδηγού συσκευής που έχουν υλοποιηθεί εσωτερικά του αρθρώματος.

Οι συναρτήσεις του προτύπου είναι στην πραγματικότητα δείκτες οι οποίοι αντιστοιχίζονται με τις διευθύνσεις των συναρτήσεων που τις υλοποιούν μέσα σε μια δομή τύπου struct file_operations. Η αντιστοίχηση των συναρτήσεων με τα πρότυπα τους γίνεται με ευκολία μέσω μιας σύνταξης που προσφέρει ο μεταφραστής GNU gcc και έχει την εξής μορφή:

```
struct file_operations fops = {
       open: cdev_open,
       release: cdev_release,
       mmap: cdev_mmap,
       ioctl: cdev_ioctl,
       owner: THIS_MODULE
};
```

Η δομή struct file_operations παρέχεται στον πυρήνα με την συνάρτηση

```
register_chrdev(cdev_major, "cdev", &fops)
```

Στην παράμετρο fops παρέχεται η δομή με τις αντιστοιχίσεις που έχουν γίνει ανάμεσα στις κλήσεις συστήματος και τις υλοποιήσεις αυτών στον οδηγό συσκευής.

4.4.2 Κλήση συστήματος open

Με την κλήση συστήματος *open()*, ο χρήστης δηλώνει την επιθυμία του στο λειτουργικό σύστημα να χρησιμοποιήσει τη λειτουργικότητα που προσφέρει μια συσκευή συστήματος στη διεργασία του. Κάθε μια συσκευή χαρακτήρα που προσφέρει το λειτουργικό σύστημα Linux, αναπαριστάται από ένα αρχείο στον κατάλογο /dev. Με την κλήση συστήματος open, προς ένα τέτοιο αρχείο, επιστρέφεται στον χρήστη ένας περιγραφέας αρχείου (File Descriptor), ο οποίος και θα χρησιμοποιείται στην συνέχεια από το χρήστη στης υπόλοιπες λειτουργίες που θα επιτελέσει στο αρχείο αυτό και κατ' επέκταση στην συσκευή που αυτό αναπαριστά.

Η κλήση συστήματος *open()* από τη διεργασία χρήστη καταλήγει, μετά από μεταγωγή περιβάλλοντος και πιθανή προεπεξεργασία της από τον πυρήνα, στην κλήση της αντίστοιχης υλοποίησης της open που έχει δηλωθεί από τον οδηγό συσκευής. Στη υλοποίηση μας η open δεν προσφέρει κάποια ουσιαστική λειτουργία

στον οδηγό συσκευής, απλά κρατάει στατιστικά στοιχεία για τον αριθμό των διεργασιών στο άρθρωμα.

4.4.3 Κλήση συστήματος release

Η κλήση συστήματος *close()*, ενημερώνει τον πυρήνα του λειτουργικού συστήματος για την πρόθεση του χρήστη να τερματίσει τη χρήση μιας συσκευής που έχει νωρίτερα ανοίξει με την κλήση συστήματος *open()*. Στην κλήση συστήματος *close()* παρέχεται σαν όρισμα ο περιγραφέας αρχείου που επιστράφηκε νωρίτερα στη διεργασία από την *open()*. Η κλήση συστήματος *close()* ενεργοποιεί την υλοποίηση της συνάρτησης προτύπου *release()* στον οδηγό συσκευής χαρακτήρα.

Σε αντίθεση με την *open()* η υλοποίηση της *release()* ελευθερώνει τη μνήμη πυρήνα που έχει δεσμεύσει το αρθρωμα για τη διεργασία, καθώς και διαγράφει από τις δομές διαχείρισης του αρθρώματος τις καταχωρήσεις που υπάρχουν για την διεργασία που τερμάτισε.

4.4.4 Κλήση συστήματος mmap

Η κλήση συστήματος mmap αντιστοιχεί ιδεατές περιοχές μνήμης διεργασίας χρήστη, σε περιοχές μνήμης πυρήνα που ορίζονται από τον οδηγό συσκευής που υλοποιεί την κλήση. Η κλήση συστήματος mmap αποτελεί βασικό παράγοντα για τη μείωση της αντιγραφής δεδομένων ανάμεσα σε περιοχές φυσικής μνήμης συστήματος.

Επισκόπηση αντιστοιχιζόμενων περιοχών στην υλοποίηση μας

Κάθε διεργασία που αποφασίζει να χρησιμοποιήσει τη λειτουργικότητα της κάρτας δικτύου οφείλει να καλέσει αρχικά μια συνάρτηση βιβλιοθήκης που θα ενημερώσει τον οδηγό συσκευής της κάρτας για το αίτημα χρήσης της συσκευής από τη διεργασία. Κύριο μέλημα της αρχικοποίησης που επιτελεί η διαδικασία βιβλιοθήκης είναι η αντιστοίχηση μνήμης πυρήνα σε μνήμη περιοχής χρήστη μέσω της κλήσης συστήματος *mmap()* που προσφέρει ο οδηγός συσκευής και υποστηρίζει το λειτουργικό σύστημα. Κατά την κλήση της *mmap()* από την περιοχή χρήστη προς τον οδηγό συσκευής, ο κώδικας της mmap στον οδηγό συσκευής πρέπει να φέρει εις πέρας την αντιστοίχηση μνήμης που επιζητά με την κλήση της η διαδικασία βιβλιοθήκης περιοχής χρήστη. Οι περιοχές μνήμης που προσπελαύνει είναι τριών τύπων:

1. *Περιοχές μνήμης που μοιράζεται και με άλλες διεργασίες (Shared Kernel Area).*

   Στην περιοχή αυτή τοποθετούνται δεδομένα που παρέχουν πληροφορίες και μετρητές που όλες οι διεργασίες προσπελαύνουν κατά τη χρήση της κάρτας δικτύου.

2. *Περιοχές μνήμης CBL και RFA.*

   Είναι οι περιοχές που έχουν ανατεθεί στην κάρτα δικτύου για εκτέλεση εντολών ενεργειών και τοποθέτηση εισερχόμενων Ethernet Frames. Η περιοχές αυτές μνήμης είναι συνεχόμενες μεταξύ τους και επιπλέον συνεχόμενες με την διαμοιραζόμενη περιοχή όλων των διεργασιών, ώστε να μειωθούν οι κλήσεις αντιστοίχησης μνήμης που χρειάζεται η διεργασία.

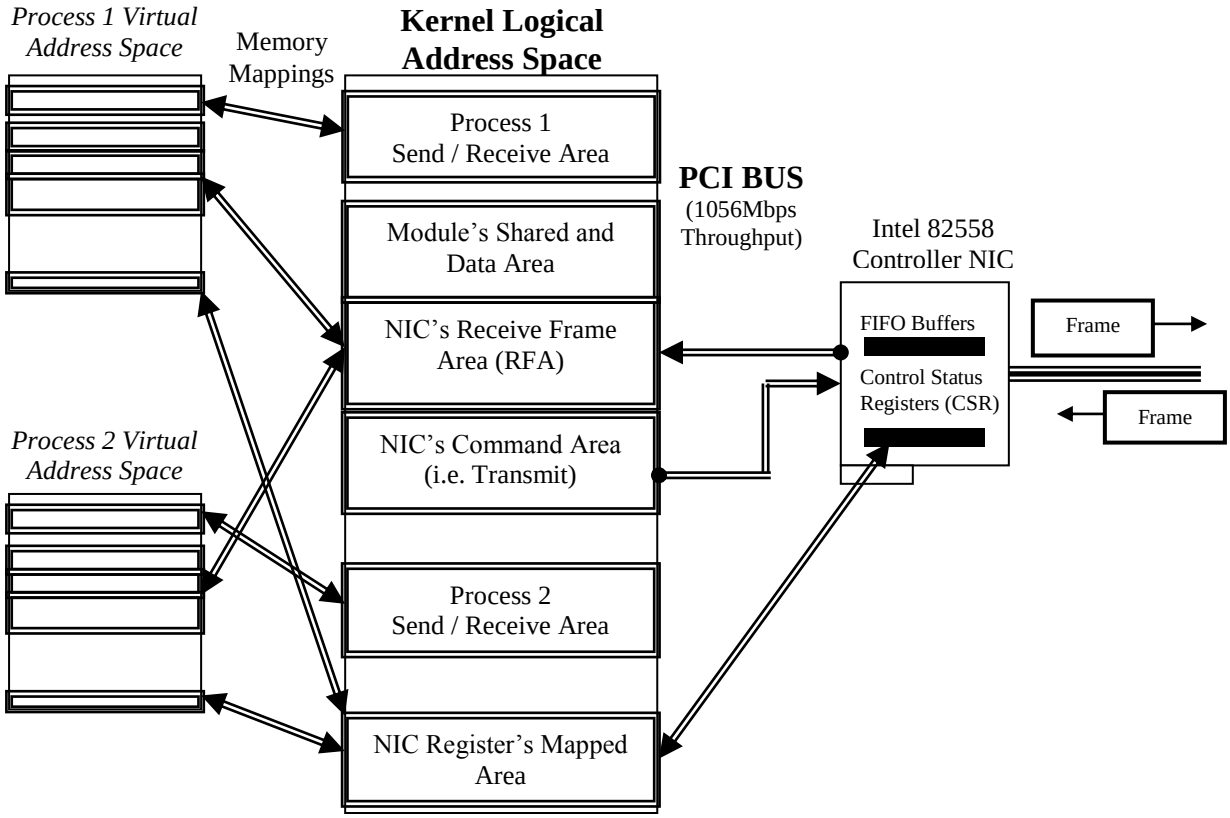
3. *Περιοχές μνήμης αποκλειστικής χρήσης από την διεργασία (Send/Receive Process Area).*

   Είναι μια συνεχόμενη περιοχή μνήμης όπου στις πρώτες σελίδες ο χρήστης τοποθετεί δεδομένα προς αποστολή. Στις επόμενες σελίδες ο οδηγός συσκευής τοποθετεί δεδομένα προερχόμενα από το δίκτυο προς τη διεργασία χρήστη.

4. *Περιοχές μνήμης αντιστοιχιζόμενες σε καταχωρητές της κάρτας δικτύου (Control Status Registers Area).*

   Προσπελαύνοντας τους καταχωρητές της κάρτας ο χρήστης ενεργοποιεί το Control Unit της κάρτας δικτύου για την έναρξη αποστολής δεδομένων στο δίκτυο.

Σχηματικά το μοντέλο λειτουργικότητας του οδηγού συσκευής με την κάρτα δικτύου και τις διεργασίες χρήστη παρουσιάζεται στο σχήμα 4-2.



Σχήμα 4-2: Αντιστοιχίσεις Μνήμης και Λειτουργικότητα Αρθρώματος

Για να γίνει η αντιστοίχηση και προς τις περιοχές μνήμης η κλήση συστήματος mmap πρέπει να κλιθεί τρεις φορές από την βιβλιοθήκη χρήστη, σαν μέρος της διαδικασίας αρχικοποίησης της διεργασίας του χρήστη με την οδηγό συσκευής της κάρτας δικτύου. Κύριο μέλημα της υλοποίησης της mmap μέσα στο άρθρωμα είναι η εύρεση ποιας περιοχής αιτείται την αντιστοίχηση ο χρήστης. Όταν βρεθεί η περιοχή προς αντιστοίχηση και έχουν γίνει έλεγχοι για την εγκυρότητα των παραμέτρων της διεργασίας, κατάλληλη δομή καταχωρείται στον πυρήνα του λειτουργικού συστήματος. Η δομή αυτή περιέχει δείκτες σε συναρτήσεις του αρθρώματος που θα κληθούν σε περίπτωση αιτήσεων διαχείρισης της περιοχής καθώς και αιτήσεων του χρήστη για προσπέλαση των δεδομένων που περιλαμβάνονται σε διευθύνσεις των περιοχών που αντιστοιχίσθηκαν.

4.4.5 Κλήση συστήματος ioctl

Η κλήση συστήματος *ioctl()* προσφέρει τη δυνατότητα στο χρήστη να ρυθμίσει παραμέτρους της συσκευής ή να επιτελέσει λειτουργίες στη συσκευή, οι οποίες δεν επιτελούνται άμεσα από τις προσφερόμενες από το πρότυπο κλήσεις συστήματος. Στο άρθρωμα μας υλοποιούνται τρεις λειτουργίες για την κλήση *ioctl()* που έχουν σκοπό την ανταλλαγή παραμέτρων επικοινωνίας της διεργασίας με το άρθρωμα:

1. *Αρχικοποίηση χρήστη.*

   Με αυτό τη λειτουργία αυτή δηλώνεται η επιθυμία του χρήστη να χρησιμοποιήσει την λειτουργικότητα του αρθρώματος.

2. *Καταχώρηση πόρτας λήψης δεδομένων.*

   Με αυτό τη λειτουργία αυτή δηλώνεται η επιθυμία του χρήστη να δηλώσει ένα UDP port στο οποίο αναμένει να λάβει δεδομένα ή να δηλώσει ένα μοναδικό κωδικό πρωτοκόλλου για την υλοποίηση ιδιωτικών πρωτοκόλλων σε επίπεδο χρήστη.

3. *Διαγραφή πόρτας λήψης δεδομένων.*

   Με αυτό τη λειτουργία αυτή δηλώνεται η επιθυμία του χρήστη να τερματίσει τη χρήση ενός UDP port για λήψη δεδομένων ή να τερματίσει την ισχύ μοναδικού κωδικού πρωτοκόλλου για την υλοποίηση ιδιωτικών πρωτοκόλλων σε επίπεδο χρήστη.

Κλήση ioctl αρχικοποίησης χρήστη

Αυτή η *ioctl()* κλήση συστήματος επιτρέπει την ανταλλαγή βασικών παραμέτρων μεταξύ της διεργασίας χρήστη και του αρθρώματος. Παράμετροι ανταλλάσσονται και από τις δύο μεριές, μέσω μια δομής που αντιγράφεται αρχικά από την περιοχή χρήστη προς την περιοχή πυρήνα του αρθρώματος. Η δομή αυτή περιέχει τις παραμέτρους που αιτείται η διεργασία χρήστη προς το άρθρωμα και περιλαμβάνουν:

- *Το μέγεθος της περιοχής αποστολής δεδομένων της διεργασίας στον πυρήνα*
- *Το μέγεθος της περιοχής λήψης δεδομένων της διεργασίας στον πυρήνα.*

Το άθροισμα των δυο αυτών παραμέτρων αποτελεί και το μέγεθος, σε σελίδες συστήματος, που το αρθρωμα αιτείται με *kmalloc()* από την περιοχή μνήμης του πυρήνα και το οποίο στη συνέχεια θα αντιστοιχίσει μέσω της *mmap()* στην διεργασία χρήστη. Για την διατήρηση των παραμέτρων αυτών, αλλά και της διεύθυνσης μνήμης που επιστρέφει η *kmalloc()*, εξάγεται ένα *struct openPorts* από την λίστα ελεύθερων δομών και αφού καταχωρηθούν σε αυτό οι τιμές διαχείρισης για την διεργασία, εισάγεται στη λίστα κατειλημμένων δομών. Στο *struct openPorts* επιπλέον εισάγεται το pid της διεργασίας ώστε να γνωρίζουμε σε ποια διεργασία αυτό ανήκει. Το pid της διεργασίας δεν χρειάζεται να το παρέχει η διεργασία ρητώς, μιας και το παρέχει ο πυρήνας μέσω της δομής *current->pid*.
Στη συνέχεια στη δομή που περιέχονται οι παραμέτρους αρχικοποίησης καταχωρούνται από τον πυρήνα οι εξής παράμετροι:

- *Η φυσική διεύθυνση της περιοχής μνήμης της διεργασίας στον πυρήνα*

  Δεδομένου ότι η αποστολή δεδομένων γίνεται απευθείας από περιοχή διεργασίας χρήστη, είναι απαραίτητο οι ιδεατές διευθύνσεις να αντιστοιχισθούν σε φυσικές κατά την δημιουργία δομών αποστολής δεδομένων. Γνωρίζοντας την αρχική φυσική διεύθυνση μπορούμε να υπολογίσουμε τις υπόλοιπες σαν offsets της αρχικής.

- *Το μέγεθος τη διαμοιραζόμενης περιοχής διεργασιών*

  Δεδομένο του ότι η RFA και CB περιοχές της κάρτας δικτύου αποτελούν συνέχεια της διαμοιραζόμενης περιοχής του αρθρώματος, παρέχεται στην διεργασία χρήστη το μέγεθος αυτής, ώστε στη συνέχεια να γνωρίζει την αρχή της RFA περιοχής.

- *Το μέγεθος της RFA*

  Στην αρχή της περιοχής λήψης δεδομένων χρήστη στον πυρήνα, υλοποιείται ένα bit map μεγέθους ίσο με τον αριθμό των RFDs της RFA. Αν κάποιο λαμβανόμενο Frame διατηρηθεί στην RFA αντί να μεταφερθεί στην περιοχή λήψης δεδομένων χρήστη το αντίστοιχο bit σημειώνεται. Για να υπολογιστεί το μέγεθος αυτού του bit map παρέχεται στη διεργασία, σαν αριθμό σελίδων, το μέγεθος της RFA.

- *Το μέγεθος της CBL*

  Παρόμοια υλοποιείται ένα bit map στην αρχή της περιοχής αποστολής δεδομένων μεγέθους ίσο με τον αριθμό των CBs. Σε αυτό σημειώνεται αν υπήρχε επιτυχής αποστολή στο Ethernet Frame που δημιουργήθηκε στο CB. Για τον υπολογισμό του μεγέθους του bit map παρέχεται στη διεργασία το μέγεθος της CBL περιοχής.

Κλήση ioctl καταχώρησης πόρτας λήψης δεδομένων

Χρησιμοποιώντας την κλήση αυτή, η διεργασία χρήστη δηλώνει το UDP port ή το ιδιωτικό πρωτόκολλο που θα χρησιμοποιήσει κατά την λήψη δεδομένων. Σε αυτή την περίπτωση οι παράμετροι της κλήσης συστήματος μεταφέρονται μόνο από την πλευρά της διεργασίας χρήστη προς τον οδηγό συσκευής. Οι παράμετροι που μεταφέρονται είναι τα εξής:

- *Ο τύπος της λήψης δεδομένων*

  Παρέχεται 0 αν πρόκειται για UDP port ή 1 αν πρόκειται για κωδικό πρωτοκόλλου.

- *Το UDP port ή τον κωδικό πρωτοκόλλου λήψης*

Γίνεται μια αναζήτηση από το άρθρωμα στη λίστα κατειλημμένων δομών διαχείρισης διεργασιών για την εξασφάλιση της μοναδικότητας της UDP port ή του κωδικού πρωτοκόλλου και στη συνέχεια τα δεδομένα καταχωρούνται στη δομή διαχείρισης του αρθρώματος για την διεργασία.

Κλήση ioctl διαγραφής πόρτας λήψης δεδομένων

Χρησιμοποιώντας την κλήση αυτή, η διεργασία χρήστη ενημερώνει το άρθρωμα για την πρόθεση της να τερματίσει τη χρήση UDP port ή κωδικού πρωτοκόλλου που μέχρι πρότεινος χρησιμοποιούσε για την λήψη δεδομένων. Οι παράμετροι που μεταφέρονται κατά την κλήση προς το άρθρωμα είναι οι εξής:

- *Ο τύπος της λήψης δεδομένων*

  Παρέχεται 0 αν πρόκειται για UDP port ή 1 αν πρόκειται για κωδικό πρωτοκόλλου.

- *Το UDP port ή τον κωδικό πρωτοκόλλου λήψης*

Γίνεται μια αναζήτηση από το άρθρωμα στη λίστα κατειλημμένων δομών διαχείρισης διεργασιών και αφού βρεθεί η κατάλληλη δομή με την UDP port ή τον κωδικό πρωτοκόλλου και διαπιστωθεί η κατοχύρωση τους από την συγκεκριμένη διεργασία, επιτελείται η διαγραφή τους.

4.5 Χειριστής διακοπών

Ο χειριστής διακοπών είναι η συνάρτηση που το άρθρωμα κατά την αρχικοποίηση του καταχωρεί στον πυρήνα σε αντιστοίχηση με τον αριθμό αίτησης διακοπής που έχει ανατεθεί από το PCI στην περιφερειακή συσκευή της κάρτα δικτύου. Κάθε αίτηση διακοπή από την κάρτα δικτύου προς τον επεξεργαστή λαμβάνεται από τον πυρήνα του λειτουργικού συστήματος, οπότε και καλείται η συνάρτηση του χειριστή διακοπών για τον έλεγχο και αντιμετωπίσει των αίτιων της αίτηση διακοπής.

Η συνάρτηση του χειριστή διακοπών πρέπει με σύντομους ελέγχους, στους καταχωρητές της κάρτας δικτύου, να συμπεράνει την αιτία δημιουργίας της αίτησης διακοπής και να προβεί στις απαραίτητες ενέργειες που θα ικανοποιήσουν την αίτηση διακοπής. Μια αίτηση διακοπής μπορεί να επισημαίνει στον οδηγό συσκευής περισσότερα από ένα γεγονότα που πρέπει να αντιμετωπιστούν.

Οι έλεγχοι γίνονται στο πεδίο STAT/ACK των καταχωρητών του CSR που έχουν αντιστοιχιστεί στην μνήμη του συστήματος. Στο πεδίο αυτό δηλώνεται η αιτία δημιουργίας της αίτησης διακοπής, σαν μια μάσκα από bits όπου το καθένα

αντιστοιχεί σε ένα γεγονός που οφείλει ο χειριστής διακοπών να αντιμετωπίσει. Όταν ο χειριστής διακοπών αντιμετωπίσει όλες τις αιτίες ενεργοποίησης της αίτησης διακοπή, επιβεβαιώνει την λήψη και αντιμετώπιση της αίτησης διακοπής, γράφοντας 1 σε κάθε bit του πεδίου STAT/ACK. Με τον τρόπο αυτό απενεργοποιείται η γραμμή της αίτησης διακοπής και η κάρτα μπορεί πλέον να στείλει νέα αίτηση διακοπής για κάποιο μελλοντικό αίτιο.

Στο χειριστή διακοπών μας, για την αντιμετώπιση της περίπτωσης εμφάνισης νέου αιτίου αίτησης διακοπής (ενεργοποίησης κάποιου bit στο STAT/ACK του CSR) κατά την εξυπηρέτηση των αρχικών αιτίων αίτησης διακοπής, οι έλεγχοι των αιτίων αίτησης διακοπής επιτελούνται μέσα σε ένα βρόγχο, τον οποίο εγκαταλείπουμε μόνο κατά την μη ύπαρξη νέου αιτίου αίτησης διακοπής.

Τα βήματα που υλοποιούν τους ελέγχους για την αιτία αποστολής της αίτησης διακοπής είναι τα εξής:

1. Διαβάζει από το CSR το πεδίο STAT/ACK που δείχνει μέσω ενός bit mask την αιτία που στάλθηκε η διακοπή. Πιθανές αιτίες είναι η άφιξη ενός Ethernet Frame και η μεταφορά του στην RFA, η ολοκλήρωση από το CU μιας εντολής που είχαμε εισάγει στη CBL περιοχή καθώς και η πλήρωση της RFA περιοχής με Frames από το δίκτυο οπότε και εμφανίζεται έλλειψη πόρων που μας απαγορεύει την μεταφορά άλλων Frames από το δίκτυο. Είναι πιθανό το IRQ που στάλθηκε να καλύπτει ταυτόχρονα πολλές αιτίες οπότε και θα είναι πολλαπλά bits ενεργοποιημένα στο STAT/ACK πεδίο του CSR. Στη συνέχεια γίνονται έλεγχοι για ποιες αιτίες υπαγόρευσαν την αποστολή του IRQ.
2. Αν το IRQ ειδοποιεί για άφιξη νέου frame (FR IRQ), ο χειριστής διακοπών:
   2.1. Ελέγχει αν το Ethernet Frame που παρέλαβε περιέχει IP δεδομένα ή περιέχει έγκυρο κωδικό κατοχυρωμένου ιδιωτικού πρωτοκόλλου. Στη περίπτωση ιδιωτικού πρωτοκόλλου τα δεδομένα μεταφέρονται άμεσα στην περιοχή λήψης δεδομένων της διεργασίας.
   2.2. Αν το πρωτόκολλο είναι IP ελέγχεται περαιτέρω στον IP Header, αν το μεταφερόμενο πρωτόκολλο είναι το UDP, διαφορετικά απορρίπτει το Fame. Στη συνέχεια αναζητά στη λίστα με τα κατοχυρωμένα UDP ports των διεργασιών ώστε να βρει σε πια διεργασία απευθύνονται τα δεδομένα.
   2.3. Αντιγράφει τα δεδομένα, αν η πολιτική που επέλεξε η διεργασία το επιβάλει, στο χώρο λήψης δεδομένων της διεργασία στον πυρήνα. Αν ο χώρος παραλαβής της διεργασίας είναι γεμάτος ή αν η διεργασία διάλεξε με ευθύνη της να αφήνει τα frames στο RFA δεν γίνεται αντιγραφή και σημειώνεται ένα αντίστοιχο bit για τη μη μεταφορά των δεδομένων στην αρχή της περιοχής λήψης.

3. Αν το IRQ ειδοποιεί για τέλος εκτέλεσης εντολών από το CU (CNA IRQ), ο χειριστής διακοπών:
    3.1. Ο οδηγός συσκευής ελέγχει μια μεταβλητή που κρατάει το τελευταίο offset στην περιοχή CBL στην οποία εκτελέσθηκαν εντολές και από το σημείο αυτό και μέχρι να συναντήσει το EL bit ενεργοποιημένο ελέγχει αν όλες οι εντολές εκτελέστηκαν επιτυχώς. Στη συνέχεια πρέπει να ειδοποιήσει όλους του χρήστες ότι τα δεδομένα τους στάλθηκαν ή όχι. Για να γίνει αυτό χρειάζεται :
        3.1.1. Ελέγχοντας στο CBL τη διεύθυνση των δεδομένων βρίσκει τη σελίδα χρήστη στον πυρήνα στην οποία ανήκουν.
        3.1.2. Σε μια δομή στην αρχή της σελίδας χρήστη στον πυρήνα σημειώνει επιτυχία διαφορετικά αποτυχία. Η δομή αυτή είναι ένα bit mask με τόσες θέσεις όσες και οι θέσεις στο CBL για εντολές. Γενικά χρειάζεται 2*|CBL| bits μιας και το 00 σημαίνει κενό, το 11 επιτυχία και το 10 αποτυχία.
    3.2. Ελέγχει αν τυχόν υπάρχουν νέες εντολές προς εκτέλεση και σε ποιο offset στο CBL αυτές βρίσκονται. Νέες εντολές προς εκτέλεση πιθανώς να έχουν δημιουργηθεί από τους χρήστες όσο χρόνο το CU της κάρτας εκτελούσε προηγούμενες εντολές. Σε αυτή την περίπτωση ο οδηγός συσκευής θέτει το General Pointer πεδίο στο CSR στο offset των εντολών προς εκτέλεση και δίνει την εντολή start στο πεδίο CUC του CSR.
4. Αν το IRQ ειδοποιεί για τη μη ύπαρξη ελευθέρου χώρου στο RFA (RNR IRQ), ο χειριστής διακοπών :
    4.1. Ο οδηγός συσκευής διατρέχει όλη τη λίστα με τα RFDs και κάνει κάποιους καθαρισμούς σε bits των επικεφαλίδων
    4.2. Στέλνει το σήμα SIGUSR2 στις διεργασίες χρήστη οπότε και ειδοποιούνται για τον καθαρισμό του χώρου. Αν κάποια διεργασία είχε αφήσει στο χώρο αυτό δεδομένα ειδοποιείται να μην τα προσπελάσει διότι πιθανά θα έχουν αντικατασταθεί από νέα.
5. Εφόσον έχουν εξεταστεί όλες οι πιθανές αιτίες για την υποβολή του IRQ, ο χειριστής διακοπών επιβεβαιώνει την αντιμετώπιση τους γράφοντας 1 σε όλα τα bits του πεδίου STAT/ACK του CSR. Από αυτό το σημείο και μετά η συσκευή μπορεί να στείλει πλέον νέο IRQ.

Εκτός από τις αιτίες υποβολής αίτησης διακοπής που παρουσιάστηκαν, ο ελεγκτής της κάρτας δικτύου μπορεί να δημιουργήσει αίτησης διακοπής και για άλλες αιτίες, οι οποίες όμως δεν αφορούν την υλοποίηση μας. Σε αυτή την περίπτωση ο χειριστής

διακοπών δεν προβαίνει σε καμία ενέργεια, απλά επιβεβαιώνει την λήψη της αίτηση διακοπής.

4.6 Ανακεφαλαίωση

Σε αυτό το κεφάλαιο παρουσιάστηκαν τα αρθρώματα και η χρήση τους σαν οδηγοί συσκευών του λειτουργικού συστήματος. Ένα άρθρωμα που υλοποιεί εσωτερικά του οδηγό συσκευής, κατά την δυναμική φόρτωση του στον πυρήνα οφείλει να καταχωρήσει την λειτουργικότητα του στον πυρήνα σαν συγκεκριμένο τύπο οδηγού συσκευής. Η κατοχύρωση αυτή συνοδεύεται από μια δομή με δείκτες σε συναρτήσεις του αρθρώματος, οι οποίες αποτελούν τις υλοποιήσεις των συναρτήσεων που επιβάλει το πρότυπο του οδηγού συσκευής.

Αν η φυσική συσκευή, που οδηγεί το άρθρωμα, προβλέπει την χρήση αιτήσεων διακοπής για την επικοινωνία της με τον πυρήνα του λειτουργικού συστήματος, το άρθρωμα καταχωρεί στον συγκεκριμένο αριθμό αίτηση διακοπής που έχει ανατεθεί στην συσκευή και μια συνάρτηση του σαν χειριστή διακοπών της συσκευής.

Το άρθρωμα της υλοποίησης μας δεν αποτελεί μόνο το μέσο επικοινωνίας του πυρήνα με την φυσική συσκευή, αλλά αποδίδει επιπλέον λειτουργικότητα στις διεργασίες χρήστη που επιθυμούν να χρησιμοποιήσουν την κάρτα δικτύου. Για το λόγο αυτό υλοποιεί και συναρτήσεις διαχείριση δομών των διεργασιών που χρησιμοποιούν το άρθρωμα. Δεν αποτελεί λοιπόν ένα τυπικό οδηγό συσκευής, αλλά το συνδετικό επίπεδο για την υλοποίηση επικοινωνία επιπέδου χρήστη με απευθείας χρήση της περιφερειακής συσκευής.

Κεφάλαιο 5
Οι Δομές και Πολιτικές Διαχείρισης Δεδομένων

5.1 Η δομή διαχείρισης παραμέτρων διεργασιών

Ο οδηγός συσκευής έχει προσαρμοστεί για άμεση επικοινωνία με τις διεργασίες επιπέδου χρήστη, παρακάμπτοντας το κλασσικό μονοπάτι μεταφοράς των δεδομένων μέσα από τον πυρήνα του λειτουργικού συστήματος. Λόγω αυτού του τρόπου επικοινωνίας, ο οδηγός συσκευής αναλαμβάνει να διατηρήσει στοιχεία διαχείρισης των διεργασιών που χρησιμοποιούν την λειτουργικότητα της κάρτας δικτύου.

Η δομή στον οδηγό συσκευής που διατηρεί τα στοιχεία κάθε διεργασίας η οποία έχει εκδηλώσει ενδιαφέρον για επικοινωνία με τον οδηγό συσκευής είναι η *struct openPorts*. Στη δομή αυτή περιλαμβάνονται πεδία που αφορούν τόσο παραμέτρους της επικοινωνίας με απομακρυσμένες διεργασίες, όσο και παραμέτρους διαχείρισης της διεργασίας στο τοπικό σύστημα του οδηγού συσκευής. Στους παραμέτρους επικοινωνίας της διεργασίας περιλαμβάνονται τα στοιχεία που χρησιμοποιούν τα πρωτόκολλα για τη διαφοροποίηση του τελικού προορισμού των δεδομένων μέσα στο υπολογιστικό σύστημα. Για τη στοίβα πρωτοκόλλων UDP/IP η διαφοροποίηση του τελικού αποδέκτη γίνεται μέσω του ζευγαριού {IP διεύθυνση παραλήπτη,UDP port παραλήπτη}. Η IP διεύθυνση της κάρτας είναι δεδομένη για την επικοινωνία οπότε η κάθε διεργασία κατοχυρώνει στον οδηγό συσκευής τα UDP ports στα οποία θα παραλαμβάνει τα δεδομένα.

Ο οδηγός συσκευής επιπλέον υποστηρίζει τη δυνατότητα των διεργασιών για χρήση ιδιωτικών πρωτοκόλλων βάση των οποίων θα επικοινωνούν οι απομακρυσμένες διεργασίες χρήστη. Τα ιδιωτικά πρωτόκολλα ενσωματώνονται στο Ethernet Frame αμέσως μετά την επικεφαλίδα που θέτει το επίπεδο μεταφοράς (Ethernet Header). Αυτό που δεικνύει τον τύπο του πρωτοκόλλου που περιλαμβάνεται στο Ethernet Frame είναι ένας μοναδικός κωδικός αριθμός στο πεδίο TYPE της Ethernet επικεφαλίδας. Για παράδειγμα αν τα δεδομένα που περιλαμβάνονται στο Ethernet Frame βασίζονται στο πρωτόκολλο IP, ο κωδικός αριθμός αυτός είναι 0x0800. Κατοχυρώνοντας ένα μοναδικό κωδικό αριθμό η διεργασία για ολόκληρο το τοπικό δίκτυο μπορεί να περιλάβει ιδιωτικά πρωτόκολλα μέσα στο Ethernet Frame. Για την κατοχύρωση αυτή ενημερώνεται ο οδηγός

συσκευής της κάρτας δικτύου όπου τοποθετεί τον κωδικό του ιδιωτικού πρωτοκόλλου στη δομή με τις παραμέτρους επικοινωνίας της διεργασίας.

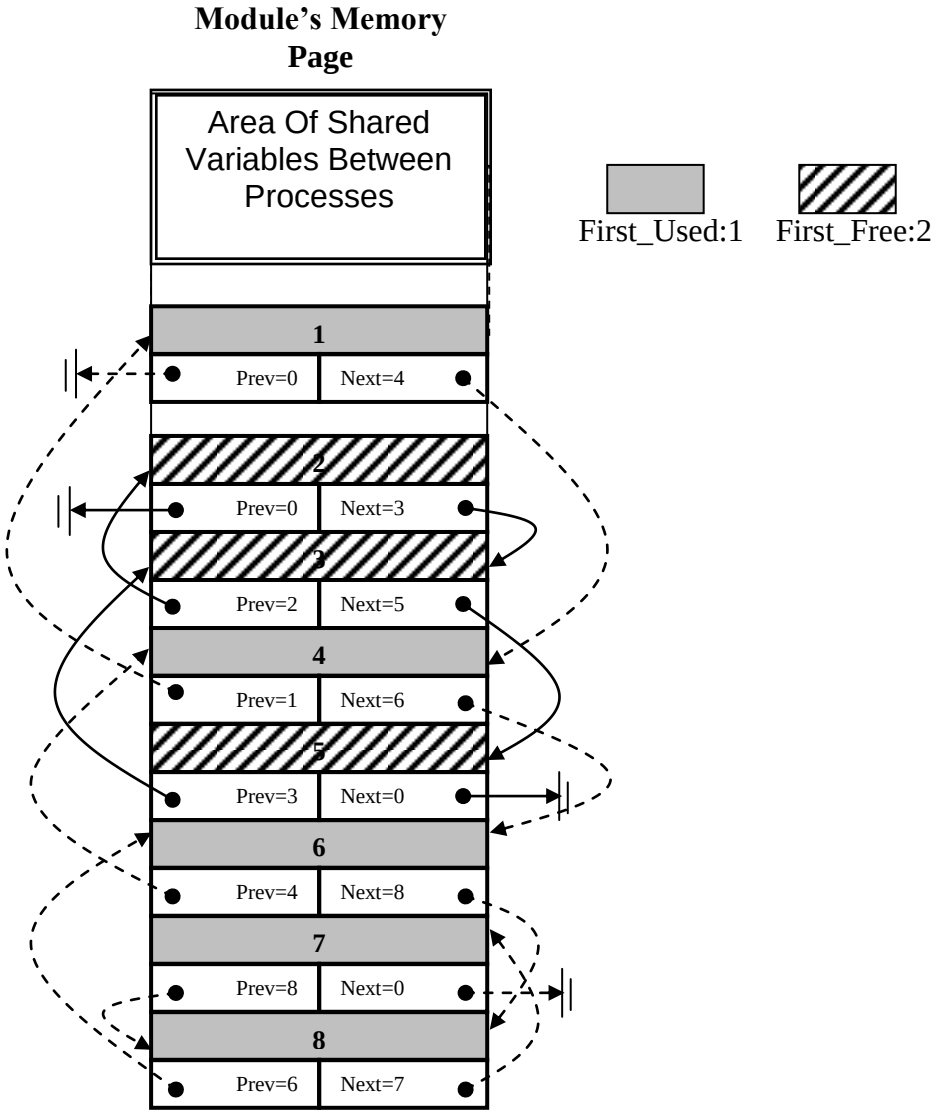
Ο ένας τρόπος επικοινωνίας δεν αποκλείει τον άλλο. Μια διεργασία μπορεί να χρησιμοποιεί την στοίβα πρωτοκόλλων IP/UDP για κάποια κατηγορία δεδομένων (πιθανώς για δεδομένα αρχικοποίησης) και κάποιο ιδιωτικό πρωτόκολλο για την καθεαυτή μεταφορά των δεδομένων.

Εκτός από τις παραμέτρους επικοινωνίας που αφορούν τα πρωτόκολλα μεταφοράς δεδομένων, στη δομή της διεργασίας αποθηκεύονται και παράμετροι που αφορούν τη διαχείριση των διεργασιών από τον οδηγό συσκευής. Τέτοια στοιχεία αφορούν τη διεύθυνση μνήμης πυρήνα από την οποία ξεκινά η ιδιωτική περιοχή αποστολής και λήψης της διεργασίας του χρήστη, καθώς και το μέγεθος της καθεμίας από αυτές τις περιοχές σαν αριθμός από σελίδες συστήματος.

Οι δομές αυτές τοποθετούνται σε σελίδα μνήμης πυρήνα που δεσμεύει ο οδηγός συσκευής κατά την αρχικοποίηση του. Η σελίδα αυτή διαμορφώνεται από τον οδηγό συσκευής σε κομμάτια μνήμης μεγέθους ίσο με το μέγεθος της δομής διαχείρισης διεργασιών *struct openPorts*. Σχηματίζεται με αυτό τον τρόπο μια λίστα από άδειες δομές διαχείρισης διεργασιών. Η δομές αριθμούνται διαδοχικά από το 1 έως το PAGE_SIZE/SIZEOF_STRUCT ανάλογα με τη θέση τους στη λίστα. Ο οδηγός συσκευής διατηρεί δυο δείκτες που αφορούν τις δομές διαχείρισης διεργασιών. Ο ένας δείκτης κρατάει τον αριθμό της πρώτης ελεύθερης δομής στη λίστα με τις άδειες δομές. Ο άλλος δείκτης κρατάει τον αριθμό της πρώτης κατειλημμένης δομής σχηματίζοντας μια λίστα χρησιμοποιημένων δομών, στις οποίες περιέχονται έγκυρα δεδομένα διεργασιών χρήστη. Οι δυο λίστες είναι διπλά συνδεδεμένες ώστε να είναι δυνατή σε ένα βήμα η προσθήκη και η αποδέσμευση κάποιας δομής από αυτές.

Για εξοικονόμηση χώρου που καταλαμβάνει η κάθε δομή στη σελίδα μνήμης που δεσμεύει ο οδηγός συσκευής, η διπλή σύνδεση της λίστας γίνεται χρησιμοποιώντας σαν δείκτες προηγούμενης και επόμενης δομής όχι απόλυτες διευθύνσεις μνήμης, αλλά τους αριθμούς που αύξοντα αριθμούνται οι δομές κατά την σειριακή αρχικοποίηση τους σε μια συνεχόμενη λίστα ελευθέρων δομών. Για την κράτηση κάθε αριθμού αρκεί 1 Byte σε σχέση με τα 4 Bytes που χρειάζεται μια διεύθυνση μνήμης. Εξοικονόμηση χώρου στη δομή επιτυγχάνεται επιπλέον με την συγχώνευση παραμέτρων που απαιτούν για την αποθήκευση τους λιγότερο του ενός Byte, όπως π.χ ο αριθμός των σελίδων αποστολής και λήψης δεδομένων της διεργασίας στην περιοχή του πυρήνα. Οι δύο αυτές παράμετροι χρειάζονται 4 bits η καθεμία για την αποθήκευση τους (λόγω περιορισμού στο μέγεθος τους που επιβάλει το άρθρωμα), οπότε και συγχωνεύονται σε μια μεταβλητή 1 Byte.

Σχηματικά οι δύο λίστες, ελεύθερων και κατειλημμένων δομών *struct openPorts* με παραμέτρους των διεργασιών, συνδέονται μεταξύ τους, μετά από κάποια περίοδο χρήσης, όπως φαίνεται στο παράδειγμα του σχήματος 5-1.



Σχήμα 5-1: Υλοποίηση λίστας ελεύθερων και κατειλημμένων δομών struct openPorts

Κατά την έναρξη της χρήσης του αρθρώματος γίνεται αρχικοποίηση της περιοχής που θα κρατηθούν οι δομές διαχείρισης παραμέτρων των διεργασιών. Ο δείκτης First_Free παίρνει την τιμή 1 και ο δείκτης First_Used παίρνει την τιμή 0 που δεικνύει τη μη ύπαρξη δομής.

Στη συνέχεια κάθε φορά που μια διεργασία καλεί την κλήση συστήματος αρχικοποίησης που παρέχει το άρθρωμα στις διεργασίες χρήστη, η πρώτη ελεύθερη δομή από τη λίστα των ελευθέρων κατοχυρώνεται στη διεργασία οπότε και προστίθεται πρώτη στη λίστα των κατειλημμένων. Αντίστροφα όταν μια διεργασία

καλέσει την κλήση συστήματος *close()* ή όταν τερματιστεί η λειτουργία της, στον οδηγό συσκευής γίνεται μια διερεύνηση στην λίστα των κατειλημμένων δομών και αφαιρείται από αυτή η δομή που περιέχει τα στοιχεία της διεργασίας. Η δομή που αφαιρέθηκε προστίθεται πρώτη στη λίστα των ελεύθερων δομών.

Αν μια διεργασία έχει ανάγκη για δημιουργία περισσοτέρων των τριών UDP ports για την επικοινωνία της, κατοχυρώνει νέα δομή για τις επιπλέον παραμέτρους επικοινωνίας, οπότε και στο τέλος ελευθερώνονται όλες οι δομές που έχει κατοχυρώσει η διεργασία. Στο άρθρωμα ο αριθμός των δομών *struct openPorts* είναι μεταβλητός και μπορεί να διαφοροποιηθεί κατά την αρχικοποίηση του.

5.2 Πολιτική διαχείρισης λαμβανομένων δεδομένων

Ο ελεγκτής της κάρτας δικτύου απαιτεί για την λειτουργία του την δέσμευση κάποιας περιοχής μνήμης συστήματος από τον οδηγό συσκευής, ώστε να εναποτίθοναι σε αυτή τα λαμβανόμενα Ethernet Frames (Receive Frame Area – RFA). Η περιοχή μνήμης αυτή δεσμεύεται, διαμορφώνεται κατάλληλα σε δομές παραλαβής δεδομένων που προστάζει ο οδηγός συσκευής και η φυσική διεύθυνση της αρχής της αποστέλλεται στον ελεγκτή της κάρτας δικτύου. Από το σημείο αυτό και έπειτα ο ελεγκτής της κάρτας δικτύου χρησιμοποιεί το χώρο της περιοχής αυτής για ασύγχρονη εναπόθεση λαμβανόμενων δεδομένων από το δίκτυο.

Η περιοχή λήψης των δεδομένων έχει πεπερασμένο μέγεθος και εναποτίθονται σε αυτή όλα τα λαμβανόμενα Ethernet Frames, ασχέτως της διεργασίας χρήστη που αυτά απευθύνονται. Αυτό έχει ως αποτέλεσμα την μη δυνατότητα συγκράτησης σε αυτή των λαμβανομένων Ethernet Frames για μεγάλο χρονικό διάστημα διότι υπάρχει άμεσος κίνδυνος της πλήρωσης του χώρου μνήμης της περιοχής, με δυσάρεστο συνεπακόλουθο την πιθανή απώλεια νέων λαμβανόμενων Ethernet Frames. Για να αποφευχθεί η κρίσιμη κατάσταση της πλήρωσης του χώρου λήψης δεδομένων της κάρτας δικτύου, τα λαμβανόμενα Ethernet Frames αφού πιστοποιηθούν από το άρθρωμα ότι ανήκουν σε κάποια καταχωρημένη διεργασία, αντιγράφονται στο χώρο λήψης δεδομένων της διεργασίας στον πυρήνα (πολιτική μιας αντιγραφής).

Η επιλογή αυτή, της αντιγραφής των λαμβανομένων δεδομένων σε διαφορετικές περιοχές πυρήνα ανάλογα της διεργασίας που αυτά απευθύνονται, εμφανίζει αρκετά πλεονεκτήματα:

- *Η διεργασία χρήστη μπορεί να επεξεργαστεί τα δεδομένα όποια στιγμή αυτή το κρίνει χρήσιμο, χωρίς να επιβάλλεται η άμεση επεξεργασία τους κατά την λήψη από την κάρτα δικτύου.*

- *Πολλές φορές λαμβανόμενα δεδομένα επιβάλλουν την διατήρηση τους για πολλαπλές προσπελάσεις τους από τη διεργασία. Με τη μεταφορά τους σε ιδιωτικό χώρο της διεργασίας εξασφαλίζεται η ασφαλής διατήρηση τους.*

- *Έντονες ανταλλαγές μηνυμάτων μεταξύ απομακρυσμένων διεργασιών επιβάλουν την ταχύτατη μεταφορά μικρού μεγέθους δεδομένων. Το μέγεθος του MTU του Ethernet Frame είναι 1500 bytes οπότε και ανάλογος χώρος έχει δεσμευτεί σε κάθε buffer της περιοχής λαμβανομένων δεδομένων του ελεγκτή της κάρτας δικτύου. Αν διατηρήσουμε μικρά μηνύματα σε buffers 1500 bytes έχουμε μεγάλη σπατάλη χώρου. Η σύντομη αντιγραφή τους σε ιδιωτικούς buffers μεταβλητού μεγέθους επιτρέπει την αποφυγή σπατάλης χώρου.*

Το μόνο εμφανές μειονέκτημα της πολιτικής αντιγραφής των δεδομένων σε περιοχή λήψης διεργασίας παραλήπτη είναι το κόστος της αντιγραφής των δεδομένων από μία περιοχή μνήμης σε κάποια άλλη. Η καθυστέρηση αυτή παίζει σημαντικό ρόλο κατά την συχνή αντιγραφή μεγάλου όγκου δεδομένων τα οποία ανταλλάσσονται μεταξύ των κόμβων επικοινωνίας. Πολλές φορές σε τέτοιους τύπους επικοινωνίας τα μεταφερόμενα δεδομένα δεν επιβάλουν πολλαπλή επεξεργασία και μόνιμη διατήρηση τους. Κρίσιμο ρόλο παίζει συχνά η όσο το δυνατόν μικρότερη καθυστέρηση στην προσπέλαση των δεδομένων και η άμεση απόρριψη τους. Τέτοιες απαιτήσεις τίθονται σε επικοινωνίες τύπου Πραγματικού Χρόνου (Real Time), οπού και αργοπορημένα δεδομένα δεν έχουν πρακτική ωφέλεια και επιπλέον τα λαμβανόμενα δεδομένα δεν χρειάζονται μόνιμη αποθήκευση.

Για να καλυφθούν αποδοτικά οι διαφορετικές απαιτήσεις ανάλογα με τον τύπο επικοινωνίας των απομακρυσμένων διεργασιών, ο οδηγός συσκευής επιτρέπει στη διεργασία χρήστη να επιλέξει την πολιτική τοποθέτησης λαμβανομένων δεδομένων ανάλογα με τις ανάγκες της. Η επιλογή γίνεται από το χρήστη κατά την κατοχύρωση του UDP port για την λήψη δεδομένων ή του κωδικού ιδιωτικού πρωτοκόλλου που θα χρησιμοποιηθεί. Η κατοχύρωση του χρήστη ενός UDP port κοντά στο τέλος της περιοχής των πιθανών UDP ports, αυτόματα σημαίνει την αποδοχή του χρήστη της πολιτικής διατήρησης δεδομένων στη περιοχή RFA, χωρίς την αντιγραφή τους στην περιοχή λήψης διεργασίας (πολιτική μηδενικής αντιγραφής). Σε αντίθετη περίπτωση τα δεδομένα αντιγράφονται κάθε φορά στην περιοχή λήψης της διεργασίας.

Σε περίπτωση μη αντιγραφή των λαμβανομένων δεδομένων, οι διεργασίες χρήστη πρέπει να έχουν άμεση πρόσβαση στην περιοχή απόθεσης λαμβανόμενων

Ethernet Frames από την κάρτα δικτύου. Αυτό επιτυγχάνεται μέσω αντιστοίχησης μνήμης κατά την αρχικοποίηση τους. Βέβαια με αυτό τον τρόπο δεν εξασφαλίζεται η διατήρηση των δεδομένων της διεργασίας, οπότε και υπάρχει περίπτωση πιθανής απώλειας τους αν επέλθει πλήρωση της περιοχής λήψης δεδομένων.

5.3 Πολιτική διαχείρισης περιοχής λήψης δεδομένων διεργασίας

Η κοινή πολιτική κατά την λήψη δεδομένων από την κάρτα δικτύου είναι η αντιγραφή των δεδομένων από τον οδηγό συσκευής στην ιδιωτική περιοχή λήψης δεδομένων της διεργασίας (ΠΛΔΔ) στην οποία απευθύνονται τα δεδομένα. Η ΠΛΔΔ προσπελάζεται τόσο από τον οδηγό συσκευής κατά την αντιγραφή των δεδομένων από την περιοχή RFA, όσο και από την διεργασία για την τελική προσπέλαση και επεξεργασία των δεδομένων .

Λόγω της προσπέλασης της ιδιωτικής περιοχής χρηστή από δύο οντότητες, την διεργασία χρήστη και τον οδηγό συσκευής, εύκολα μπορούν να δημιουργηθούν συνθήκες ανταγωνισμού μεταξύ του οδηγού συσκευής και της διεργασίας χρήστη. Μέσω μιας κατάλληλα διαμορφωμένης πολιτικής πρόσβασης στην ΠΛΔΔ οι ανεπιθύμητες αλληλεπιδράσεις από τις συνθήκες ανταγωνισμού οφείλουν να αντιμετωπισθούν. Επιπλέον η διαχείριση του ελεύθερου χώρου της περιοχής οφείλει να είναι αποδοτική ώστε να μην δημιουργείται τμηματοποίηση της περιοχής σε μικρά κομμάτια ελεύθερου χώρου τα οποία εύκολα μένουν ανεκμετάλλευτα.

Συνήθως η πολιτική μεταφοράς των λαμβανόμενων μηνυμάτων από την περιοχή RFA στην ΠΛΔΔ επιλέγεται για ανταλλαγή μηνυμάτων που θέλουμε να επεξεργαστούμε πολλές φορές οπότε και πρέπει να τα διατηρήσουμε σε ασφαλές μέρος. Τα μηνύματα αυτά οφείλουν να μεταδίδονται όσο το δυνατόν με μικρότερη καθυστέρηση από τη μια απομακρυσμένη διεργασία στην άλλη ώστε να είναι έγκαιρα διαθέσιμα προς επεξεργασία. Το μέγεθος των μηνυμάτων είναι συχνά μικρό και για το λόγο αυτό απαιτούμε και ελάχιστη καθυστέρηση στη μεταφορά τους.

Αν η μεταφορά των μηνυμάτων γίνει βάση της στοίβας πρωτοκόλλων UDP/IP μέσω του επιπέδου μεταφοράς η Ethernet, στα μηνύματα μοιραία ενσωματώνονται επικεφαλίδες που θέτουν τα πρωτόκολλα μεταφοράς. Πολλές φορές το συνολικό μέγεθος των επικεφαλίδων αυτών είναι άμεσα συγκρινόμενο με το μέγεθος του μηνύματος. Όπως είναι φυσικό οι επικεφαλίδες δεν είναι απαραίτητες στη διεργασία ή τουλάχιστον είναι ελάχιστα απαραίτητες και συνεπώς γρήγορα άχρηστες. Η ελευθέρωση του χώρου που καταλαμβάνουν αυτές οι επικεφαλίδες επιφέρει και βέλτιστη αξιοποίηση του χώρου της ΠΛΔΔ.

Βάση αυτής της παρατήρησης τρεις διαφορετικές προσέγγισης μπορούν να ακολουθηθούν κατά την μεταφορά των Ethernet Frames από την RFA στην ΠΛΔΔ:

Απλή μεταφορά

Τα δεδομένα τοποθετούνται εξολοκλήρου κατά την παραλαβή τους από το δίκτυο στην ΠΛΔΔ. Η τοποθέτηση γίνεται σειριακά από το σημείο που αρχίζει ο κενός χώρος στην ΠΛΔΔ. Οι κεφαλίδες των πρωτοκόλλων τοποθετούνται ολόκληρες, μιας και το Frame μεταφέρεται επακριβώς με τη μορφή που παραλήφθηκε από το δίκτυο.

Κύριο μειονέκτημα αυτής της πολιτικής τοποθέτησης δεδομένων είναι ότι ο χώρος που καταλαμβάνουν οι κεφαλίδες των πρωτοκόλλων αμέσως πριν τα χρήσιμα δεδομένα δεν είναι αξιοποιήσιμος μιας και η μετέπειτα διαγραφή τους θα δημιουργούσε έντονη τμηματοποίηση στην περιοχή. Αποτέλεσμα είναι η γρήγορη πλήρωση του χώρου της ΠΛΔΔ. Παρά το εμφανές μειονέκτημα της η πολιτική αυτή έχει αρκετά πλεονεκτήματα που την κάνει απαραίτητη σε κάποιες περιπτώσεις.

Πλεονέκτημα της πολιτικής αυτής είναι η απλότητα. Το μόνο που χρειάζεται να γνωρίζει ο οδηγός συσκευής κατά την τοποθέτηση των δεδομένων είναι το offset του ελεύθερου χώρου μέσα στην ΠΛΔΔ. Επιπλέον διατηρώντας τις κεφαλίδες των πρωτοκόλλων δίνεται η δυνατότητα στο χρήστη να επεξεργαστεί με όποιο τρόπο θέλει τα πρωτόκολλα οπότε και να αντιμετωπίσει το πιθανό IP Fragmentation που επιφέρει η μεταφορά μεγάλων πακέτων δεδομένων σε τοπικό δίκτυο ή η επικοινωνία μεταξύ δικτύων διαφορετικού MTU (Maximum Transfer Unit).

Επιπρόσθετο πλεονέκτημα της πολιτικής αυτής είναι το γεγονός ότι η μεταφορά του Ethernet Frame γίνεται χωρίς προηγούμενη επεξεργασία από τον οδηγό συσκευής. Αυτό επιτρέπει στον χρήστη να αναπτύξει νέα πρωτόκολλα επικοινωνίας, τοποθετώντας δικές του κεφαλίδες στα προς μεταφορά δεδομένα, χωρίς το άρθρωμα του οδηγού συσκευής να είναι αναγκασμένο να γνωρίζει λεπτομέρειες της υλοποίησης των πρωτοκόλλων.

Πολιτική συμπύκνωσης κεφαλίδων πρωτοκόλλων

Η πολιτική αυτή τοποθετεί πριν από τα λαμβανόμενα δεδομένα, μικρές επικεφαλίδες οι οποίες περιλαμβάνουν επιλεκτικά πεδία από τις κεφαλίδες των πρωτοκόλλων. Τα πλέον απαραίτητα στοιχεία που χρειάζεται μια διεργασία για να πιστοποιήσει την προέλευση των δεδομένων είναι:

{Destination Port, Source Port, Source IP ,Length of Data}

Πλεονέκτημα της πολιτικής αυτής είναι ο ελάχιστος χώρος που καταλαμβάνεται από τις κεφαλίδες των πρωτοκόλλων στην ΠΛΔΔ (10 Bytes από τα 42 Bytes των τριών κεφαλίδων των πρωτοκόλλων). Επιπλέον η πολιτική είναι αρκετά απλή στην τοποθέτηση των δεδομένων στην ΠΛΔΔ, μιας και απαραίτητο είναι μόνο το offset του κενού χώρου μέσα στην περιοχή.

Μειονέκτημα της πολιτικής είναι η αδυναμία πλήρους επεξεργασίας των πρωτοκόλλων από την διεργασία χρήστη, μιας και δεν μπορούμε να κρίνουμε σε κάθε περίπτωση τα πεδία των κεφαλίδων το πρωτοκόλλων που πιθανώς χρειάζεται μια διεργασία να επεξεργαστεί.

Υλοποίηση σωρού τοποθέτησης

Η πολιτική αυτή επιτρέπει την μεταφορά αυτούσιων των επικεφαλίδων πρωτοκόλλων και δεδομένων στην ΠΛΔΔ αλλά σε διαφορετικές περιοχές, υλοποιώντας ένα σωρό. Τα δεδομένα που μεταφέρονται μέσω των πρωτοκόλλων μεταφέρονται την επάνω περιοχή του σωρού, ενώ οι επικεφαλίδες των πρωτοκόλλων τοποθετούνται στην κάτω περιοχή. Ο τρόπος αυτός επιτρέπει την βιβλιοθήκη επικοινωνίας να διαγράφει τις επικεφαλίδες που έχει επεξεργαστεί, εξοικονομώντας χώρο στην ΠΛΔΔ, χωρίς να επηρεάζει τα δεδομένα που μετεφέρθησαν.

Πλεονέκτημα της πολιτικής αυτής είναι η ευελιξία που προσφέρει στην εξοικονόμηση χώρου στην ΠΛΔΔ. Επιπλέον επιτρέπει την πλήρη επεξεργασία των κεφαλίδων των πρωτοκόλλων.

Μειονέκτημα είναι η πολυπλοκότητά της κατά την απόθεση σε διαφορετική περιοχή των επικεφαλίδων και την μετέπειτα διαγραφή τους από τη διεργασία χρήστη. Η διαγραφή αυτή και η επαναχρησιμοποίηση του χώρου που ελευθερώνεται πρέπει να γίνει με τέτοιο τρόπο ώστε να αποφευχθούν οι συνθήκες ανταγωνισμού μεταξύ οδηγού συσκευής και διεργασίας χρήστη.

Η υλοποίηση του σωρού τοποθέτησης των δεδομένων εμφανώς υπερκαλύπτει την επιλεκτική τοποθέτηση στοιχείων των επικεφαλίδων που προτείνει η δεύτερη πολιτική. Η απλή μεταφορά είναι επιβεβλημένη κατά την υλοποίηση ιδιωτικών πρωτοκόλλων σε επίπεδο χρήστη.

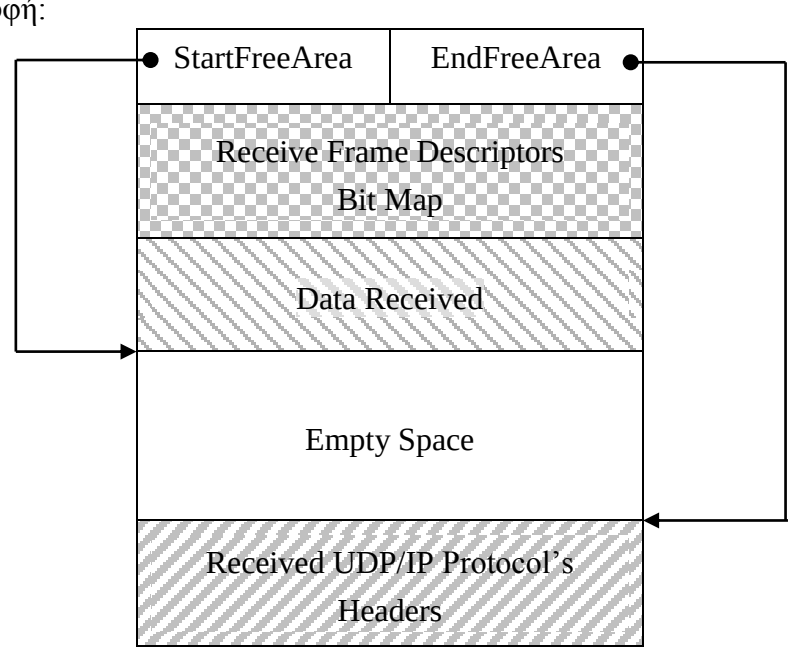
Ο οδηγός συσκευής κάνει χρήση των δυο αυτών πολιτικών (απλή μεταφορά και σωρός τοποθέτησης) ανάλογα με τα δεδομένα που έχει παραλάβει η κάρτα δικτύου και έχουν τοποθετηθεί στην περιοχή RFA. Αν τα δεδομένα που περιλαμβάνονται στο Ethernet Frame ακολουθούν τα πρωτόκολλα UDP/IP η τοποθέτηση τους γίνεται βάση της πολιτικής υλοποίησης του σωρού ώστε εύκολα να αφαιρεθούν από τη διεργασία χρήστη οι επικεφαλίδες των πρωτοκόλλων. Αν τα δεδομένα αφορούν ιδιωτικό πρωτόκολλο της διεργασίας, ακολουθείται η σειριακή μεταφορά τους στην επάνω περιοχή του σωρού, σε κενό χώρο.

5.4 Υλοποίηση διαχείρισης περιοχής λήψης δεδομένων διεργασίας

Για την υλοποίηση και των δύο πολιτικών μεταφοράς των δεδομένων πρέπει να υπάρχει μια ομοιόμορφη αντιμετώπιση της περιοχής λήψης δεδομένων συμβατή με

οποιαδήποτε πολιτική και αν εφαρμοστεί, καθώς δύναται να υπάρχει μίξη πολιτικών για διαφορετικές επικοινωνίες χρήστη με απομακρυσμένους χρήστες.

Η διαμόρφωση της ιδιωτικής περιοχής λήψης δεδομένων της διεργασίας έχει την εξής μορφή:



Σχήμα 5-2: Περιοχή λήψης δεδομένων διεργασίας στον πυρήνα

Στα πρώτα 4 Bytes της περιοχής λήψης δεδομένων διεργασίας τοποθετούνται τα offsets για την αρχή και το τέλος της ελεύθερου χώρου μέσα στην περιοχή. Τα πρώτα 2 Bytes αποτελούν το offset αρχής ελεύθερου χώρου στην περιοχή, τα επόμενα 2 Bytes αποτελούν το offset τέλους ελεύθερου χώρου στην περιοχή λήψης δεδομένων διεργασίας. Χρησιμοποιώντας 2 Bytes έχουμε μια περιοχή $[0,2^{16} - 1] = [0,65535]$ στην οποία κυμαίνονται τα offsets, άρα 65536/4096 = 16 σελίδες συστήματος, όπου είναι και το μέγιστο μέγεθος που μπορεί να αιτηθεί ο χρήστης για περιοχή λήψης δεδομένων.

Στη συνέχεια ακολουθεί ένα bit map στο οποίο κάθε bit αντιστοιχεί σε ένα RFD της περιοχής RFA. Η περιοχή αυτή είναι μεταβλητού μεγέθους, ανάλογα με τον αριθμό από RFDs που έχει δεσμεύσει ο οδηγός συσκευής για την κάρτα δικτύου. Το ανάλογο bit στο bit map μαρκάρεται για να υποδείξει την λήψη κάποιου Ethernet Frame το οποίο απευθύνεται στη διεργασία και ο οδηγός συσκευής το έχει διατηρήσει στη περιοχή RFA, χωρίς να το έχει μεταφέρει στην περιοχή λήψης δεδομένων της διεργασίας. Στη μη μεταφορά του frame συνάγουν 2 παράγοντες:

- *Η πολιτική που έχει επιλέξει ο χρήστης για τα συγκεκριμένα frames*
- *Η πιθανή αδυναμία μεταφοράς του frame στη περιοχή λήψης λόγω έλλειψης ελευθέρου χώρου.*

Αμέσως μετά ακολουθεί η υλοποίηση του σωρού, με τα δεδομένα από τα Ethernet Frames να τοποθετούνται στην κορφή του σωρού και τις κεφαλίδες από τα πρωτόκολλα να τοποθετούνται στον τέλος του σωρού. Ενδιάμεσα είναι ο κενός χώρος που θα τοποθετηθούν τα νέα δεδομένα που παραλαμβάνονται.

Τα δεδομένα που αποτελούν το περιεχόμενο του Ethernet Frame τοποθετούνται σειριακά στην επάνω περιοχή του σωρού. Για τη διαχείριση των δεδομένων αυτών τοποθετούνται επικεφαλίδες 4Bytes πριν από κάθε τοποθέτηση δεδομένων στην επάνω περιοχή του σωρού. Οι επικεφαλίδες μας επιτρέπουν άμεσα να ξέρουμε το μέγεθος των δεδομένων που ακολουθούν καθώς και την πιθανή ύπαρξη κεφαλίδων πρωτοκόλλων στο τέλος του σωρού. Οι επικεφαλίδες έχουν την εξής μορφή:

2 Bytes	2 Bytes
Μέγεθος Δεδομένων που Ακολουθούν	Offset Κεφαλίδων Πρωτοκόλλων UDP/IP

Η τοποθέτηση των κεφαλίδων των πρωτοκόλλων στο τέλος του σωρού δεν επιβάλλει την ύπαρξη κάποιας επιπλέον επικεφαλίδας πριν από αυτά με την προϋπόθεση ότι οι UDP/IP/Ethernet κεφαλίδες που τοποθετούνται έχουν το σύνηθες μέγεθος των 8+20+14=42 Bytes. Διαφοροποίηση αυτού του μεγέθους μπορεί να εισέλθει λόγο του πρωτοκόλλου IP, το οποίο επιτρέπει μεταβλητό μέγεθος στην επικεφαλίδα του. Παρόλα αυτά αν ληφθεί ένα πακέτο με IP κεφαλίδα διαφορετικού μεγέθους από 20 Bytes αυτό αντιμετωπίζεται σαν άγνωστο πρωτόκολλο και τοποθετείται ολόκληρος στην αρχή της περιοχής λήψης δεδομένων.

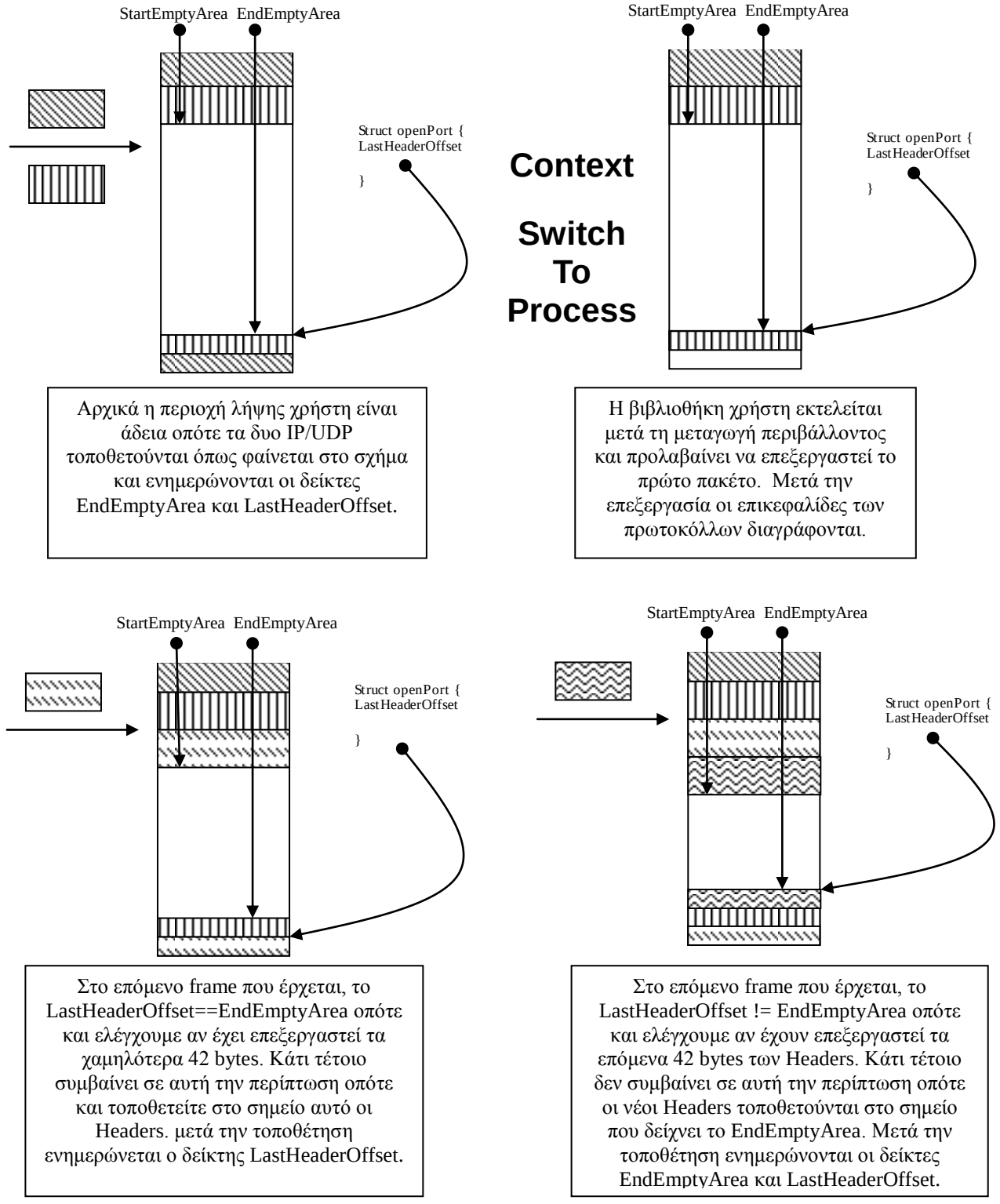
Η λήψη ενός Ethernet Frame τύπου IP και η τοποθέτηση των κεφαλίδων του στο κάτω μέρος της περιοχής λήψης δεδομένων χρήστη ακολουθεί την εξής διαδικασία (η οποία φαίνεται και στο Σχήμα ...):

1. *Στη λίστα struct openPorts που διατηρεί το άρθρωμα με τα διαχειριστικά στοιχεία και παραμέτρους των διάφορων διεργασιών χρήστη, το πεδίο LastHeaderOffset διατηρεί το offset της περιοχής λήψης δεδομένων διεργασίας στο οποίο τοποθετήθηκαν οι τελευταίες επικεφαλίδες των πρωτοκόλλων Ethernet/IP/UDP. Ελέγχεται αν το offset αυτό είναι ίδιο με το offset που δεικνύει το τέλος της άδειας περιοχής μέσα στην περιοχή λήψης χρήστη και το οποίο λαμβάνεται από τα πρώτα 4 Bytes της περιοχής λήψης χρήστη.*

2. *Αν τα δύο offset είναι ίδια ελέγχουμε εάν η βιβλιοθήκη επικοινωνίας επιπέδου χρήστη έχει επεξεργαστεί κάποιες επικεφαλίδες παλαιότερων πακέτων οπότε και έχει ελευθερώσει χώρο από το τέλος της περιοχής λήψης δεδομένων. Για την διαπίστωση αυτή, από τα χαμηλότερα 42 Bytes της περιοχής λήψης χρήστη, ελέγχονται τα Bytes που αντιστοιχούν στο πεδίο TYPE του Ethernet Header. Αν το βρουν 0 σημαίνει ότι οι επικεφαλίδες έχουν επεξεργαστεί ήδη και η τοποθέτηση αρχίζει από αυτό το σημείο. Μετά την τοποθέτηση των επικεφαλίδων ενημερώνεται το πεδίο LastHeaderOffset της δομής openPorts.*

3. *Αν τα δύο offset είναι διαφορετικά, σημαίνει ότι βρισκόμαστε σε ενδιάμεσο σημείο περιοχής που είχαν παλαιότερα τοποθετηθεί επικεφαλίδες οπότε και ελέγχουμε αν έχει επεξεργαστεί η αμέσως επόμενη περιοχή με επικεφαλίδες μεγέθους 42 Bytes. Βασιζόμαστε πάλι στο πεδίο TYPE Ethernet Header το οποίο αν είναι 0 δεικνύει κενό χώρο οπότε και τοποθετούνται εκεί οι νέες επικεφαλίδες. Διαφορετικά η τοποθέτηση γίνεται στο τέλος του άδειου χώρου της περιοχής λήψης δεδομένων χρήστη. Η τοποθέτηση γίνεται από το EndEmptyAreaOffset-42 έως EndEmptyAreaOffset.*

4. *Για κάθε τμήμα ή ολόκληρο frame που τοποθετούμε στην περιοχή, τοποθετούμε πριν 4 Bytes που χρησιμοποιούμε για να καταχωρίσουμε μέγεθος του. Με τον τρόπο αυτό, όταν η βιβλιοθήκη θα κληθεί να παραλάβει νέα δεδομένα από την ΠΛΔΔ θα γνωρίζει το χώρο που αυτά καταλαμβάνουν πριν συναντήσει την επόμενη επικεφαλίδα. Κατά την διάσχιση των επικεφαλίδων όταν βρει επικεφαλίδα με 0 στο πεδίο μεγέθους σημαίνει τη μη ύπαρξη νέων δεδομένων. Επιπλέον η επικεφαλίδα περιέχει και το offset που πιθανά βρίσκεται οι κεφαλίδες πρωτοκόλλων για τα δεδομένα που ακολουθούν. Όταν ολόκληρο το frame έχει τοποθετηθεί στην περιοχή που ακολουθεί την επικεφαλίδα το offset έχει την τιμή 0.*



Σχήμα 5-3: Υλοποίηση σωρού κατά την μεταφορά δεδομένων στην περιοχή διεργασίας

Κεφάλαιο 6
Πειραματικές Μετρήσεις

6.1 Το υλικό

Χρησιμοποιώντας το άρθρωμα που υλοποιεί τον οδηγό συσκευής και τη διεπαφή των διεργασιών χρήστη με την κάρτα δικτύου, επιτελέστηκε αποστολή και λήψη δεδομένων μεταξύ δυο συνδεδεμένων υπολογιστικών συστημάτων με σκοπό την μέτρηση της απόδοσης της επικοινωνίας επιπέδου χρήστη. Τα υπολογιστικά συστήματα που χρησιμοποιήθηκαν για τις μετρήσεις είχαν τα εξής τεχνικά χαρακτηριστικά:

- *Επεξεργαστή Intel Pentium 4 2,66 GHz και Intel Pentium 4 1,6 GHz*
- *512 Mbytes μνήμη*
- *Λειτουργικό σύστημα Linux 2.4*
- *Κάρτα δικτύου Intel Pro/100+ με ελεγκτή Intel 82558*

Η σύνδεση των δυο συστημάτων έγινε back-to-back με crossover UTP καλώδιο.

6.2 Μέτρηση καθυστέρησης

Η μέτρηση της καθυστέρησης κατά την μεταφορά μηνύματος μεταξύ των κόμβων του δικτύου έγινε με την υλοποίηση της Ping-Pong μεθόδου. Κατά την μέθοδο αυτή, ο αποστολέας στέλνει ένα μήνυμα στον παραλήπτη, ο οποίος και το αναμεταδίδει πίσω στον αποστολέα. Η μέτρηση έγινε βάση του εξής αλγορίθμου:

Αποστολέας

```
countFrames=0;
while(countFrames<totalFrames) {
   rdtsc();
   sendFrame();
   while(!rcvFrame()) ;
   rdtsc();
   countFrames++;
}
```

Παραλήπτης

```
countFrames=0;
while(countFrames<totalFrames) {
    while(!rcvFrame()) ;
    sendFrame();
    countFrames++;
}
```

Μετρήσεις με την παραπάνω μέθοδο έγιναν για πολλούς διαφορετικούς αριθμούς επαναλήψεων για κάθε συγκεκριμένο μέγεθος μηνύματος. Ανεξάρτητα από τον αριθμό των επαναλήψεων η τιμή της καθυστέρησης σταθεροποιείται από τις πρώτες επαναλήψεις χωρίς να εμφανίζει ιδιαίτερες μεταβολές στη συνέχεια. Τα αποτελέσματα που παρουσιάζονται είναι για 30 επαναλήψεις. Στο διάγραμμα εμφανίζεται και η καθυστέρηση που εμφανίζεται κατά την υλοποίηση της ίδιας μεθόδου χρησιμοποιώντας τα προσφερόμενα από το λειτουργικό σύστημα UDP Sockets.

Σχήμα 6-1: Καθυστέρηση μεταφοράς μηνύματος

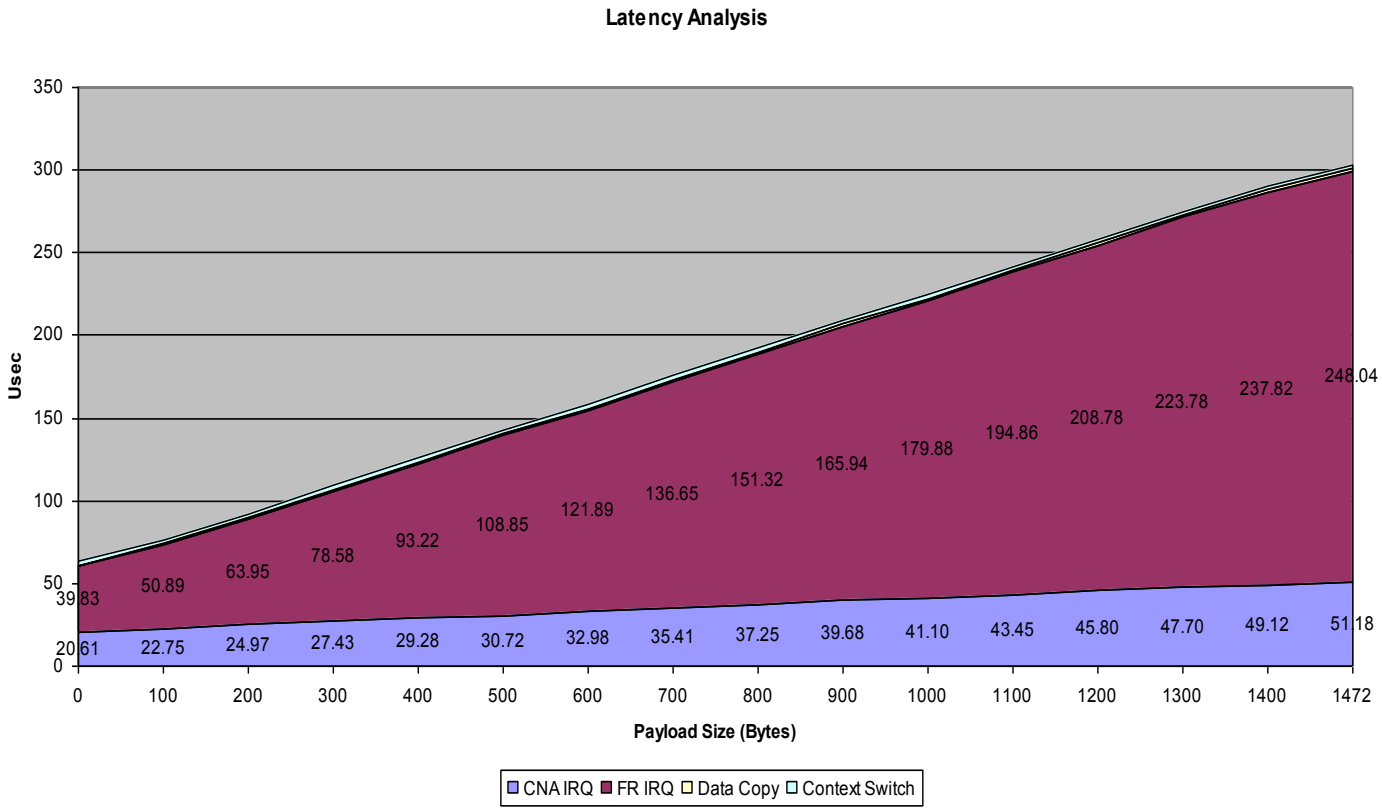
Κατά μέσο όρο εμφανίζεται μια διαφορά της τάξης των 28 usec υπέρ της μεταφοράς μηνύματος σε επίπεδο χρήστη σε σχέση με την χρήση UDP Sockets του λειτουργικού συστήματος. Η τιμή αυτή αφορά τη χρήση της κλασσικής υλοποίησης του αρθρώματος η οποία βασίζεται στην εμφάνιση IRQs για την διαχείριση ασύγχρονων γεγονότων που εμφανίζονται στην κάρτα δικτύου.

Για να διαπιστώσουμε τις επιμέρους καθυστερήσεις κατά την μεταφορά μηνύματος σε επίπεδο χρήστη, τμηματοποιήσαμε το μονοπάτι μεταφοράς μηνύματος

σε επιμέρους στάδια και κάναμε μετρήσεις για την διάρκεια του κάθε σταδίου. Οι μετρήσεις έγιναν είτε από το άρθρωμα με χρήση της εντολής *rdtsc()*, η οποία διαβάζει κύκλους ρολογιού από τους καταχωρητές του επεξεργαστή, είτε από επίπεδο χρήστη με χρήση inline assembly για πρόσβαση στους ίδιους καταχωρητές. Την διαδικασία της αποστολής ενός μηνύματος την χωρίσαμε στα εξής επιμέρους στάδια:

- *Στο στάδιο από τη δημιουργίας κεφαλίδας πρωτοκόλλων και δομής αποστολής Frame από επίπεδο χρήστη μέχρι την έναρξη της αποστολής του Frame.*
- *Στο στάδιο από την έναρξη αποστολή του Frame έως τη στιγμή που θα λάβουμε CNA IRQ, το οποίο δεικνύει το πέρας εκτέλεσης της εντολής αποστολής από την κάρτα δικτύου.*
- *Στο στάδιο μέχρι την εμφάνιση του FR IRQ που δεικνύει την λήψη Ethernet Frame από την κάρτα δικτύου.*
- *Στο στάδιο της μεταφοράς του μηνύματος σε περιοχή μνήμης διεργασίας στον πυρήνα.*
- *Στο στάδιο που η εκτέλεση επανέρχεται στην διεργασία αποστολέα.*

Στο σχήμα 6-2 εμφανίζεται η συνεισφορά κάθε σταδίου στη συνολική καθυστέρηση.



Σχήμα 6-2: Ανάλυση της καθυστέρησης μεταφοράς μηνύματος στον αποστολέα

Όπως παρατηρούμε σημαντική είναι η καθυστέρηση από την έναρξη της αποστολή έως την εμφάνιση του CNA IRQ, σε σημείο σε μικρού μεγέθους μηνύματα να εμφανίζεται σαν το μισό ολόκληρης της καθυστέρησης. Επιπλέον αν σκεφτούμε ότι η ίδια καθυστέρηση εμφανίζεται και στο παραλήπτη κατά την αναμετάδοσης του μηνύματος προς τον αποστολέα, τότε ο παράγοντας αυτός μοιάζει να κυριαρχεί στα μικρού μεγέθους μηνύματα. Η καθυστέρηση αυτή συμπεραίνουμε ότι οφείλεται σε καθυστερήσεις ενεργοποίησης του DMA και της CU μηχανής της κάρτας δικτύου μιας και σε μετρήσεις (οι οποίες δεν φαίνονται στο σχήμα) η καθυστέρηση στην ολοκλήρωση των βημάτων σε επίπεδο χρήστη είναι αμελητέα (της τάξης του 1,5 usec).

Για την επίτευξη χαμηλότερης καθυστέρησης κατά την αποστολή μηνύματος μετρήσεις έγιναν με απενεργοποίηση των IRQ στην κάρτα δικτύου και την χρήση polling για την εύρεση δεδομένων που απευθύνονται στη διεργασία χρήστη απευθείας από την περιοχή RFA. Σε αυτή την περίπτωση και εφόσον δεν εμφανίζονται γεγονότα που να επιβάλλουν την μεταγωγή του συστήματος σε περιοχή πυρήνα όλη η εργασία επιτελείται από περιοχή χρήστη. Οι μετρήσεις σε αυτή την περίπτωση είναι αρκετά χαμηλότερες και εμφανίζονται επίσης στο σχήμα 6-1.

Γενικά η καθυστέρηση που επιτυγχάνει η υλοποίηση μας κατά την μεταφορά μηνύματος εμφανίζονται αρκετά χαμηλές και είναι στα ίδια επίπεδα με εργασίες που έχουν εμφανιστεί στο παρελθόν, ειδικά αν αναλογιστούμε ότι η κάρτα δικτύου που χρησιμοποιήσαμε είναι εν γένει αργότερη σε σχέση με άλλες κάρτες δικτύου, όπως φαίνεται και από την υλοποίηση του Bobnet [6] όπου συγκρίνονται διάφορες κάρτες δικτύου. Ειδικά με την χρήση polling οι επιδόσεις της υλοποίησης μας εμφανίζονται καλύτερες από τις περισσότερες από τις υπόλοιπες υλοποιήσεις.

6.3 Μέτρηση throughput

Για την μέτρηση του throughput που επιτυγχάνεται με τη χρήση επικοινωνιών επιπέδου χρήστη ακολουθήθηκε ο παρακάτω αλγόριθμος:

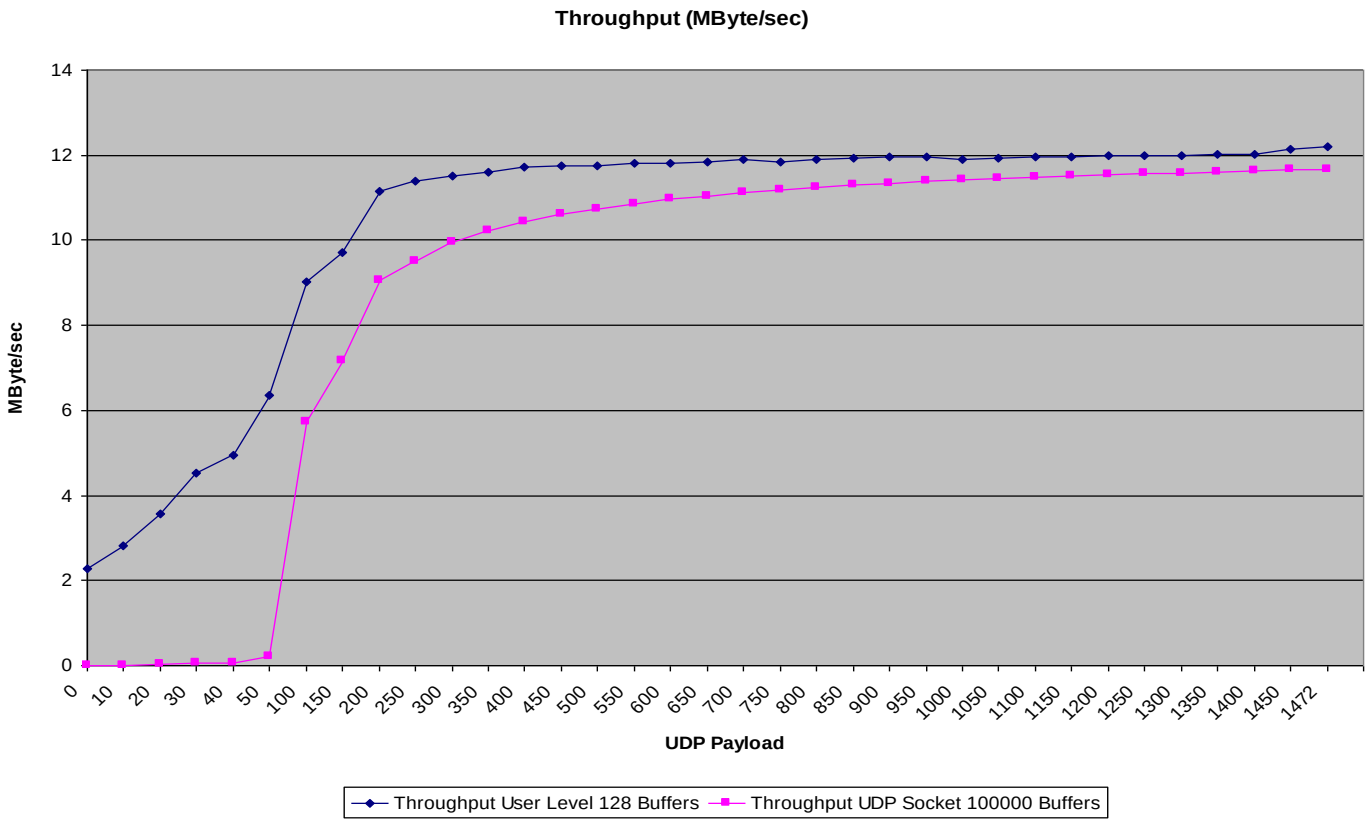
Αποστολέας

```
countBufs=0;
while(countBufs<totalBufs) {
  sendFrame();
  countBufs++;
}
```

Παραλήπτης

```
countBufs=0;
first=1;
while(countBufs<totalBufs) {
  while(!rcvFrame());
  if(first) {
    rdtsc();
    first=0; }
  countBufs++;
}
rdtsc();
```

Η μέτρηση του throughput γίνεται στον παραλήπτη βάση του χρόνου που χρειάζεται η διεργασία παραλήπτη ώστε να λάβει ένα συγκεκριμένο αριθμό από μηνύματα σταθερού μεγέθους. Η ίδια μέθοδος υλοποιήθηκε και με UDP Sockets και τα αποτελέσματα των μετρήσεων εμφανίζονται στο σχήμα 6-3.



Σχήμα 6-3: Throughput κατά την μεταφορά μηνυμάτων

Όπως φαίνεται και στο σχήμα, κατά την επικοινωνία από επίπεδο χρήστη, επιτυγχάνεται υψηλό throughput ακόμα και με μεταφορά περιορισμένου αριθμού μικρών μηνυμάτων. Αυτό είναι άμεσο επακόλουθο της χαμηλής καθυστέρησης που εμφανίζεται κατά την μεταφορά ενός μηνύματος. Το μέγιστο throughput που επιτυγχάνουμε είναι της τάξης των 12,3 MBytes/sec, τιμή πολύ κοντά στη θεωρητική μέγιστη τιμή του Fast Ethernet των 12,5 Mbytes/sec.

Αντίθετα με χρήση UDP Sockets, το throughput που επιτυγχάνεται εμφανίζει σταθερή τιμή μόνο κατά την αποστολή μεγάλου αριθμού μηνυμάτων καθώς κατά την αποστολή μικρού αριθμού μηνυμάτων οι μετρήσεις εμφανίζουν μεγάλες διαφοροποιήσεις μεταξύ τους. Αυτό οφείλεται στην άμεση εξάρτηση της επικοινωνίας από τον πυρήνα του λειτουργικού συστήματος και την εκάστοτε φόρτο που εμφανίζεται σε αυτόν κάθε χρονική στιγμή. Ειδικά για μικρά μηνύματα τα UDP Sockets εμφανίζουν πολύ μικρό throughput ακόμα και κατά την αποστολή μεγάλου

αριθμού μηνυμάτων. Το μέγιστο throughput που εμφανίζεται κατά την χρήση UDP Sockets είναι της τάξης των 11,65 Mbytes/sec.

Κεφάλαιο 7
Μελλοντική Εργασία

7.1 Ανακεφαλαίωση

Δεδομένης της ανάγκης για επικοινωνία μεταξύ απομακρυσμένων οντοτήτων, το κόστος αυτής εμφανίζεται πολλές φορές αρκετά υψηλό σε σχέση με τις δυνατότητες που προσφέρει το φυσικό μέσο. Στην εργασία μας διερευνήσαμε τους παράγοντες που αποτελούν τροχοπέδη για την επίτευξη επικοινωνίας χαμηλής καθυστέρησης και προτείναμε ένα σύστημα για επικοινωνία από επίπεδο διεργασίας χρήστη, χωρίς την παρέμβαση του πυρήνα του λειτουργικού συστήματος.

Όπως επιβεβαιώσαμε στις μετρήσεις μας οι επικοινωνίες αυτού του τύπου εμφανίζουν μικρό κόστους στην καθυστέρηση κατά την μεταφορά μηνύματος καθώς και δίνουν δυνατότητα στον χρήστη να προσαρμόσει τους παραμέτρους της επικοινωνίας ανάλογα με τις εκάστοτε απαιτήσεις του. Επιπλέον διαπιστώσαμε ότι το πρόβλημα της επίτευξης χαμηλής καθυστέρησης δεν είναι μονοσήμαντο και παράγοντες που συχνά παραβλέπονται, όπως για παράδειγμα το υλικό δικτύωσης, εμφανίζουν διαφοροποίηση στην καθυστέρηση, ειδικά κατά την προσπάθεια επίτευξης εξαιρετικά χαμηλού κόστους επικοινωνίας.

7.2 Μελλοντική Εργασία

Το σύστημα για επικοινωνία από επίπεδο χρήστη που υλοποιήθηκε παρέχει τη δυνατότητα στις διεργασίες χρήστη για μεταφορά μηνύματος με χαμηλή καθυστέρηση, οι τιμές τις οποίας είναι ισάξιες αν όχι κάποιες φορές καλύτερες από αυτές παλαιότερων υλοποιήσεων. Παρόλα αυτά επιτρέπει επιπλέον διερεύνηση για επίτευξη ακόμα καλύτερων επιδόσεων, τόσο από άποψη πολιτικών διαχείρισης δεδομένων των διεργασιών, όσο και από την διερεύνηση εξειδικευμένων παραμέτρων και δυνατοτήτων που προσφέρει η κάρτα δικτύου. Επιπλέον βελτιστοποίηση του κώδικα σε επίπεδο μεταφραστή μπορεί να επιφέρει βελτίωση και στη συνολική απόδοση του συστήματος.

Η βιβλιοθήκη που υλοποιήθηκε σε επίπεδο χρήστη επιτρέπει στις διεργασίες την επικοινωνία τους με την κάρτα δικτύου. Νέες διαδικασίες είναι δυνατόν να

υλοποιηθούν στην βιβλιοθήκη ώστε να εμφανίζεται πλουσιότερη από δυνατότητες προς τις διεργασίες χρήστη.

Τέλος, λόγο των διαμοιραζόμενων περιοχών αποστολή και λήψης δεδομένων της κάρτας δικτύου μεταξύ των διεργασιών στο σύστημα μας, δύναται να εμφανιστεί θέμα ασφάλειας των δεδομένων που στέλνονται ή παραλαμβάνονται από το δίκτυο. Οφείλουν να διερευνηθούν τρόποι αντιμετώπισης της πιθανής αλλοίωσης των διαμοιραζόμενων δεδομένων των περιοχών αυτών.

Βιβλιογραφία

[1] M. Boosten, R. W. Dobinson, and P. D. V. van der Stok, "MESH: Messaging and scheduling for fine-grain parallel processing on commodity platforms," *In Proceedings of PDPTA,* June 1999.

[2] D. P. Bovet and M. Cesati. "Understanding the Linux Kernel, 2nd Edition," *O'Reilly*, ISBN: 0-596-00213-0, December 2002.

[3] G. Ciaccio, "Optimal Communication Performance on Fast Ethernet with GAMMA," *In 10 IPPS/SPDP Workshop on Parallel and Distributed Processing*, pp. 534 - 548, 1998.

[4] D. D. Clark, V. Jacobson, J. Romkey and H. Salwen, "An Analysis of TCP Processing Overhead," IEEE Communications Magazine, pp. 23-29, June 1989.

[5] D. E. Comer, "Internetworking with TCP/IP Vol.1: Principles, Protocols, and Architecture (4th Edition)," *Pearson Education*, ISBN: 0-130-18380-6, January 2000.

[6] C. Csanady and P. Wyckoff, "Bobnet: High Performance Message Passing for Commodity Networking Components," *Proceedings of the 2 nd International Conference on Parallel and Distributed Computing and Networks*, Brisbane, Australia, December 1998.

[7] D. Dunning, G. Regnier, G. McAlpine, D. Cameron, B. Shubert, F. Berry, A. M. Merritt, E. Gronke and C. Dodd, "The Virtual Interface Architecture," *IEEE Micro*, pages 66-76, March/April 1998.

[8] T. von Eicken, A. Basu, V. Buch, and W. Vogels, "U-Net: A User-Level Network Interface for Parallel and Distributed Computing," *In Proceedings of the 14th ACM Symposium on Operating Systems Principles*, pages 40-53, December 1995.

[9] T. von Eicken, D. Culler, S. Goldstein, and K. Schauser, "Active messages: A mechanism for integrated communication and computation," *In Proceedings of the Nineteenth International Symposium on Computer Architecture*, ACM Press, 1992.

[10] T. von Eicken and W. Vogels, “Evolution of the Virtual Interface Architecture,” *IEEE Computer*, Volume 31, Number 11, pages 61- 68, November 1998.

[11] Intel, “Intel 8255x 10/100 Mbps Ethernet Controller Family, Open Source Software Developer Manual, Revision 1.0,” January 2003.

[12] J. Kay and J. Pasquale, “The importance of non-data touching processing overheads in TCP/IP,” *Proc. ACM Communications Architectures and Protocols Conf. (SIGCOMM),* San Francisco, CA, pp. 259-269, September 1993.

[13] C. M. Lee, A. T. C. Tam, and C. L. Wang, “Directed point: An efficient communication subsystem for cluster computing,” *In International Conference on Parallel and Distributed Computing Systems (IASTED),* Oct. 1998.

[14] R. P. Martin, “HPAM: An Active Message Layer for a Network of HP Workstations,” *In Proceedings of the Hot Interconnectes Symposium,* 1994.

[15] Myricom’s Homepage http://www.myri.com

[16] S. Pakin, M. Lauria, and A. Chien, “High performance messaging on workstations: Illinois Fast Messages (FM) for Myrinet,” *In Supercomputing '95*, San Diego, CA, December 1995.

[17] I. Pratt and K. Fraser, “*Arsenic: A UserAccessible Gigabit Ethernet Interface*,” In IEEE INFOCOM, pages 67 - 76, April 2001.

[18] L. Prylli and B. Tourancheau, “BIP: A New Protocol Designed for High Performance Networking on Myrinet,” *In IPPS/SPDP Workshops*, pages 472–485, 1998.

[19] Steven H. Rodrigues, Thomas E. Anderson, and David E. Culler, “High-performance local area communication with fast sockets,” *In Proceedings of Usenix Annual Technical Conference,* 1997.

[20] A. Rubini and J. Corbet, “Linux Device Drivers, 2nd Edition,” *O’Reilly*, ISBN: 0-596-00008-1, June 2001.

[21] P. Shivam, P. Wyckoff, and D. Panda, “*EMP: Zero-copy OSbypass NIC-driven gigabit ethernet message passing*,” In Proceedings of SC01, November 2001.

[22] M. Verma and T. Chiueh, “Pupa: A Low-Latency Communication System for Fast Ethernet,” *In Proc. Workshop PC-NOW IPPS/SPDP'98*, number 1388 in Lecture Notes in Computer Science, Orlando, Florida, April 1998.

[23] M. Welsh, A. Basu, and T. Von Eicken, “Low-latency communication over Fast Ethernet,” *In Lecture Notes in Computer Science*, volume 1123, 1996.