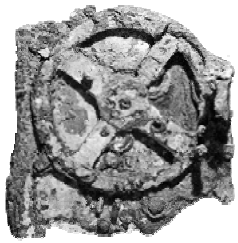
Τμήμα Ηλεκτρονικών Μηχανικών και
Μηχανικών Η/Υ Πολυτεχνείου Κρήτης



Εργαστήριο Μικροεπεξεργαστών και Υλικού

Διπλωματική Εργασία

" Έλεγχος Συμφόρησης σε Επαναχρησιμοποιήσιμο Σχεδιασμό του Πρωτοκόλλου TCP/IP για Αναδιατασσόμενη Λογική"

Κοϊδής Ιωσήφ

Επιβλέποντες Καθηγητές

Καθ. Απόστολος Δόλλας

Καθ. Μιχάλης Πατεράκης

Αν. Καθ. Διονύσιος Πνευματικάτος

Σεπτέμβρης 2003

Ευχαριστίες

Θα ήθελα να ευχαριστήσω, πάνω από όλα, τους γονείς μου και την οικογένειά μου για την καθημερινή τους συμπαράσταση και βοήθεια, τόσο κατά τη διάρκεια της διπλωματικής μου εργασίας όσο και καθ' όλη τη διάρκεια των σπουδών μου.

Παράλληλα, αισθάνομαι την ανάγκη να ευχαριστήσω τον επιβλέποντα καθηγητή αυτής της εργασίας, Κ. Απόστολο Δόλλα, όχι μόνο για τη δυνατότητα που μου προσέφερε να αποκτήσω πολύτιμες γνώσεις και τεχνογνωσία στο ερευνητικό του πεδίο, αλλά και για την αμέριστη βοήθειά του στην διπλωματική μου εργασία. Ευχαριστώ επίσης τον Κ. Διονύσιο Πνευματικάτο και τον Κ. Μιχάλη Πατεράκη για τη συμβολή τους σε θέματα σχεδίασης και υλοποίησης, καθώς και τον Κ. Μάρκο Κιμιωνή, που με την βοήθειά του συνέβαλλε στην υλοποίηση του συστήματος. Τέλος, θα ήθελα να ευχαριστήσω τους φίλους και συνεργάτες μου –προπτυχιακούς και μεταπτυχιακούς φοιτητές – του Εργαστηρίου Μικροεπεξεργαστών και Υλικού, για τις συμβουλές τους και για την στήριξη που μου παρείχαν κατά τη διάρκεια αυτής της εργασίας.

Με εκτίμηση

Στους Γονείς μου.

Περιεχόμενα

Κεφάλαιο 1ο - Εισαγωγή

Τα τελευταία χρόνια γινόμαστε όλοι μάρτυρες της αλματώδους ανάπτυξης της τεχνολογίας, μέσω της οποίας οι τελικοί χρήστες απολαμβάνουν υπηρεσίες υψηλής ποιότητας. Έχει περάσει αρκετός καιρός από την πρώτη σύνδεση δύο υπολογιστών σε δίκτυο επικοινωνίας. Από τότε οι τομείς των τηλεπικοινωνιών και των υπολογιστικών συστημάτων συμπορεύονται και αλληλοσυμπληρώνονται. Η χρήση του δικτύου μεγαλώνει συνεχώς και ταυτόχρονα μεγαλώνουν και οι απαιτήσεις για την απόδοσή του και τις υπηρεσίες του. Η εξάπλωση της χρήσης του δικτύου και η εισαγωγή της έννοιας της δικτύωσης σε διάφορους τομείς της καθημερινής ζωής προβλέπεται να δημιουργήσει μοναδικά συστήματα που θα προσφέρουν πληθώρα έξυπνων υπηρεσιών και διευκολύνσεων.

Η ανάγκη για δικτύωση των υπολογιστών και γενικότερα συστημάτων οδήγησε στη σχεδίαση πρωτοκόλλων και στην υλοποίησή τους. Αρχικά αυτά προσαρτήθηκαν στα λειτουργικά συστήματα των υπολογιστών, στη συνέχεια όμως η υλοποίηση στράφηκε και σε άλλες κατευθύνσεις. Έτσι έχουν δημιουργηθεί ολοκληρωμένα κυκλώματα ASIC (Application Specific Integrated Circuit) που υποστηρίζουν δημοφιλή πρωτόκολλα, έχουν δημιουργηθεί προγράμματα που προσφέρουν τη λειτουργικότητα των πρωτοκόλλων σε μικροελεγκτές ή μικροεπεξεργαστές συγκεκριμένων εφαρμογών και, τέλος, έχουν υλοποιηθεί IP (Intellectual Property) cores με κάποια γλώσσα περιγραφής υλικού (HDL) υψηλού επιπέδου που υλοποιούν τις λειτουργίες πρωτοκόλλων. Αυτά τα IP cores μπορούν να προγραμματίζουν ολοκληρωμένα κυκλώματα αναδιατασσόμενης λογικής (FPGAs, Field Programmable Gate Arrays) δίνοντάς τους τη δυνατότητα υποστήριξης πρωτοκόλλων δικτύων.

Σκοπός αυτής της διπλωματικής εργασίας είναι η σχεδίαση και υλοποίηση έλεγχου συμφόρησης σε Επαναχρησιμοποιήσιμο Σχεδιασμό του Πρωτοκόλλου TCP/IP, το οποίο θα στοχεύει στον προγραμματισμό ενός ολοκληρωμένου κυκλώματος αναδιατασσόμενης λογικής, FPGA. Η επιλογή ανάπτυξης IP core προκύπτει από την ευελιξία που προσφέρει, το μικρό χρόνο ανάπτυξης, αλλά και την προσέγγιση του στη λογική της σχεδίασης σε hardware. Η υλοποίηση του IP core γίνεται με τη χρήση εργαλείων CAD που μεταφέρουν την περιγραφή της αρχιτεκτονικής της σχεδίασης από μια γλώσσα περιγραφής υλικού, όπως η VHDL, σε μορφή που μπορεί να μεταφερθεί σε κάποιο προγραμματιζόμενο ολοκληρωμένο κύκλωμα. Παράλληλα γίνεται χρήση εργαλείων που προσομοιώνουν την

περιγραφόμενη αρχιτεκτονική για να διαπιστωθεί τόσο η λειτουργικότητά της σε λογικό επίπεδο όσο και σε επίπεδο τήρησης των χρονικών περιορισμών.

Τα πρωτόκολλα που υποστηρίζονται σε αυτή την εργασία είναι τα εξής: πρωτόκολλο για το επίπεδο MAC, το πρωτόκολλο ARP (Address Resolution Protocol), το IP (Internet Protocol), το ICMP (Internet Control Message Protocol), το UDP (User Datagram Protocol) και το TCP (Transmission Control Protocol). Αυτή η διπλωματική εργασία αποτελεί συνέχεια και εξέλιξη της διπλωματικής εργασίας του κ. Γιάννη Ζήση. Στην εργασία του κυρίου Ζήση υλοποιήθηκε ένα IP core που υποστηρίζει το πρωτόκολλο TCP/IP δίνοντάς του την δυνατότητα δημιουργίας πολλαπλών συνδέσεων καθώς και την δυνατότητα παραλληλισμού των διεργασιών αποστολής και λήψης τμημάτων TCP. Πιο συγκεκριμένα, το πρωτόκολλο TCP/IP στην εργασία του εκείνη μπορεί να κάνει αίτηση και να δέχεται αιτήσεις για σύνδεση με μέγιστο αριθμό αυτόν τον δεκαέξι συνδέσεων, πάραυτα δεν επιτρέπεται στο πρωτόκολλο να στέλνει και να λαμβάνει πολλαπλά τμήματα TCP με αποτέλεσμα η χρήση του δικτύου να μην είναι αποδοτική κάτι το οποίο σημαίνει ότι το δίκτυο παραμένει ανενεργό για πολύ μεγάλα χρονικά διαστήματα.

Παρατηρούμε λοιπόν πως αν και έχουμε πετύχει πολύ μεγάλη μείωση στον χρόνο επεξεργασίας των συνδέσεων εφόσον πλέον η επεξεργασία γίνεται από εξειδικευμένο υλικό και όχι από έναν γενικής χρήσης επεξεργαστή παρόλα αυτά δεν γίνεται καθόλου αποδοτική χρήση του μέσου μετάδοσης και του δικτύου γενικότερα το οποίο έχει ως αποτέλεσμα πολύ μεγαλύτερους χρόνους για ανταλλαγή δεδομένων μεταξύ των συνδέσεων. Στόχος λοιπόν της παρούσας διπλωματικής εργασίας είναι να συμβάλει στην αύξηση των δυνατοτήτων του υπάρχοντος IP core. Οι επιπλέον δυνατότητες που υποστηρίζονται στο εξελιγμένο αυτό IP core είναι οι εξής. Υποστηρίζεται πλέον η δυνατότητα αποστολής και λήψης πολλαπλών τμημάτων TCP από την σχεδίαση μαζί με τους αντίστοιχους αλγορίθμους οι οποίοι συνοδεύουν το πρωτόκολλο TCP/IP για την αποφυγή συμφόρησης του δικτύου, επίσης υπάρχει πλέον η δυνατότητα υποστήριξης μεγαλύτερου αριθμού συνδέσεων από την πλευρά της σχεδίασης οι οποίες μπορούν να φτάσουν μέχρι τον αριθμό των τριάντα δύο, η σχεδίαση πλέον μπορεί να βρίσκεται χωρίς προβλήματα μέσα σε ένα δίκτυο διακοσίων πενήντα έξη συστημάτων ενώ στην προηγούμενη σχεδίαση ο αριθμός αυτός ήταν τέσσερα, τέλος υλοποιήθηκε και η διεπαφή του συστήματος με το MII κάτι το οποίο είναι απαραίτητο ώστε να μπορέσει να συνδεθεί η σχεδίαση στο δίκτυο μέσω ενός Ethernet Tranceiver.

Η χρησιμότητα της υλοποίησης του TCP/IP stack σε hardware μέσω ενός IP core προκύπτει από το γεγονός ότι μπορεί να προσφερθεί δυνατότητα δικτύωσης σε συστήματα που δεν βρίσκονται κοντά σε Ηλεκτρονικό Υπολογιστή με δυνατότητα εύκολης προσαρμοστικότητας. Παράλληλα μπορεί να παρέχει την ταχύτητα που συνεπάγεται μια

σχεδίαση σε hardware με αποκλειστική ενασχόληση με το πρωτόκολλο καθεαυτό, χωρίς απώλειες σε επεξεργαστική ισχύ. Κάποιες εφαρμογές που είναι δυνατό να αξιοποιήσουν το TCP/IP stack είναι εφαρμογές για home networking όπου διάφορες συσκευές μπορούν να είναι συνδεδεμένες στο διαδίκτυο για παρακολούθηση και έλεγχο. Ακόμα μπορεί να χρησιμοποιηθεί σε κάποια συσκευή που θα μπορεί να κάνει content based routing. Μια τέτοια εφαρμογή δρομολογεί πακέτα με βάση το περιεχόμενό τους (payload) γρήγορα χωρίς την παρέμβαση προγράμματος προς τους κατάλληλους προορισμούς.

Στη συνέχεια του κειμένου, στο 2^{ο} κεφάλαιο, γίνεται μια αναφορά στη σχετική έρευνα που υπάρχει στον τομέα της υλοποίησης του TCP/IP stack πέρα από την υλοποίηση για επεξεργαστές γενικού σκοπού. Στο 3^{ο} κεφάλαιο αναλύονται και θέτονται οι προδιαγραφές και η δομή της παρούσας αρχιτεκτονικής. Στο 4^{ο} κεφάλαιο παρατίθεται η μελέτη της σχεδίασης καθώς και οι αλγόριθμοι οι οποίοι υποστηρίζονται και στο 5^{ο} κεφάλαιο παρατίθεται η υλοποίηση της σχεδίασες και οι σχεδιαστηκές επιλογές οι οποίες παρθηκαν. Στο 6^{ο} κεφάλαιο παρατίθενται αποτελέσματα για τη λειτουργία και την απόδοσή της από τις προσομοιώσεις που έγιναν καθώς και ένα τυπωμένο κύκλωμα Ethernet Tranceiver το οποίο υλοποιήσαμε με σκοπό την διασύνδεση της σχεδίασης στο δίκτυο. Τέλος, στο 7^{ο} κεφάλαιο αναλύονται τα συμπεράσματα που προέκυψαν από την εργασία αυτή και περιγράφονται οι μελλοντικές επεκτάσεις που μπορούν να προστεθούν.

Κεφάλαιο 2ο - Σχετική έρευνα

Τον τελευταίο καιρό πολλές εταιρίες και ορισμένα εκπαιδευτικά ιδρύματα ασχολούνται με την υλοποίηση ή μεταφορά ήδη υπαρχόντων και πολυχρησιμοποιημένων πρωτοκόλλων δικτύων σε χαμηλότερο επίπεδο, ενώ μέχρι τώρα υλοποιούνται με κάποιο πρόγραμμα που τρέχει σε επεξεργαστή γενικού σκοπού. Η μεταφορά αυτή προκύπτει από την ανάγκη για τη δυνατότητα δικτύωσης ενσωματωμένων (embedded) συστημάτων με πρωτόκολλα όπως το TCP/IP αλλά και για την αύξηση της απόδοσης των υπολογιστικών συστημάτων αφαιρώντας το φορτίο αυτών των πρωτοκόλλων από τη διαθέσιμη επεξεργαστική ισχύ. Γενικά έχουν παρατηρηθεί τρεις τρόποι υλοποίησης του πρωτοκόλλου TCP/IP σε χαμηλό επίπεδο. Ένας τρόπος είναι σε μικροελεγκτές με πρόγραμμα γραμμένο για τον εκάστοτε μικροελεγκτή, ένας άλλος τρόπος με την κατασκευή ολοκληρωμένου κυκλώματος ASIC που επιτελεί μόνο αυτή τη λειτουργία, και τέλος η χρήση ολοκληρωμένων κυκλωμάτων αναδιατασσόμενης λογικής FPGAs που προγραμματίζονται με κάποιο IP (Intellectual Property) core που παρέχει τη λειτουργία του δικτύου σε συνδυασμό με εφαρμογές που το χρησιμοποιούν.

Μεταξύ αυτών των τρόπων διακρίνονται πλεονεκτήματα και μειονεκτήματα. Η χρήση μικροελεγκτή ή μικροεπεξεργαστή προσφέρει μια φθηνή λύση και ευκολία ανάπτυξης του προγράμματος που υλοποιεί το TCP/IP stack. Εφόσον πρόκειται για πρόγραμμα, παρουσιάζει πολλές ομοιότητες με τις υλοποιήσεις που υπάρχουν για επεξεργαστές γενικού σκοπού, από όπου μπορεί να χρησιμοποιηθεί η γενικότερη δομή. Η ιδιαιτερότητα του κάθε μικροελεγκτή όμως απαιτεί την προσαρμογή του προγράμματος στο σύνολο εντολών που υποστηρίζει και είναι δυνατόν να χρειάζονται αλλαγές ακόμα και στη μεταφορά του προγράμματος μεταξύ μικροελεγκτών της ίδιας εταιρίας. Ένα ακόμα πλεονέκτημα που παρουσιάζουν οι μικροελεγκτές είναι η σχετική ευκολία στις αλλαγές του προγράμματος είτε για επέκταση είτε για υποστήριξη διαφορετικών πρωτοκόλλων, όπως για παράδειγμα το IPv6. Παρ' όλες τις ευκολίες που προσφέρει η ανάπτυξη του stack πρωτοκόλλων σε μικροελεγκτές, παραμένει χαμηλή η ταχύτητα επεξεργασίας, ενώ και η συνολική απόδοση παραμένει σε χαμηλά επίπεδα, καθώς απαιτείται η εκτέλεση μεγάλου αριθμού εντολών για την επίτευξη της λειτουργικότητας.

Η υλοποίηση του TCP/IP stack σε ASIC έχει πραγματοποιηθεί από κάποιες εταιρίες που θέτουν ως στόχο την ταχύτητα και την απόδοση. Προσφέρουν διεπαφές ικανές να συνδέσουν το ολοκληρωμένο κύκλωμα είτε με μικροελεγκτή είτε με κάποιο bus υπολογιστή, αυξάνοντας την

συνολική απόδοση του συστήματος. Η ταχύτητα που μπορεί να προσφέρει ένα ολοκληρωμένο κύκλωμα κατασκευασμένο για συγκεκριμένη λειτουργία είναι δεδομένη, καθώς κάθε κύκλος εκτέλεσης είναι αφιερωμένος στη λειτουργία αυτή. Παράλληλα η αρχιτεκτονική που χτίζεται γύρω από το πρωτόκολλο επικεντρώνεται αποκλειστικά στην υποστήριξή του, χωρίς περιττές δυνατότητες ενώ μπορεί να προσφέρει και χαμηλή κατανάλωση που είναι ζητούμενη σε ενσωματωμένα συστήματα. Οι αλλαγές όμως είναι αδύνατες σε ένα τέτοιο ολοκληρωμένο κύκλωμα μειώνοντας την ευελιξία του, ενώ για να είναι οικονομικά συμφέρουσα λύση πρέπει να παραχθεί μεγάλος αριθμός τεμαχίων.

Μεταξύ των δύο παραπάνω λύσεων στέκεται η υλοποίηση του TCP/IP stack σαν IP core, με τη χρήση μιας γλώσσας περιγραφής υλικού που στοχεύει σε συσκευές επαναπρογραμματιζόμενης λογικής (π.χ. FPGAs). Επειδή οι γλώσσες περιγραφής υλικού υπόκεινται σε διεθνή πρότυπα, η σχεδίαση που περιγράφουν δεν δεσμεύεται από τους κατασκευαστές αυτών των συσκευών. Η χρήση μιας γλώσσας περιγραφής υλικού, όπως η VHDL, προσφέρει τη δυνατότητα της δημιουργίας μιας αρχιτεκτονικής που υλοποιεί τις λειτουργίες του πρωτοκόλλου, με υποσυστήματα που επιτελούν συγκεκριμένη εργασία, χωρίς περίσσεια λογική. Παράλληλα η χρήση μιας γλώσσας προσφέρει πάντα τη δυνατότητα διατήρησης ενός υψηλού επιπέδου που παρέχει με τη σειρά του τη δυνατότητα πραγματοποίησης αλλαγών και προσαρμογής του συστήματος στις διάφορες ανάγκες. Ο επαναπρογραμματισμός των συσκευών μπορεί να είναι άμεσος μεταφέροντας άμεσα και όλες τις αλλαγές στη σχεδίαση. Η ταχύτητα διατηρείται σε ικανοποιητικά επίπεδα, ενώ με κατάλληλη μελέτη μπορεί να κρατηθεί και η κατανάλωση σε χαμηλά επίπεδα. Ένα ακόμα πλεονέκτημα της δημιουργίας ενός IP core είναι η αυτούσια χρήση του και η ολοκλήρωσή του σε μεγαλύτερα συστήματα με περισσότερες δυνατότητες που βρίσκονται στην ίδια συσκευή.

2.1 Hynix Semiconductors

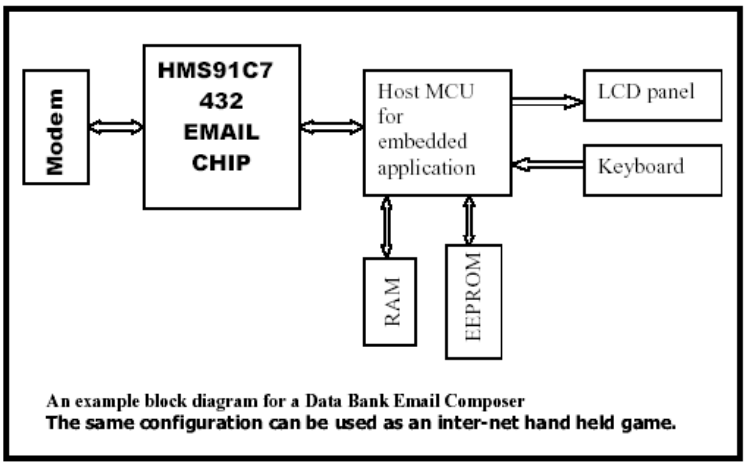
Η εταιρία Hynix [26] έχει κατασκευάσει ένα ολοκληρωμένο κύκλωμα CMOS, το HMS91C7432 που περιλαμβάνει πλήρη υλοποίηση του TCP/IP stack, για τη δικτύωση ενσωματωμένων εφαρμογών. Ενσωματώνει ένα υποσύστημα αποστολής ροών χαρακτήρων ASCII με τη μορφή e-mail, ενώ για τη σύνδεση με το δίκτυο περιλαμβάνει ένα υποσύστημα υλοποίησης του πρωτοκόλλου PPP που μπορεί να συνδεθεί με ένα κοινό modem.

Οι διεπαφές που προσφέρει είναι απλές και λειτουργικές καθώς για τη σύνδεση με τις εφαρμογές υπάρχει ένα bus εύρους 8 bits για δεδομένα, συνοδευόμενο από ένα bus με σήματα ελέγχου εύρους 4 bits. Για τη σύνδεση με κάποιο modem υπάρχει μια σειριακή θύρα.

Οι λειτουργίες που προσφέρει το ολοκληρωμένο αυτό είναι οι εξής:

- Πλήρης υλοποίηση του πρωτοκόλλου TCP/IP.
- Ενσωματωμένη λειτουργία λήψης και αποστολής e-mail.
- Υποστήριξη του πρωτοκόλλου SMTP.
- Υποστήριξη του πρωτοκόλλου POP3.
- Υποστήριξη του πρωτοκόλλου PPP για συνδέσεις dial-up.
- Υποστήριξη του πρωτοκόλλου DNS για επίλυση των URL από δυναμικό DNS εξυπηρετητή.
- Ενσωματωμένος οδηγός για σειριακό modem.
- Υποστήριξη modem V.90 56Kflex.
- Διεπαφή εύρους 8 bit για τις εφαρμογές των χρηστών.

Μια τυπική εφαρμογή με τη χρήση του ολοκληρωμένου κυκλώματος HMS91C7432, όπως προτείνεται από την εταιρεία που το κατασκευάζει, είναι η σύνθεση και αποστολή e-mail από ηλεκτρονική ατζέντα. Στο *Σχήμα 2.1* δίδεται το διάγραμμα αυτής της εφαρμογής. Στο εγχειρίδιο του HMS91C7432 ωστόσο, δεν δίνονται πληροφορίες για τον αριθμό των συνδέσεων που μπορεί να διατηρεί ταυτόχρονα.



Σχήμα 2.1 Διάγραμμα μιας τυπικής εφαρμογής με το HMS91C743.

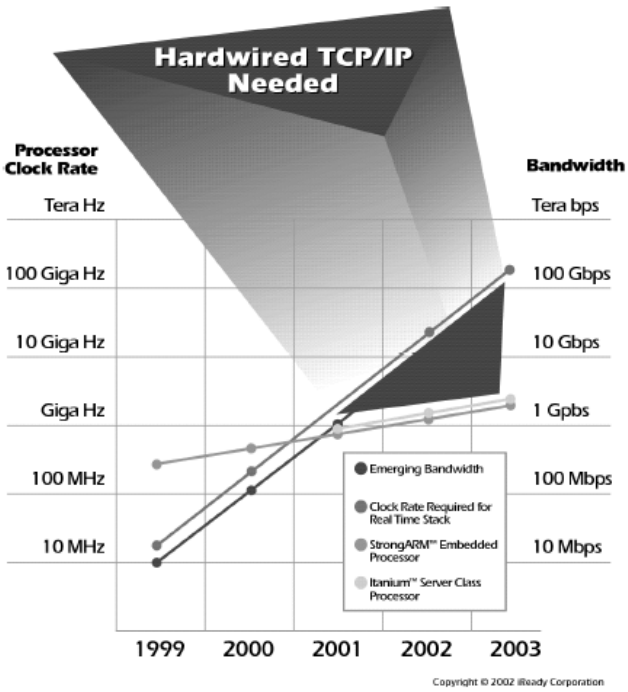
2.2 To project EthernetMAX

Το project EthernetMAX ξεκίνησε από τη συνεργασία δύο εταιριών με μεγάλη πείρα στην κατασκευή ολοκληρωμένων κυκλωμάτων. Η National Semiconductor [27] και η iReady [28] συνεργάζονται με σκοπό να παραχθεί ένα ολοκληρωμένο κύκλωμα ASIC που θα υλοποιεί

πλήρως το TCP/IP stack, περιλαμβάνοντας και το φυσικό επίπεδο, το οποίο είναι πολύ γρήγορο Ethernet. Ο διαμοιρασμός της εργασίας έχει γίνει ως εξής:

- Η εταιρία National Semiconductor υλοποιεί το φυσικό επίπεδο και το επίπεδο MAC, που είναι Gigabit Ethernet, εκμεταλλευόμενη την πείρα που έχει σε αναλογικό hardware.
- Η εταιρία iReady υλοποιεί το πρωτόκολλο TCP/IP καθώς έχει ήδη κατασκευάσει κάποιο core που προσφέρει τη λειτουργία του πρωτοκόλλου TCP/IP σε hardware, και το έχει πατεντάρει.

Ο προβληματισμός για τη δημιουργία ενός τέτοιου ολοκληρωμένου προκύπτει από τις εξής έρευνες όπως αναφέρουν οι δύο κατασκευαστές. Η αύξηση του ρυθμού μετάδοσης δεδομένων στο φυσικό μέσο προχωράει με γρηγορότερους ρυθμούς από ότι η εξέλιξη των επεξεργαστών. Αποτέλεσμα αυτού του γεγονότος είναι οι επεξεργαστές γενικού σκοπού να ξοδεύουν πολύ χρόνο στο πρωτόκολλο TCP/IP, που χρησιμοποιείται ευρέως, και κυρίως όταν εξυπηρετούν εφαρμογές πραγματικού χρόνου. Αναφέρεται ακόμα ότι οι παρόντες επεξεργαστές δεν μπορούν να αντεπεξέλθουν στο ρυθμό μετάδοσης των 10 Gbps, καθώς από ειδικούς εκτιμάται ότι ένας επεξεργαστής γενικού σκοπού απαιτεί ταχύτητα ρολογιού 1 MHz ανά 1 Mbps

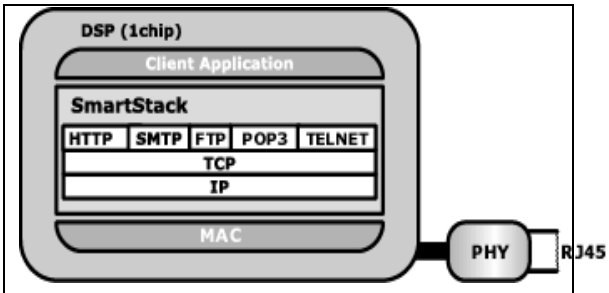


Σχήμα 2.2 Γράφημα απαιτήσεων επεξεργαστικής ισχύος σε σχέση με τους ρυθμούς μετάδοσης δεδομένων.

ρυθμού μετάδοσης στο πρωτόκολλο TCP/IP. Η εταιρία iReady ισχυρίζεται ότι με το ολοκληρωμένο κύκλωμα που κατασκευάζεται και με ταχύτητα ρολογιού στα 200 MHz, θα μπορεί να εξυπηρετήσει τέτοιους ρυθμούς, ενώ δεν δίνονται επιπλέον λεπτομέρειες για το πλήθος των συνδέσεων που μπορεί να υποστηρίξει το σύστημα αυτό και για τα ειδικότερα χαρακτηριστικά της υλοποίησης. Στο γράφημα του *Σχήματος 2.2* [29] διαφαίνεται η ανάγκη για την υλοποίηση του TCP/IP σε hardware.

2.3 eDevice Inc.

Η εταιρία eDevice [30] έχει κατασκευάσει το SmartStack™ το οποίο αποτελεί μια υλοποίηση του TCP/IP stack με πρόγραμμα, προσφέροντας ένα εύκολο τρόπο στους κατασκευαστές ολοκληρωμένων συστημάτων να προσθέσουν στα προϊόντα τους δυνατότητα σύνδεσης με το δίκτυο. Η προσέγγιση αυτής της εταιρείας παρουσιάζει μια ιδιαιτερότητα. Όλο το stack, από το MAC μέχρι και τις εφαρμογές που υποστηρίζει είναι υλοποιημένο σε ένα επεξεργαστή DSP. Στο *Σχήμα 2.3* παρουσιάζεται η εσωτερική οργάνωση των επιπέδων των πρωτοκόλλων που υποστηρίζει η υλοποίηση του SmartStack.



Σχήμα 2.3 Διάγραμμα του SmartStack.

Τα πρωτόκολλα που υποστηρίζει το SmartStack είναι το ARP στο επίπεδο MAC, το IP, το UDP, το TCP και σε επίπεδο εφαρμογών το HTTP, το SMTP, το POP3, το FTP και το TELNET. Υποστηρίζει μία σύνδεση TCP που μπορεί να χρησιμοποιηθεί είτε για εφαρμογές τύπου πελάτη, είτε για εφαρμογές τύπου εξυπηρετητή. Η επικοινωνία με το συνδεδεμένο εξοπλισμό γίνεται με τη χρήση απλού προγράμματος που παρέχει εντολές τύπου AT. Η σύνδεση με το δίκτυο επιτυγχάνεται με την προσθήκη ενός ολοκληρωμένου κυκλώματος Transceiver το οποίο υλοποιεί το φυσικό επίπεδο. Η εταιρία eDevice παρέχει το ίδιο ολοκληρωμένο, με

αλλαγμένο πρόγραμμα όπου στη θέση του επιπέδου MAC έχει μπει υποσύστημα σύνδεσης με modem και έχει προστεθεί το πρωτόκολλο PPP στο stack πριν το επίπεδο IP.

2.4 CMX Systems Co.

Η CMX [31] είναι μια από τις αρκετές εταιρίες που προσφέρουν υλοποιήσεις του TCP/IP stack σε πρόγραμμα για διάφορους μικροελεγκτές και μικροεπεξεργαστές. Καλύπτει μεγάλη γκάμα των μικροελεγκτών του εμπορίου 8 bit και 16 bit, όπως οι 8051, Atmel AVR, MicroChip PIC18Cxxx, STMicroElectronics ST10 και άλλους. Γι' αυτή την κατηγορία υπάρχει το προϊόν CMX-MicroNet το οποίο προσφέρει τις παρακάτω λειτουργίες. TCP, PPP, UDP, SLIP, IP, ARP, BOOTP, HTTP Web Server, FTP Server, TFTP Client, SMTP Client και DHCP Client.

Η εταιρία υποστηρίζει ακόμα και μικροεπεξεργαστές 16 bit και 32 bit με το προϊόν RTX που είναι ένας πυρήνας λειτουργικού συστήματος πραγματικού χρόνου (RTOS). Μαζί με το πακέτο TCP/IP που χρησιμοποιεί τις δυνατότητες του πυρήνα, υποστηρίζεται το TCP/IP stack. Κάποιοι από τους μικροεπεξεργαστές που υποστηρίζονται είναι οι εξής: Siemens 80C16x family, Mitsubishi M16C, PowerPC, 80386/486/586 και άλλοι. Η χρήση του TCP/IP πρωτοκόλλου στο επίπεδο εφαρμογών επιτυγχάνεται με την κλήση συναρτήσεων που είναι υλοποιημένες στη γλώσσα C, ενώ με τη χρήση κατάλληλου compiler γίνεται μετάφραση στον κώδικα μηχανής του εκάστοτε μικροελεγκτή ή μικροεπεξεργαστή που στοχεύει η εφαρμογή. Η δημιουργία των sockets γίνεται σύμφωνα με τους διαθέσιμους πόρους του κάθε μικροελεγκτή/μικροεπεξεργαστή, χωρίς η εταιρεία CMX να δίνει συγκεκριμένο αριθμό για την κάθε συσκευή που υποστηρίζει, ενώ δεν δίνει λεπτομέρειες ούτε για τις απαιτήσεις πόρων που μπορεί να έχει η δημιουργία μιας σύνδεσης, ώστε να μπορεί κανείς να προϋπολογίσει το κόστος σε πόρους καθώς και τι απομένει για τις εφαρμογές.

2.5 Altera, Nios embedded processor

Η εταιρία Altera [37] ειδικεύεται κατεξοχήν στην κατασκευή FPGAs και λογισμικού CAD για τη σχεδίαση cores που μπορούν να προγραμματίσουν τις FPGAs και να τους δώσουν κάποια λειτουργικότητα. Στα πλαίσια αυτά η Altera ανέπτυξε ένα core που υλοποιεί ένα "soft" 32 bit επεξεργαστή, τον Nios. Μεταξύ των πολλών δυνατοτήτων που έχει ο επεξεργαστής αυτός όταν τρέχει σε ένα FPGA προστίθεται και η δυνατότητα σύνδεσής του σε δίκτυο Ethernet ενώ

παρέχονται εργαλεία για την ανάπτυξη εφαρμογών. Ειδικότερα η εταιρία Altera παρέχει τις εξής δυνατότητες δικτύωσης του επεξεργαστή Nios [32].

- Πλατφόρμα που μπορεί να δεχθεί μια επιπλέον κάρτα που υλοποιεί το φυσικό επίπεδο και το επίπεδο MAC με τη χρήση ενός Transceiver.
- Βιβλιοθήκη συναρτήσεων της γλώσσας C που υποστηρίζει το TCP/IP stack, χωρίς την παρουσία λειτουργικού συστήματος. Οι παρεχόμενες συναρτήσεις υποστηρίζουν τα εξής επιμέρους πρωτόκολλα.
    - Απευθείας πρόσβαση σε πλαίσια Ethernet.
    - Πρωτόκολλο ARP.
    - Πρωτόκολλο IP.
    - Πρωτόκολλο ICMP.
    - Πρωτόκολλο UDP.
    - Πρωτόκολλο TCP.

Πέρα από τα γενικότερα χαρακτηριστικά που παρέχει η εταιρεία Altera για τη δυνατότητα διαδικτύωσης του soft επεξεργαστή Nios δεν προσφέρει λεπτομέρειες ειδικότερα για τον αριθμό των συνδέσεων που μπορεί να υποστηρίξει, ούτε για τους διαθέσιμους πόρους στον επεξεργαστή με την παράλληλη υποστήριξη του πρωτοκόλλου TCP/IP.

2.6 To project XCoNet

Αυτή η εργασία προέκυψε με τη συμβολή της εταιρίας Xilinx και του πανεπιστημίου της Χαβάης και απέκτησε την επωνυμία "*Xilinx Chips on the Net*" [33]. Για τους σκοπούς της φτιάχτηκε μια πλατφόρμα που αποτελούνταν από ένα FPGA, μνήμη SRAM 512Kbytes, μία οθόνη LCD, και δυνατότητα σύνδεσης σε δίκτυο Ethernet μέσω ενός Transceiver. Θεωρείται ως η πρώτη προσπάθεια για τη σύνδεση απευθείας κάποιου συστήματος αμιγώς hardware, υλοποιημένο σε FPGA με κάποιο δίκτυο. Υποστηρίζει το TCP/IP stack, χωρίς όμως να παρέχονται λεπτομέρειες για τα χαρακτηριστικά της κατασκευής όπως οι διεπαφές με τα υψηλότερα επίπεδα, τον αριθμό των συνδέσεων που υποστηρίζονται και ποιες εφαρμογές δοκιμάστηκαν.

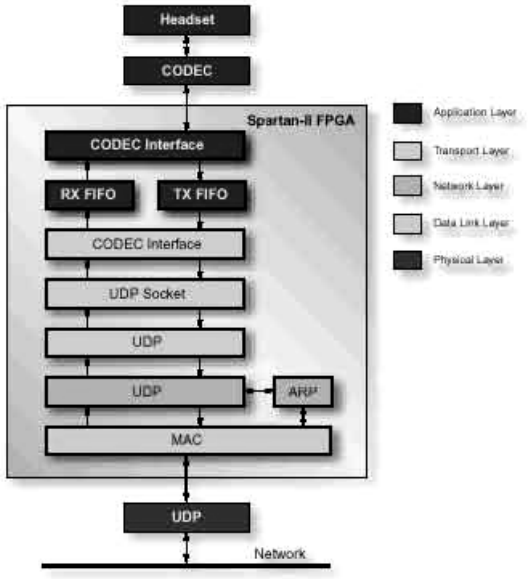
2.7 Το VHDL IP Stack, στο πανεπιστήμιο του Queensland

Στο πανεπιστήμιο του Queensland στην Αυστραλία φτιάχτηκε ένα IP core που υλοποιεί μέχρι κάποιο σημείο τη λειτουργία του πρωτοκόλλου TCP/IP [34]. Υλοποιεί το επίπεδο Ζεύξης

Δεδομένων για δίκτυο Ethernet, το πρωτόκολλο ARP για την αντιστοίχηση διευθύνσεων IP σε διευθύνσεις Ethernet και το πρωτόκολλο IP. Ακόμα υποστηρίζει το πρωτόκολλο ICMP μόνο για πακέτα echo request και αποστολή απαντήσεων echo reply, ενώ η σχεδίαση αυτή μπορεί να δέχεται πακέτα UDP και να τα αποθηκεύει σε κάποια θέση μνήμης. Όλη η σχεδίαση έγινε σε γλώσσα VHDL και δοκιμάστηκε σε μια πλατφόρμα της εταιρίας Xess με ένα FPGA XCV300 της εταιρίας Xilinx. Για το φυσικό επίπεδο χρησιμοποιείται ένα ολοκληρωμένο κύκλωμα Transceiver που λειτουργεί στα 10 Mbps. Στη σχεδίαση αυτή δόθηκε αρκετή βαρύτητα στο πρωτόκολλο IP το οποίο υποστηρίζει τμηματοποίηση πακέτων IP, αλλά μπορεί να εξυπηρετεί ταυτόχρονα μόνο μέχρι 2 διαφορετικές διευθύνσεις IP. Επίσης δεν προσφέρει σημαντική υποστήριξη στο επίπεδο μεταφοράς, πέρα από την αποθήκευση πακέτων UDP.

2.8 Insight Electronics

Η εταιρία Insight Electronics ανέπτυξε ένα IP core το οποίο υλοποιεί το stack των πρωτοκόλλων από το επίπεδο Ζεύξης Δεδομένων για δίκτυο Ethernet, ως το επίπεδο μεταφοράς για το πρωτόκολλο UDP [35]. Επιπλέον έχει φτιαχτεί μια εφαρμογή που ολοκληρώνει τη σχεδίαση και παρέχει τη δυνατότητα για μετάδοση φωνής μέσω IP (Internet Protocol). Στο *Σχήμα 2.4* φαίνεται η εσωτερική δομή της σχεδίασης του IP core.

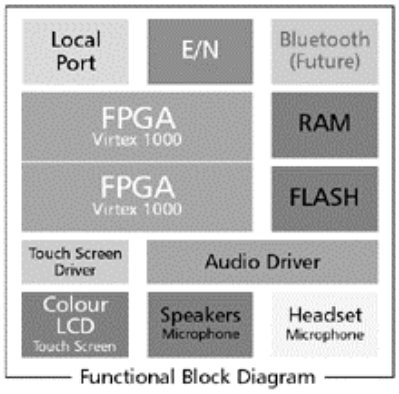


Σχήμα 2.4 Διάγραμμα της εσωτερικής δομής του core VoIP.

Αναλυτικότερα το φυσικό επίπεδο υποστηρίζεται από ένα ολοκληρωμένο κύκλωμα Transceiver που συνδέεται σε δίκτυο Ethernet και το επίπεδο MAC υλοποιείται στο IP core. Στο ίδιο επίπεδο υλοποιείται και το πρωτόκολλο ARP ώστε να μπορεί να αντιστοιχίζει η σχεδίαση τις διευθύνσεις IP σε διευθύνσεις Ethernet. Τα πρωτόκολλα IP και UDP υλοποιούνται επίσης στο IP core και είναι υπεύθυνα για να παραδίδουν καθαρά δεδομένα στην εφαρμογή. Η εφαρμογή υλοποιεί μετάδοση VoIP (Voice over IP), λαμβάνοντας δεδομένα από ένα ολοκληρωμένο κύκλωμα CODEC (COder/DECoder) και παραδίδοντας δεδομένα σε αυτό το ολοκληρωμένο που περιέχονται στα πακέτα UDP. Το ολοκληρωμένο κύκλωμα CODEC αναλαμβάνει τη μετατροπή του αναλογικού σήματος φωνής σε ψηφιακά δεδομένα και αντίστροφα. Το IP core τοποθετείται ολόκληρο, συμπεριλαμβανομένης και της εφαρμογής VoIP, σε ένα FPGA Spartan II της εταιρίας Xilinx και όλο το σύστημα βρίσκεται σε μια πλατφόρμα που έχει κατασκευάσει η εταιρία. Αν δύο τέτοιες πλατφόρμες συνδεθούν σε ένα δίκτυο Ethernet ή και στο Internet, μπορούν δύο χρήστες πλέον να συνομιλήσουν.

2.9 Η πλατφόρμα MMT 2000

Η πλατφόρμα MMT 2000 [36] είναι μια συσκευή που προέκυψε με τη συνεργασία τριών εταιριών. Οι εταιρίες αυτές είναι η Xilinx [38], η Celoxica [39] και η Marconi [40] με τη συμβολή των οποίων κατασκευάστηκε μια πλατφόρμα που αποτελείται από δύο FPGAs Virtex 1000, μνήμη SRAM, μνήμη FLASH, έγχρωμη οθόνη αφής LCD, ολοκληρωμένο κύκλωμα Transceiver στα 10 Mbps και μπορεί να δεχθεί ακουστικά και μικρόφωνο. Στο *Σχήμα 2.5* φαίνεται το διάγραμμα της πλατφόρμας που περιγράφεται εδώ.



Σχήμα 2.5 Διάγραμμα της συσκευής MMT 2000.

Στην πλατφόρμα MMT 2000 έχουν «κατέβει» διάφορα IP cores που έχουν υλοποιηθεί σε γλώσσα C με το περιβάλλον Handel-C που έχει αναπτύξει η Celoxica. Έχει δημιουργηθεί ένα IP core που υποστηρίζει το TCP/IP stack ολοκληρωμένο και μπορεί να δεχθεί μία σύνδεση TCP και δύο συνδέσεις UDP. Ακόμα έχει φτιαχτεί μια εφαρμογή VoIP χαμηλής καθυστέρησης που έχει ολοκληρωθεί με το πρωτόκολλο TCP/IP. Η συσκευή υποστηρίζει τον προγραμματισμό του ενός FPGA από το άλλο μέσω του δικτύου, ενώ άλλα IP cores έχουν δοκιμαστεί και έχουν να κάνουν με την αναπαραγωγή MP3, την αξιοποίηση της οθόνης για ενδείξεις και παιχνίδια. Μελλοντικές προσθήκες στην πλατφόρμα αναφέρονται στην υποστήριξη της τεχνολογίας Bluetooth και τη δυνατότητα παροχής βοήθειας από μακριά μέσω του δικτύου και της οθόνης LCD.

2.10 Το TCP/IP stack στο Πολυτεχνείο Κρήτης

Στο Πολυτεχνείο Κρήτης και συγκεκριμένα στο Εργαστήριο Μικροεπεξεργαστών και Υλικού, έγινε αντιληπτή η ανάγκη για μεταφορά του TCP/IP stack σε hardware, καθώς και τα πλεονεκτήματα που μπορεί να προσφέρει, αρχικά με τη χρήση του σε ενσωματωμένα συστήματα, αλλά και σε γενικότερες εφαρμογές. Τα τελευταία τέσσερα χρόνια έχει ξεκινήσει μια εργασία για την υλοποίηση του TCP/IP stack με τη μορφή ενός IP core καθώς έχουν αναγνωριστεί τα πλεονεκτήματά του όπως έχουν αναφερθεί στην αρχή του κεφαλαίου, ενώ τα εργαλεία που υπάρχουν στη διάθεση του εργαστηρίου προσφέρουν τη δυνατότητα για την ανάπτυξη IP cores και την επαλήθευση της σωστής λειτουργίας τους, τουλάχιστο σε επίπεδο προσομοιώσεων.

Η δεύτερη φάση ανάπτυξης του TCP/IP stack υλοποιήθηκε στα πλαίσια της διπλωματικής εργασίας του κυρίου Γιάννη Ζήση. Με το αυξανόμενο ενδιαφέρον για την υλοποίηση όσο το δυνατόν μεγαλύτερου μέρους του TCP/IP stack σε IP core, κρίθηκε θεμιτό να εξελιχθεί η υλοποίηση της διπλωματικής εργασίας του κ. Κάχρη, ώστε να περιλαμβάνει περισσότερη λειτουργικότητα, με χρησιμοποίηση κομματιών από το ήδη υπάρχον IP core. Παρατηρώντας τις σχετικές εργασίες σε παρόμοια ζητήματα τέθηκε ως βασικός στόχος η υποστήριξη πολλών sockets και συνδέσεων, ώστε να παρέχονται σε hardware δυνατότητες που να πλησιάζουν κατά το δυνατό τις δυνατότητες του software, με ταυτόχρονη επιτάχυνση των διεργασιών που εκτελούνται.

Σε σχέση με την πρώτη φάση της υλοποίησης του IP core τοποθετήθηκε μνήμη στη θέση των καταχωρητών που φυλάσσουν την πληροφορία της μίας σύνδεσης. Η χρήση μνήμης η οποία χωρίζεται εσωτερικά σε κατάλληλες δομές, προσφέρει τη δυνατότητα για την υποστήριξη πολλών συνδέσεων καθώς και την ευχέρεια επέκτασης, ανάλογα με τις απαιτήσεις του

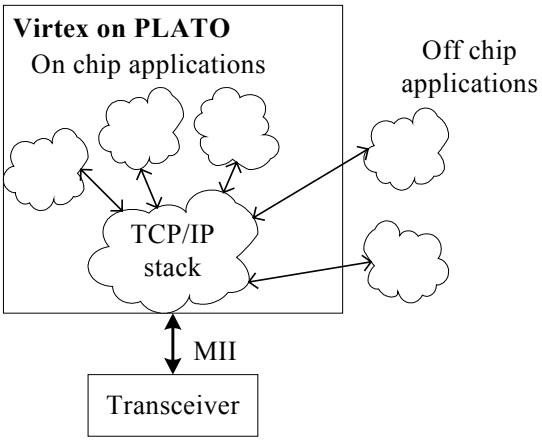
συστήματος που υλοποιείται. Επίσης για την υποστήριξη πολλών συνδέσεων υλοποιείται, με χρήση μνήμης, ένας πίνακας (πίνακας ARP) αντιστοίχησης των διευθύνσεων IP σε διευθύνσεις Ethernet. Η τελική σχεδίαση μπορούσε πλέον να επικοινωνεί όχι μόνο με περισσότερες από μία απομακρυσμένες εφαρμογές, αλλά και με διαφορετικούς απομακρυσμένους σταθμούς.

Ακόμα τέθηκε ως στόχος η δυνατότητα παραλληλισμού στο εσωτερικό της σχεδίασης για να υποστηρίζεται Full Duplex τρόπος λειτουργίας, καθώς και για να εκτελούνται ανεξάρτητες διαδικασίες ταυτόχρονα, με καλή εκμετάλλευση των δυνατοτήτων που προσφέρουν οι σχεδιάσεις σε hardware. Τέλος θεωρήθηκε ότι είναι απαραίτητο η σχεδίαση να παρέχει μια αφαιρετική διεπαφή στο επίπεδο εφαρμογών, για τη διαχείριση των sockets. Έτσι οι εφαρμογές μπορούν να αξιοποιήσουν όλες τις παραπάνω δυνατότητες του TCP/IP stack μέσω ενός περιορισμένου αριθμού εντολών. Θέτοντας όλους τους παραπάνω στόχους επιδιώκεται η δημιουργία ενός IP core που θα είναι κατάλληλο για ενσωματωμένα συστήματα που απαιτούν περισσότερες από μία συνδέσεις, με δυνατότητα επέκτασης του υποστηριζόμενου αριθμού συνδέσεων. Παράλληλα θα μπορεί να χρησιμοποιηθεί σε πιο μοντέρνες και εξεζητημένες εφαρμογές, όπως οι Network Processors, οι οποίοι έχουν αυξημένες απαιτήσεις σε επεξεργαστική ισχύ και στον αριθμό των συνδέσεων που μπορούν να δεχτούν.

Το IP core που σχεδιάστηκε σε εκείνη την εργασία διατηρεί όλα τα χαρακτηριστικά γενικότητας ώστε να μπορεί να απεικονιστεί σε οποιοδήποτε ολοκληρωμένο κύκλωμα αναδιατασσόμενης λογικής FPGA που διαθέτει τους απαραίτητους πόρους. Ειδικότερα όμως η ανάπτυξή του βασίζεται στο FPGA XCV1000bg560 της οικογένειας Virtex το οποίο υπάρχει στην πλατφόρμα PLATO [10] του εργαστηρίου. Η συγκεκριμένη πλατφόρμα με το ολοκληρωμένο κύκλωμα αναδιατασσόμενης λογικής προσφέρει αρκετά πλεονεκτήματα.

- Το FPGA είναι αρκετά μεγάλο σε επαναπρογραμματιζόμενες μονάδες (CLBs) και μπορεί να φιλοξενήσει, μαζί με το TCP/IP stack, εφαρμογές μέσα στο ίδιο ολοκληρωμένο κύκλωμα.
- Η πλατφόρμα παρέχει συνδέσεις με εξωτερικά κυκλώματα για να μπορούν να υποστηριχθούν εφαρμογές εκτός του ολοκληρωμένου κυκλώματος. Κάποιες από αυτές τις συνδέσεις προορίζονται για την προσαρμογή ενός ολοκληρωμένου κυκλώματος Transceiver που υλοποιεί το φυσικό επίπεδο.
- Περιλαμβάνει εξωτερική μνήμη για χρήση είτε από το TCP/IP stack (πίνακες, buffers) είτε για χρήση από τις εφαρμογές, π.χ. αποθήκευση δεδομένων ή αρχείων.

Στο *Σχήμα 2.6* παρατίθεται ένα διάγραμμα που περιγράφει παραστατικά την υλοποίηση του TCP/IP stack στην πλατφόρμα PLATO με το ολοκληρωμένο κύκλωμα FPGA XCV1000bg560.



Σχήμα 2.6 Το TCP/IP stack και εφαρμογές στο FPGA XCV1000bg560 της πλατφόρμας PLATO.

Η τρίτη φάση ανάπτυξης του TCP/IP stack υλοποιείται στα πλαίσια της διπλωματικής εργασίας που περιγράφεται στο παρόν κείμενο. Έχοντας λοιπόν φτάσει σε ένα πολύ καλό επίπεδο στην ανάπτυξη του IP Core κρίθηκε αναγκαίο να δούμε το IP Core και από την πλευρά του δικτύου και συγκεκριμένα από την σκοπιά της απόδοσης πλέον του πρωτοκόλλου έτσι όπως έχει υλοποιηθεί. Το IP Core λοιπόν στην δεύτερη φάση της αναπτυξής του είχε πλέον επιλύσει σημαντικά προβλήματα της πρώτης φάσης με κυριότερα αυτά της δυνατότητας πολλών συνδέσεων, Full Duplex τρόπου λειτουργίας, αλλά και την δυνατότητα εκκίνησης μιας σύνδεσης από την μεριά της σχεδίασης. Πάραυτα είχε ένα μεγάλο μειονέκτημα η αποστολή και η λήψη τμημάτων TCP μπορούσε να γίνει μόνο με ένα τμήμα την φορά γεγονός το οποίο ρίχνει πάρα πολύ το throughput του πρωτοκόλλου. Έτσι λοιπόν αν και κερδίζουμε πάρα πολλά από την λειτουργία του TCP/IP stack σε ειδικό hardware έχοντας πολύ μικρό χρόνο επεξεργασίας, χάνουμε πάρα πολλά σε ότι αφορά την γενικότερη απόδοση του πρωτοκόλλου από τους χρόνους που πρέπει να διανύσουν τα τμήματα μέσα στο δίκτυο αφού δεν του επιτρέπεται να στέλνει πολλά τμήματα μαζί με την μορφή "παραθύρου". Έτσι λοιπόν στην παρούσα διπλωματική εργασία υλοποιήθηκε και ενσωματώθηκε στην ήδη υπάρχουσα μορφή του IP Core ο μηχανισμός congestion control του TCP/IP ο οποίος επιτρέπει στο πρωτόκολλο να στέλνει πολλά τμήματα μαζί με την μορφή "παραθύρου", αλλά και οι αλγόριθμοι οι οποίοι το συνοδεύουν ώστε να ρυθμίζεται ανάλογα το ποσό δεδομένων τα οποία στέλνει ο σταθμός εργασίας, έτσι ώστε να γίνεται η μέγιστη δυνατή αξιοποίηση των πόρων του δικτύου χωρίς παράλληλα να

δημιουργούνται προβλήματα υπερφόρτωσης του δικτύου τα οποία θα οδηγούσαν σε δραματική μείωση της απόδοσής του. Έτσι λοιπόν το IP Core στην παρούσα του φάση υποστηρίζει πλήρως το πρωτόκολλο TCP/IP και μπορεί να κάνει και την καλύτερη δυνατή αξιοποίηση του δικτύου πάνω στο οποίο είμαστε συνδεδεμένοι ανάλογα με τις συνθήκες που επικρατούν κάθε φορά, ανάλογα δηλαδή τον φορτό τον οποίο έχει το δίκτυο κάποια δεδομένη χρονική στιγμή.

Κεφάλαιο 3ο - Προδιαγραφές του συστήματος υλικού και η 2η γενιά του συστήματος

3.1 Η δεύτερη γενιά του συστήματος

Η δεύτερη γενιά του IP core στο Πολυτεχνείο Κρήτης και συγκεκριμένα στο εργαστήριο μικροεπεξεργαστών και υλικού υλοποιήθηκε στα πλαίσια της διπλωματικής του κυρίου Γιάννη Ζήση [16].

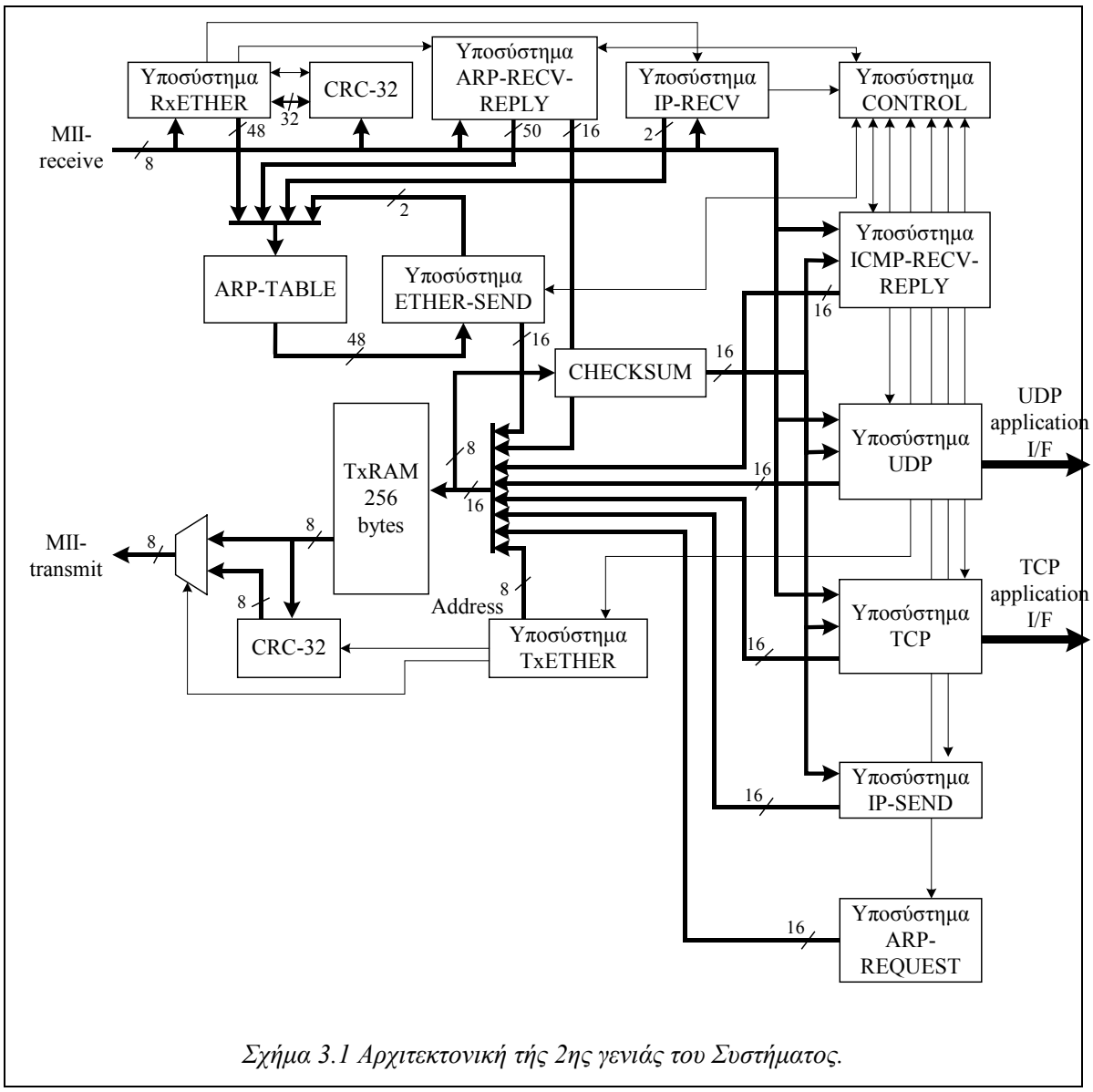
Στην ενότητα αυτή παρατίθεται η αρχιτεκτονική της συνολικής σχεδίασης της δεύτερης γενιάς του συστήματος, στην οποία φαίνονται όλα τα υποσυστήματα που την απαρτίζουν. Στο *Σχήμα 3.1* δίνεται μια εικόνα τη αρχιτεκτονικής καθώς και το data path που μεταφέρει τα δεδομένα μεταξύ των υποσυστημάτων, ενώ παρουσιάζονται συνοπτικά και τα σήματα του control path που καθοδηγούν την επεξεργασία των δεδομένων από το κατάλληλο υποσύστημα σε κάθε περίπτωση.

Πιο αναλυτικά τα δεδομένα εισέρχονται στο σύστημα από τη διεπαφή MII-receive η οποία είναι εύρους 8 bit. Το πρώτο υποσύστημα που λαμβάνει αυτή την πληροφορία είναι το RxETHER που συγχρονίζεται στη λήψη του πλαισίου Ethernet και αναγνωρίζει αν αυτό μπορεί να εισέλθει στη σχεδίαση.

Στη συνέχεια τα δεδομένα του πλαισίου εισέρχονται στο CRC-32 το οποίο υπολογίζει ως το τέλος του πλαισίου ένα πολυώνυμο, που από την τιμή του ελέγχεται αν το πλαίσιο είναι αμιγές ή αν έχουν προκύψει σφάλματα κατά τη διάδοσή του στο δίκτυο.

Το υποσύστημα ARP-RECV-REPLY ενεργοποιείται μόνο αν τα δεδομένα του πλαισίου αναφέρονται σε ARP. Τα δεδομένα που γίνονται δεκτά από το ARP-RECV-REPLY είναι τόσο αιτήσεις για ARP, όσο και απαντήσεις σε αιτήσεις για ARP που έγιναν από τη σχεδίαση. Στην πρώτη περίπτωση αν προκύψει ότι πρέπει να δοθεί απάντηση το ARP-RECV-REPLY σε συνεννόηση με το υποσύστημα ελέγχου (CONTROL) γράφει στη μνήμη μετάδοσης TxRAM την απάντηση αυτή. Στη δεύτερη περίπτωση το υποσύστημα προσπελαύνει τον πίνακα ARP (ARP-TABLE) και τον ενημερώνει, ενώ παράλληλα ενημερώνει το υποσύστημα που είχε κάνει αίτηση για ARP ότι η απάντηση που περιμένει έφτασε στη σχεδίαση.

Αν τα δεδομένα του πλαισίου δεν αναφέρονται σε ARP και περιέχουν πακέτο του πρωτοκόλλου IP τότε ενεργοποιείται το υποσύστημα IP-RECV που φροντίζει σε συνεργασία με το RxETHER να ενημερώσει τον πίνακα ARP ενώ ταυτόχρονα δείχνει στο υποσύστημα CONTROL ποιο πρωτόκολλο περικλείει (ICMP, UDP ή TCP).



Σχήμα 3.1 Αρχιτεκτονική τής 2ης γενιάς του Συστήματος.

Στη συνέχεια τα δεδομένα εισέρχονται σε ένα από τα υποσυστήματα ICMP-RECV-REPLY, UDP ή TCP ανάλογα με την εντολή του υποσυστήματος CONTROL. Σε αυτό το σημείο υλοποιείται το υψηλότερο επίπεδο του προτύπου OSI για τη σχεδίαση και τα δεδομένα υφίστανται την τελευταία επεξεργασία μέσα στο σύστημα. Αν πρόκειται για πακέτο του πρωτοκόλλου ICMP και πιο συγκεκριμένα πακέτο echo τότε το ICMP-RECV-REPLY φροντίζει

να αποστείλει απάντηση γράφοντας στη μνήμη TxRAM ένα πακέτο echo reply. Για τα πακέτα τύπου ICMP δεν υπάρχει επικοινωνία με το επίπεδο των εφαρμογών.

Τα πακέτα που αναφέρονται στα πρωτόκολλα UDP ή TCP εισέρχονται στο κατάλληλο υποσύστημα και υπόκεινται σε κάποια επεξεργασία. Από την επεξεργασία αυτή προκύπτει αν υπάρχει εφαρμογή που να αναμένει τα δεδομένα των πακέτων ή αν υπάρχει εφαρμογή που να μπορεί να τα δεχτεί οπότε και προωθούνται μέσω της διεπαφής του αντίστοιχου υποσυστήματος στο επίπεδο εφαρμογών.

Η αποστολή δεδομένων από τη σχεδίαση προς το δίκτυο, γίνεται με την εγγραφή των πακέτων από κάθε υποσύστημα στη μνήμη TxRAM. Τα υποσυστήματα που πρέπει να εισάγουν κάποιο άθροισμα ελέγχου στα πακέτα τους, το επιτυγχάνουν ενεργοποιώντας το υποσύστημα CHECKSUM καθώς γράφουν στη μνήμη μετάδοσης. Το υποσύστημα CHECKSUM υπολογίζει το άθροισμα ελέγχου, επιστρέφει την τιμή του και το αντίστοιχο σύστημα που το ενεργοποίησε, φροντίζει να γράψει την τιμή του στη μνήμη TxRAM, στην κατάλληλη θέση.

Το υποσύστημα IP-SEND αναλαμβάνει να φτιάξει μια επικεφαλίδα του πρωτοκόλλου IP για το πακέτο που έχει γραφεί στη μνήμη μετάδοσης και δραστηριοποιείται αμέσως, από κάποιο από τα υποσυστήματα ICMP-RECV-REPLY ή UDP ή TCP.

Το υποσύστημα ETHER-SEND δραστηριοποιείται αμέσως μετά το υποσύστημα IP-SEND και αναλαμβάνει αντίστοιχα να εισάγει μια επικεφαλίδα το πρωτοκόλλου Ethernet. Παράλληλα συμβουλεύεται τον πίνακα ARP για να λάβει μια διεύθυνση Ethernet που να αντιστοιχίζεται στη διεύθυνση IP του πακέτου που έχει γραφεί στη μνήμη TxRAM.

Το υποσύστημα ARP-REQUEST ενεργοποιείται όταν είτε το UDP ή το TCP ανοίγει καινούρια σύνδεση για λογαριασμό μιας εφαρμογής πελάτη από το επίπεδο εφαρμογών. Δημιουργεί λοιπόν ένα πακέτο που περιέχει αίτηση για ARP έχοντας εισάγει σε αυτό την ανάλογη πληροφορία και το γράφει στη μνήμη μετάδοσης.

Στη διαδικασία αποστολής πακέτων από το σύστημα στο δίκτυο τελευταίο ενεργοποιείται το υποσύστημα TxETHER το οποίο αναλαμβάνει να μεταφέρει τα δεδομένα που υπάρχουν στη μνήμη TxRAM στο δίκτυο, παρακολουθώντας για συγκρούσεις και φροντίζοντας για τις επαναλήψεις των προσπαθειών μετάδοσης. Κατά τη διάρκεια της μετάδοσης τα δεδομένα που εξέρχονται από τη μνήμη διέρχονται από το υποσύστημα CRC-32 που υπολογίζει ένα πολυώνυμο το οποίο προσαρτάται στο τέλος του πλαισίου που μεταδίδεται, μέσω του πολυπλέκτη που υπάρχει στην έξοδο του συστήματος. Το πλαίσιο καταλήγει στο δίκτυο μέσω της διεπαφής εύρους 8 bit MII-transmit.

Στη συνέχει κοιτάμε χωριστά κάθε υποσύστημα χωρίς να δίνουμε πολλές λεπτομέρειες για την εσωτερική λειτουργία και δομή του καθενός, κυρίως για να δείξουμε τι αλλαγές έχουν

γίνει στην τρίτη πλέον γενιά του συστήματος σε σχέση με την δεύτερη, ποια υποσυστήματα είναι ίδια, ποια άλλαξαν, ποια προστέθηκαν και για ποιους λόγους κάναμε τις επιλογές τις οποίες κάναμε ενώ η ανάλυση ακολουθεί τη διαβάθμιση των επιπέδων OSI από το χαμηλότερο προς τα υψηλότερα.

3.2 Η διεπαφή MII, MIIinterface

Το υποσύστημα MIIinterface είναι ένα νέο υποσύστημα το οποίο προστέθηκε στο IP core. Το υποσύστημα αυτό αναλαμβάνει να υλοποιήσει τη διασύνδεση μεταξύ του IP core και του interface το οποίο χρησιμοποιείται για την διασύνδεση ολοκληρωμένων κυκλωμάτων Ethernet transceivers σύμφωνα με το πρότυπο IEEE 802.3. Συγκεκριμένα οι Ethernet transceivers υλοποιούν ένα απλό interface για επικοινωνία με άλλες συσκευές και αυτό αποτελείται εν τάχη από τα εξής σήματα :

- TX_CLK : Το συγκεκριμένο σήμα (Transmit Clock Output) είναι το ρολόι το οποίο παρέχεται από τον transceiver προς την εξωτερική συσκευή για να υπάρχει συντονισμός κατά την διάρκεια κάποιας μετάδοσης
- TXD[3:0] : Το συγκεκριμένο σήμα (Transmit Data Input) είναι το data bus εύρους τεσσάρων bit το οποίο χρησιμοποιείται για την μετάδοση των δεδομένων προς το δίκτυο. Τα δεδομένα φεύγουν προς το δίκτυο σε κάθε θετική ακμή του TX_CLK.
- TX_EN : Το συγκεκριμένο σήμα (transmit enable) όταν είναι ενεργοποιημένο υποδηλώνει ότι υπάρχουν έγκυρα δεδομένα προς μετάδοση.
- TX_ER : Το συγκεκριμένο σήμα (Transmit Error Input) όταν ενεργοποιηθεί στέλνει ένα συγκεκριμένο σήμα στο δίκτυο το οποίο υποδηλώνει ότι έχει συμβεί κάποιο λάθος κατά την διάρκεια της μετάδοσης, για να είναι έγκυρη η ενεργοποίηση του συγκεκριμένου σήματος πρέπει να είναι ενεργοποιημένο το σήμα TX_EN.
- COL : Το συγκεκριμένο σήμα (Collision Output) ενεργοποιείται όταν ανιχνευτεί από τον transceiver κάποια σύγκρουση πακέτων στο δίκτυο.
- RX_CLK : Το συγκεκριμένο σήμα (Receive Clock Output) είναι το ρολόι το οποίο παρέχεται από τον transceiver προς την εξωτερική συσκευή για να υπάρχει συντονισμός κατά την διάρκεια λήψης δεδομένων.
- RXD[3:0] : Το συγκεκριμένο σήμα (Receive Data Output) είναι το data bus εύρους τεσσάρων bit το οποίο χρησιμοποιείται για την λήψη δεδομένων από το

δίκτυο. Τα δεδομένα εισέρχονται από το δίκτυο σε κάθε θετική ακμή του RX_CLK.

- CRS : Το συγκεκριμένο σήμα (Carrier Sense Output) ενεργοποιείται όταν εντοπιστούν έγκυρα δεδομένα από την πλευρά του δικτύου.
- RX_DV : Το συγκεκριμένο σήμα (Receive Data Valid Output) ενεργοποιείται όταν υπάρχουν έγκυρα και αποκωδικοποιημένα δεδομένα στην είσοδο RXD.
- RX_ER : Το συγκεκριμένο σήμα (Receive Error Output) ενεργοποιείται όταν ανιχνευτεί κάποιας μορφής λάθος από την πλευρά του δικτύου κατά την διάρκεια κάποιας λήψης.

Από τα παραπάνω μπορούμε γρήγορα να δούμε το πρόβλημα το οποίο δημιουργείται και το οποίο δεν είναι άλλο από το γεγονός ότι ο transceiver δέχεται και στέλνει δεδομένα κατά "τετράδες" bit ενώ η σχεδίαση μας δουλεύει με bytes. Εδώ λοιπόν έρχεται το υποσύστημα MIIinterface να γεφυρώσει αυτό το χάσμα μεταξύ των δύο συστημάτων. Περαιτέρω λεπτομέρειες για την λειτουργία του υποσυστήματος θα αναφέρουμε σε παρακάτω κεφάλαιο.

3.3 Το υποσύστημα λήψης πλαισίων Ethernet, RxETHER

Το υποσύστημα αυτό είναι υπεύθυνο για τη λήψη πλαισίων Ethernet (frames) και επιτελεί τέσσερις λειτουργίες.

1. Έλεγχο της διεύθυνσης Ethernet, ώστε να διαπιστωθεί αν το εισερχόμενο πακέτο αναφέρεται στο σύστημα αυτό.
2. Έλεγχο του είδους του πλαισίου, αν αυτό αναφέρεται σε όλα τα συστήματα του τοπικού δικτύου (broadcast), και αν είναι πλαίσιο που κάνει αίτηση για ARP (Address Resolution Protocol) για κάποια διεύθυνση IP. Ανάλογα ενεργοποιεί το υποσύστημα που είναι υπεύθυνο για να ελέγξει αν πρέπει να δοθεί απάντηση ή όχι.
3. Έλεγχο αν πρόκειται για πλαίσιο που αναφέρεται στο επίπεδο IP και ενεργοποίηση του αντίστοιχου υποσυστήματος για να αρχίσει η λήψη του από αυτό.
4. Έλεγχο του CRC που υπάρχει στο τέλος του πλαισίου για αν διαπιστωθεί ότι αυτό λήφθηκε ακέραιο και χωρίς σφάλματα.

Παροχή των δεδομένων που πρέπει να αποθηκευθούν στον πίνακα ARP, ώστε να παραμένει ενημερωμένος. Το συγκεκριμένο υποσύστημα παρέμεινε ώς έχει και στην τρίτη γενιά της σχεδίασης.

3.4 Το υποσύστημα αποστολής πλαισίων Ethernet, ETHER-SEND

Το υποσύστημα που περιγράφεται σε αυτή την ενότητα είναι υπεύθυνο για την προσθήκη της επικεφαλίδας του πρωτοκόλλου Ethernet, ώστε να υπάρχει ένα πλήρες πλαίσιο στη μνήμη μετάδοσης το οποίο μπορεί να μεταδοθεί στο δίκτυο. Με αυτή τη λειτουργία υλοποιείται το επίπεδο λογικής ζεύξης (LLC-Logical Link Control). Το πλαίσιο που δημιουργείται περικλείει κάποιο πακέτο IP το οποίο έχει ήδη γραφεί στη μνήμη. Το συγκεκριμένο υποσύστημα παρέμεινε ώς έχει και στην τρίτη γενιά της σχεδίασης.

3.5 Το υποσύστημα μετάδοσης πλαισίων Ethernet, TxETHER

Το υποσύστημα αυτό αποτελεί ουσιαστικά τη διεπαφή μεταξύ του φυσικού επιπέδου και του επιπέδου ζεύξης δεδομένων για την μετάδοση πλαισίων στο δίκτυο και υλοποιεί το πρωτόκολλο MAC του επιπέδου ζεύξης δεδομένων. Πριν τη μετάδοση του πλαισίου που βρίσκεται αποθηκευμένο στη μνήμη TxRAM, ελέγχει την ύπαρξη φέροντος στο κανάλι από την ένδειξη του σήματος *crs* που παρέχεται από τη διεπαφή MII. Αν το κανάλι δεν είναι κενό τότε μπαίνει σε κατάσταση αναμονής μέχρι να διαπιστωθεί ότι γίνεται κενό.

Μόλις προκύψει κενό, διαβάζει το μήκος του πλαισίου από την πρώτη θέση της μνήμης και στη συνέχεια ξεκινάει τη μετάδοση μεταφέροντας δεδομένα στη διεπαφή MII. Ταυτόχρονα ενεργοποιεί το υποσύστημα υπολογισμού του CRC-32, η τιμή του οποίου προστίθεται στο τέλος του πλαισίου.

Καθ' όλη τη διάρκεια μετάδοσης ελέγχεται αν προκύπτει σύγκρουση εξ αιτίας παράλληλης προσπάθειας μετάδοσης από άλλο σταθμό που βρίσκεται στο ίδιο δίκτυο Ethernet. Αν εντοπιστεί σύγκρουση τότε εκπέμπεται ένα σήμα που διασφαλίζει τη γνωστοποίησή της σε όλους τους σταθμούς του δικτύου Ethernet. Το συγκεκριμένο υποσύστημα παρέμεινε ώς έχει και στην τρίτη γενιά της σχεδίασης.

3.6 Το υποσύστημα λήψης πακέτων που αναφέρονται σε ARP, ARP-RECV-REPLY

Το υποσύστημα αυτό αναλαμβάνει να ανιχνεύσει αν το εισερχόμενο πακέτο περιέχει ARP request ή ARP response για τη διεύθυνση IP του τοπικού συστήματος. Η ενεργοποίησή του γίνεται από το υποσύστημα λήψης πλαισίων Ethernet, μόλις βρεθεί σε κάποιο από αυτά ότι ο τύπος του πρωτοκόλλου που περιέχουν αναφέρεται σε ARP. Αμέσως τότε αρχίζει το διάβασμα στην επικεφαλίδα του και αν διαπιστωθεί ότι πρόκειται για ARP request, αναλαμβάνει να ετοιμάσει ένα πακέτο που να περιέχει ARP response και να το στείλει στο σύστημα που έκανε

την αίτηση ενημερώνοντάς το για την τρέχουσα αντιστοίχηση των διευθύνσεων Ethernet και IP της σχεδίασης.

Αν το πακέτο που εισέρχεται στο σύστημα περιέχει ARP response τότε ελέγχεται αν η απάντηση αυτή αναφέρεται στο τοπικό σύστημα (παρατηρείται η διεύθυνση IP στην οποία στοχεύει) και ενημερώνεται ο πίνακας ARP σύμφωνα με την εισερχόμενη πληροφορία των διευθύνσεων Ethernet και IP του συστήματος που την απέστειλε. Το συγκεκριμένο υποσύστημα παρέμεινε ώς έχει και στην τρίτη γενιά της σχεδίασης.

3.7 Το υποσύστημα αποστολής πακέτων που περιέχουν αιτήσεις ARP, ARP-REQUEST

Οι εφαρμογές για τα πρωτόκολλα υψηλότερων επιπέδων (UDP ή TCP), αν επιχειρήσουν να επικοινωνήσουν για πρώτη φορά με κάποιο απομακρυσμένο σύστημα πρέπει να ζητήσουν ARP για τη διεύθυνση IP τους, ώστε να ενημερωθεί ο πίνακας ARP με την αντίστοιχη διεύθυνση Ethernet. Έτσι διασφαλίζεται ότι τα πακέτα που αποστέλλονται από τις εφαρμογές των υψηλότερων επιπέδων οδηγούνται στο σωστό απομακρυσμένο σύστημα. Το συγκεκριμένο υποσύστημα αναλαμβάνει αυτήν την εργασία να δημιουργήσει και να στείλει δηλαδή ένα πακέτο ARP πρός μια διεύθυνση IP και να ενημερώσει το υποσύστημα το οποίο έκανε αίτηση για ARP όταν έρθει η απάντηση. Το συγκεκριμένο υποσύστημα παρέμεινε ώς έχει και στην τρίτη γενιά της σχεδίασης.

3.8 Το υποσύστημα λήψης πακέτων IP, IP-RECV

Το υποσύστημα λήψης πακέτων IP, ενεργοποιείται από το υποσύστημα Ethernet και αμέσως αρχίζει να διαβάζει το πακέτο, απομονώνοντας τη χρήσιμη πληροφορία που υπάρχει σ' αυτό. Η υλοποίηση του πρωτοκόλλου στο επίπεδο IP έχει σαν στόχο να ανιχνεύσει τον τύπο του πακέτου, να αποθηκεύσει τη διεύθυνση IP της πηγής του πακέτου και να συμβάλλει στην ενημέρωση του πίνακα ARP. Διαβάζοντας λοιπόν το πακέτο ενεργοποιεί ένα σήμα που υποδεικνύει στο υποσύστημα CONTROL τον τύπο του, και αυτό με τη σειρά του αναλαμβάνει να ενεργοποιήσει το κατάλληλο υποσύστημα, είτε το ICMP, είτε το UDP, είτε το TCP.

Η ενεργοποίηση του πρωτοκόλλου του υψηλότερου επιπέδου γίνεται πριν τελειώσει η επικεφαλίδα του IP πακέτου, για να μπορέσουν και τα υπόλοιπα υποσυστήματα να λάβουν την πληροφορία για τις διευθύνσεις IP, γιατί είναι αναγκαία τόσο στο παραπάνω επίπεδο (UDP, TCP), όσο και στο υποσύστημα ICMP, για να μπορέσει να απαντήσει, αν πρόκειται για ICMP

echo request, στη συγκεκριμένη διεύθυνση IP. Η διεύθυνση IP της πηγής του πακέτου χρησιμοποιείται επίσης από το υποσύστημα που αναλύεται εδώ, για να διευθυνσιοδοτηθεί ο πίνακας ARP ώστε να εισαχθεί ως δεδομένο η διεύθυνση Ethernet, η οποία παρέχεται από το υποσύστημα λήψης πλαισίων Ethernet. Ουσιαστικά το υποσύστημα IP καθορίζει και την ενεργοποίηση της εγγραφής στον πίνακα ARP, επειδή αυτό γνωρίζει πότε η διεύθυνση είναι έγκυρη.

Μια ακόμη λειτουργία που αναλαμβάνει το υποσύστημα IP-RECV είναι ο υπολογισμός του μεγέθους ενός τμήματος από κάποια ροή TCP και η παροχή της πληροφορίας αυτής στο πρωτόκολλο TCP. Το συγκεκριμένο υποσύστημα παρέμεινε ώς έχει και στην τρίτη γενιά της σχεδίασης.

3.9 Ο πίνακας ARP

Ο πίνακας ARP είναι μια μονόπορτη σύγχρονη στατική μνήμη με ένα σήμα ενεργοποίησης (Enable) και ένα σήμα για γράψιμο (active high) ή διάβασμα (active low), το Write En. Χρησιμοποιείται και στην τρίτη γενιά του συστήματος ακριβώς όπως και στην δεύτερη, έχει δηλαδή ακριβώς την ίδια λειτουργικότητα και τις ίδιες διασυνδέσεις με τα υπόλοιπα υποσυστήματα τα οποία τον χρησιμοποιούν. Ο πίνακας ARP όμως στην δεύτερη γενιά του συστήματος είχε ένα μεγάλο μειονέκτημα είχε μόνο τέσσερις θέσεις μνήμης κάτι το οποίο είχε ως αποτέλεσμα να μην μπορεί το IP core να συνδεθεί στο ίδιο δίκτυο με παραπάνω από τέσσερις άλλες συσκευές οι οποίες κιόλας θα έπρεπε να έχουν διαφορετικά τα τελευταία 2 bit της διεύθυνσης IP τους. Στην παρούσα φάση λοιπόν του συστήματος ο αντίστοιχος πίνακας είναι 256 θέσεων. Θα εξηγήσουμε σε επόμενο κεφάλαιο πιο εκτενώς την λειτουργικότητα του πίνακα ARP καθώς και γιατί επιλέξαμε να τον υλοποιήσουμε κατ'αυτόν τον τρόπο.

3.10 Το υποσύστημα αποστολής πακέτων IP, IP-SEND

Το υποσύστημα αυτό αναλαμβάνει να προσθέσει μια επικεφαλίδα IP σε δεδομένα που έχουν γραφεί ήδη από άλλο πρωτόκολλο στη μνήμη μετάδοσης TxRAM τοποθετώντας την κατάλληλη πληροφορία σε αυτήν. Το υποσύστημα του πρωτοκόλλου IP δεν είναι προσπελάσιμο απευθείας από κάποια εφαρμογή αλλά ο χειρισμός του βρίσκεται πλήρως στο υποσύστημα κεντρικού ελέγχου, CONTROL. Η πληροφορία που απαιτεί το υποσύστημα IP ώστε να δημιουργήσει σωστή επικεφαλίδα είναι το μέγεθος των δεδομένων που θα συμπεριλάβει, τη διεύθυνση IP για την οποία αυτά προορίζονται και τον τύπο του πρωτοκόλλου που περικλείει.

Τα δεδομένα που συμπεριλαμβάνει ένα πακέτο IP μπορεί να είναι είτε ένα πακέτο UDP, είτε ένα τμήμα TCP, είτε ένα πακέτο ICMP echo reply. Το συγκεκριμένο υποσύστημα παρέμεινε ως έχει και στην τρίτη γενιά της σχεδίασης.

3.11 Το υποσύστημα λήψης και αποστολής πακέτων ICMP, ICMP-RECV-REPLY

Η δεύτερη γενιά της σχεδίασης υποστηρίζει τη λήψη και την αποστολή πακέτων ICMP echo request και echo reply. Όταν διαπιστωθεί από το πρωτόκολλο IP ότι πρόκειται για πακέτο αυτού του τύπου ενεργοποιείται το υποσύστημα ICMP το οποίο διαβάζει κάποια από τα πεδία τα οποία χρειάζονται για να δοθεί σωστά η απάντηση. Πρώτα επιβεβαιώνεται ότι πρόκειται για πακέτου echo request, και στη συνέχεια φυλάσσονται τα πεδία Identifier και Sequence Number τα οποία πρέπει να επιστραφούν αυτούσια στην απάντηση.

Το πρωτόκολλο ICMP θεωρείται ότι βρίσκεται στο επίπεδο δικτύου μαζί με το πρωτόκολλο IP, ουσιαστικά όμως στην υλοποίηση το αντιλαμβανόμαστε σαν ένα πρωτόκολλο υψηλότερου επιπέδου, για να ενταχθεί στο συνολικό μηχανισμό της σχεδίασης. Παρέχεται έτσι η δυνατότητα να μην προστίθεται περίσσεια λογική και να γίνεται καλύτερη εκμετάλλευση της διαχείρισης που προσφέρει το υποσύστημα κεντρικού ελέγχου. Το συγκεκριμένο υποσύστημα παρέμεινε ως έχει και στην τρίτη γενιά της σχεδίασης.

3.12 Το υποσύστημα κεντρικού ελέγχου, CONTROL

Στη σχεδίαση που υλοποιήθηκε υπάρχει ένα κεντρικό σύστημα ελέγχου για τον καθορισμό της ροής της πληροφορίας των πακέτων που εισέρχονται στο σύστημα αλλά και των πακέτων που εξέρχονται από αυτό. Υλοποιεί δηλαδή τη διαδικασία απόπλεξης των εισερχόμενων πακέτων και τη διαδικασία πολυπλεξίας των εξερχόμενων πακέτων μεταξύ του επιπέδου δικτύου και του επιπέδου μεταφοράς. Επιπλέον ασκεί τον κεντρικό έλεγχο για την εγγραφή στην μνήμη μετάδοσης (TxRAM), καθώς χτίζεται ένα πακέτο για να σταλθεί από το σύστημα στο τοπικό δίκτυο. Το υποσύστημα Control χρησιμοποιήθηκε στην παρούσα φάση της σχεδίασης ακριβώς με την ίδια λογική όπως και στην δεύτερη γενιά έγιναν όμως κάποιες παρεμβάσεις στην λειτουργία του ώστε να εναρμονίζεται με τις νέες συνθήκες οι οποίες προκύπτουν λόγω της παρουσίας του congestion control στην σχεδίασή μας.

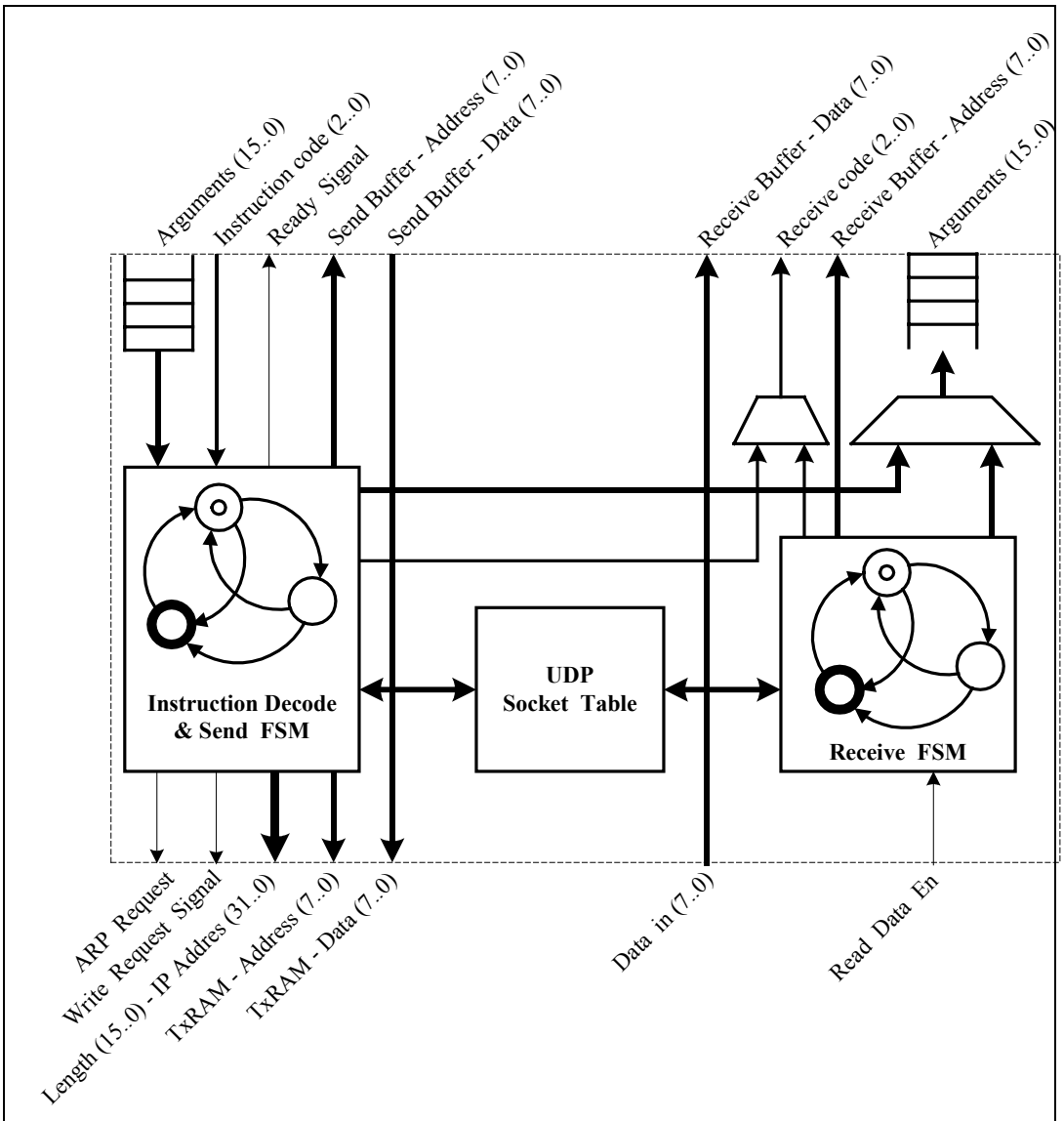
3.13 Το υποσύστημα λήψης και αποστολής μηνυμάτων UDP

Το υποσύστημα UDP είναι υπεύθυνο για τη λήψη και την αποστολή μηνυμάτων UDP μεταξύ της σχεδίασης και ενός απομακρυσμένου σταθμού για περισσότερες από μία εφαρμογές. Παράλληλα προσφέρει τη δυνατότητα επικοινωνίας με το επίπεδο εφαρμογών με την ανταλλαγή κάποιων εντολών που συνοδεύονται από τα αντίστοιχα ορίσματά τους.

Το υποσύστημα UDP αποτελείται από τα εξής επιμέρους υποσυστήματα.

- Μια FIFO για τα ορίσματα των εντολών που δίνει το επίπεδο εφαρμογής στο UDP.
- Μια FIFO για τα ορίσματα των απαντήσεων και των εντολών που επιστρέφει το UDP στο επίπεδο εφαρμογής.
- Ένα FSM που αποκωδικοποιεί τις εντολές, τις εκτελεί, αποστέλλει μηνύματα UDP και επιστρέφει απαντήσεις σε κάποιες από τις εντολές.
- Ένα FSM που ενεργοποιείται κατά τη λήψη κάποιου μηνύματος, ελέγχει αν γίνεται δεκτό, αποθηκεύει τα δεδομένα του στη μνήμη λήψης και δίνει μια εντολή στο επίπεδο εφαρμογής που ειδοποιεί για την άφιξη του μηνύματος.
- Τον πίνακα που καταγράφονται τα sockets.
- Πολυπλέκτες για την επιστροφή απαντήσεων και εντολών προς το επίπεδο εφαρμογών.

Στο *Σχήμα 3.2* φαίνεται παραστατικά η εσωτερική δομή του υποσυστήματος UDP. Το υποσύστημα UDP κρατήθηκε ως έχει και στην παρούσα φάση της σχεδίασης μίας και ο στόχος αυτής της διπλωματικής ήταν η προσθήκη congestion control στην υλοποίηση του TCP πρωτοκόλλου. Οι αλλαγές οι οποίες έγιναν στα άλλα υποσυστήματα τα οποία αποτελούν αυτήν την σχεδίαση ήταν αυτά τα οποία ήταν απαραίτητο να γίνουν ώστε να ενσωματωθεί το congestion control με την υπόλοιπη σχεδίαση. Όπως καταλαβαίνουμε λοιπόν το συγκεκριμένο υποσύστημα παρέμεινε ως έχει και στην τρίτη γενιά της σχεδίασης.



Σχήμα 3.2 Εσωτερική Δομή και διεπαφές του υποσυστήματος UDP.

3.14 Το υποσύστημα TCP

Το υποσύστημα TCP είναι το υποσύστημα στο οποίο επικεντρωθήκαμε κυρίως στα πλαίσια αυτής της διπλωματικής, μιας και ο έλεγχος συμφόρησης είναι βασικό κομμάτι της υλοποίησης του πρωτοκόλλου TCP/IP. Το υποσύστημα TCP λοιπόν της σχεδίασής μας, σχεδιάστηκε σχεδόν εξολοκλήρου από την αρχή. Ο λόγος για τον οποίο λέμε σχεδόν είναι γιατί επιλέξαμε να ακολουθήσουμε την ίδια αρχιτεκτονική η οποία είχε χρησιμοποιηθεί και στην δεύτερη γενιά. Χρησιμοποιήσαμε δηλαδή την ίδια δομή για το υποσύστημα TCP αλλά έχει αλλάξει εξολοκλήρου από πλευράς λειτουργικότητας και λειτουργίας, ώστε να μπορέσει να υποστηρίξει το congestion control. Το υποσύστημα TCP θα το αναλύσουμε ενδελεχώς σε μετέπειτα κεφάλαιο με πολύ περισσότερες λεπτομέρειες οι οποίες θα αναφέρονται στην υλοποίηση και στην λειτουργικότητά του.

Κεφάλαιο 4ο - Διαχείρηση του Congestion Control

Σε αυτό το κεφάλαιο θα αναφερθούμε και θα περιγράψουμε το πώς γίνεται ο έλεγχος συμφόρησης του δικτύου (Congestion Control) από το πρωτόκολλο TCP. Πιο συγκεκριμένα θα αναφερθούμε εκτενέστερα στην αναγκαιότητα της ύπαρξής του ως αναπόσπαστο κομμάτι του πρωτοκόλλου, και θα περιγράψουμε και θα αναλύσουμε διεξοδικά τους αλγορίθμους οι οποίοι το υλοποιούν. Επίσης θα μιλήσουμε για το πώς πρέπει να συμπεριφέρεται το TCP όταν κάποιες συνδέσεις έχουν μείνει ανενεργές για μεγάλο χρονικό διάστημα, καθώς και κάποιες μεθόδους οι οποίες μπορούν να χρησιμοποιηθούν στην δημιουργία και αποστολή επιβεβαιώσεων.

Ο έλεγχος συμφόρησης του δικτύου γίνεται με τέσσερις αλγορίθμους. Αυτοί ονομαστικά είναι ο slow start, ο congestion avoidance, ο fast retransmit και ο fast recovery. Οι αλγόριθμοι αυτοί χρησιμοποιούνται ταυτοχρόνως από το πρωτόκολλο ανάλογα με τις τιμές που έχουν κάποιες μεταβλητές. Παρακάτω θα προχωρήσουμε στην επεξήγηση κάποιων όρων τους οποίους θα χρησιμοποιήσουμε παρακάτω για να περιγράψουμε την λειτουργία των παραπάνω αλγορίθμων.

- Τμήμα TCP (Segment) : Είναι ένα οποιοδήποτε πακέτο TCP/IP το οποίο μπορεί να μεταφέρει δεδομένα ή κάποια επιβεβαίωση ή και τα δύο.
- Μέγιστο μέγεθος πακέτου αποστολέα (Sender Maximum Segment Size) : Είναι το μέγιστο μέγεθος πακέτου το οποίο μπορεί ο αποστολέας να αποστείλει προς κάποιον αποδέκτη. Η τιμή του μπορεί να βασιστεί στο μέγιστο μέγεθος πακέτου το οποίο μπορεί να διακινηθεί μέσω του δικτύου, το μέγιστο μέγεθος πακέτου του δέκτη ή άλλους παράγοντες. Το μέγεθος αυτό δεν περιλαμβάνει την επικεφαλίδα και τα options.
- Μέγιστο μέγεθος πακέτου δέκτη (Receiver Maximum Segment Size) : Είναι το μέγιστο μέγεθος πακέτου το οποίο ο δέκτης μπορεί να αποδεχθεί. Αυτή είναι και η τιμή την οποία στέλνει ο δέκτης στο MSS (Maximum Segment Size) κατά την διαδικασία σύναψης της σύνδεσης μεταξύ δύο απομακρυσμένων σταθμών. Αν δεν χρησιμοποιείται η δυνατότητα του MSS κατά την διαδικασία σύναψης της σύνδεσης θεωρούμε πώς το αντίστοιχο νούμερο είναι 536 bytes. Πάλι το μέγεθος αυτό δεν περιλαμβάνει την επικεφαλίδα και τα options.
- Πακέτο μέγιστου μεγέθους (Full – Sized Segment) : Είναι το πακέτο το οποίο περιέχει το μέγιστο επιτρεπτό αριθμό bytes

- Παράθυρο δέκτη (Receiver Window) : Η πιο πρόσφατη τιμή την οποία έχει δώσει ο δέκτης για το παράθυρο δεδομένων το οποίο μπορεί να δεχθεί. Και λέγοντας παράθυρο εννοούμε τον χώρο τον οποίο έχει διαθέσιμο στους buffers του ο δέκτης για να δεχθεί δεδομένα TCP.
- Παράθυρο συμφόρησης (Congestion Window) : Είναι μια μεταβλητή η οποία κρατείται από το πρωτόκολλο TCP η οποία περιορίζει το ποσό των δεδομένων τα οποία μπορούν να αποσταλούν πρός το δίκτυο. Κάθε χρονική στιγμή το TCP δεν μπορεί να στείλει δεδομένα με sequence number μεγαλύτερο από το άθροισμα του μεγαλύτερου επιβεβαιωμένου sequence number και του μικρότερου αριθμού εκ των παραθύρου συμφόρησης και του παραθύρου του δέκτη.
- Αρχικό παράθυρο (Initial Window) : Είναι η αρχική τιμή του παραθύρου συμφόρησης του αποστολέα αφού έχει ολοκληρωθεί η διαδικασία σύναψης της σύνδεσης μεταξύ αποστολέα και δέκτη.
- Παράθυρο απώλειας (Loss Window) : Είναι η τιμή του παραθύρου συμφόρησης αφού ο αποστολέας καταλάβει ότι έχει χαθεί κάποιο πακέτο μέσω του χρονομετρητή αναμετάδοσης.
- Παράθυρο επανεκκίνησης (Restart Window) : Είναι η τιμή του παραθύρου συμφόρησης εφόσον το TCP ξαναξεκινάει μεταδώσεις μετά από ένα χρονικό διάστημα όπου έμεινε ανενεργό.
- Flight Size : Είναι το ποσό των δεδομένων τα οποία έχουν σταλεί αλλά δεν έχουν επιβεβαιωθεί ακόμα.

Στην παρακάτω ενότητα θα περιγράψουμε τους τέσσερις αλγορίθμους που υλοποιούν τον έλεγχο συμφόρησης τον slow start, τον congestion avoidance και τέλος τους fast retransmit και fast recovery. Ενδεικτικά εδώ πρέπει να αναφέρουμε ότι στο RFC2581 [23], το οποίο περιγράφει το congestion control αναφέρεται πώς μπορεί σε κάποιες περιπτώσεις ο αποστολέας TCP να ωφεληθεί αν είναι πιο συντηρητικός στις ενέργειες του απ'ότι αφήνουν οι παραπάνω αλγόριθμοι αλλά σε καμία περίπτωση δεν πρέπει να είναι πιο επιθετικός απ'όσο ορίζουν οι αλγόριθμοι αυτοί.

4.1 Slow Start και Congestion Avoidance

Οι αλγόριθμοι slow start και congestion avoidance *πρέπει* να χρησιμοποιούνται από τον αποστολέα TCP ώστε να ελέγχουν το ποσό των ανεπιβεβαίωτων δεδομένων τα οποία έχουν στείλει στο δίκτυο. Για να υλοποιηθούν αυτοί οι αλγόριθμοι προστίθενται δύο μεταβλητές στο TCP ανά σύνδεση.

Το παράθυρο συμφόρησης το οποίο είναι ένα όριο από την πλευρά του αποστολέα για το μέγεθος των δεδομένων τα οποία μπορεί να στείλει στο δίκτυο ο αποστολέας πριν πάρει κάποια επιβεβαίωση. Από την άλλη έχουμε το παράθυρο του δέκτη το οποίο είναι ένα όριο από την μεριά του δέκτη για το μέγεθος των δεδομένων τα οποία μπορεί να υπάρχουν ανεπιβεβαίωτα στο δίκτυο, πιο συγκεκριμένα είναι ο χώρος τον οποίο έχει ελεύθερο ο δέκτης στους buffers του για να αποθηκεύσει τα δεδομένα τα οποία θα του στείλει ο αποστολέας. Είναι λογικό λοιπόν ότι το ελάχιστο του παραθύρου συμφόρησης και του παραθύρου του δέκτη είναι αυτό το οποίο καθορίζει και το μέγεθος των ανεπιβεβαίωτων δεδομένων τα οποία μπορούν να υπάρχουν στο δίκτυο. Είναι σαφές ότι αν στείλουμε κι άλλα δεδομένα στο δίκτυο πέρα από το παράθυρο συμφόρησης τότε επιβαρύνουμε το δίκτυο, ενώ αν όλοι οι αποστολείς συμπεριφέρονταν με αυτόν τον τρόπο τότε η συμφόρηση του δικτύου θα οδηγούσε σε συνεχείς συγκρούσεις μεταξύ πακέτων και σε υπερχείλιση των buffers στους δρομολογητές με άμεσο αποτέλεσμα το να μειωθεί δραματικά η απόδοση του δικτύου. Από την άλλη άν ένας αποστολέας στείλει περισσότερα δεδομένα απ'όσα υποδεικνύει το παράθυρο του δέκτη τότε είναι σαφές ότι θα υπερχειλίσουν οι buffers του δέκτη με άμεσο αποτέλεσμα κάποια πακέτα ενώ θα φτάσουν σωστά στον δέκτη αυτός να τα απορρίψει αφού δεν θα έχει χώρο για να αποθηκεύσει τα καινούρια δεδομένα.

Η δεύτερη μεταβλητή η οποία προστίθεται στο TCP ανά σύνδεση είναι το slow start threshold, η μεταβλητή αυτή χρησιμοποιείται για να αποφασίσουμε αν θα χρησιμοποιήσουμε κάποια δεδομένη στιγμή τον slow start αλγόριθμο ή τον congestion avoidance για την αποστολή δεδομένων στο δίκτυο.

Το να ξεκινήσουμε την αποστολή δεδομένων από έναν αποστολέα TCP σε ένα δίκτυο στο οποίο επικρατούν άγνωστες συνθήκες απαιτεί από τον αποστολέα να διερευνήσει σιγά σιγά το δίκτυο για να βρεί την διαθέσιμη χωρητικότητα, έτσι ώστε να αποφύγει την συμφόρηση του δικτύου με το να στείλει ένα ακατάλληλα μεγάλο μέγεθος δεδομένων. Έτσι λοιπόν ο slow start αλγόριθμος χρησιμοποιείται στην αρχή για αυτόν τον σκοπό ή μετά από απώλεια κάποιου πακέτου μετά όταν κάνει timeout ο retransmission timer.

Το αρχικό παράθυρο, η αρχική τιμή του παραθύρου συμφόρησης δηλαδή, πρέπει να είναι μικρότερο η ίσο με δύο πακέτα μέγιστου μεγέθους. Εδώ μπορούμε να αναφέρουμε τελείως

εγκυκλοπαιδικά ότι σε κάποιες πειραματικές μορφές του TCP επιτρέπεται και μεγαλύτερος αριθμός στην θέση του αρχικού παράθυρου, ο οποίος δίνεται από την παρακάτω εξίσωση :

IW = min (4*SMSS, max (2*SMSS, 4380 bytes))

με αυτήν την προσθήκη ο αποστολέας TCP μπορεί να χρησιμοποιήσει αρχικό παράθυρο τριών και τεσσάρων πακέτων, αρκεί πάντα το συνολικό μέγεθος των δεδομένων τα οποία αποστέλλει στο δίκτυο να μην υπερβαίνουν τα 4380 bytes. Αυτή βέβαια η αλλαγή δεν αποτελεί επίσημα μέρος του πρωτοκόλλου TCP/IP απλά το αναφέρουμε για να δείξουμε και κάποιες ερευνητικές προσπάθειες βελτίωσης της απόδοσης του πρωτοκόλλου που γίνονται προς αυτήν την κατεύθυνση.

Η αρχική τιμή τώρα του slow start threshold μπορεί να είναι αυθαίρετα υψηλή, (για παράδειγμα κάποιες υλοποιήσεις χρησιμοποιούν την τιμή του παραθύρου η οποία προβάλλεται στο δίκτυο) αλλά αυτό μπορεί να μειωθεί σε περίπτωση που ανιχνευθεί συμφόρηση στο δίκτυο. Ο αλγόριθμος slow start χρησιμοποιείται όταν η τιμή του παραθύρου συμφόρησης είναι μικρότερη από αυτήν του slow start threshold, ενώ ο congestion avoidance όταν είναι μεγαλύτερη. Όταν οι δύο αυτές τιμές είναι ίσες τότε ο αποστολέας μπορεί να χρησιμοποιήσει όποιον από τους δύο αλγόριθμους επιθυμεί.

Κατά την διάρκεια του slow start το TCP αυξάνει το παράθυρο συμφόρησης κατά το πολύ ένα πακέτο μέγιστου μεγέθους για κάθε επιβεβαίωση που λαμβάνει η οποία επιβεβαιώνει καινούρια δεδομένα. Ο slow start σταματάει όταν η τιμή του παραθύρου συμφόρησης γίνει μεγαλύτερη από την τιμή του slow start threshold ή αν παρουσιαστεί συμφόρηση στο δίκτυο.

Κατά την διάρκεια του congestion avoidance, το παράθυρο συμφόρησης αυξάνεται κατά ένα πακέτο μέγιστου μεγέθους κατά την χρονική διάρκεια την οποία χρειάζεται ένα πακέτο να φτάσει στον παραλήπτη του και να επιστρέψει η επιβεβαίωση στον αποστολέα. Ο congestion avoidance συνεχίζει καθ'όλη την διάρκεια της μετάδοσης έως ότου παρατηρηθεί συμφόρηση στο δίκτυο. Ένας τύπος ο οποίος χρησιμοποιείται συνήθως για να υπολογίζεται το μέγεθος του παραθύρου συμφόρησης κατά την διάρκεια του congestion avoidance βρίσκεται στην παρακάτω εξίσωση :

CWND += SMSS*SMSS/CWND

Αυτή η διόρθωση της τιμής του παραθύρου συμφόρησης γίνεται σε κάθε νέα επιβεβαίωση την οποία λαμβάνει το TCP, η οποία όμως επιβεβαιώνει καινούρια δεδομένα. Η

παραπάνω εξίσωση λοιπόν παρέχει μια αποδεκτή προσέγγιση στην αρχή που διατυπώσαμε προ ολίγου ότι δηλαδή κατά την διάρκεια του congestion avoidance, το παράθυρο συμφόρησης αυξάνεται κατά ένα πακέτο μέγιστου μεγέθους κατά την χρονική διάρκεια την οποία χρειάζεται ένα πακέτο να φτάσει στον παραλήπτη του και να επιστρέψει η επιβεβαίωση στον αποστολέα.

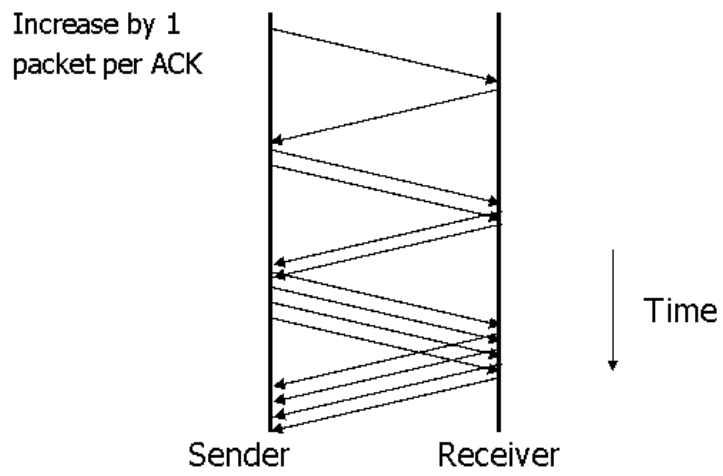
Διαισθητικά καταλαβαίνουμε όμως ότι και ο congestion avoidance δεν μπορεί να αυξάνει το παράθυρο συμφόρησης επ'άπειρον. Όπως είναι λογικό λοιπόν αν οι συνθήκες του δικτύου είναι καλές υπάρχει η περίπτωση να γίνει η αποστολή όλων τον δεδομένων χωρίς να παρατηρηθεί συμφόρηση οπότε και δεν υπάρχει κάποιο πρόβλημα, η πιο συνήθης περίπτωση όμως είναι κάποια στιγμή να παρατηρηθεί συμφόρηση στο δίκτυο την οποία ο αποστολέας την αντιλαμβάνεται μετά από την απώλεια κάποιου πακέτου. Στην επόμενη παράγραφο θα περιγράψουμε τις ενέργειες τις οποίες κάνει το TCP σε περίπτωση που χαθεί κάποιο πακέτο, ένδειξη του ότι υπάρχει συμφόρηση στο δίκτυο.

Έστω λοιπόν ότι βρισκόμαστε κατά την διάρκεια κάποιας μετάδοσης και ο retransmission κάνει timeout, οπότε και αντιλαμβανόμαστε ότι έχει χαθεί κάποιο από τα πακέτα τα οποία έχουμε ήδη στείλει προς τον παραλήπτη. Σε μια τέτοια περίπτωση είτε η μετάδοση ελέγχετε από τον slow start είτε από τον congestion avoidance αλγόριθμο η αντίδραση του TCP είναι η ίδια. Σε μια τέτοια περίπτωση λοιπόν το TCP μειώνει την τιμή της μεταβλητής slow start threshold στο μισό της τιμής του μεγέθους των δεδομένων τα οποία μπορούν να βρίσκονται την συγκεκριμένη στιγμή ανεπιβεβαίωτα στο δίκτυο η οποία όπως αναφέραμε και παραπάνω είναι κάθε στιγμή ο μικρότερος αριθμός εκ των παραθύρου συμφόρησης και του παραθύρου του δέκτη, ή σε δύο πακέτα μεγίστου μεγέθους ανάλογα με το πια τιμή είναι μεγαλύτερη από τις δύο (παίρνει δηλαδή την μεγαλύτερη από τις δύο τιμές και όχι την μικρότερη), δίνει στο παράθυρο συμφόρησης την τιμή του παραθύρου απώλειας η οποία είναι ένα πακέτο μέγιστου μεγέθους άσχετα με το ποια είναι η τιμή του αρχικού παραθύρου και η μετάδοση συνεχίζεται από το πακέτο το οποίο χάθηκε με τον αλγόριθμο slow start να ελέγχει πάλι την μετάδοση.

Εδώ πρέπει να αναφέρουμε πώς και οι δύο αυτοί αλγόριθμοι είναι υλοποιημένοι και υποστηρίζονται πλήρος από την σχεδίασή μας. Παραπάνω λεπτομέρειες θα δούμε στο επόμενο κεφάλαιο όπου και θα αναλύσουμε την αρχιτεκτονική και την λειτουργία της σχεδίασής μας.

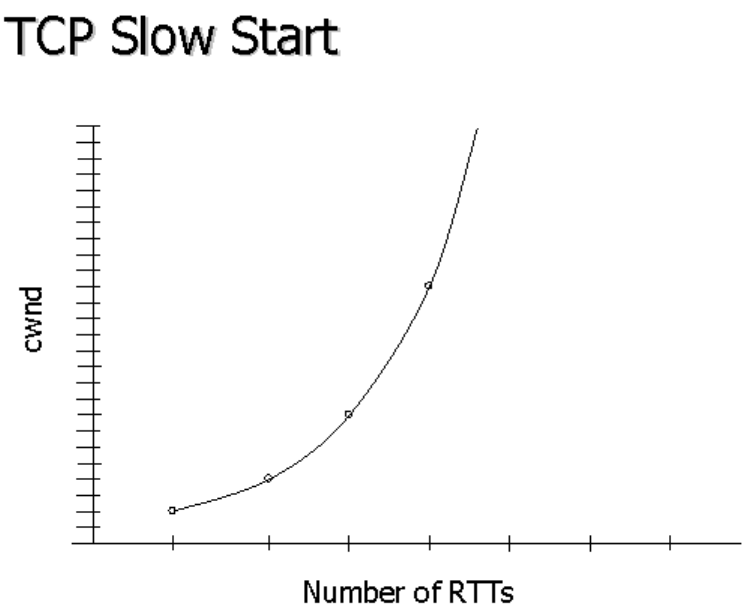
Παρακάτω θα παραθέσουμε κάποια διαγράμματα τα οποία δείχνουν την συμπεριφορά των παραπάνω αλγορίθμων που περιγράψαμε, τα οποία και θα σχολιάσουμε για καλύτερη κατανόηση από την πλευρά του αναγνώστη.

4.1.1 Παραδείγματα Slow Start



Σχήμα 4.1

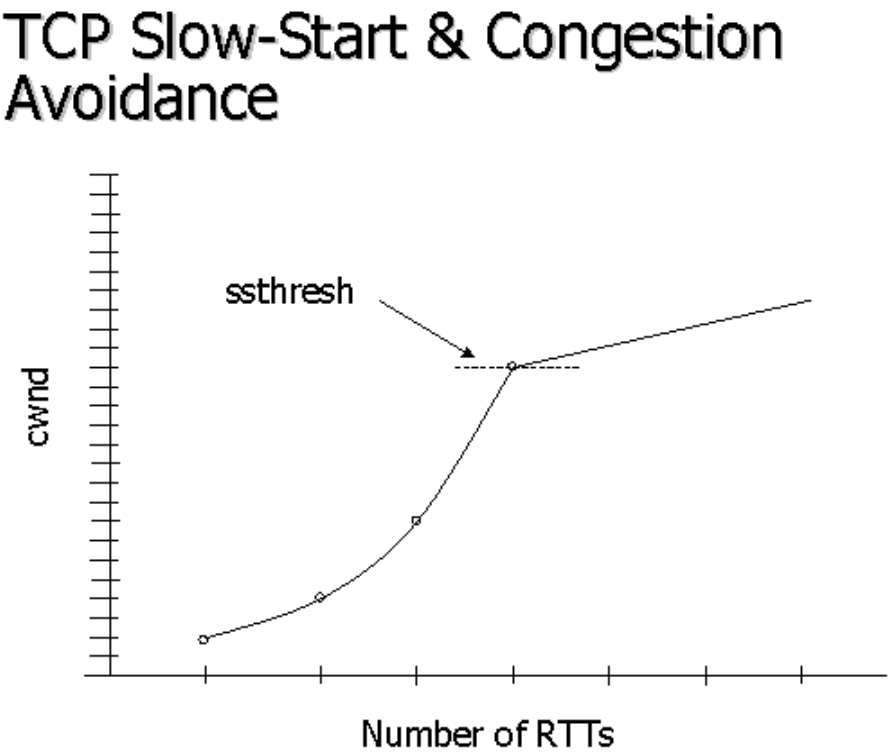
Παραπάνω στο *σχήμα 4.1* παρατηρούμε το πώς αυξάνεται ο αριθμός των πακέτων τα οποία στέλνει ο αποστολέας στο δίκτυο με την πάροδο του χρόνου σε σχέση με τις επιβεβαιώσεις για καινούρια δεδομένα τις οποίες δέχεται από τον δέκτη. Βλέπουμε λοιπόν ότι ο αποστολέας στην αρχή στέλνει ένα πακέτο, μόλις πάρει πίσω επιβεβαίωση γι'αυτό στέλνει δύο πακέτα, μόλις επιβεβαιωθούν και αυτά τέσσερα. Παρατηρούμε λοιπόν ότι κατά την διάρκεια του slow start ο αριθμός των πακέτων τα οποία αφήνουν τον αποστολέα προς το δίκτυο αυξάνεται εκθετικά δηλαδή 1,2,4,8,16… έως ότου βέβαια την τιμή του slow start threshold όπου και αναλαμβάνει την μετάδοση ο congestion avoidance.



Σχήμα 4.2

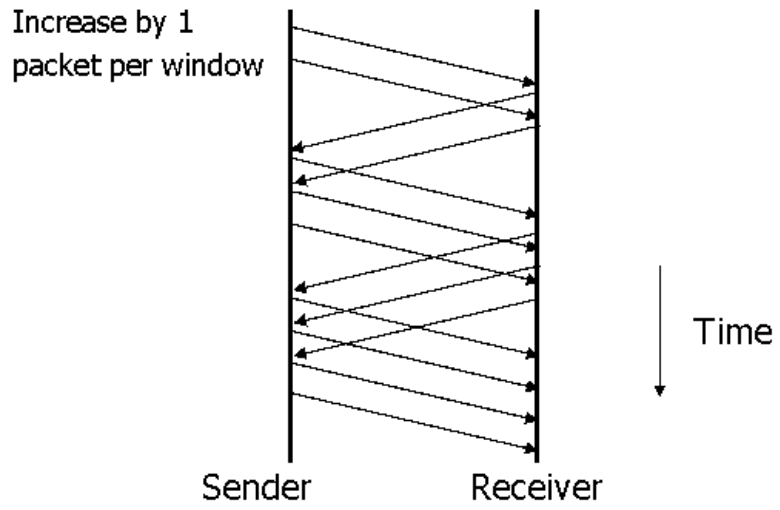
Παραπάνω στο *Σχήμα 4.2* βλέπουμε το πώς αυξάνεται το παράθυρο συμφόρησης σε σχέση με τον χρόνο που χρειάζεται ένα πακέτο να πάει από τον αποστολέα στον δέκτη και να γυρίσει πίσω η επιβεβαίωσή του. Σ'αυτό εδώ το σχήμα μπορούμε να δούμε και καλύτερα την εκθετική αυτή αύξηση του αριθμού των πακέτων τα οποία ο αποστολέας μπορεί να στείλει στο δίκτυο.

4.1.2 Παραδείγματα Congestion Avoidance



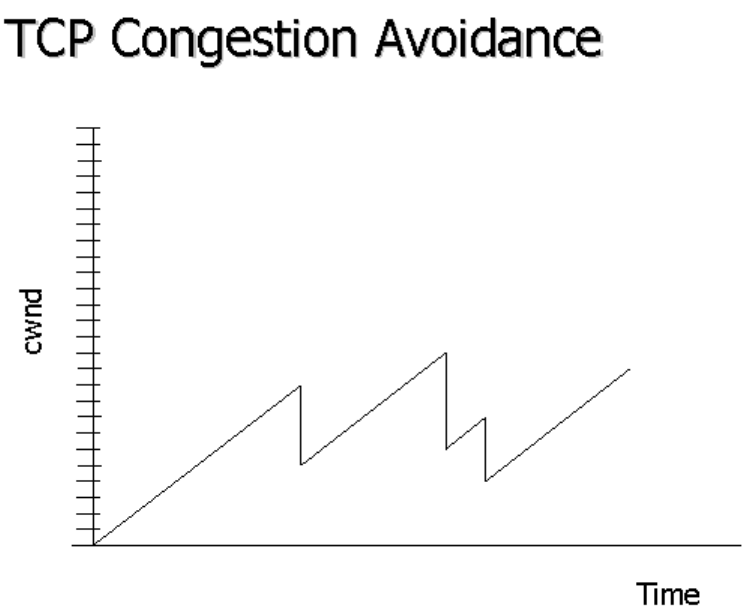
Σχήμα 4.3

Παραπάνω στο *Σχήμα 4.3* βλέπουμε σχηματικά πώς αυξάνεται το παράθυρο συμφόρησης σε σχέση με τον χρόνο που χρειάζονται ένα πακέτο και μια επιβεβαίωση να διασχίσουν το δίκτυο. Βλέπουμε λοιπόν ότι στην αρχή της μετάδοσης ο αλγόριθμος slow start είναι αυτός που ελέγχει την μετάδοση και παρατηρούμε και την εκθετική αύξηση του αριθμού των πακέτων έως την τιμή ssthresh (slow start threshold), όπου και αναλαμβάνει ο αλγόριθμος congestion avoidance. Κατά την διάρκεια του congestion avoidance βλέπουμε πώς ο αριθμός των πακέτων αυξάνεται γραμμικά πλέον, για κάθε παράθυρο δεδομένων που στέλνουμε στο δίκτυο δηλαδή αν έρθουν σωστά όλες οι επιβεβαιώσεις για αυτά τα δεδομένα τότε το παράθυρο συμφόρησης αυξάνει την τιμή του κατά ένα πακέτο μέγιστου μεγέθους.



Σχήμα 4.4

Στο *σχήμα 4.4* βλέπουμε πιο αναλυτικά πώς συμπεριφέρεται ο αλγόριθμος congestion avoidance σε σχέση με τις επιβεβαιώσεις οι οποίες έρχονται στον αποστολέα με την πάροδο του χρόνου. Πιο συγκεκριμένα βλέπουμε πως αρχικά ο αποστολέας στέλνει ένα παράθυρο δύο πακέτων, στην συνέχεια λαμβάνει τις επιβεβαιώσεις για τα πακέτα αυτά οπότε και αυξάνει το παράθυρο συμφόρησής του κατά ένα και στέλνει τα επόμενα τρία πακέτα. Ομοίως και στην αποστολή του επόμενου παραθύρου αυξάνει πάλι τον αριθμό κατά ένα και στέλνει τα επόμενα τέσσερα πακέτα. Βλέπουμε λοιπόν ότι ο αλγόριθμος congestion avoidance αυξάνει την τιμή του παραθύρου συμφόρησης κατά ένα πακέτο μέγιστου μεγέθους μετά από κάθε αποστολή και σωστή λήψη των επιβεβαιώσεων ενός παραθύρου, ανεξάρτητα με το ποια είναι η τιμή του παραθύρου αυτού, αντίθετα με τον slow start ο οποίος διπλασιάζει κάθε φορά την τιμή του εκάστοτε παραθύρου.



Σχήμα 4.5

Στο *σχήμα 4.5* βλέπουμε πως ρυθμίζει ο αλγόριθμος congestion avoidance το παράθυρο συμφόρησης κατά την διάρκεια του χρόνου. Αυτό που βλέπουμε πρώτη φορά σε αυτό το σχήμα είναι την παρουσία κάποιων απωλειών οι οποίες μας εξαναγκάζουν σε ρυθμίσεις της μεταβλητής slow start threshold. Από το *σχήμα 4.5* λείπει το κομμάτι αυτό του αλγορίθμου slow start ο οποίος και αναλαμβάνει τον έλεγχο της μετάδοσης μετά από την απώλεια κάποιου πακέτου. Πάραυτα στο παραπάνω σχήμα μπορούμε να δούμε την τιμή της μεταβλητής slow start threshold να παίρνει την τιμή του μισού του παραθύρου συμφόρησης την χρονική στιγμή της απώλειας και τον αλγόριθμο congestion avoidance να αναλαμβάνει από αυτά τα σημεία τον έλεγχο της μετάδοσης μέχρι την επόμενη απώλεια πάλι.

4.2 Fast Retransmit και Fast Recovery

Ο δέκτης TCP θα πρέπει να στέλνει άμεσα μια διπλότυπη επιβεβαίωση κάθε φορά που λαμβάνει ένα πακέτο εκτός σειράς. Ο σκοπός αυτής της διπλότυπης επιβεβαίωσης είναι να ειδοποιήσει τον αποστολέα ότι κάποιο πακέτο έχει ληφθεί εκτός σειράς και παράλληλα να του πει και πιο sequence number είναι αυτό το οποίο περιμένει. Από την πλευρά του αποστολέα, διπλότυπες επιβεβαιώσεις μπορεί να προκληθούν από ένα μεγάλο αριθμό προβλημάτων του δικτύου.

Κατ'αρχήν μπορεί να προκληθούν από πακέτα τα οποία έχουν απορριφθεί, σε αυτήν την περίπτωση όλα τα πακέτα τα οποία έφυγαν μετά το πακέτο το οποίο απορρίφθηκε θα προκαλέσουν την δημιουργία διπλότυπων επιβεβαιώσεων. Διπλότυπες επιβεβαιώσεις μπορούν επίσης να προκληθούν από επαναδιάταξη των πακέτων από το δίκτυο (πράγμα όχι και τόσο σπάνιο σε κάποιες διαδρομές του δικτύου). Και τέλος διπλότυπες επιβεβαιώσεις μπορεί να προκληθούν από δημιουργία πανομοιότυπων πακέτων δεδομένων ή και επιβεβαιώσεων από το δίκτυο. Επιπροσθέτως ο δέκτης TCP θα πρέπει να στέλνει άμεσα επιβεβαίωση στον αποστολέα όταν κάποιο εισερχόμενο πακέτο συμπληρώσει ολόκληρο ή ένα μέρος του κενού το οποίο έχει δημιουργηθεί στον συρμό των πακέτων. Αυτή η ενέργεια θα δώσει πιο γρήγορα πληροφορίες στον αποστολέα ο οποίος προσπαθεί να ανακάμψει μετά από μια απώλεια, απ'ότι το να περιμένουμε να κάνει timeout ο retransmission timer.

Ο αποστολέας TCP θα έπρεπε να χρησιμοποιεί τον αλγόριθμο fast retransmit για να εντοπίσει και να αποκαταστήσει την απώλεια κάποιου ή κάποιων πακέτων, βασιζόμενος στις εισερχόμενες διπλότυπες επιβεβαιώσεις. Ο αλγόριθμος fast retransmit χρησιμοποιεί την άφιξη τριών διπλότυπων επιβεβαιώσεων (τέσσερις ίδιες επιβεβαιώσεις χωρίς την άφιξη κάποιων άλλων παρεμβαλλόμενων πακέτων) ως ένδειξη απώλειας κάποιου πακέτου. Αφού λοιπόν ο αποστολέας λάβει τρεις διπλότυπες επιβεβαιώσεις προχωρά σε επαναμετάδοση του πακέτου το οποίο φαίνεται να έχει χαθεί χωρίς να περιμένει τον retransmission timer να κάνει timeout.

Από την στιγμή λοιπόν που ο αλγόριθμος fast retransmit στέλνει το πακέτο το οποίο φαίνεται να έχει χαθεί, τον έλεγχο της μετάδοσης αναλαμβάνει ο αλγόριθμος fast recovery έως ότου σταματήσουν αν έρχονται πλέον διπλότυπες επιβεβαιώσεις. Ο λόγος για τον οποίο δεν προτείνεται η χρησιμοποίηση του αλγορίθμου slow start είναι για τον εξής λόγο, δεδομένου ότι για να προκληθεί κάποια επιβεβαίωση έστω και διπλότυπη από την μεριά του δέκτη αυτό σημαίνει ότι κάποιο από τα πακέτα τα οποία έστειλε ο αποστολέας έφτασε εκεί σωστά αλλά εκτός σειράς, οπότε και το συγκεκριμένο πακέτο δεν καταναλώνει πλέον πόρους από το δίκτυο,

έτσι λοιπόν το TCP μπορεί να χρησιμοποιήσει προς όφελός του τον ελεύθερο χώρο ο οποίο; ξέρει ότι υπάρχει στο δίκτυο για να επανακάμψει πιο γρήγορα από την απώλεια.

Οι αλγόριθμοι fast retransmit και fast recovery υλοποιούνται συνήθως μαζί σύμφωνα με τον τρόπο που ακολουθεί.

1. Όταν έρθει η τρίτη διπλότυπη επιβεβαίωση, δίνουμε στην μεταβλητή ssthresh την τιμή

   ssthresh = max (Flightsize/2, 2*SMMS)

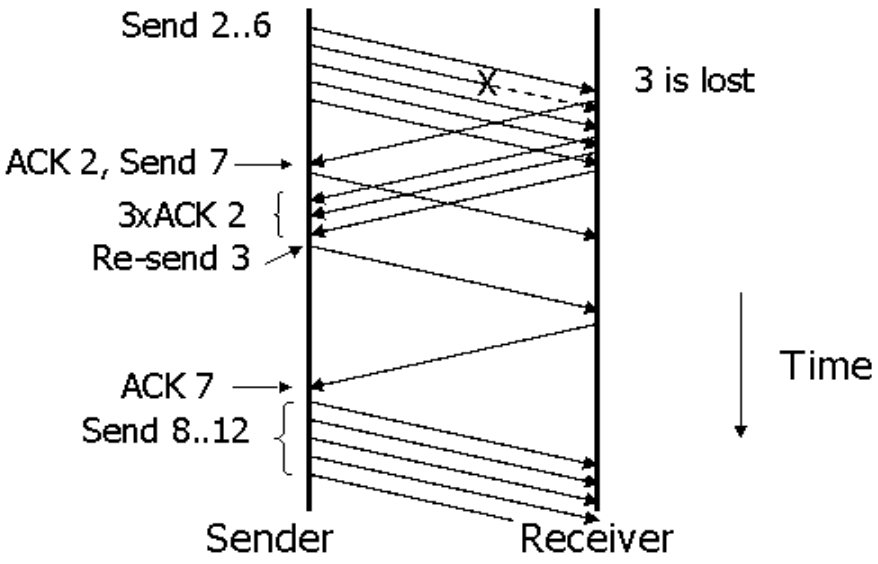
   όπου flightsize είναι το μέγεθος του παραθύρου το οποίο χρησιμοποιούμε για αποστολή εκείνη την στιγμή και όχι την τιμή του παραθύρου συμφόρησης καθώς αυτή μπορεί να πάρει τιμές μεγαλύτερες από την τιμή του παραθύρου του δέκτη.
2. Μεταδίδουμε το χαμένο πακέτο και θέτουμε την τιμή του παραθύρου συμφόρησης στην τιμή ssthresh + 3 SMSS. Αυτή η ενέργεια διογκώνει τεχνητά το παράθυρο συμφόρησης με τον αριθμό των πακέτων τα οποία έχουν φύγει ήδη από το δίκτυο μιας και σ'αυτά οφείλονται οι τρεις διπλότυπες επιβεβαιώσεις τις οποίες λάβαμε, και κατά πάσα πιθανότητα τα έχει κιόλας αποθηκεύσει ο δέκτης.
3. Για κάθε επιπλέον διπλότυπη επιβεβαίωση την οποία λαμβάνουμε αυξάνουμε και την τιμή του παραθύρου συμφόρησης κατά ένα πακέτο μέγιστου μεγέθους, χρησιμοποιώντας την ίδια λογική την οποία περιγράψαμε παραπάνω.
4. Μεταδίδουμε ένα ακόμα πακέτο άν αυτό μας επιτρέπεται από την τιμή του παραθύρου συμφόρησης ή την αντίστοιχη τιμή του παραθύρου του δέκτη.
5. Όταν έρθει κάποια επιβεβαίωση η οποία επιβεβαιώνει καινούρια δεδομένα τότε μειώνουμε την τιμή του παραθύρου συμφόρησης στην τιμή της μεταβλητής ssthresh "ξεφουσκώνοντας" έτσι κατά κάποιο τρόπο το παράθυρο συμφόρησης.

Τα παραπάνω είναι τα βήματα τα οποία υλοποιούν τους δύο αλγορίθμους. Εδώ πρέπει να αναφέρουμε ότι από την δικιά μας σχεδίαση υποστηρίζεται μόνο το κομμάτι του fast retransmit και αυτό ως ένας πιο γρήγορος τρόπος να αντιληφθούμε μια απώλεια. Το κομμάτι του fast recovery δεν έχει υλοποιηθεί και έτσι σε μια τέτοια περίπτωση ο αλγόριθμος slow start αναλαμβάνει πάλι τον έλεγχο της μετάδοσης. Περισσότερες λεπτομέρειες για το θέμα αυτό θα αναφέρουμε στο επόμενο κεφάλαιο όπου και θα αναλύσουμε την αρχιτεκτονική και την λειτουργία της σχεδίασής μας. Επίσης πρέπει να πούμε πως γενικότερα ο fast retransmit δεν δουλεύει καλά για πολλαπλές απώλειες πακέτων μέσα στο ίδιο παράθυρο δεδομένων. Επίπλέον πρέπει να αναφέρουμε ότι οι αλγόριθμοι fast retransmit και fast recovery δεν είναι εκ των ων ούκ άνευ στον έλεγχο συμφόρησης του πρωτοκόλλου TCP και αυτό σημαίνει ότι μπορεί να

υλοποιηθεί κάποιο TCP και χωρίς να έχει ενσωματωμένη την λειτουργικότητα των δύο αυτών αλγορίθμων και να είναι πάλι συμβατό με το πρότυπο. Γενικά όμως καλό είναι να υπάρχουν αφού δίνουν ταχύτητα και κάνουν πιο ευέλικτο το ίδιο το πρωτόκολλο.

Παρακάτω θα δούμε πάλι ένα παράδειγμα με τον αλγόριθμο fast retransmit για να καταλάβουμε καλύτερα την λειτουργία του.

4.2.1 Παραδείγματα Fast Retransmit



Σχήμα 4.6

Στο *σχήμα 4.6* βλέπουμε ένα παράδειγμα μετάδοσης δεδομένων μέσω του πρωτοκόλλου TCP όπου και ενεργοποιείται ο αλγόριθμος fast retransmit. Ο αποστολέας TCP λοιπόν στέλνει ένα παράθυρο δεδομένων τα πακέτα 2 έως 6. Στην διαδρομή για κάποιο λόγο χάνεται το πακέτο

νούμερο 3. Ο δέκτης λαμβάνει σωστά το πακέτο 2 και το επιβεβαιώνει λαμβάνει όμως μετά τα πακέτα 4, 5 και 6 εκτός σειράς οπότε για κάθε πακέτο εκτός σειράς που λαμβάνει επιβεβαιώνει πάλι το τελευταίο σωστό πακέτο το οποίο έλαβε και αυτό είναι το 2. Ο αποστολέας TCP τώρα έχει στείλει ήδη το πακέτο νούμερο 7 όμως λαμβάνει τις τρεις διπλότυπες επιβεβαιώσεις οπότε καταλαβαίνει ότι έχει χαθεί το πακέτο νούμερο 3 και ενεργοποιείται έτσι ο αλγόριθμος fast retransmit ο οποίος και το ξαναστέλνει αμέσως. Ο δέκτης μόλις λάβει το πακέτο νούμερο 3 μιας και πλέον έχει λάβει σωστά όλα τα πακέτα μέχρι και το νούμερο 7 επιβεβαιώνει το 7 (όταν το TCP επιβεβαιώνει ένα νούμερο πακέτου σημαίνει ότι έχει λάβει όλα τα δεδομένα μέχρι εκείνο το σημείο σωστά) οπότε και η μετάδοση ξανασυνεχίζεται με το καινούριο παράθυρο νέων δεδομένων.

4.3 Επανεκινώντας Ανενεργές Συνδέσεις

Ένα γνωστό πρόβλημα του έλεγχου συμφόρησης του TCP έτσι όπως περιγράφηκε παραπάνω είναι ότι μπορεί να αφήσει ένα πολύ μεγάλο όγκο δεδομένων να εισαχθεί στο δίκτυο έπειτα από κάποια χρονική διάρκεια κατά την οποία η σύνδεση παρέμεινε ανενεργή με αποτέλεσμα την πιθανή δημιουργία προβλήματος συμφόρησης στο δίκτυο δεδομένου ότι έχουν αλλάξει πλέον τα δεδομένα του δικτύου το οποίο να μην μπορεί πλέον να υποστηρίξει έναν τόσο μεγάλο όγκο δεδομένων αν και πριν μπορούσε. Έτσι λοιπόν αυτό που γίνεται στην συγκεκριμένη περίπτωση είναι αν το TCP δεν έχει στείλει δεδομένα για χρόνο ίσο με τον χρόνο που χρειάζεται ένα πακέτο να φτάσει στον παραλήπτη του και να επιστρέψει η επιβεβαίωσή του, τότε ο αλγόριθμος slow start αναλαμβάνει τον έλεγχο της μετάδοσης, αφού πρώτα έχουμε μειώσει την τιμή του παραθύρου συμφόρησης και την έχουμε κάνει ίδια με την τιμή του παραθύρου επανεκκίνησης. Πιο συγκεκριμένα η τιμή του παραθύρου επανεκκίνησης για το συγκεκριμένο πρότυπο ορίζεται ίσο με το αρχικό παράθυρο της σύνδεσης.

Στην σχεδίασή μας αν και έχουμε λάβει υπ'όψιν μας αυτήν την ιδιαιτερότητα του TCP και έχουμε σχεδιάσει έτσι τους πόρους που χρησιμοποιούμε ώστε να την υποστηρίζουν δεν έχει γίνει υλοποίηση του κατάλληλου μηχανισμού ο οποίος θα τους ενεργοποιεί. Πάραυτα έχουμε λύσει το συγκεκριμένο πρόβλημα με άλλον τρόπο ο οποίος παραμένει όμως πλήρως συμβατός με τις απαιτήσεις του πρωτοκόλλου TCP/IP. Περισσότερες λεπτομέρειες για το θέμα αυτό θα αναφέρουμε στο επόμενο κεφάλαιο όπου και θα αναλύσουμε την αρχιτεκτονική και την λειτουργία της σχεδίασής μας.

4.4 Δημιουργία Επιβεβαιώσεων

Το TCP δίνει την δυνατότητα στον δέκτη να καθυστερήσει κάποια επιβεβαίωση και ο λόγος είναι μήπως και έρθουν και άλλα πακέτα από την ίδια ροή και τα επιβεβαιώσει όλα με μία επιβεβαίωση. Πάραυτα υπάρχουν και κάποιοι περιορισμοί σε αυτήν την καθυστέρηση καθώς αυτή δεν πρέπει να υπερβαίνει τα 500ms και πρέπει να δημιουργείται μία επιβεβαίωση κάθε δεύτερο πακέτο που λαμβάνει ο δέκτης.

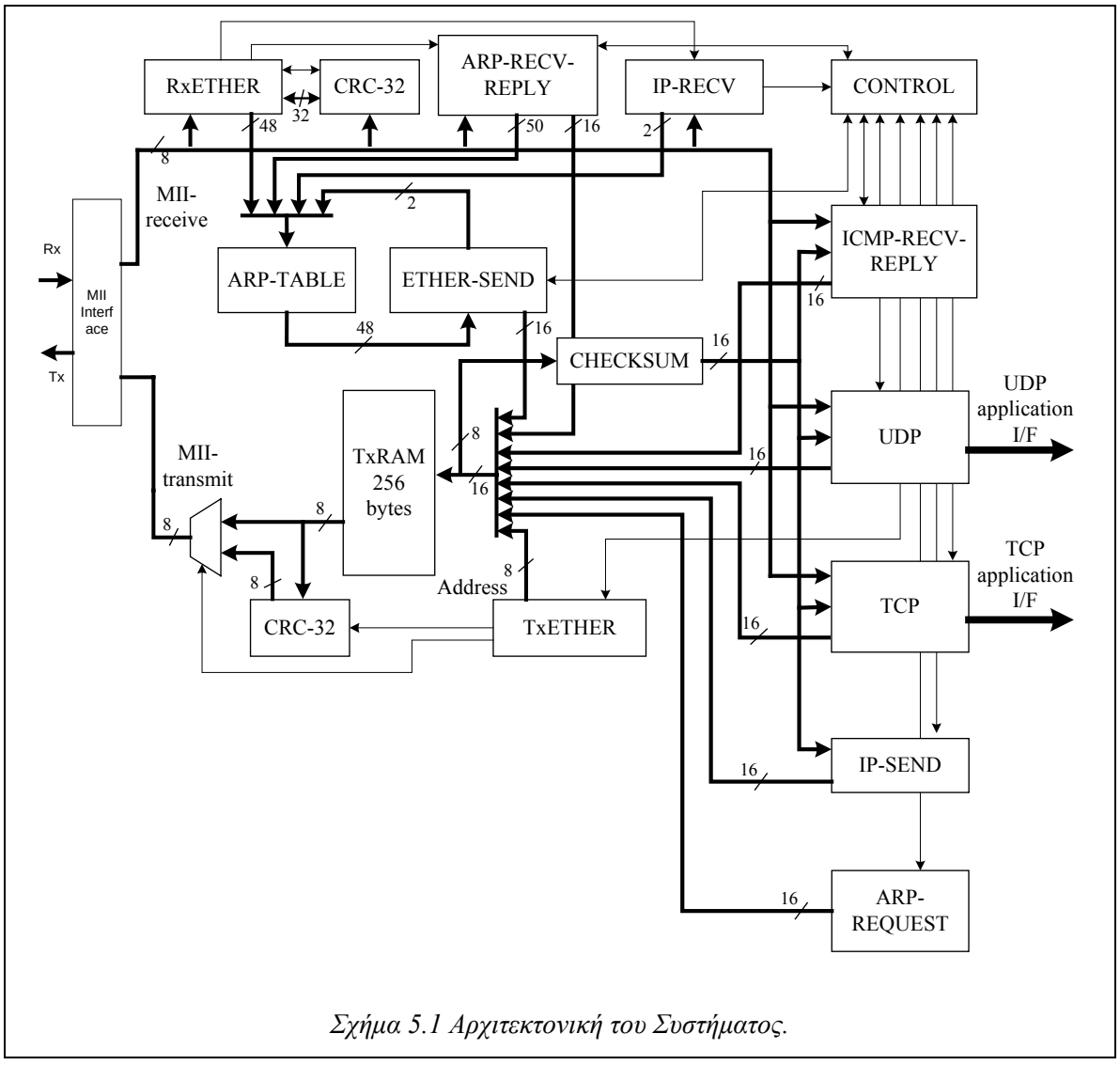
Πακέτα τα οποία λαμβάνονται εκτός σειράς πρέπει να επιβεβαιώνονται αμέσως με διπλότυπες επιβεβαιώσεις έτσι ώστε να επιταχυνθεί η διαδικασία ανάκαμψης από την απώλεια με την ενεργοποίηση του αλγορίθμου fast retransmit. Επίσης αμέσως πρέπει να στέλνονται και οι επιβεβαιώσεις οι οποίες επιβεβαιώνουν κάποιο πακέτο το οποίο είχε χαθεί και είχε δημιουργήσει έτσι κενό στην ροή των δεδομένων.

Τέλος ο δέκτης δεν πρέπει να στέλνει χωρίς λόγο διπλότυπες επιβεβαιώσεις εκτός και αν πρέπει να ανανεώσει την τιμή του παραθύρου του που έχει κοινοποιήσει στον αποστολέα καθώς έχει δημιουργήσει κενό χώρο καταναλώνοντας δεδομένα προς το επίπεδο της εφαρμογής.

Κεφάλαιο 5ο: Σχεδίαση και υλοποίηση της τρίτης γενιάς της σχεδίασης

Σε αυτό το κεφάλαιο θα παρουσιάσουμε και θα αναλύσουμε εις βάθος την τρίτη γενιά της σχεδίασης που έγινε στα πλαίσια της διπλωματικής εργασίας. Θα μιλήσουμε μόνο για τα υποσυστήματα αυτά τα οποία άλλαξαν ή προστέθηκαν σε σχέση με την δεύτερη γενιά αφού θεωρούμε πώς όσα έμειναν ίδια καλύπτονται αρκετά καλά από την διπλωματική εργασία του κ. Γιάννη Ζήση [16] όπου και μπορεί να ανατρέξει κάποιος για περισσότερες λεπτομέρειες απ'όσες ενδεικτικά αναφέραμε εμείς στο παρόν κείμενο και συγκεκριμένα στο τρίτο κεφάλαιο αυτής της διπλωματικής εργασίας έτσι ώστε να περιγράψουμε εν τάχη την δεύτερη γενιά της σχεδίαση και να κάνουμε και μια μικρή εισαγωγή στις αλλαγές που έχουν γίνει πλέον.

Παρακάτω στο *σχήμα 5.1* βλέπουμε την αρχιτεκτονική της σχεδίασης στην παρούσα της μορφή. Με μια πρώτη ματιά βλέπουμε πως αν ανατρέξουμε στο τρίτο κεφάλαιο αυτής της διπλωματικής εργασίας και κάνουμε την σύγκριση με την δεύτερη γενιά εκ πρώτης όψεως η μόνη αλλαγή η οποία έχει γίνει είναι η προσθήκη του υποσυστήματος MII Interface, κατά τα άλλα η σχεδίαση συνεχίζει να βασίζεται στην ίδια αρχιτεκτονική με τις ίδιες διασυνδέσεις μεταξύ των υποσυστημάτων. Η συγκεκριμένη παρατήρηση είναι σωστή και απολύτως δικαιολογημένη, πρέπει όμως να θυμηθούμε ότι χρησιμοποιήσαμε αυτούσια την αρχιτεκτονική η οποία είχε προταθεί για την σχεδίαση από την διπλωματική εργασία του κυρίου Γιάννη Ζήση [16], με την διαφορά ότι αλλάξαμε σχεδόν άρδην την λειτουργικότητα όσων υποσυστημάτων χρειαζόταν έτσι ώστε να ενσωματώσουμε στην υπάρχουσα σχεδίαση τον έλεγχο συμφόρησης του δικτύου, που υποστηρίζει το πρωτόκολλο TCP/IP. Έτσι λοιπόν έχουμε μπροστά μας μια σχεδόν απαράλλακτη "εμφανισιακά" σχεδίαση, λειτουργικά όμως έχουν αλλάξει πάρα πολλά πράγματα. Σε αυτό το κεφάλαιο λοιπόν όπως προείπαμε θα αναλύσουμε και θα δείξουμε την λειτουργία όσων υποσυστημάτων προστέθηκαν και όσων άλλαξαν στην τρίτη γενιά της σχεδίασης.



Σχήμα 5.1 Αρχιτεκτονική του Συστήματος.

5.1 Η διεπαφή MII, MIIinterface

Στο τρίτο κεφάλαιο του κειμένου αυτού κάναμε μια πρώτη εισαγωγή για την διεπαφή MII Interface την οποία προσθέσαμε στην σχεδίαση. Το υποσύστημα MII Interface λοιπόν προστέθηκε στην τρίτη γενιά της σχεδίασης και ο λόγος είναι ότι χρειαζόταν ένας τρόπος να συνδεθεί η σχεδιασή μας, με τις εισόδους τις οποίες είχε με κάποιον Ethernet transceiver, ο οποίος αναλαμβάνει και τον ρόλο της σύνδεσής της σχεδίασής μας με το δίκτυο, με τις επαφές τις οποίες αυτός παρέχει.

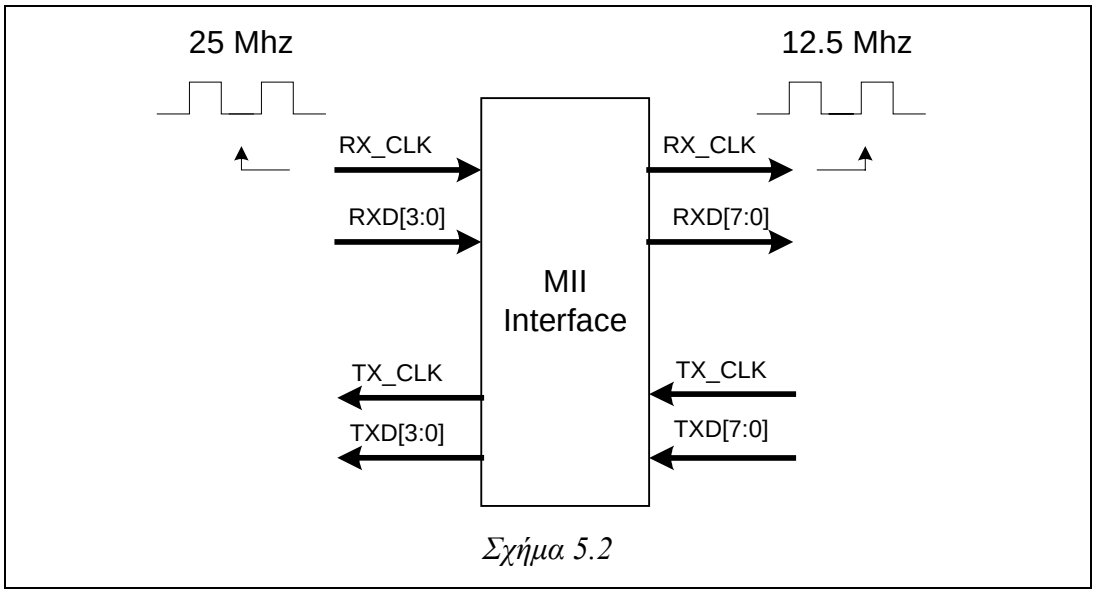
Ο Ethernet transceiver λοιπόν παρέχει την σύνδεση της σχεδίασής μας με το φυσικό δίκτυο. Πιο συγκεκριμένα λοιπόν παρέχει δυνατότητα σύνδεσης σε δίκτυο είτε 10 είτε 100 Mbps, δυνατότητες μετάδοσης είτε με half duplex είτε με full duplex τρόπο, πράγμα το οποίο σημαίνει ότι μπορεί να συνδεθεί σε οποιοδήποτε δίκτυο το οποίο χαρακτηρίζεται από τα παραπάνω δεδομένα. Επίσης παρέχει autonegotiation αλγορίθμους χάριν στους οποίους μπορεί να διαπραγματεύεται μόνο του με το δίκτυο τον τρόπο λειτουργίας τον οποίο το κάθε δίκτυο υποστηρίζει καθώς και την δυνατότητα επιβολής κάποιου συγκεκριμένου τρόπου λειτουργίας αν εμείς το επιθυμούμε. Η σημαντικότερη λειτουργία του όμως είναι το γεγονός ότι κωδικοποιεί κατά Manchester τα δεδομένα έτσι ώστε να μεταδοθούν στην συνέχεια μέσω του φυσικού μέσου,

Τα σήματα τα οποία μας παρέχει ο Ethernet transceiver τα αναφέραμε και τα αναλύσαμε στο τρίτο κεφάλαιο της διπλωματικής μας τα αναφέρουμε πάλι εν τάχη και θα προχωρήσουμε στην επεξήγηση της χρησιμότητας αυτού του υποσυστήματος και του έργου το οποίο επιτελεί. Τα σήματα λοιπόν αυτά είναι :

- TX_CLK
- TXD[3:0]
- TX_CLK.
- TX_EN
- TX_ER
- COL
- RX_CLK
- RXD[3:0]
- CRS
- RX_DV
- RX_ER

Η χρησιμότητα και το έργο το οποίο επιτελεί το υποσύστημα MII Interface φαίνεται στο *σχήμα 5.2* το οποίο ακολουθεί. Οι Ethernet Transceivers λοιπόν για να λειτουργήσουν σε δίκτυο 100Mbps πρέπει να τους δώσουμε ρολόι 25 Mhz. Ενδεχόμενο πρόβλημα υπεισέρχεται από το γεγονός ότι αυτού του είδους οι Ethernet Transceivers χρησιμοποιούν το MII Interface το οποίο παρέχει τα δεδομένα από το δίκτυο και δέχεται τα δεδομένα σε κομμάτια των τεσσάρων bit και όχι σε bytes, που είναι το μικρότερο μέγεθος δεδομένων το οποίο χρησιμοποιεί η σχεδίασή μας.

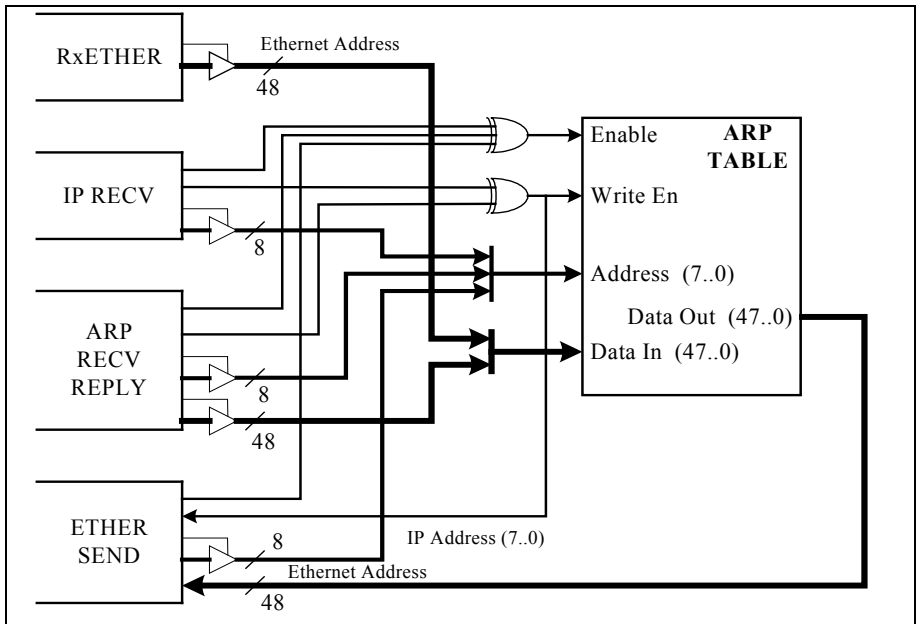
Για να λύσουμε λοιπόν αυτό το πρόβλημα αυτό που κάνουμε με το συγκεκριμένο υποσύστημα είναι να τροφοδοτήσουμε την σχεδίασή μας με ακριβώς το μισό ρολόι απ'αυτό που μας παρέχει ο Ethernet Transceiver, τον ρόλο αυτό του υποδιπλασιαμού της συχνότητας του παρεχόμενου παλμού του ρολογιού αναλαμβάνει το υποσύστημα αυτό όπως και το χρέος να παίρνει τις τετράδες δεδομένων τα οποίο μας παρέχονται από το ολοκληρωμένο με συχνότητα 25Mhz και να τα παραδίδει στην σχεδίασή μας σε μορφή bytes στα 12,5 Mhz πλέον.



Σχήμα 5.2

5.2 Ο πίνακας ARP

Ο πίνακας ARP είναι μονόπορτη σύγχρονη στατική μνήμη με ένα σήμα ενεργοποίησης (Enable) και ένα σήμα για γράψιμο (active high) ή διάβασμα (active low), το Write En. Η διεύθυνση της μνήμης είναι εύρους 8 bits και δέχεται τα 8 LSB της τιμής της διεύθυνσης IP για την οποία πρέπει να αντιστοιχίζεται μια διεύθυνση Ethernet. Τα δεδομένα εισόδου και εξόδου από τη μνήμη είναι εύρους 48 bits και φέρουν τις διευθύνσεις Ethernet που αποθηκεύονται ή επιστρέφονται. Επειδή υπάρχουν πολλά σήματα που καταλήγουν σε ένα (για παράδειγμα στη διεύθυνση της μνήμης) χρησιμοποιούνται tri-state buffers, ώστε να απομονώνονται τα σήματα που δεν δίνουν τιμή τη συγκεκριμένη στιγμή. Στο *Σχήμα 5.3* φαίνονται όλα τα σήματα που συνδέονται στις εισόδους και την έξοδο της μνήμης που υλοποιεί τον πίνακα ARP, καθώς και η λογική με την οποία έχει συνδεθεί με όλα τα υποσυστήματα της σχεδίασης τα οποία τον χρησιμοποιούν.



Σχήμα 5.3 Ο πίνακας ARP.

Ο πίνακας ARP χρησιμοποιούνταν και στην δεύτερη γενιά της σχεδίασης ακριβώς με την ίδια λογική που χρησιμοποιούμε και εμείς. Το βασικό του μειονέκτημα το οποίο μας οδήγησε στο να παρέμβουμε στο συγκεκριμένο υποσύστημα ήταν το γεγονός ότι η μνήμη η οποία τον υλοποιούσε είχε μόνο τέσσερις θέσεις μνήμης. Απόρροια του γεγονότος αυτού ήταν σε πραγματικές συνθήκες λειτουργίας (όταν θα συνδέαμε π.χ. την σχεδίαση σε ένα τοπικό δίκτυο) η

σχεδίασή μας να μην δουλέψει σωστά αφού ο πίνακας ARP δεν θα μπορούσε να διακρίνει μεταξύ δύο IP διευθύνσεων οι οποίες αν και διαφορετικές θα είχαν τα δύο τελευταία bit τους ίδια.

Βρισκόμαστε λοιπόν μπροστά σε έναν πολύ μεγάλο περιορισμό της σχεδίασης καθώς το σύστημά μας για να δουλέψει σωστά θα έπρεπε να βρίσκεται σε ένα υποδίκτυο το οποίο θα απαρτιζόταν το πολύ από άλλους τέσσερις υπολογιστές ή άλλα συστήματα με δυνατότητα δικτύωσης τα οποία όμως θα έπρεπε και να έχουν διαφορετικά μεταξύ τους τα τελευταία δύο bit της διεύθυνσης IP τους (2 LSBs).

Η λύση την οποία εμείς δώσαμε δεν είναι και η πιο σωστή αφού δεν εξαλείψαμε το συγκεκριμένο πρόβλημα απλά το περιορίσαμε, αυξάνοντας όμως τις δυνατότητες τις σχεδίασής μας κατά πολύ. Αυτό το οποίο κάναμε λοιπόν ήταν να αυξήσουμε το μέγεθος της συγκεκριμένης μνήμης η οποίο παίζει τον ρόλο του ARP table σε 256 θέσης κάτι το οποίο σημαίνει και αύξηση των bit διευθυνσιοδότησής της από δύο σε οκτώ. Πλέον λοιπόν η σχεδίασή μας έχει την δυνατότητα να βρίσκεται μέσα σε ένα υποδίκτυο 256 υπολογιστών ή και άλλων συσκευών με δυνατότητα δικτύωσης χωρίς κανένα πρόβλημα αφού ένα subnet μπορεί να έχει μέχρι και 256 συσκευές και είμαστε και πλέον σίγουροι ότι και τα οκτώ τελευταία bit της διεύθυνσης IP τους (8 LSBs), θα είναι διαφορετικά μεταξύ τους.

Όπως προαναφέραμε, μετριάσαμε το πρόβλημα αλλά δεν το λύσαμε τελείως κι αυτό διότι στον σημερινό κόσμο όπου όλα τα δίκτυα συνδέονται μεταξύ τους μέσω του Internet, καταλαβαίνουμε ότι το πρόβλημα μπορεί να εμφανιστεί πάλι, αφού κάποιες διευθύνσεις IP μπορεί να είναι διαφορετικές αλλά να έχουν ίδια και τα οκτώ τελευταία bit.

Για να μπορεί να υποστηρίξει λοιπόν πλήρως τη διαδικασία αντιστοίχισης ο πίνακας ARP θα πρέπει να χρησιμοποιηθεί μια CAM (Content Address Memory) σε συνεργασία με την υπάρχουσα δομή του πίνακα ARP κατ'αυτόν τον τρόπο θα ταυτίζεται ολόκληρη η διεύθυνση IP και θα έχουμε μοναδική αντιστοιχία σε κάποια διεύθυνση Ethernet. Ο λόγος για τον οποίο δεν προχωρήσαμε σε μια τέτοια υλοποίηση είναι καθαρά θέμα χρόνου. Το θέμα της διπλωματικής αυτής εργασίας είναι η υλοποίηση ελέγχου συμφόρησης δικτύου για το πρωτόκολλο TCP/IP οπότε δεν μας έμεινε χρόνος για την ορθή υλοποίηση μιας τέτοιας δομής η οποία εκτός από τις αλλαγές που θα επέφερε στους χρονισμούς σχεδόν όλης της σχεδίασης, θα χρειαζόταν και ένα πολύ πιο πολύπλοκο σύστημα ελέγχου το οποίο θα εγγυούταν για την ορθή λειτουργία μιας τέτοιας δομής. Έτσι λοιπόν προτιμήσαμε να πάμε σε μία λύση που απλά μετριάζει το πρόβλημα, χωρίς να το λύνει απέχοντας όμως πολύ από άποψη λειτουργικότητας σε σχέση με την δεύτερη γενιά της σχεδίασης.

5.3 Το υποσύστημα κεντρικού ελέγχου, Control

Στην δεύτερη γενιά της σχεδίασης υπήρχε ένα κεντρικό σύστημα ελέγχου για τον καθορισμό της ροής της πληροφορίας των πακέτων που εισέρχονται στο σύστημα αλλά και των πακέτων που εξέρχονται από αυτό. Υλοποιεί δηλαδή τη διαδικασία απόπλεξης των εισερχόμενων πακέτων και τη διαδικασία πολυπλεξίας των εξερχόμενων πακέτων μεταξύ του επιπέδου δικτύου και του επιπέδου μεταφοράς. Επιπλέον ασκεί τον κεντρικό έλεγχο για την εγγραφή στην μνήμη μετάδοσης (TxRAM), καθώς χτίζεται ένα πακέτο για να σταλθεί από το σύστημα στο τοπικό δίκτυο.

Αναλυτικότερα το υποσύστημα παρατήρησης ροών από πακέτα IP, ελέγχοντας το πεδίο που περιέχει την πληροφορία του είδους του πακέτου, παρέχει ένα σήμα στο κεντρικό υποσύστημα ελέγχου το οποίο περιγράφει την πληροφορία αυτή.

Στη σχεδίαση γίνονται δεκτά στην είσοδο πακέτα ICMP echo request (ping), πακέτα UDP, και πακέτα TCP. Το υποσύστημα ελέγχου με βάση αυτή την πληροφορία ενεργοποιεί αντίστοιχα το κατάλληλο υποσύστημα, ώστε να συνεχίσει αυτό να δέχεται το υπόλοιπο του εισερχόμενου πακέτου. Η ενεργοποίηση αυτή παραμένει σε όλη τη διάρκεια εισόδου του πακέτου, οπότε γίνεται και ο έλεγχος αν προέκυψε σφάλμα από το CRC-32 στο εισερχόμενο πλαίσιο. Το κομμάτι αυτό του υποσυστήματος το οποίο ασκεί έλεγχο στα εισερχόμενα πλαίσια Ethernet κρατήθηκε αυτούσιο και στην τρίτη γενιά της σχεδίασης.

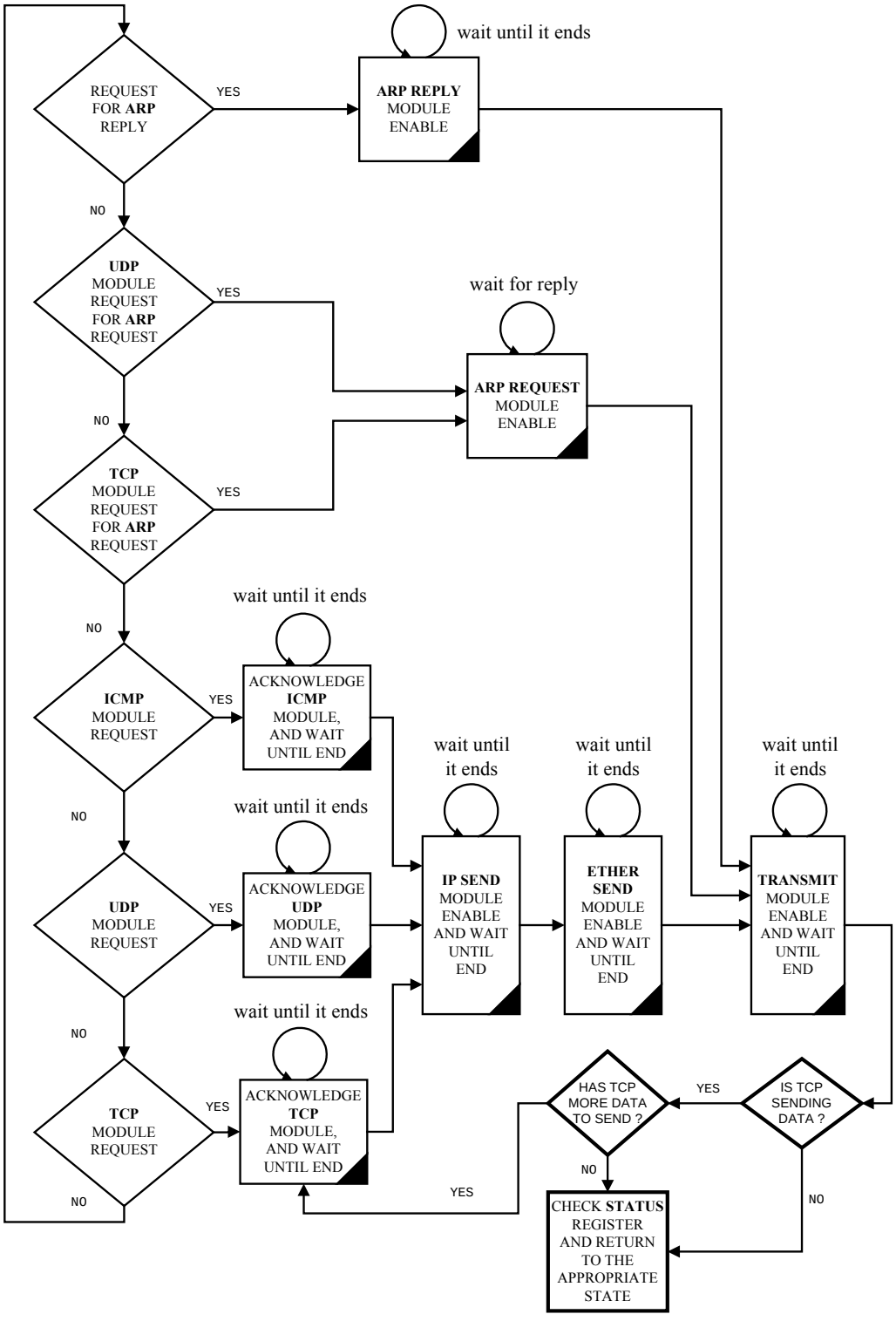
Ο έλεγχος εγγραφής στη μνήμη μετάδοσης είναι απαραίτητο να εκτελείται από το υποσύστημα CONTROL, επειδή τη μνήμη αυτή ζητούν να την προσπελάσουν διάφορα υποσυστήματα, και κάποια από αυτά για να γράψουν διαφορετικού είδους πληροφορία κάθε φορά. Αναλυτικότερα οι προσπελάσεις που είναι πιθανόν να γίνουν στη μνήμη μετάδοσης είναι οι εξής :

1. Από το υποσύστημα απάντησης σε πακέτο τύπου ARP (Address Resolution Protocol).
2. Από το υποσύστημα απάντησης σε πακέτο τύπου ICMP echo request, για ICMP echo reply.
3. Από το υποσύστημα UDP για να σταλθεί πακέτο τύπου ARP στο δίκτυο, για την αντιστοίχηση κάποιας διεύθυνσης IP απομακρυσμένου συστήματος με τη διεύθυνση Ethernet.
4. Από το υποσύστημα UDP για την αποστολή datagram πακέτου κάποιας εφαρμογής.
5. Από το υποσύστημα TCP για να σταλθεί πακέτο τύπου ARP στο δίκτυο, για την αντιστοίχιση κάποιας διεύθυνσης IP απομακρυσμένου συστήματος με τη διεύθυνση Ethernet.

6. Από το υποσύστημα TCP για την αποστολή δεδομένων από κάποια εφαρμογή.
7. Από το υποσύστημα TxETHER που υλοποιεί το MAC (Multiple Access Control) για τη μετάδοση ενός πλαισίου στο τοπικό δίκτυο.
8. Τέλος προσπελαύνεται και από τα υπόλοιπα υποσυστήματα των επιπέδων του δικτύου, όπως το υποσύστημα IP_SEND και το υποσύστημα ETHER_SEND, για την προσθήκη της αντίστοιχης επικεφαλίδας.

Οι πιθανές ταυτόχρονες προσβάσεις στη μνήμη εξόδου από τα υποσυστήματα που υπάρχουν στη σχεδίαση καθιστούν αναγκαία την ύπαρξη ενός μηχανισμού ανταλλαγής σημάτων επικοινωνίας (handshaking), ώστε το κάθε υποσύστημα να γράφει στη μνήμη μόνο όταν είναι ελεύθερη και δεν την προσπελαύνει κανένα από τα υπόλοιπα υποσυστήματα. Έτσι υπάρχουν διαθέσιμα σήματα για αιτήσεις πρόσβασης και αντίστοιχα σήματα επιβεβαίωσης ότι είναι ελεύθερη για εγγραφή, καθώς και σήματα που ενεργοποιούν απευθείας κάποια υποσυστήματα για εγγραφή, ώστε να ολοκληρωθεί το πλαίσιο που θα μεταδοθεί με την προσθήκη της κατάλληλης επικεφαλίδας για κάθε πρωτόκολλο . Τα σήματα αυτά είναι τόσα, όσες είναι και οι περιπτώσεις προσπέλασης της μνήμης, που απαριθμήθηκαν παραπάνω. Κάθε υποσύστημα πρέπει να διατηρεί την αίτησή του μέχρι να λάβει την επιβεβαίωση ότι μπορεί να γράψει στη μνήμη ελεύθερα. Αν τερματίσει την αίτηση χωρίς να λάβει επιβεβαίωση, τότε αυτή δεν θα πραγματοποιηθεί, μέχρι την επόμενη αίτηση από το ίδιο υποσύστημα. Ο μηχανισμός παρακολούθησης των αιτήσεων και παροχής επιβεβαιώσεων υλοποιείται με ένα αλγόριθμο εκ περιτροπής κατά τον οποίο ελέγχονται όλα τα υποσυστήματα με τη σειρά και κυκλικά αν κάνουν αίτηση για εγγραφή στη μνήμη μετάδοσης. Μόλις διαπιστωθεί μια αίτηση παρέχεται η αντίστοιχη επιβεβαίωση, και αρχίζει η εγγραφή από το αιτών υποσύστημα, ενώ σταματάει ο έλεγχος για άλλες αιτήσεις μέχρι να τελειώσει όλη η διαδικασία εγγραφής στη μνήμη εξόδου, αλλά και η μετάδοση στο δίκτυο των δεδομένων που εισαχθήκανε σε αυτήν. Όταν τελειώσει αυτή η διαδικασία, τότε προχωράει ο μηχανισμός στον επόμενο έλεγχο, κ.ο.κ.

Σε αυτό εδώ το σημείο έγινε και η δική μας παρέμβαση στο υποσύστημα control. Στην δεύτερη γενιά της σχεδίασης το υποσύστημα TCP μπορούσε να στείλει μόνο ένα πακέτο την φορά και κατόπιν περίμενε για την επιβεβαίωση του συγκεκριμένου πακέτου για να προχωρήσει στην αποστολή του επόμενου. Έτσι ο έλεγχος της μνήμης μετάδοσης έφευγε από το υποσύστημα TCP το οποίο έπρεπε πάλι να κάνει αίτηση στο υποσύστημα control ώστε να μπορέσει να στείλει και το επόμενο πακέτο της ροής TCP. Η παρέμβασή μας λοιπόν έγκειται στο εξής προστέθηκαν κάποια επιπλέον σήματα ελέγχου μεταξύ του υποσυστήματος control και του υποσυστήματος TCP τα οποία εξυπηρετούν τον εξής σκοπό :



Σχήμα 5.4 Διάγραμμα ροής του υποσυστήματος CONTROL

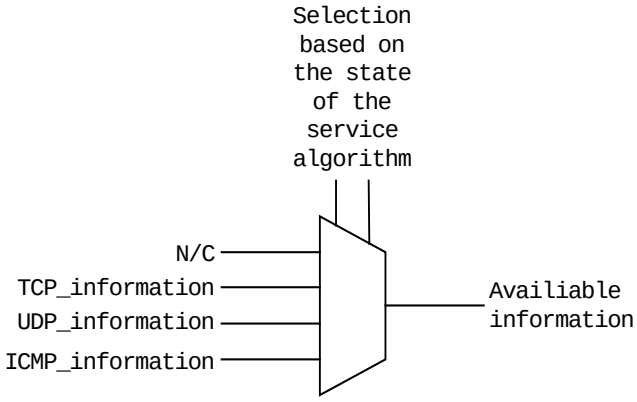
Το υποσύστημα TCP αφού πάρει τον έλεγχο της μνήμης μετάδοσης προχωράει στην δημιουργία του πακέτου στην μνήμη μετάδοσης όταν τελειώσει την δουλειά του το TCP, αν έχει να στείλει και άλλα πακέτα μέσα στο ίδιο παράθυρο κάνει αίτηση στο υποσύστημα control και ζητάει τον έλεγχο της μνήμης μετάδοσης αμέσως μετά το τέλος της αποστολής του πακέτου προς το δίκτυο και πριν δοθεί ο έλεγχος σε κάποιο άλλο υποσύστημα. Έτσι το υποσύστημα control συνεχίζει την εργασία του δίνοντας τον έλεγχο της μνήμης μετάδοσης στα άλλα υποσυστήματα τα οποία πρέπει να προσθέσουν κάποια επικεφαλίδα στο εξερχόμενο πακέτο και μετά το τέλος αυτής της διεργασίας ελέγχει αν το TCP έχει κάνει αίτηση για να ξαναπάρει τον έλεγχο της μνήμης. Αν τώρα το TCP έχει πλέον εξαντλήσει τον αριθμό των πακέτων τα οποία μπορεί να στείλει μέσα στο ίδιο παράθυρο, τότε μετά την μετάδοση και του τελευταίου πακέτου προς το δίκτυο, το control συνεχίζει τον κυκλικό αλγόριθμο ελέγχου που υλοποιεί και ελέγχει αν κάποιο άλλο υποσύστημα έχει κάνει αίτηση για χρήση της μνήμης μετάδοσης. Παραπάνω στο *σχήμα 5.4* βλέπουμε την λειτουργία του υποσυστήματος control, όσον αφορά τον έλεγχο της μνήμης μετάδοσης.

Πρέπει να σημειωθεί ότι υπάρχει ένας καταχωρητής κατάστασης για να μπορεί το σύστημα να επανέλθει στη σωστή κατάσταση μετά από μια διαδικασία εγγραφής. Όταν σε κάποιο έλεγχο αίτησης βρεθεί ότι ζητείται να γίνει εγγραφή στη μνήμη μετάδοσης, τότε γράφεται στον καταχωρητή κατάστασης μια τιμή η οποία περιγράφει μοναδικά την αίτηση που εξυπηρετείται, ώστε όταν τελειώσει όλη η διαδικασία , το υποσύστημα CONTROL να μπορέσει να συνεχίσει την παρακολούθηση των αιτήσεων από το σημείο που είχε μείνει.

Παράλληλα με κάθε αίτηση κάποιου υποσυστήματος είναι απαραίτητο να δίδεται και κάποια πληροφορία σχετικά με τα δεδομένα που θέλει να γράψει, όπως η διεύθυνση IP, του απομακρυσμένου συστήματος με το οποίο θέλει να επικοινωνήσει ,τόσο στην περίπτωση πακέτου με δεδομένα όσο και στην περίπτωση δημιουργίας πλαισίου τύπου ARP, καθώς και πληροφορία για το μέγεθος του πακέτου που περιέχει δεδομένα. Η πληροφορία που παρέχεται από τα υποσυστήματα ICMP, UDP και TCP χρησιμοποιείται από τα υποσυστήματα που βρίσκονται από το επίπεδο του δικτύου και κάτω, για τη δημιουργία των αντίστοιχων επικεφαλίδων. Γι’ αυτό το λόγο η πληροφορία πρέπει να παραμένει διαθέσιμη και σταθερή σε όλη τη διάρκεια εγγραφής και μετάδοσης, για να δομηθεί σωστά το τελικό πλαίσιο που θα μεταδοθεί.

Το υποσύστημα ελέγχου έχοντας γνώση ποιο υποσύστημα εξυπηρετεί μπορεί και υλοποιεί εσωτερικά ένα πολυπλέκτη 4 σε 1 που επιλέγει τη σωστή πληροφορία για διαμοιρασμό στο σύστημα. Ο πολυπλέκτης αυτός σε συνδυασμό με τον αλγόριθμο εξυπηρέτησης αιτήσεων επιτυγχάνει τη συνολική πολυπλεξία των πρωτοκόλλων προς το κοινό μέσο μετάδοσης, δηλαδή

τη μνήμη μετάδοσης και το σύστημα που υλοποιεί το φυσικό επίπεδο του δικτύου. Ο πολυπλέκτης και τα σήματά του φαίνονται στο *Σχήμα 5.5*. Το κομμάτι αυτό του υποσυστήματος control παρέμεινε ως έχει από την δεύτερη γενιά της σχεδίασης.



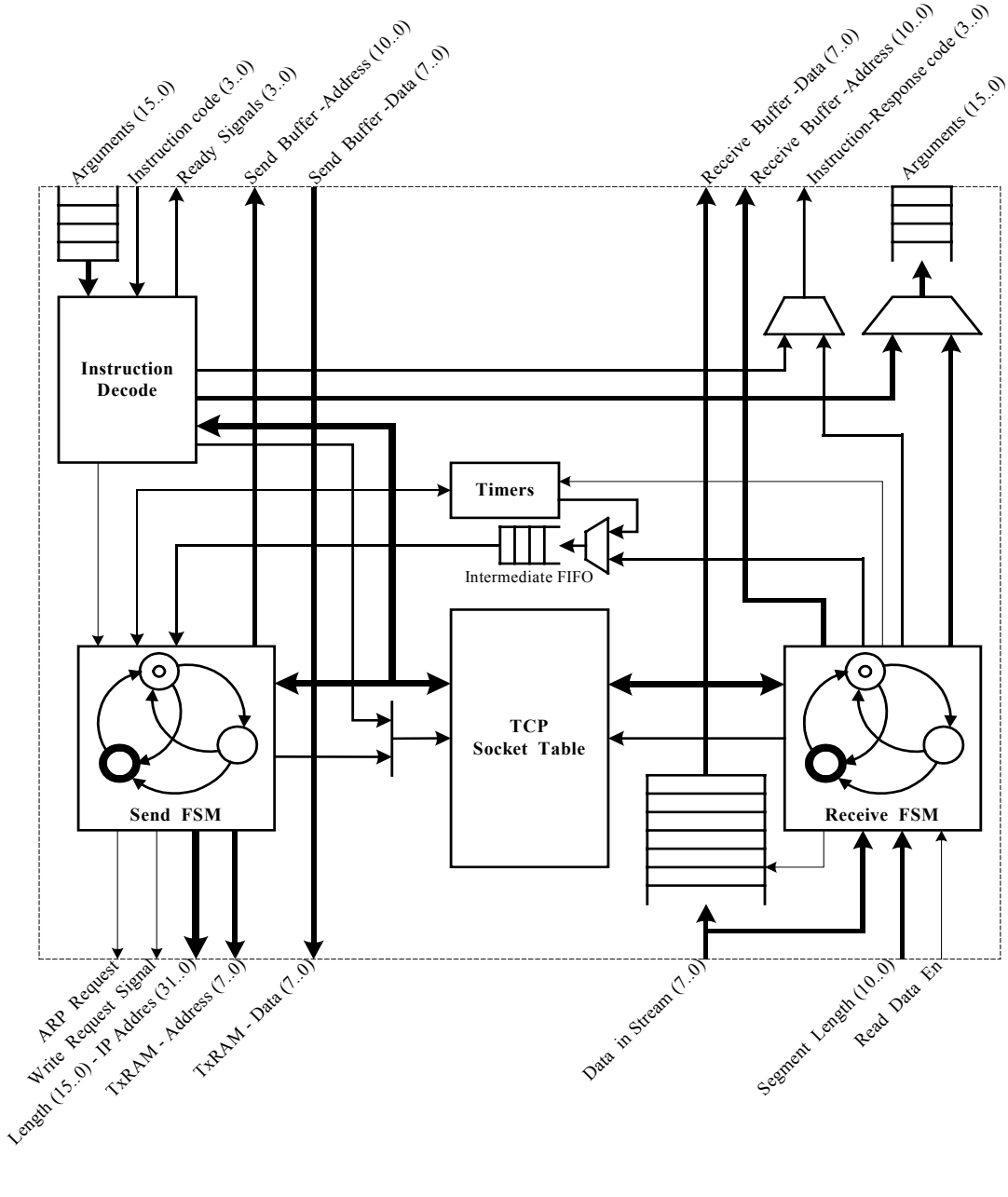
Σχήμα 5.5 Πολυπλεξία δεδομένων από το υποσύστημα CONTROL.

5.4 Το υποσύστημα TCP

Το υποσύστημα TCP λοιπόν βασίστηκε στην αρχιτεκτονική η οποία είχε προταθεί στην δεύτερη γενιά της σχεδίασης όπως έχουμε ήδη αναφέρει σε προηγούμενο κεφάλαιο. Μπορεί λοιπόν να έχουν παραμείνει και στην παρούσα φάση της σχεδίασης τα ίδια δομικά στοιχεία έχει αλλάξει όμως πλήρως η λειτουργικότητά τους ώστε να μπορέσουμε να υποστηρίξουμε τον έλεγχο συμφόρησης του δικτύου. Παρακάτω θα δώσουμε μια συνοπτική περιγραφή του TCP και κατόπιν θα προχωρήσουμε σε αναλυτική περιγραφή των υποσυστημάτων και της λειτουργικότητάς τους.

Το υποσύστημα TCP είναι υπεύθυνο για το χειρισμό ροών τμημάτων TCP είτε αυτά αποστέλλονται από τη σχεδίαση προς ένα απομακρυσμένο σταθμό, είτε εισέρχονται στη σχεδίαση. Επίσης αναλαμβάνει τη διαχείριση των συνδέσεων, δηλαδή τη δημιουργία τους, το κλείσιμο αυτών καθώς και την αξιόπιστη μεταφορά των δεδομένων μεταξύ δύο εφαρμογών, σύμφωνα με τις οδηγίες που παρέχει το RFC 793. Στο *Σχήμα 5.6* παρατίθεται η εσωτερική δομή του υποσυστήματος TCP και τα επιμέρους υποσυστήματα που το αποτελούν είναι τα εξής:

- Μια FIFO για τη λήψη ορισμάτων που ακολουθούν τις εντολές από το επίπεδο εφαρμογών.
- Μια FIFO για την αποστολή ορισμάτων που επιστρέφονται με τις απαντήσεις του TCP προς τις εφαρμογές.
- Το υποσύστημα Instruction Decode που αποκωδικοποιεί τις εντολές των εφαρμογών και δρομολογεί την εκτέλεσή τους.
- Το υποσύστημα που αποστέλλει τμήματα TCP στο δίκτυο.
- Το υποσύστημα με τους timers για το μέτρημα των χρόνων μέχρι την επιβεβαίωση κάποιου τμήματος.
- Το υποσύστημα που δημιουργεί εσωτερικές εντολές για αναμεταδώσεις και επιβεβαιώσεις που πρέπει να γίνουν. Οι εντολές αυτές δρομολογούνται μέσω της ενδιάμεσης FIFO που υπάρχει.
- Το υποσύστημα που αναλαμβάνει τη λήψη τμημάτων TCP και την προώθηση τους στη μνήμη λήψης.
- Μια FIFO απόζευξης των εισερχόμενων δεδομένων για να δοθεί απαραίτητος χρόνος στο προηγούμενο υποσύστημα να ελέγξει την κατάσταση της σύνδεσης και να αποφασίσει για τον χειρισμό τους.
- Τον πίνακα που αποθηκεύονται τα sockets που δημιουργούνται καθώς και οι τιμές των παραμέτρων της σύνδεσης.



Σχήμα 5.6 Εσωτερική δομή του υποσυστήματος TCP.

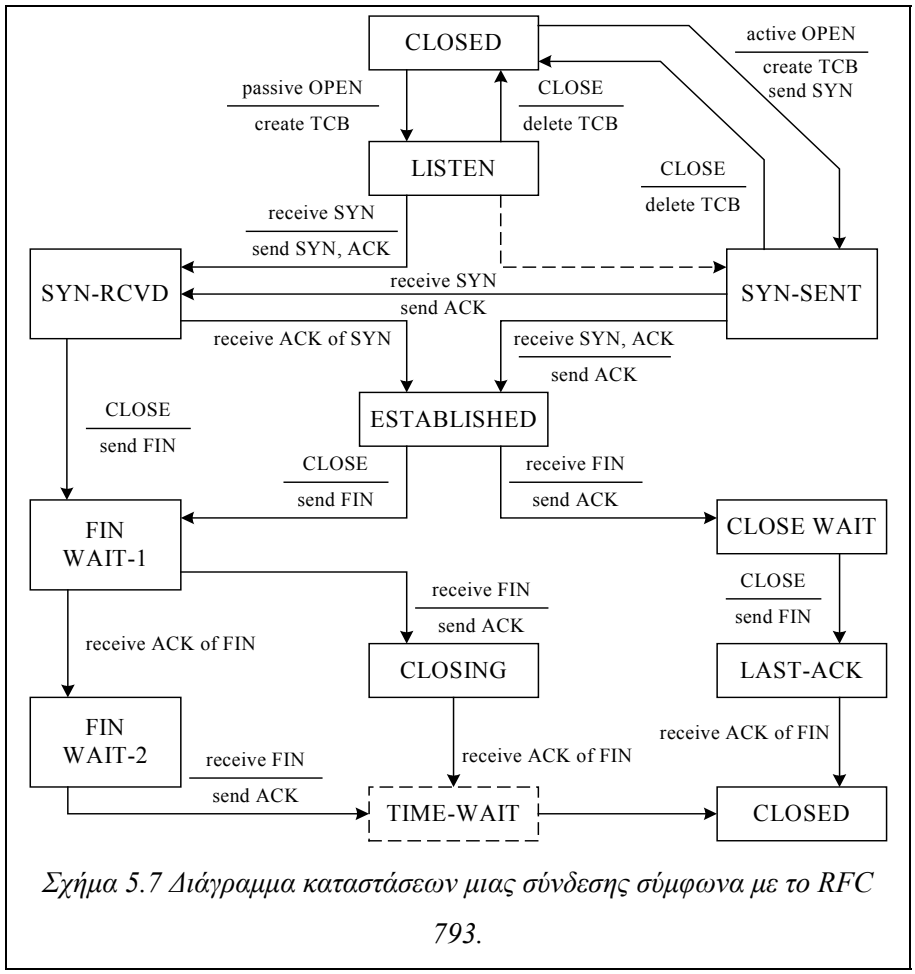
Στο *Σχήμα 5.7* δίδεται το διάγραμμα καταστάσεων όπως προτείνεται από το RFC 793, όπου φαίνεται η κάθε σύνδεση σε ποια κατάσταση μπορεί να βρεθεί ανάλογα με το τμήμα TCP που έρχεται στη σχεδίαση, και ποια είναι η επόμενη κατάσταση στην οποία μπορεί να μεταβεί. Το διάγραμμα καταστάσεων αυτό δεν έχει υλοποιηθεί σαν αυτόνομο FSM, αλλά η κατάσταση περιγράφεται από μια τιμή σε συγκεκριμένη θέση στο TCB με επιπλέον πληροφορία για τη

σύνδεση αυτή. Ο τρόπος αυτός αναπαράστασης δίνει τη δυνατότητα η αρχιτεκτονική σχεδίαση του υποσυστήματος να μπορεί να χειριστεί περισσότερες από μία συνδέσεις.

Η δράση λοιπόν του υποσυστήματος TCP είναι συνάρτηση των εντολών που παρέχει η εφαρμογή για μια σύνδεση, ή η λήψη κάποιου τμήματος TCP, και η κατάσταση που βρίσκεται αυτή η σύνδεση. Το διάγραμμα καταστάσεων περιγράφει σε κάθε περίπτωση ποια θα είναι αυτή η δράση και ποια θα είναι η επόμενη κατάσταση στην οποία θα βρεθεί η σύνδεση. Η μετάβαση από την κατάσταση LISTEN στην κατάσταση SYN-SENT δεν υποστηρίζεται από την υλοποίηση καθώς δεν θεωρήθηκε απαραίτητο μια σύνδεση τύπου εξυπηρετητή να μπορεί να μετατραπεί σε σύνδεση τύπου πελάτη. Επίσης η κατάσταση TIME-WAIT δεν υλοποιήθηκε και παρακάμπτεται κατά τη μετάβαση από τις καταστάσεις CLOSING και FIN WAIT-2 αφού αυτές καταλήγουν αμέσως στην κατάσταση CLOSED. Ο ρόλος της TIME-WAIT είναι η τοποθέτηση καθυστέρησης μεταξύ του κλεισίματος μιας σύνδεσης και της δυνατότητας να ανοιχτεί ξανά, για να αποφεύγονται λάθη από πακέτα IP με τμήματα TCP που περιφέρονται στο δίκτυο από προηγούμενες πιθανές λανθασμένες δρομολογήσεις. Η κατάσταση αυτή παραλείφθηκε αφενός γιατί παρουσίαζε δυσκολία στην υποστήριξή της και αφετέρου γιατί δεν επηρεάζει τη σωστή λειτουργία του πρωτοκόλλου, παρά έχει αποτρεπτικό ρόλο σε πιθανά σφάλματα. Ο ρόλος αυτός μπορεί να επιτευχθεί αν τίθεται ο περιορισμός ότι οι εφαρμογές δεν θα πρέπει να ανοίγουν αμέσως συνδέσεις που έκλεισαν πριν από μικρό χρονικό διάστημα. Στον *Πίνακα 5.1* παρατίθενται οι δυαδικές τιμές που αναπαριστούν τις διαφορετικές καταστάσεις του προηγούμενου διαγράμματος. Οι τιμές αυτές ακολουθούν κωδικοποίηση one-hot αφενός γιατί υπάρχει χώρος στη θέση μνήμης (16 bits) και αφετέρου γιατί το bit που έχει τιμή ‘1’ χρησιμοποιείται για να ενεργοποιήσει απευθείας κάποια διαδικασία, χωρίς να είναι απαραίτητο να γίνει σύγκριση ολόκληρης της τιμής που περιγράφει την κατάσταση.

Όνομα κατάστασης	**Δυαδική τιμή αναπαράστασης**
CLOSED	“0000000000000001”
LISTEN	“0000000000000010”
SYN-SENT	“0000000000000100”
SYN-RECEIVED	“0000000000001000”
ESTABLISHED	“0000000000010000”
FIN WAIT-1	“0000000000100000”
FIN WAIT-2	“0000000001000000”
CLOSING	“0000000010000000”
CLOSE WAIT	“0000001000000000”
LAST-ACK	“0000010000000000”

Πίνακας 5.1 Καταστάσεις των συνδέσεων και αναπαράσταση σε δυαδική τιμή.



Σχήμα 5.7 Διάγραμμα καταστάσεων μιας σύνδεσης σύμφωνα με το RFC 793.

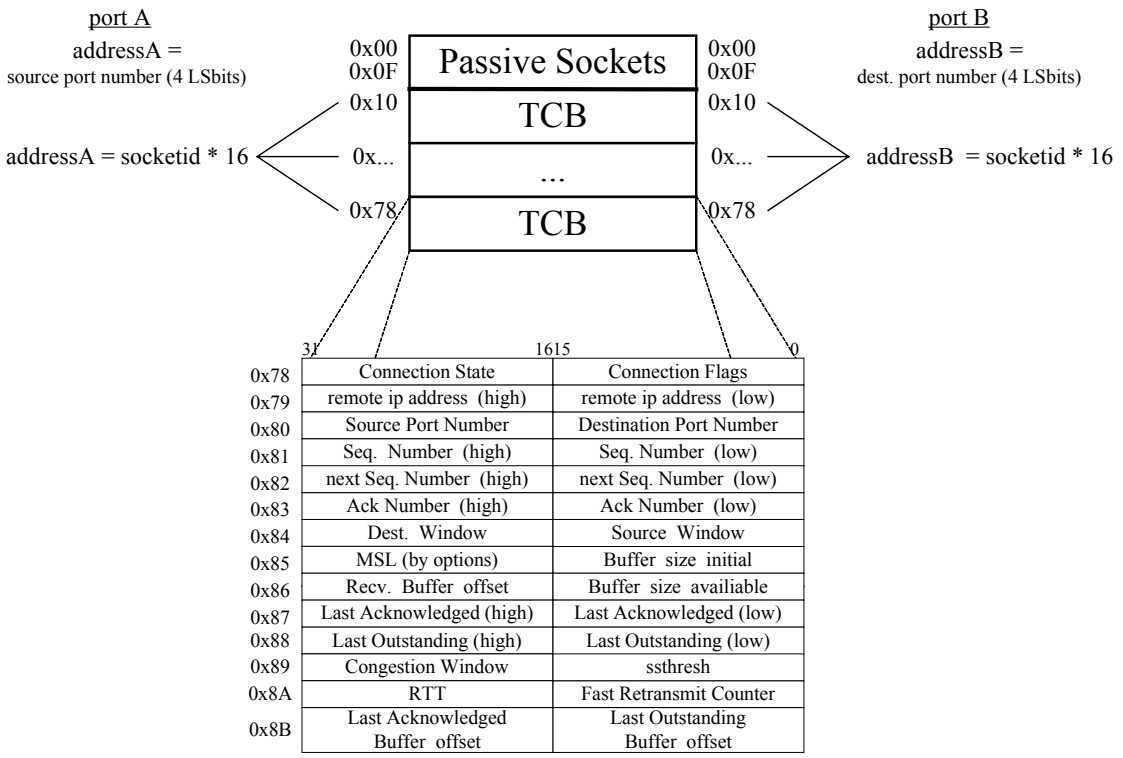
5.4.1 Sockets και εντολές στο πρωτόκολλο TCP

Στο TCP υπάρχουν εντολές για δημιουργία socket τόσο για εφαρμογές εξυπηρετητή όσο και για εφαρμογές πελάτη με τη διαφορά ότι η δημιουργία ενός socket από εφαρμογή πελάτη ενεργοποιεί και το μηχανισμό για την καθιέρωση σύνδεσης. Κάθε σύνδεση είναι μοναδική στο δίκτυο αφού αποτελείται από ένα ζεύγος από sockets μεταξύ των δυο εφαρμογών που επικοινωνούν. Πιο αναλυτικά όταν το socket περιλαμβάνει την πληροφορία για τη διεύθυνση IP του απομακρυσμένου σταθμού, το destination port number της απομακρυσμένης εφαρμογής και το local port number της τοπικής εφαρμογής, τότε το ζευγάρι που προκύπτει με το συμπληρωματικό socket για κάποιο απομακρυσμένο χρήστη ορίζει μοναδικά μια σύνδεση. Έτσι λοιπόν σε ένα TCB στον πίνακα των socket φυλάσσεται αυτή η πληροφορία, καθώς και η κατάσταση της σύνδεσης. Η δυαδική τιμή του πρώτου πεδίου του TCB περιγράφει την

κατάσταση της σύνδεσης και παίζει πολύ σημαντικό ρόλο στη μετάδοση δεδομένων μεταξύ των εφαρμογών. Άλλα στοιχεία που φυλάσσονται σε ένα TCB είναι το Sequence Number, το Next Sequence Number, το Acknowledgment Number και τα Window Size των δύο sockets. Ακόμα φυλάσσεται η διεύθυνση της μνήμης για λήψη τμημάτων TCP από τη σύνδεση αυτή, ο εναπομένων χώρος στη μνήμη για τη συγκεκριμένη λήψη και η τιμή το M.M.T. για τα εξερχόμενα τμήματα. Επίσης στο TCB φυλάσσεται πληροφορία για το τελευταίο Sequence Number που έχει επιβεβαιωθεί και η διεύθυνση όπου υπάρχουν στην μνήμη αυτά τα δεδομένα, το τελευταίο Sequence Number από το οποίο περιμένουμε επιβεβαίωση και η διεύθυνση όπου υπάρχουν στην μνήμη αυτά τα δεδομένα, οι τιμές των μεταβλητών congestion window και slow start threshold, ο χρόνος που χρειάζεται ένα τμήμα TCP να μεταβεί στον προορισμό του και να έρθει πίσω η επιβεβαίωσή του και τέλος η τιμή ενός counter που χρησιμοποιούμε για την ενεργοποίηση του αλγορίθμου Fast Retransmit. Στο *Σχήμα 5.8* παρουσιάζεται η δομή του πίνακα των sockets αλλά και η εσωτερική δομή ενός TCB.

Όπως φαίνεται στο σχήμα η μνήμη που περιέχει τα TCBs είναι δίπορτη για παράλληλη προσπέλαση από τα FSMs που αποστέλλουν και λαμβάνουν τμήματα TCP. Το socketid διευθυνσιοδοτεί ανά 16 θέσεις μνήμης και η προσπέλαση των ενδιάμεσων διευθύνσεων γίνεται με τα 4 LSBits του address bus της μνήμης. Η τιμή του socketid διαμορφώνεται ως εξής:
socketid = *remote ip address(bit 0) & destination port number(bit 1,0) & source port number(bit 1,0)*.
Έτσι περιέχει τα χαρακτηριστικά του socket το οποίο μπορεί να δημιουργήσει μοναδική σύνδεση με το συμπληρωματικό του ζευγάρι.

Με τον τρόπο τον οποίο έχουμε κάνει την υλοποίηση της μνήμης η οποία περιέχει τα TCBs μπορούμε να υποστηρίξουμε μέχρι 32 συνδέσεις TCP με το κατάλληλο μέγεθος μνήμης. Το σύστημα είναι δηλαδή έτσι σχεδιασμένο ώστε με ελάχιστες αλλαγές να μπορεί να υποστηρίξει 32 συνδέσεις TCP. Εμείς στην δική μας υλοποίηση υλοποιήσαμε χώρο για 20 συνδέσεις TCP και αυτό για να δείξουμε την δυνατότητά μας να πάμε σε μεγαλύτερα νούμερα συνδέσεων σε σχέση με την δεύτερη γενιά της σχεδίασης (13 συνδέσεις). Δεν φτάσαμε την σχεδίαση στα όριά της μόνο και μόνο για ευκολία στην προσομοίωση της μιας και πρόκειται για μια πολύ χρονοβόρα διαδικασία, και με αυτόν τον τρόπο καταφέραμε να κερδίσουμε πολύτιμο χρόνο στην υλοποίηση της ίδιας της σχεδίασης, ξέροντας όμως πως ότι ισχύει για τις 20 συνδέσεις ισχύει και για τις 32 με μοναδικό αντίκυπο να μειωθεί η μέγιστη συχνότητα στην οποία μπορεί να λειτουργήσει η σχεδίαση.



Σχήμα 5.8 Πίνακας των sockets και η δομή των TCB για το πρωτόκολλο TCP.

Στις 16 χαμηλότερες θέσεις της μνήμης που περιέχει τα TCBs φυλάσσονται τα sockets που ανοίγουν από εφαρμογές τύπου εξυπηρετητή και στα οποία αναμένουν αιτήσεις για σύνδεση από απομακρυσμένες εφαρμογές τύπου πελάτη. Κάθε τέτοιο socket (passive) καταλαμβάνει μόνο μια γραμμή που στα 16 MSbits φυλάγεται η κατάσταση της σύνδεσης και στα 16 LSbits φυλάγεται η τιμή του source port number της τοπικής εφαρμογής που άνοιξε το socket. Η προσπέλαση αυτών των θέσεων μνήμης γίνεται με χρήση των 4 LSbits της τιμής source port number από την πλευρά A και με τα 4 LSbits του destination port number από την πλευρά B. Μια αίτηση για σύνδεση εξετάζεται αν θα γίνει δεκτή μόνο αν το εισερχόμενο τμήμα SYN έχει ίδιο destination port number με το source port number που υπάρχει στην αντίστοιχη θέση στη μνήμη για τα passive sockets και αν η κατάσταση της σύνδεσης είναι LISTEN.

Οι εντολές που υποστηρίζει το υλικό και παρέχουν πρόσβαση στο πρωτόκολλο TCP είναι παρόμοιες με αυτές που προσφέρει το μοντέλο προγραμματισμού. Λεπτομερέστερα υπάρχουν δυαδικοί κωδικοί που χαρακτηρίζουν την κάθε εντολή, οι ίδιοι κωδικοί επιστρέφονται όταν η εντολή εκτελείται σωστά και το συμπλήρωμά τους όταν προκύπτει σφάλμα. Τα ορίσματα διακινούνται ομοίως μέσω FIFOs εύρους 16-bit. Στις εντολές του πρωτοκόλλου TCP έχουν

προστεθεί κάποιες που ολοκληρώνουν τη λειτουργία. Έτσι υπάρχει η εντολή receive η οποία καθορίζει ουσιαστικά σε ποια διεύθυνση μνήμης μπορεί να δεχτεί τα δεδομένα της σύνδεσης και πόσο είναι το μέγεθός του χώρου της. Ακόμα υπάρχει η εντολή bind με την οποία ο χρήστης κάνει δεκτή την αίτηση μιας απομακρυσμένης εφαρμογής για την καθιέρωση σύνδεσης. Από την πλευρά των απαντήσεων έχουν προστεθεί απαντήσεις-εντολές όπως η connection_request που δείχνει ότι κάποια απομακρυσμένη εφαρμογή κάνει αίτηση για σύνδεση, και όπως η close_wait που δείχνει ότι κάποια απομακρυσμένη εφαρμογή τερματίζει τη σύνδεση, ώστε να μπορέσει να τερματιστεί και τοπικά ελευθερώνοντας το TCB. Τότε η εφαρμογή πρέπει να χρησιμοποιήσει την εντολή close_active_socket. Η εντολή close_passive_socket χρησιμοποιείται για το κλείσιμο των sockets που αναμένουν αιτήσεις για σύνδεση από απομακρυσμένες εφαρμογές. Στους *Πίνακες 5.2* και *5.3* παρατίθενται όλες οι εντολές, με τα ορίσματα που δέχονται, και μια μικρή περιγραφή. Στο TCP γίνονται όλοι οι απαραίτητοι έλεγχοι για τη δημιουργία ενός socket, ενώ κατά τη λήψη και την αποστολή τμημάτων ελέγχεται πάντα η κατάσταση της σύνδεσης και διαβάζονται ή ενημερώνονται οι μεταβλητές που είναι αποθηκευμένες στο TCB του socket.

Συνοπτικά η υλοποίηση του πρωτοκόλλου TCP σε υλικό, περιλαμβάνει FSMs για την αποστολή και λήψη δεδομένων, μνήμες για την δημιουργία των πινάκων των sockets, υποσυστήματα που αποκωδικοποιούν τις εντολές και στέλνουν απαντήσεις στο υψηλότερο επίπεδο, και FIFOs για τη διακίνηση των ορισμάτων μεταξύ του επιπέδου εφαρμογής και του πρωτοκόλλου.

Εντολές από το επίπεδο εφαρμογής προς το πρωτόκολλο TCP.			
Όνομα εντολής	**Κωδικός**	**Ορίσματα**	**Περιγραφή**
Open_passive_socket	0001	1. source port number	Ανοίγει socket για εφαρμογές εξυπηρετητή. Αναμένονται αιτήσεις για σύνδεση.
open_active_socket	0010	1. remote ip address 2. destin. port number 3. source port number 4. congestion window 5. ssthresh	Ανοίγει socket για εφαρμογές πελάτη. Στέλνεται αίτηση για σύνδεση.
Send_segment	0011	1. socketid 2. buffer offset 3. byte count 4. congestion window 5. ssthresh	Αποστέλλει δεδομένα που βρίσκονται σε κάποια θέση στη μνήμη αποστολής σε κάποιο προορισμό (πληροφορία από το TCB).
receive_segment	0100	1. socketid 2. buffer offset 3. byte count	Δείχνει το χώρο μνήμης που πρέπει να γράφονται τα τμήματα που καταφθάνουν από μια σύνδεση, καθώς και το διαθέσιμο μέγεθος.
Bind	0101	1. remote ip address 2. destin. port number 3. source port number	Κάνει δεκτή κάποια αίτηση για σύνδεση και δημιουργεί νέο TCB για τη σύνδεση αυτή.
close_active_socket	0111	1. socketid	Κλείνει ένα socket από εφαρμογή πελάτη, στέλνεται FIN και ελευθερώνει χώρο στον πίνακα.
Close_passive_socket	1111	1. socketid	Κλείνει ένα socket από εφαρμογή εξυπηρετητή και ελευθερώνει χώρο στον πίνακα των sockets.

Πίνακας 5.2 Εντολές και ορίσματα για το πρωτόκολλο TCP

Απαντήσεις-εντολές από το πρωτόκολλο TCP στο επίπεδο εφαρμογής.			
Όνομα	**Κωδικός**	**Ορίσματα**	**Περιγραφή**
suc_open_server_socket	0001	1. socketid	Δείχνει επιτυχία στη δημιουργία του TCB και επιστρέφει το socketid.
err_open_server_socket	1110	1. source port number	Δείχνει αποτυχία στη δημιουργία του TCB σύνδεσης και επιστρέφει το source port number.
suc_open_client_socket	0010	1. socketid	Δείχνει επιτυχία στην καθιέρωση σύνδεσης και επιστρέφει το socketid.
err_open_client_socket	1101	1. remote ip address 2. destin. port number 3. source port number	Δείχνει αποτυχία στην καθιέρωση και επιστρέφει τα ορίσματα που δόθηκαν με την εντολή.
success_send	0011	1. socketid	Δείχνει επιτυχία αποστολής ενός μηνύματος και επιστρέφει το socketid.
error_send	1100	1. socketid	Δείχνει αποτυχία αποστολής του μηνύματος και επιστρέφει το socketid.
success_receive	0100	1. socketid, PSH 2. buffer offset 3. byte count	Δείχνει στο υψηλότερο επίπεδο ότι έχει φτάσει μήνυμα, σε κάποια θέση στη μνήμη λήψης, και το μέγεθός του.
error_receive	1011	1. socketid	Δείχνει στο υψηλότερο ότι δεν έχει ληφθεί το μήνυμα (προφανώς λόγω timeout).
connection_request	1111	1. socketid (listen) 2. remote ip address 3. destin. port number	Εντολή που ζητάει από το επίπεδο εφαρμογής να επικυρώσει μια αίτηση, με τα χαρακτηριστικά των ορισμάτων, για σύνδεση.
close_wait	1001	1. socketid	Εντολή που δείχνει στο επίπεδο εφαρμογής ότι κάποια σύνδεση έχει κλείσει από το άλλο άκρο.

Πίνακας 5.3 Απαντήσεις - εντολές και ορίσματα για το πρωτόκολλο TCP

5.4.2 Το υποσύστημα Instruction Decode

Το υποσύστημα Instruction Decode επιτελεί τις εξής λειτουργίες:

- Αποκωδικοποιεί τις εντολές που έρχονται στο σύστημα από το επίπεδο εφαρμογών, και χειρίζεται τα ορίσματα που ακολουθούν.
- Προετοιμάζει και δρομολογεί την εκτέλεσή τους ενημερώνοντας κάποιες τιμές στο TCB.
- Επιστρέφει κωδικούς λάθους στην περίπτωση που δεν είναι δυνατόν να εκτελεστεί κάποια από τις εντολές.

Η εντολή open_passive_socket(source port number) ανοίγει ένα socket που περιμένει να δεχθεί αιτήσεις για σύνδεση από εφαρμογές τύπου πελάτη. Τα passive sockets τοποθετούνται στις 16 πρώτες θέσεις του socket table από τη διεύθυνση 0x00 ως τη διεύθυνση 0x0F. Το κάθε passive socket καταχωρείται σε μια θέση της μνήμης όπου στα 16 MSBits καταχωρείται η κατάσταση της σύνδεσης και στα 16 LSBits καταχωρείται η τιμή του ορίσματος source port number. Για να ανοίξει ένα νέο passive socket ελέγχεται η κατάσταση που υπάρχει ήδη γραμμένη στη θέση μνήμης όπου θα γραφεί το νέο socket. Αν η τιμή αυτή δεν αναπαριστά την κατάσταση CLOSED σημαίνει ότι κάποια άλλη εφαρμογή έχει καλύψει αυτή τη θέση μνήμης και επιστρέφεται ένας κωδικός λάθους καθώς και η τιμή του source port number για το οποίο έγινε η προσπάθεια δημιουργίας socket. Αν είναι όμως CLOSED τότε στη θέση αυτή γράφεται η τιμή που αναπαριστά την κατάσταση LISTEN και ταυτόχρονα γράφεται η τιμή του source port number που ήρθε σαν όρισμα. Επιστρέφεται ένας κωδικός επιτυχίας με όρισμα το socketid αυτού του passive socket, το οποίο είναι πλέον έτοιμο να δεχθεί αιτήσεις για συνδέσεις από απομακρυσμένες εφαρμογές.

Η εντολή open_active_socket(remote ip address, destination port number, source port number, congestion window, ssthresh) ανοίγει ένα socket από εφαρμογή τύπου πελάτη και ζητάει να κάνει σύνδεση με κάποια απομακρυσμένη εφαρμογή τύπου εξυπηρετητή. Τα active sockets αποθηκεύονται στα διαθέσιμα TCBs που υπάρχουν στο socket table. Το κάθε TCB έχει μέγεθος 16 θέσεων μνήμης οπότε η διευθυνσιοδότηση κάποιου από αυτά γίνεται ως εξής:

remote ip address(LSBit) & destination port number (2 LSBits) & source port number (2 LSBits), όπου ο τελεστής & σημαίνει συνένωση. Επίσης πολλαπλασιάζεται με το σταθερό αριθμό 16 (ο πολλαπλασιασμός γίνεται με τέσσερα swift right του αριθμού) ώστε να προσπελαστεί η πρώτη θέση του TCB όπου φυλάσσεται η κατάσταση της σύνδεση και κάποια flags. Αν η κατάσταση της σύνδεσης δεν είναι CLOSED τότε σημαίνει ότι το συγκεκριμένο TCB χρησιμοποιείται ήδη,

και επιστρέφεται ένας κωδικός λάθους με ορίσματα αυτά που λήφθηκαν με την εντολή ανοίγματος του socket. Αν αντίθετα η κατάσταση είναι CLOSED ακολουθείται η εξής διαδικασία.

- Ζητείται ARP προς το τοπικό δίκτυο για τη διεύθυνση IP που υπήρχε στα ορίσματα της εντολής. Το υποσύστημα TCP μπαίνει σε αναμονή όσον αφορά την αποστολή τμημάτων μέχρι να έρθει απάντηση στην αίτηση για ARP, ενώ μπορεί να λαμβάνει τμήματα TCP απρόσκοπτα.
- Στη συνέχεια ενημερώνονται οι τιμές του TCB τόσο από τα ορίσματα της εντολής όσο και από κάποιες αρχικές τιμές για τα στοιχεία Sequence number, Next sequence number και Acknowledgment number.
- Έπειτα ειδοποιείται το υποσύστημα που αποστέλλει τμήματα, να στείλει ένα τμήμα SYN.

Επιτυχία για την επίτευξη σύνδεσης επιστρέφεται στο επίπεδο εφαρμογών όταν έρθει η αναμενόμενη απάντηση στην αίτηση σύνδεσης από τον απομακρυσμένο σταθμό. Ο κωδικός επιτυχίας συνοδεύεται από ένα όρισμα που δίνει ένα socketid με το οποίο το επίπεδο εφαρμογών μπορεί να προσπελαύνει τη σύνδεση εφεξής. Το socketid δημιουργείται με τον ίδιο τρόπο που δημιουργείται και η διεύθυνση πρόσβασης στο TCB (περιγράφηκε προηγουμένως) χωρίς να το πολλαπλασιάσουμε με το 16. Μέχρι τη στιγμή που θα έρθει μια τέτοια απάντηση ή μέχρι να προκύψει timeout από τον timer που χρονομετρά γι' αυτό το τμήμα, το υποσύστημα δεν μπορεί να ανοίξει καινούριο active socket και το δείχνει αυτό στο επίπεδο εφαρμογών με το σήμα rdy_send_syn.

Αν προκύψει timeout για το τμήμα SYN που ζητάει σύνδεση σε απομακρυσμένη εφαρμογή, επιστρέφεται κωδικός λάθους στο επίπεδο εφαρμογής και ελευθερώνεται το TCB γράφοντας στη θέση της κατάστασης σύνδεσης την τιμή που αντιστοιχεί σε CLOSED.

Η εντολή send_segment(socketid, buffer offset, byte count, congestion window, ssthresh) στέλνει μια ροή από bytes που βρίσκεται στη μνήμη αποστολής, στη θέση που περιγράφεται από το όρισμα buffer offset και έχει μέγεθος ίσο με την τιμή του ορίσματος byte count. Η ροή στέλνεται σε απομακρυσμένη εφαρμογή σύμφωνα με την πληροφορία που υπάρχει στο TCB που αντιστοιχεί στην τιμή του ορίσματος socketid. Το υποσύστημα Instruction Decode αρχικά ελέγχει αν στο TCB η κατάσταση γι' αυτό το socket είναι ESTABLISHED. Αν δεν είναι ESTABLISHED επιστρέφεται κωδικός λάθους με όρισμα το socketid. Αν η κατάσταση είναι ESTABLISHED τότε ειδοποιείται το υποσύστημα να στείλει αυτή τη ροή σύμφωνα πάντα με την πληροφορία που υπάρχει για τη σύνδεση στο TCB, ενώ αυτό αναλαμβάνει την τμηματοποίηση της ροής καθώς και τις αναμεταδώσεις τμημάτων που δεν επιβεβαιώνονται. Το υποσύστημα

TCP, μόλις ξεκινήσει τη διαδικασία αποστολής μιας ροής, δεν δύναται να στείλει άλλη ροή ταυτόχρονα και ειδοποιεί γι' αυτό το επίπεδο εφαρμογών με το σήμα rdy_send.

Η εντολή receive_segment(socketid, buffer offset, byte count) παρέχεται από το επίπεδο εφαρμογών στο υποσύστημα TCP το οποίο μέσω του Instruction Decode την εκτελεί. Ουσιαστικά η εκτέλεση της εντολής περιλαμβάνει μόνο την ενημέρωση του TCB που αντιστοιχεί στο socketid που υπάρχει σαν πρώτο όρισμα. Η ενημέρωση των πεδίων του TCB receive buffer offset και initial buffer size γίνεται αντίστοιχα από τα ορίσματα buffer offset και byte count που συνοδεύουν την εντολή. Με βάση αυτές τις τιμές που υπάρχουν καταχωρημένες γίνονται δεκτά τμήματα που εισέρχονται στο σύστημα TCP και αναφέρονται στη συγκεκριμένη σύνδεση, και αποθηκεύονται στη μνήμη λήψης. Όταν εξαντληθεί ο χώρος που περιγράφεται από αυτές τις τιμές μετά τη λήψη ενός ή περισσότερων τμημάτων τότε επιστρέφεται στο επίπεδο εφαρμογών εντολή για την άφιξη δεδομένων με πληροφορία για τη σύνδεση στην οποία αναφέρονται και για το χώρο που βρίσκονται στη μνήμη λήψης. Κάθε εφαρμογή που θέλει να λαμβάνει ροές TCP πρέπει να δίνει τέτοιες εντολές στο υποσύστημα TCP ώστε να υπάρχει πάντα χώρος δεσμευμένος για την αποθήκευση των τμημάτων στη μνήμη λήψης.

Η εντολή bind(remote ip address, destination port number, source port number) κάνει δεκτή μια αίτηση για σύνδεση από απομακρυσμένη εφαρμογή. Η εντολή bind δίδεται από το επίπεδο εφαρμογών στο υποσύστημα TCP μόνο όταν ειδοποιηθεί ότι κάποια εφαρμογή απομακρυσμένου σταθμού θέλει να συνδεθεί με κάποια εφαρμογή τοπικά η οποία έχει ανοίξει passive socket και βρίσκεται σε κατάσταση LISTEN. Με την άφιξη λοιπόν μιας τέτοιας αίτησης και αφού ελεγχθεί ότι κάποια εφαρμογή αναμένει αυτές τις αιτήσεις, δίνονται σε αυτήν όλα τα στοιχεία της απομακρυσμένης εφαρμογής που ζητάει σύνδεση. Η αποδοχή αυτής της αίτησης γνωστοποιείται με την εντολή bind, ενώ τα ορίσματά της χρησιμοποιούνται για τη δημιουργία ενός καινούριου TCB στο οποίο καταχωρείται πλέον όλη η πληροφορία για την καινούρια σύνδεση και μέσω αυτής πραγματοποιείται όλη η διακίνηση δεδομένων μεταξύ των δύο εφαρμογών.

Η εντολή close_passive_socket(socketid) δίδεται από μία εφαρμογή εξυπηρετητή για να κλείσει το socket στο οποίο δέχεται αιτήσεις για σύνδεση με κάποια απομακρυσμένη εφαρμογή. Η εκτέλεση της εντολής ουσιαστικά αλλάζει την κατάσταση του socket στο TCB που υποδεικνύεται από το όρισμα socketid που έρχεται μαζί με την εντολή.

Η εντολή close_passive_socket(socketid) ξεκινάει τη διαδικασία για το κλείσιμο μιας σύνδεσης μεταξύ δύο εφαρμογών. Όταν το υποσύστημα Instruction Decode λάβει αυτή την εντολή από το επίπεδο εφαρμογών αλλάζει την κατάσταση της σύνδεσης στο TCB που υποδεικνύει το όρισμα socketid, σύμφωνα με το διάγραμμα του *Σχήματος 5.7*. Στη συνέχεια το

υποσύστημα αποστολής τμημάτων TCP ειδοποιείται να δημιουργήσει και να στείλει ένα τμήμα FIN. Το τμήμα που σχηματίζεται, στέλνεται στην απομακρυσμένη εφαρμογή που περιγράφεται από την πληροφορία του TCB για τη σύνδεση αυτή.

5.4.3 Το υποσύστημα αποστολής τμημάτων TCP

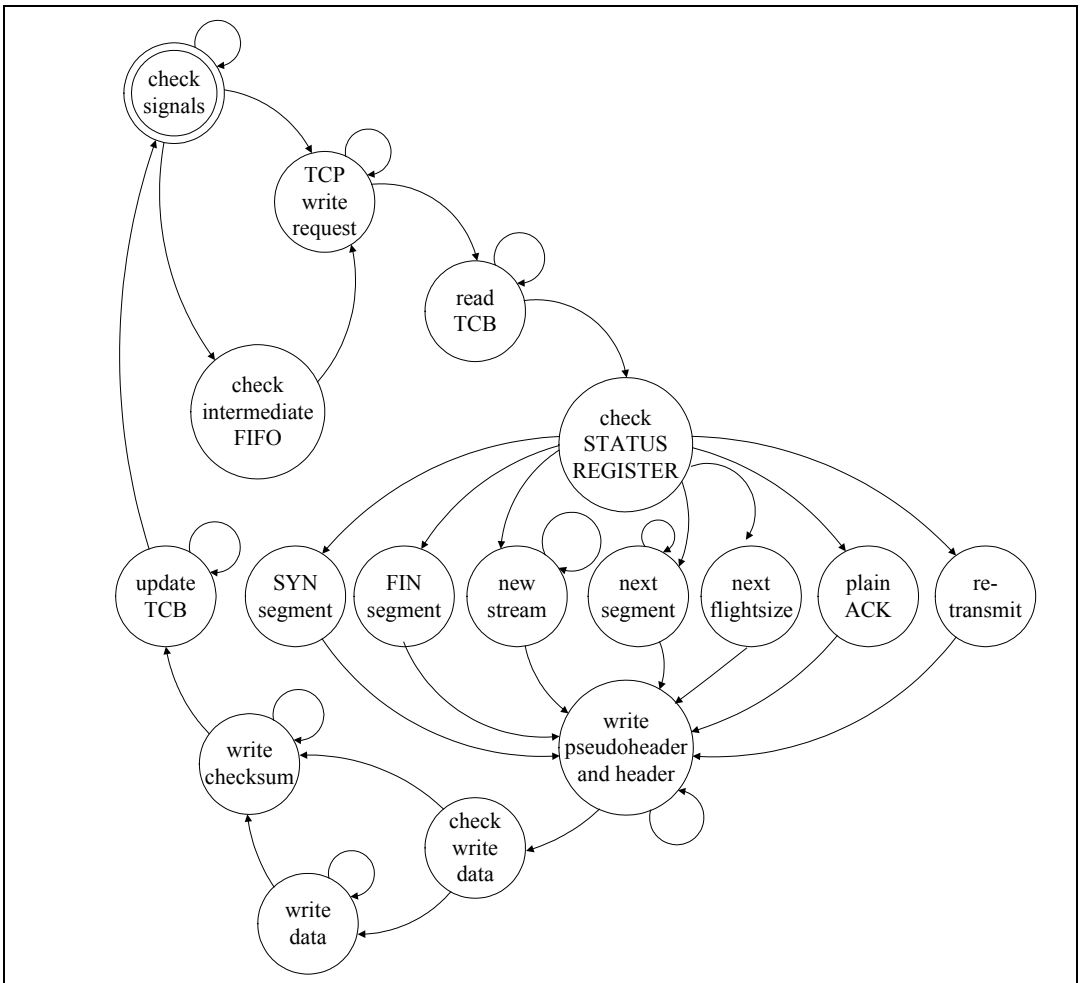
Το υποσύστημα αποστολής τμημάτων TCP αποτελείται από ένα FSM το οποίο ασχολείται με την ετοιμασία και αποστολή αυτών των τμημάτων. Τα τμήματα που αποστέλλονται είναι είτε για τη δημιουργία σύνδεσης (SYN), είτε για το κλείσιμο σύνδεσης (FIN), είτε με δεδομένα, είτε μόνο για επιβεβαίωση. Η απόφαση για τη δημιουργία του κατάλληλου τμήματος λαμβάνεται από το FSM από τις τιμές κάποιων σημάτων που προέρχονται από το υποσύστημα Instruction Decode, ή από κάποιες εντολές που υπάρχουν στην ενδιάμεση FIFO. Στο *Σχήμα 5.9* δίδεται το διάγραμμα καταστάσεων του FSM και στη συνέχεια γίνεται λεπτομερέστερη ανάλυση της λειτουργίας του.

Στην κατάσταση check signals το FSM ελέγχει αν κάποιο από τα σήματα ενεργοποίησης έχει πάρει τιμή από το υποσύστημα Instruction Decode. Ακόμα ελέγχει αν υπάρχει κάποια εντολή στην ενδιάμεση FIFO. Αν διαπιστωθεί ότι υπάρχει εντολή τότε το FSM μεταβαίνει στην κατάσταση εξαγωγής της εντολής από τη FIFO και προχωράει στην εκτέλεσή της.

Ανάλογα με το σήμα ενεργοποίησης ή την εντολή από την ενδιάμεση FIFO, δημιουργείται και αποθηκεύεται μια τιμή σε ένα STATUS καταχωρητή που οδηγεί τις μεταβάσεις του FSM. Υπάρχουν όμως κάποιες κοινές καταστάσεις από τις οποίες διέρχεται πάντα το FSM, όπως η κατάσταση κατά την οποία γίνεται αίτηση στο υποσύστημα CONTROL για πρόσβαση στη μνήμη μετάδοσης TxRAM. Στη συνέχεια γίνεται ανάγνωση του TCB από το socket table. Η διεύθυνση του TCB που διαβάζεται προκύπτει από επιλογή μεταξύ καταχωρητών σύμφωνα με την τιμή που έχει αποκτήσει ο STATUS καταχωρητής ή δίδεται απευθείας μέσω της ενδιάμεσης FIFO ώς μέρος της εντολής. Η πληροφορία που υπάρχει στο TCB χρησιμοποιείται για να δομηθεί σωστά το τμήμα TCP, καθώς από εκεί λαμβάνονται η διεύθυνση IP, οι τιμές των source και destination port number, οι τιμές τωνAcknowledgment και Sequence number αλλά και για να λειτουργήσει σωστά ο έλεγχος συμφόρησης του δικτύου κατά την διάρκεια της μετάδοσης δεδομένων.

Στη συνέχεια το FSM μεταβαίνει στην κατάσταση ελέγχου του STATUS καταχωρητή για να οδηγηθεί έπειτα στην κατάλληλη κατάσταση όπου πρέπει να δράσει διαφορετικά. Όπως φαίνεται από το διάγραμμα του σχήματος οι καταστάσεις που μπορεί να βρεθεί είναι εξής:

- Η κατάσταση SYN segment, όπου δημιουργείται ένα τμήμα για το συγχρονισμό δύο απομακρυσμένων sockets και τη δημιουργία σύνδεσης μεταξύ δύο εφαρμογών. Η δημιουργία τμήματος SYN ενεργοποιείται είτε με την εντολή open active socket, είτε με τη λήψη ενός αντίστοιχου τμήματος ώστε να ολοκληρωθεί η διαδικασία χειραψίας για την καθιέρωση της σύνδεσης.
- Η κατάσταση FIN segment, κατά την οποία δημιουργείται ένα τμήμα για τον τερματισμό μιας σύνδεσης μεταξύ δύο εφαρμογών. Η δημιουργία ενός τέτοιου τμήματος ενεργοποιείται με την εντολή close_active_socket.
- Η κατάσταση new stream όπου ξεκινάει η μετάδοση μιας καινούριας ροής δεδομένων, όπως έχει δοθεί από το επίπεδο εφαρμογών με την εντολή send. Επίσης αρχικοποιούνται οι τιμές κάποιων καταχωρητών που διατηρούν πληροφορία για τη ροή που αποστέλλεται στο δίκτυο.
- Η κατάσταση next segment κατά την οποία η σχεδίαση στέλνει το επόμενο τμήμα TCP το οποίο ανήκει στο ίδιο παράθυρο δεδομένων. Στην κατάσταση αυτή αντίθετα με τις άλλες δεν ερχόμαστε μετά από την αποκωδικοποίηση κάποιας εντολής είτε από το επίπεδο εφαρμογών είτε από την ενδιάμεση FIFO αλλά μετά από σήμα που μας δίνει το υποσύστημα control το οποίο σημαίνει ότι έχει ολοκληρωθεί η μετάδοση του προηγούμενου τμήματος προς το δίκτυο και το TCP έχει πάλι τον έλεγχο της μνήμης μετάδοσης και μπορεί να προχωρήσει στην δημιουργία του επόμενου τμήματος σε αυτήν. Συνεχίζουμε στην κατάσταση next segment έως ότου εξαντλήσουμε την τιμή του παραθύρου συμφόρησης ή της τιμής του παραθύρου του δέκτη.
- Η κατάσταση new flightsize κατά την οποία μεταδίδεται καινούριο τμήμα της ροής που ξεκίνησε να αποστέλλεται. Η αποστολή νέου παραθύρου δεδομένων πραγματοποιείται μετά τη λήψη επιβεβαίωσης η οποία επιβεβαίωνε επιτυχώς όλα τα δεδομένα τα οποία είχαμε στείλει με το προηγούμενο παράθυρο δεδομένων, η οποία γνωστοποιείται στο υποσύστημα αποστολής από το υποσύστημα λήψης μέσω της ενδιάμεσης FIFO.
- Η κατάσταση plain ACK όπου δημιουργείται ένα τμήμα το οποίο περιέχει μόνο μια επιβεβαίωση για δεδομένα που λήφθηκαν από το υποσύστημα TCP, και έχουν αποθηκευθεί στη μνήμη λήψης. Η επιβεβαίωση αυτής της μορφής δημιουργείται με την ενεργοποίηση από την αντίστοιχη εντολή της ενδιάμεσης FIFO. Μια τέτοια εντολή εισάγεται όταν λαμβάνονται δεδομένα από κάποια σύνδεση για την οποία δεν αποστέλλονται δεδομένα ώστε να προσαρτηθεί η επιβεβαίωση σε αυτά (piggyback). Έτσι σύμφωνα με τις οδηγίες του RFC 793 αποστέλλεται μεμονωμένη επιβεβαίωση στον αποστολέα των δεδομένων.



Σχήμα 5.9 Διάγραμμα καταστάσεων του υποσυστήματος αποστολής τμημάτων TCP.

- Η κατάσταση retransmit όπου πραγματοποιείται αναμετάδοση κάποιου τμήματος για το οποίο προέκυψε timeout χωρίς να ληφθεί επιβεβαίωση. Για κάθε καινούριο τμήμα με δεδομένα ή τμήμα FIN που αποστέλλεται ενεργοποιείται ένας timer που μετράει κάποια χρονική περίοδο σε milliseconds. Αν προκύψει timeout σε κάποιο από αυτούς τους timers εισάγεται μια εντολή στην ενδιάμεση FIFO που ενεργοποιεί την αναμετάδοση του αντίστοιχου τμήματος το οποίο είναι πάντα το επόμενο μετά το τελευταίο για το οποίο έχουμε λάβει επιβεβαίωση. Έτσι λοιπόν προχωρούμε στην εκ νέου μετάδοση του χαμένου τμήματος αλλά συνεχίζουμε κατόπιν και στην επαναμετάδοση όλων των άλλων τμημάτων τα οποία ακολουθούν το χαμένο τμήμα και έχουμε ήδη αποστείλει καθώς θεωρούμε ότι ο δέκτης και να τα έχει λάβει σωστά δεν τα έχει αποθηκεύσει. Κάνουμε αυτήν την θεώρηση διότι σε αντίθετη περίπτωση η υλοποίηση του ελέγχου συμφόρησης του δικτύου ο οποίος είναι και υπεύθυνος για τις μεταδόσεις και λήψεις τμημάτων TCP

θα γινόταν υπερβολικά περίπλοκος. Επίσης με την θεώρηση αυτή δεν δημιουργούμε κάποιο πρόβλημα στην ορθή λειτουργία του TCP είτε του αποστολέα είτε του δέκτη αφού και να πάρει τα τμήματα διπλά ο δέκτης απλά θα τα απορρίψει. Από αυτήν την διαδικασία χάνουμε μόνο στο throughput της σχεδίασής μας αφού υπάρχει περίπτωση να αναμεταδίδουμε τμήματα τα οποία ο δέκτης έχει ήδη λάβει σωστά. Αυτό όμως είναι κάτι το οποίο μπορούμε να το δεχτούμε μιας και σε σχέση με την δεύτερη γενιά της σχεδίασης το throughput έχει αυξηθεί κατακόρυφα.

5.4.4 Το υποσύστημα των timers και της ενδιάμεσης FIFO

Το υποσύστημα που περιγράφεται σε αυτή την ενότητα αφορά τους timers του συστήματος καθώς και την λειτουργικότητα της ενδιάμεσης FIFO. Σε κάθε σύνδεση αντιστοιχεί ένας timer και υπάρχει και μία FIFO πλάτους 7 bit και 16 θέσεων.

Ο timer λοιπόν είναι ο ακρογωνιαίος λίθος της λειτουργίας του έλεγχου συμφόρησης κα κατ'επέκταση του TCP γενικότερα. Το TCP χρησιμοποιεί μια γκάμα από timers οι οποίοι αναλαμβάνουν έναν ξεχωριστό ρόλο ο καθένας και λύνουν και συγκεκριμένα προβλήματα τα οποία προκύπτουν κατά την διάρκεια της λειτουργίας του. Οι timers αυτοί είναι : ο retransmission timer, ο quiet timer, ο persistence timer και ο Idle timer.

Ο retransmission timer είναι αυτός που διαχειρίζεται τα timeouts τα οποία μπορεί να προκύψουν όταν ξεπερνιέται ένας δοσμένος χρόνος μέσα στον οποίο έπρεπε να είχε έρθει η επιβεβαίωση για κάποιο τμήμα TCP το οποίο έχουμε προηγουμένως αποστείλει. Το χρονικό διάστημα για το timeout που αναφέραμε παρέχεται από τον ίδιο τον timer από παρατηρήσεις προηγούμενων μεταδόσεων και αποθηκεύεται στο TCB της σύνδεσης.

Αφού κλείσει κάποια σύνδεση TCP είναι δυνατόν για κάποια τμήματα να είναι ακόμα στο δίκτυο και όταν φτάσουν θα βρούνε το port για το οποίο προορίζονταν κλειστό. Έτσι ο quiet timer επιτελεί αυτόν ακριβώς τον σκοπό μετράει ένα χρονικό διάστημα κατά το οποίο μια σύνδεση που μόλις έχει κλείσει μπορεί να ξανανοίξει για λίγο και να δεχθεί τυχόν τμήματα τα οποία περιπλανώνται ακόμα στο δίκτυο. Εμείς δεν έχουμε υλοποιήσει τον συγκεκριμένο timer μιας και από την σχεδίασή μας δεν υποστηρίζεται το κομμάτι αυτό του FSM του TCP το οποίο κάνει αυτήν την δουλειά. Αντ'αυτού θέτουμε τον περιορισμό να μην προσπαθούμε να ανοίγουμε αμέσως συνδέσεις που μόλις έκλεισαν για κάποιο χρονικό διάστημα.

Ο persistence timer ενεργοποιείται στην σπάνια περίπτωση όπου το παράθυρο του δέκτη γίνει 0 και το τμήμα το οποίο θα ανανέωνε αυτήν την τιμή με μία μη μηδενική χαθεί κάτι το οποίο θα έφερνε την σύνδεση σε κατάσταση deadlock. Ο persistence timer λοιπόν σε μια τέτοια περίπτωση στέλνει τμήματα του ενός byte σε προκαθορισμένα διαστήματα για να δει αν ο δέκτης έχει ακόμα μηδενικό παράθυρο. Ο δέκτης από την άλλη απαντάει σε αυτά τα τμήματα με την τιμή του παραθύρου του έως ότου η μετάδοση ξαναρχίσει πάλι κανονικά.

Ο idle timer τίθεται σε λειτουργία όταν η σύνδεση δεν στέλνει δεδομένα. Όταν λοιπόν τεθεί σε λειτουργία αυτός ο timer και κάνει timeout τότε θεωρούμε ότι οι συνθήκες του δικτύου έχουν πλέον αλλάξει οπότε και πρέπει να αλλάξουμε τις τιμές των congestion window και ssthresh πριν ξεκινήσουμε κάποια άλλη μετάδοση. Ο idle timer είναι υλοποιημένος στην σχεδίαση αλλά δεν υπάρχει εξωτερικός έλεγχος από την σχεδίαση ο οποίος θα τον ενεργοποιεί. Αντ'αυτού έχουμε λύσει αυτό το πρόβλημα ζητώντας από την εφαρμογή που χρησιμοποιεί το TCP με κάθε εντολή που δίνει για αποστολή νέων δεδομένων να δίνει και τις τιμές των congestion window και ssthresh μαζί με τα υπόλοιπα ορίσματα.

Ο timer λοιπόν που υλοποιήσαμε λειτουργεί με modes τα οποία αντιστοιχούν σε timer για SYN που κάνει timeout σε προκαθορισμένη τιμή και κατόπιν στέλνεται μήνυμα λάθους στο επίπεδο εφαρμογών, retransmission timer ο οποίος κάνει timeout σε χρόνο που του παρέχεται από το TCP, σαν persistence timer και σαν idle timer που όπως είπαμε όμως δεν υποστηρίζεται από την σχεδίαση. Στον *πίνακα 5.4* βλέπουμε τα Modes αυτά και την αντίστοιχη λειτουργία του timer.

MODE	**FUNCTION**
00	Syn timer
01	retransmission timer
10	persistence timer
11	idle timer

Πίνακας 5.4 Modes και Functions του timer

Για τα τμήματα FIN και δεδομένων μόλις προκύψει timeout από τους αντίστοιχους timers, δημιουργείται μια εντολή που τοποθετείται στην ενδιάμεση FIFO που εκτελείται από το υποσύστημα αποστολής και αναμεταδίδεται το τμήμα που δεν επιβεβαιώθηκε. Για κάθε timeout από τους δύο αυτούς timers αυξάνει την τιμή του ένας μετρητής, ο οποίος όταν φτάσει σε μια προκαθορισμένη και σταθερή τιμή δεν επιτρέπει επανάληψη της αναμετάδοσης, καθώς θεωρείται αποτυχημένη όλη η προσπάθεια μετάδοσης. Στην περίπτωση των τμημάτων με δεδομένα

επιστρέφεται κωδικός λάθους στο επίπεδο εφαρμογών δείχνοντας ότι απέτυχε η αποστολή τους. Στην περίπτωση των τμημάτων FIN δεν επιστρέφεται κωδικός λάθους καθώς δεν υπάρχουν κωδικοί επιτυχίας και σφάλματος για την εντολή close(socketid). Σε όλες τις περιπτώσεις αποτυχημένης αποστολής τμήματος η κατάσταση της σύνδεσης μεταβαίνει στην τιμή CLOSED κάνοντας διαθέσιμο το TCB που καταλαμβάνει η σύνδεση.

Εκτός από τις εντολές για timeout και αναμετάδοση, η ενδιάμεση FIFO μεταφέρει και εντολές για αποστολή απλής επιβεβαίωσης, ή εντολή για να προχωρήσει η αποστολή της τρέχουσας ροής δεδομένων καθώς λήφθηκε επιβεβαίωση. Πιο συγκεκριμένα αν ληφθεί τμήμα TCP στη σύνδεση του οποίο δεν αποστέλλονται δεδομένα, τότε πρέπει να σταλεί ένα τμήμα που περιέχει απλά μια επιβεβαίωση για το τμήμα που λήφθηκε. Αντίστοιχα, αν ληφθεί επιβεβαίωση για το τμήμα που ανήκει στην τρέχουσα ροή που μεταδίδεται από το TCP, τότε πρέπει να ειδοποιηθεί το υποσύστημα αποστολής ώστε να προχωρήσει στη αποστολή του επόμενου παραθύρου της ροής. Για κάθε μια από τις περιπτώσεις αυτές το υποσύστημα των timers και της ενδιάμεσης FIFO δέχεται κάποια σήματα από το υποσύστημα λήψης και δημιουργεί μια εντολή που την εισάγει στη FIFO. Στον *Πίνακα 5.5* φαίνεται η μορφή των εντολών με μια συνοπτική περιγραφή.

FIFO data (7..5)	**FIFO data (4..0)**	**Περιγραφή**
“100”	Socketid	Timeout με mode 00
“101’	Socketid	Timeout με mode 01
“110”	Socketid	Timeout με mode 10
“111”	Socketid	Timeout με mode 11
“010”	Socketid	Μετάδοση επόμενου Flightsize
“001”	Socketid	Μετάδοση απλής επιβεβαίωσης

Πίνακας 5.5 Μορφή εντολών της ενδιάμεσης FIFO.

5.4.5 Το υποσύστημα λήψης πλαισίων TCP

Το υποσύστημα αυτής της ενότητας ασχολείται με τη διαχείριση των τμημάτων που εισέρχονται στη σχεδίαση και ασκεί τους ελέγχους που είναι απαραίτητοι για να ληφθούν αποφάσεις τόσο για τα δεδομένα όσο και για την πληροφορία της σύνδεσης στην οποία ανήκουν. Αποτελείται ένα υποσύστημα που λαμβάνει την επικεφαλίδα και εισάγει τα δεδομένα στη FIFO απόζευξης, ένα υποσύστημα που εξάγει τα δεδομένα από τη FIFO είτε αποθηκεύοντάς τα στη μνήμη λήψης είτε απλά αδειάζοντας τη FIFO, ένα υποσύστημα που ασκεί τους ελέγχους στην πληροφορία της σύνδεσης και ένα καταχωρητή η τιμή του οποίου καθοδηγεί τους ελέγχους αυτούς.

Το υποσύστημα που λαμβάνει την επικεφαλίδα και εισάγει τα δεδομένα στη FIFO απόζευξης υλοποιείται με ένα FSM. Μόλις διαπιστωθεί ότι το υποσύστημα IP λαμβάνει πακέτο που περιέχει τμήμα TCP ενεργοποιείται αυτό το FSM το οποίο φροντίζει να διαβάσει την επικεφαλίδα του τμήματος και να αποθηκεύσει τα επιμέρους πεδία σε καταχωρητές. Αυτό το FSM ενεργοποιεί επίσης το υποσύστημα που ασκεί τους ελέγχους, ώστε αυτό να προσπελάσει το κατάλληλο TCB και να διαβαστούν από εκεί οι τιμές που διατηρούν οι διάφορες μεταβλητές της σύνδεσης. Σε αυτή τη φάση ακόμα αποθηκεύεται η τιμή της διεύθυνσης IP του απομακρυσμένου σταθμού και υπολογίζεται το μέγεθος των δεδομένων που φέρει το τμήμα, με τη χρήση της τιμής που παρέχεται από το υποσύστημα λήψης πακέτων IP. Ο υπολογισμός επιτυγχάνεται με την ως εξής:

$$TCP\ data\ length = Segment\ Length - (Data\ Offset * 4),$$

όπου *Segment Length* είναι η τιμή που υπολογίζεται από το υποσύστημα IP-RECV και *Data Offset* η τιμή (λέξεις 32-bit) που υπάρχει στο αντίστοιχο πεδίο της επικεφαλίδας. Η τιμή του μεγέθους των δεδομένων χρησιμοποιείται αρχικά για την εισαγωγή τους στη FIFO απόζευξης.

Ο καταχωρητής (RECEIVE_CONTROL_REGISTER) που οδηγεί το υποσύστημα ελέγχου για τη δράση του, διαμορφώνει την τιμή του με βάση κάποιες λογικές συναρτήσεις. Οι είσοδοι των συναρτήσεων αυτών προέρχονται από συνδυασμό κάποιων τιμών που υπάρχουν σε καταχωρητές και αναφέρονται τόσο σε πληροφορία που λαμβάνεται από την επικεφαλίδα του τμήματος TCP που εισέρχεται στο σύστημα, όσο και σε πληροφορία που υπάρχει στο TCB της σύνδεσης που φέρει το τμήμα TCP. Οι λογικές συναρτήσεις χρησιμοποιώντας τη διαθέσιμη πληροφορία καλύπτουν όλες τις περιπτώσεις που μπορούν να προκύψουν από τη λήψη ενός τμήματος, ώστε αυτό να αντιμετωπιστεί με τον κατάλληλο τρόπο. Στη συνέχεια παρατίθενται οι συναρτήσεις αυτές.

RECEIVE_CONTROL_REGISTER (0) = SYN **and not** ACK and LISTEN **and** (target TCB is CLOSED)
RECEIVE_CONTROL_REGISTER (1) = SYN **and not** ACK **and** SYN-SENT
RECEIVE_CONTROL_REGISTER (2) = SYN **and** ACK **and** SYN-SENT
RECEIVE_CONTROL_REGISTER (3) = **not** SYN **and** ACK **and** SYN-RCVD
RECEIVE_CONTROL_REGISTER (4) = (FIN **nor** PSH) **and** ACK **and** (ESTABLISHED **or** FIN-WAIT-2)
RECEIVE_CONTROL_REGISTER (5) = **not** FIN **and** PSH **and** ACK **and** (ESTABLISHED **or** FIN-WAIT-2)
RECEIVE_CONTROL_REGISTER (6) = **not** FIN **and not** SYN **and not** RST **and** ESTABLISHED
RECEIVE_CONTROL_REGISTER (7) = FIN **and** ACK **and** ESTABLISHED
RECEIVE_CONTROL_REGISTER (8) = **not** FIN **and** ACK **and** FIN-WAIT-1
RECEIVE_CONTROL_REGISTER (9) = FIN **and** ACK **and** FIN-WAIT-1
RECEIVE_CONTROL_REGISTER (10) = FIN **and** ACK **and** FIN-WAIT-2
RECEIVE_CONTROL_REGISTER (11) = **not** FIN **and** ACK **and** CLOSING
RECEIVE_CONTROL_REGISTER (12) = **not** FIN **and** ACK **and** LAST-ACK
RECEIVE_CONTROL_REGISTER (13) = RST

Το υποσύστημα που οδηγείται από τον παραπάνω καταχωρητή έχει ως στόχο να ασκήσει όλους τους απαραίτητους ελέγχους για να μπορέσει να διαχειριστεί το εισερχόμενο τμήμα όπως πρέπει. Ενδεικτικά αναφέρεται ότι ασκείται έλεγχος στις τιμές που έχουν τα πεδία Acknowledgment και Sequence number του TCB για να διαπιστωθεί ότι το ληφθέν τμήμα είναι αυτό που αναμένεται από τη συγκεκριμένη σύνδεση. Επίσης ελέγχεται αν αυτή η σύνδεση αναμένει δεδομένα και αν υπάρχει διαθέσιμος χώρος στη μνήμη λήψης για να αποθηκευθούν αυτά. Ελέγχεται αν το τμήμα που λαμβάνεται το φέρει κάποια σύνδεση που ταυτόχρονα αποστέλλει δεδομένα. Αν συμβαίνει αυτό τότε ειδοποιείται το υποσύστημα αποστολής να προχωρήσει στην αποστολή του επόμενου τμήματος (με εντολή από την ενδιάμεση FIFO). Αν δεν πρόκειται για σύνδεση που αποστέλλει παράλληλα δεδομένα ειδοποιείται το υποσύστημα αποστολής να μεταδώσει μια απλή επιβεβαίωση (επίσης με εντολή από την ενδιάμεση FIFO). Ακόμα το υποσύστημα ελέγχου αναλαμβάνει τη διαχείριση των τμημάτων SYN, FIN, RST για την καθιέρωση συνδέσεων μεταξύ δύο εφαρμογών, το κλείσιμο συνδέσεων και την αρχικοποίησή τους σε περίπτωση σφάλματος, ελέγχοντας πάντα τις παραμέτρους που χαρακτηρίζουν τις συνδέσεις αυτές. Τέλος ενημερώνει τις τιμές των παραμέτρων και τις

αποθηκεύει στο κατάλληλο TCB, ενώ επιστρέφει απαντήσεις στο επίπεδο εφαρμογών αν χρειάζεται. Επίσης το υποσύστημα αυτό αναλαμβάνει να σταματήσει ή να ξαναξεκινήσει τον κατάλληλο timer ανάλογα με το αν επιβεβαιώθηκαν όλα τα πακέτα του προηγούμενου παραθύρου ή αν υπάρχουν και άλλα ανεπιβεβαίωτα.

Το υποσύστημα εξαγωγής δεδομένων από τη FIFO απόζευξης ενεργοποιείται από το υποσύστημα ελέγχου και αναλαμβάνει να αποθηκεύσει τα δεδομένα στη μνήμη λήψης στη θέση που υποδεικνύεται. Αν τα δεδομένα του εισερχόμενου τμήματος απορριφθούν φροντίζει να τα βγάλει από τη FIFO ώστε να μην παραμείνουν και «μολύνουν» τα δεδομένα του επόμενου τμήματος που θα ληφθεί από το σύστημα.

Κεφάλαιο 6ο :
Δοκιμή και Πειράματα

6.1 Προσομοιώσεις σχεδιάσεων Hardware

Οι προσομοιώσεις των σχεδιάσεων hardware αποτελούν απαραίτητη διαδικασία για την κατασκευή συστημάτων που λειτουργούν σωστά. Δίνουν τη δυνατότητα να διαπιστωθεί ότι μια σχεδίαση λειτουργεί σύμφωνα με τις προδιαγραφές που έχουν τεθεί αλλά και να πιστοποιηθεί ότι η απεικόνιση σε κάποια τεχνολογία διατηρεί την αναμενόμενη συμπεριφορά της σχεδίασης. Διακρίνονται τα εξής είδη προσομοιώσεων μιας σχεδίασης hardware.

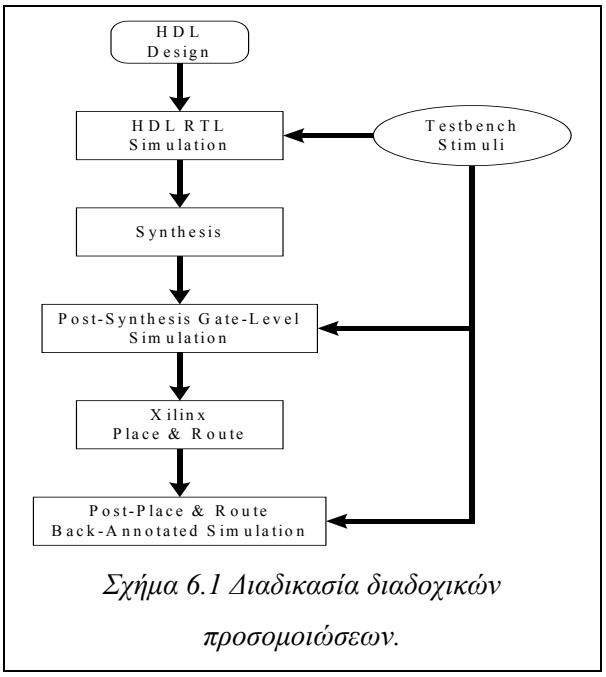
- Προσομοίωση Functional. Η προσομοίωση αυτή γίνεται στο επίπεδο του πηγαίου κώδικα της γλώσσας HDL που γράφει ο κάθε σχεδιαστής. Σκοπός της είναι να δείξει ότι η σχεδίαση λειτουργεί «λογικά». Σε υψηλό επίπεδο ενδιαφέρει να φανεί ότι η σχεδίαση πραγματοποιεί αυτά που περιγράφει ο σχεδιαστής, και σε πιο χαμηλό επίπεδο ενδιαφέρει να φανεί ότι τα σήματα και οι είσοδοι/έξοδοι των υποσυστημάτων έχουν τις τιμές που πρέπει στον ανάλογο κύκλο του ρολογιού του συστήματος. Στην περίπτωση της προσομοίωσης Functional δεν λαμβάνονται υπόψη οι χρονικές καθυστερήσεις που υπάρχουν στα πραγματικά κυκλώματα, καθώς σε αυτή τη φάση ανακαλύπτονται μόνο τα λογικά λάθη που περιέχονται στη γλώσσα περιγραφής της σχεδίασης.
- Προσομοίωση Post Synthesis. Η προσομοίωση αυτή γίνεται αφού περάσει ο πηγαίος κώδικας από τη διαδικασία Synthesis, η οποία τον μεταφράζει σε κύκλωμα με λογικές πύλες, πολυπλέκτες και flip flops. Στην εργασία αυτή, για τη διαδικασία Synthesis χρησιμοποιήθηκε το εργαλείο FPGA Express™ X.E. 3.6.1 της εταιρείας Synopsys, το οποίο περιλαμβάνεται στο περιβάλλον ISE. Σε αυτού του είδους την προσομοίωση είναι δυνατόν να συμπεριληφθεί χρονική καθυστέρηση, που απεικονίζει την καθυστέρηση που εισάγουν τα στοιχεία των κυκλωμάτων και το βάθος των επιπέδων λογικής. Δεν υπάρχει ωστόσο γνώση για την καθυστέρηση μετάδοσης των σημάτων μέσα από τους αγωγούς ούτε γνώση για τη σχετική θέση των στοιχείων του κυκλώματος πάνω στο ολοκληρωμένο κύκλωμα. Σκοπός αυτής της προσομοίωσης είναι να δείξει στο σχεδιαστή ότι η σχεδίαση μεταφράζεται σωστά σε στοιχεία κυκλώματος και να δώσει μια αρχική

πληροφορία για το critical path της σχεδίασης, όσον αφορά τα επίπεδα λογικής που χρησιμοποιούνται.

- Προσομοίωση Post Place & Route. Μετά τη διαδικασία Synthesis λαμβάνει χώρα η διαδικασία Place & Route που εφαρμόζεται από το περιβάλλον ανάπτυξης (στη συγκεκριμένη περίπτωση από το ISE 4.2). Κατά τη διαδικασία αυτή ουσιαστικά γίνεται απεικόνιση των στοιχείων των κυκλωμάτων στους διαθέσιμους πόρους του ολοκληρωμένου κυκλώματος που επιλέγεται. Η προσομοίωση σε αυτό το επίπεδο αναφέρεται ειδικά στη συγκεκριμένη υλοποίηση, στο συγκεκριμένο FPGA και ονομάζεται Back Annotation. Σε αυτή την περίπτωση περιλαμβάνεται όλη η πληροφορία τόσο για τα στοιχεία του κυκλώματος όσο και για τις καθυστερήσεις που εισάγονται από τη μετάδοση των σημάτων στους αγωγούς και τη σχετική θέση των στοιχείων στο ολοκληρωμένο κύκλωμα. Η προσομοίωση αυτή πλησιάζει πάρα πολύ τα δεδομένα της πραγματικής κατασκευής και τα αποτελέσματα αυτής διαβεβαιώνουν ότι η απεικονιζόμενη σχεδίαση έχει ~ 100% πιθανότητα να δουλέψει σε πραγματικό hardware.

Η προσομοίωση μιας σχεδίασης προϋποθέτει την παροχή τιμών στις εισόδους του συστήματος και την παρακολούθηση των εξόδων του και κάποιων εσωτερικών σημάτων, για την επαλήθευση της σωστής λειτουργίας της.

Η παροχή τιμών επιτυγχάνεται με τη σχεδίαση ενός test bench το οποίο έχει σήματα που συνδέονται στις εισόδους του συστήματος. Το test bench γράφει τις τιμές σε αυτά τα σήματα είτε γεννώντας αυτές on the fly με κάποιο τρόπο (τυχαίο, ή με κάποια συνάρτηση), είτε λαμβάνοντας αυτές από κάποιο αρχείο με διανύσματα, που περιέχουν προεπιλεγμένα δεδομένα. Για τη δοκιμή της σχεδίασης που υλοποιεί TCP/IP stack πρωτοκόλλων χρησιμοποιείται ένα αρχείο με πλαίσια Ethernet. Τα πλαίσια Ethernet περιέχουν προεπιλεγμένα δεδομένα ώστε να δοκιμαστούν και να επαληθευθούν όλα τα υποσυστήματα της σχεδίασης. Γι' αυτό το λόγο θεωρήθηκε ότι αφενός η χρήση τυχαίων τιμών δεν μπορεί να προσφέρει τη δυνατότητα επαλήθευσης όλων των υποσυστημάτων και αφετέρου η δημιουργία συνάρτησης απαιτεί μεγάλο κόστος υλοποίησης. Στο *Σχήμα 6.1* δίνεται ένα διάγραμμα που περιγράφει τη διαδικασία των διαδοχικών προσομοιώσεων.



Σχήμα 6.1 Διαδικασία διαδοχικών προσομοιώσεων.

Η παρακολούθηση των εξόδων του συστήματος αποτελεί το δεύτερο σκέλος της προσομοίωσης, αφού μέσω αυτής διαπιστώνεται η σωστή λειτουργία του. Υπάρχουν δύο βασικοί τρόποι για την παρακολούθηση των εξόδων. Ο ένας είναι η χρήση κυματομορφών που δείχνουν τις τιμές που αποκτούν τα σήματα κατά τη διάρκεια της προσομοίωσης, σε σχέση με την κυματομορφή του ρολογιού του συστήματος. Η δυνατότητα προβολής κυματομορφών προσφέρεται από τα εργαλεία προσομοίωσης και το σημαντικό πλεονέκτημά που παρέχουν αυτές, είναι η ευχέρεια παρατήρησης των σημάτων στο επίπεδο των κύκλων του ρολογιού. Τέτοιου είδους παρατήρηση χρησιμοποιείται για να επιτευχθεί ο συγχρονισμός των υποσυστημάτων που μεταφέρουν ή ανταλλάσσουν πληροφορία.

Ο δεύτερος τρόπος είναι η δημιουργία monitors τα οποία έχουν σήματα που είναι συνδεμένα στις εξόδους του συστήματος. Ο ρόλος των monitors είναι η παρακολούθηση και καταγραφή των τιμών των εξόδων σε κάποιο αρχείο. Τα monitors προσφέρονται για την παρακολούθηση των δεδομένων στις εξόδους του συστήματος σε υψηλότερο επίπεδο και όχι σε επίπεδο κύκλου. Με τη χρήση τους στη συγκεκριμένη υλοποίηση παρατηρείται η αρτιότητα των πλαισίων Ethernet που εξέρχονται από το σύστημα καθώς και η ορθότητα των δεδομένων που μεταφέρουν. Τα monitors μπορούν επίσης να ελέγχουν τις τιμές των εξόδων on the fly, με βάση κάποια συνάρτηση ή έχοντας εκ των προτέρων σε κάποιο αρχείο τα αναμενόμενα αποτελέσματα, όμως για τη δοκιμή της σχεδίασης, απλά καταγράφουν τα δεδομένα σε ένα αρχείο, που ελέγχεται μετά το τέλος της προσομοίωσης.

Η ανάπτυξη του testbench και των monitors έγινε με τη χρήση της γλώσσας VHDL, όπως και η ανάπτυξη όλης της σχεδίασης του IP core. Η διαφορά μεταξύ του πηγαίου κώδικα του IP core και του πηγαίου κώδικα για την υποστήριξη της προσομοίωσης, είναι ότι ο μεν μπορεί να περάσει από τη διαδικασία Synthesis ενώ ο δε, δεν μπορεί, καθώς δεν είναι αναγκαίο. Όπως φαίνεται από το *Σχήμα 6.1*, το testbench από τη στιγμή που αναπτύσσεται, μπορεί να χρησιμοποιηθεί χωρίς καμία αλλαγή σε όλα τα στάδια των προσομοιώσεων, δεδομένου ότι τα σήματα εισόδου και εξόδου του υψηλότερου επιπέδου της σχεδίασης παραμένουν τα ίδια. Ομοίως μπορούν να χρησιμοποιηθούν και τα monitors χωρίς καμιά αλλαγή, δεδομένου ότι παρακολουθούν μόνο εξωτερικά σήματα, αφού τα εσωτερικά μετά τη διαδικασία Synthesis αλλάζουν όνομα.

6.2 Προσομοίωση της σχεδίασης

Η προσομοίωση της σχεδίασης ήταν συνεχής και παράλληλη με την διαδικασία ανάπτυξης της σχεδίασης. Για την προσομοίωση της σχεδίασης εκτός από testbench και monitors χρησιμοποιήσαμε και κάποια do scripts τα οποία προσφέρουν έναν πιο γρήγορο και ευέλικτο τρόπο προσομοίωσης, επίσης κάποιες φορές χειροκίνητα δώσαμε κάποιες τιμές σε σήματα για να ελέγξουμε μικρές αλλαγές που γίνονταν κατά την διάρκεια της ανάπτυξης της σχεδίασης.

Η διαδικασία των προσομοιώσεων κατά την διάρκεια αυτής της διπλωματικής εργασίας μπορούν να χωριστούν σε δύο κατηγορίες. Τις προσομοιώσεις που έγιναν πριν την προσθήκη του ελέγχου συμφόρησης του δικτύου στο πρωτόκολλο TCP και στις προσομοιώσεις οι οποίες έγιναν μετά την προσθήκη του ελέγχου συμφόρησης του δικτύου στο πρωτόκολλο TCP.

Πιο συγκεκριμένα οι περιφερειακές του TCP αλλαγές οι οποίες έγιναν, δοκιμάστηκαν ξεχωριστά και με την χρήση monitors. Πιο συγκεκριμένα μετά από κάθε αλλαγή τρέχαμε το ίδιο testbench το οποίο είχαμε και προηγουμένως τρέξει και στην δεύτερη γενιά της σχεδίασης και ελέγχαμε μετά τα παραγόμενα αρχεία ως προς την ομοιότητά τους. Αν τα αρχεία ήταν όμοια προχωράγαμε στις επόμενες αλλαγές. Σε αντίθετη περίπτωση αν υπήρχαν αλλαγές στα παραγόμενα αρχεία βλέπαμε την χρονική στιγμή κατά την οποία υπήρχε η διαφορά και κατόπιν ανατρέχαμε στις κυματομορφές για περισσότερες λεπτομέρειες ώστε να αντιληφθούμε το πρόβλημα και να το λύσουμε. Πιο συγκεκριμένα οι προσομοιώσεις οι οποίες έγιναν σε αυτήν την φάση περιελάμβαναν τα εξής :

- Έγιναν αιτήσεις ARP από απομακρυσμένο σταθμό προς τη σχεδίαση και αυτή απάντησε σωστά.
- Έγιναν αποστολές ICMP echo request από απομακρυσμένο σταθμό και η σχεδίαση έστειλε την ανάλογη απάντηση ICMP echo reply.
- Έγιναν αιτήσεις ARP από τη σχεδίαση προς απομακρυσμένο σταθμό και μόλις έφτασε η σωστή απάντηση, η σχεδίαση τη δέχτηκε, τη διαχειρίστηκε και ενημέρωσε το υποσύστημα που έκανε την αίτηση να συνεχίσει τη λειτουργία του.
- Έγινε αποστολή και λήψη πακέτων UDP σε sockets που είχαν ανοίξει προηγουμένως και απόρριψη πακέτων UDP που δεν αναφέρονταν σε κάποιο ανοιχτό socket.
- Έγινε δημιουργία και κλείσιμο συνδέσεων TCP μεταξύ της σχεδίασης και απομακρυσμένου σταθμού. Δοκιμάστηκε αφενός η δημιουργία σύνδεσης από τη σχεδίαση και το κλείσιμό της από το απομακρυσμένο σύστημα και αφετέρου η δημιουργία σύνδεσης από απομακρυσμένο σύστημα και το κλείσιμό της από τη σχεδίαση. Στα πλαίσια των δοκιμών αυτών εξετάστηκε η περίπτωση να ξεκινάει η αποστολή δεδομένων, μετά την καθιέρωση σύνδεσης, είτε από τη σχεδίαση είτε από το απομακρυσμένο σύστημα, ενώ στάλθηκαν επίτηδες καθυστερημένες επιβεβαιώσεις για να διαπιστωθεί ότι λειτουργούν σωστά και οι αναμεταδώσεις μετά τα timeouts.

Αυτές οι προσομοιώσεις έγιναν με σκοπό να ελέγξουμε την σωστή λειτουργία των περιφερειακών υποσυστημάτων του TCP τα οποία αλλάξαμε και αυτά είναι ο πίνακας ARP το υποσύστημα control και το MII Interface.

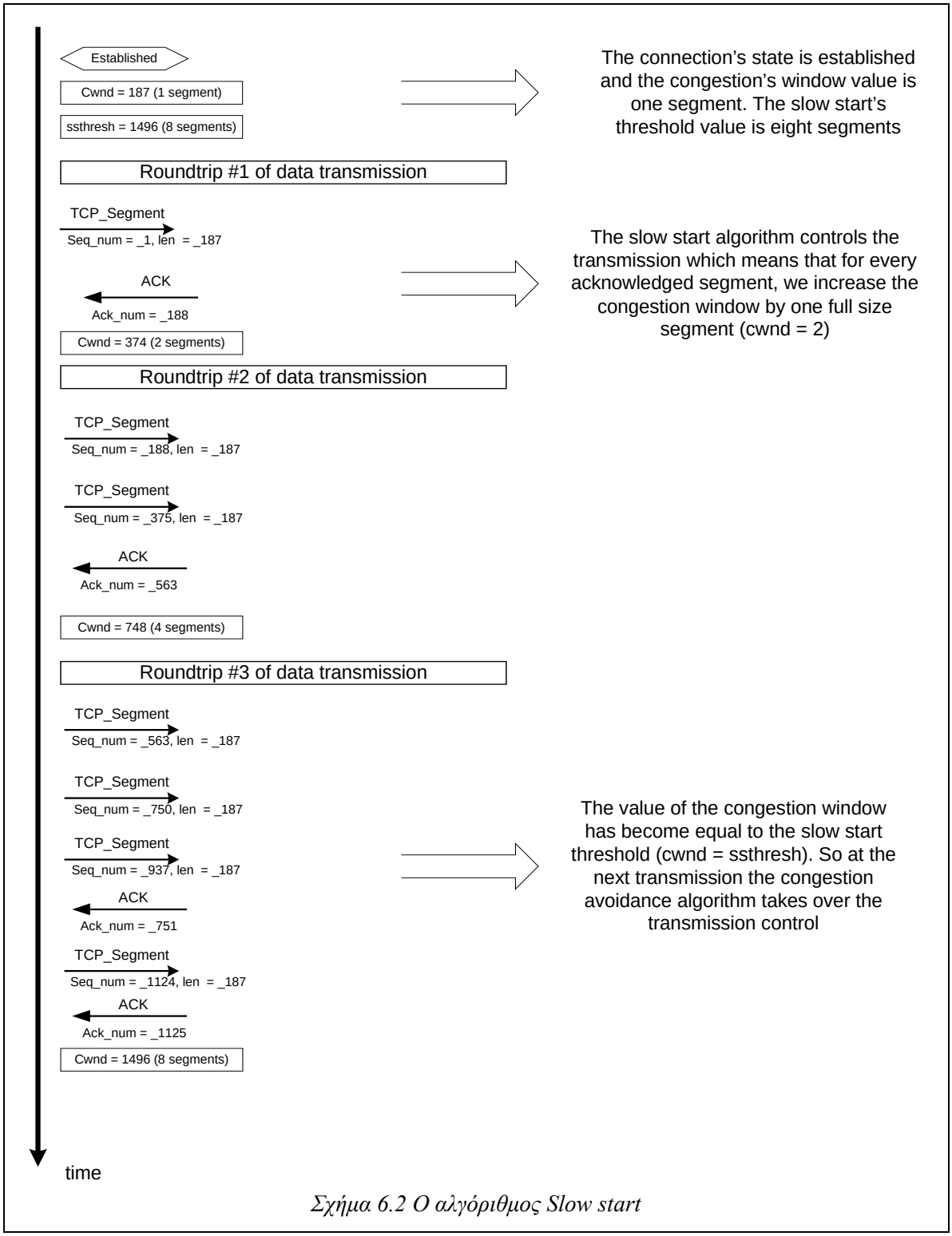
Το υποσύστημα TCP τώρα λόγω του όγκου και της πολυπλοκότητάς του ήταν επιτακτικό να γίνονται προσομοιώσεις καθ'όλη την διάρκεια της ανάπτυξής του και χωρίς αυτό να έχει λάβει ακόμα την τελική του μορφή. Επίσης εδώ πρέπει να αναφέρουμε ότι το υποσύστημα TCP το προσομοιώναμε πλέον μόνο του και όχι μαζί με όλη την άλλη σχεδίαση καταρχήν για οικονομία χρόνου και δεύτερον διότι από τις προσομοιώσεις οι οποίες είχαν γίνει κατά την πρώτη φάση της ανάπτυξης της τρίτης γενιάς της σχεδίασης μπορούσαν να μας εγγυηθούν ότι όλα τα άλλα υποσυστήματα δουλεύουν σωστά. Έτσι και δεδομένου ότι η διεπαφή του υποσυστήματος TCP, με τα υπόλοιπα υποσυστήματα είναι απόλυτα καθορισμένη, αυτό που μας ενδιέφερε ήταν να γίνονται σωστά οι διεργασίες μέσα στο υποσύστημα και τα σήματα προς όλα τα άλλα υποσυστήματα να παίρνουν τις σωστές τιμές και να συμπεριφέρονται έτσι όπως πρέπει.

Οι προσομοιώσεις που έγιναν είχαν ως σκοπό να διασφαλίσουν την σωστή λειτουργία των επιμέρους κομματιών του TCP. Με την εμπειρία την οποία είχαμε αποκτήσει πάνω στο TCP ως πρωτόκολλο διαβάζοντας τις προδιαγραφές του αλλά κυρίως μέσα από την

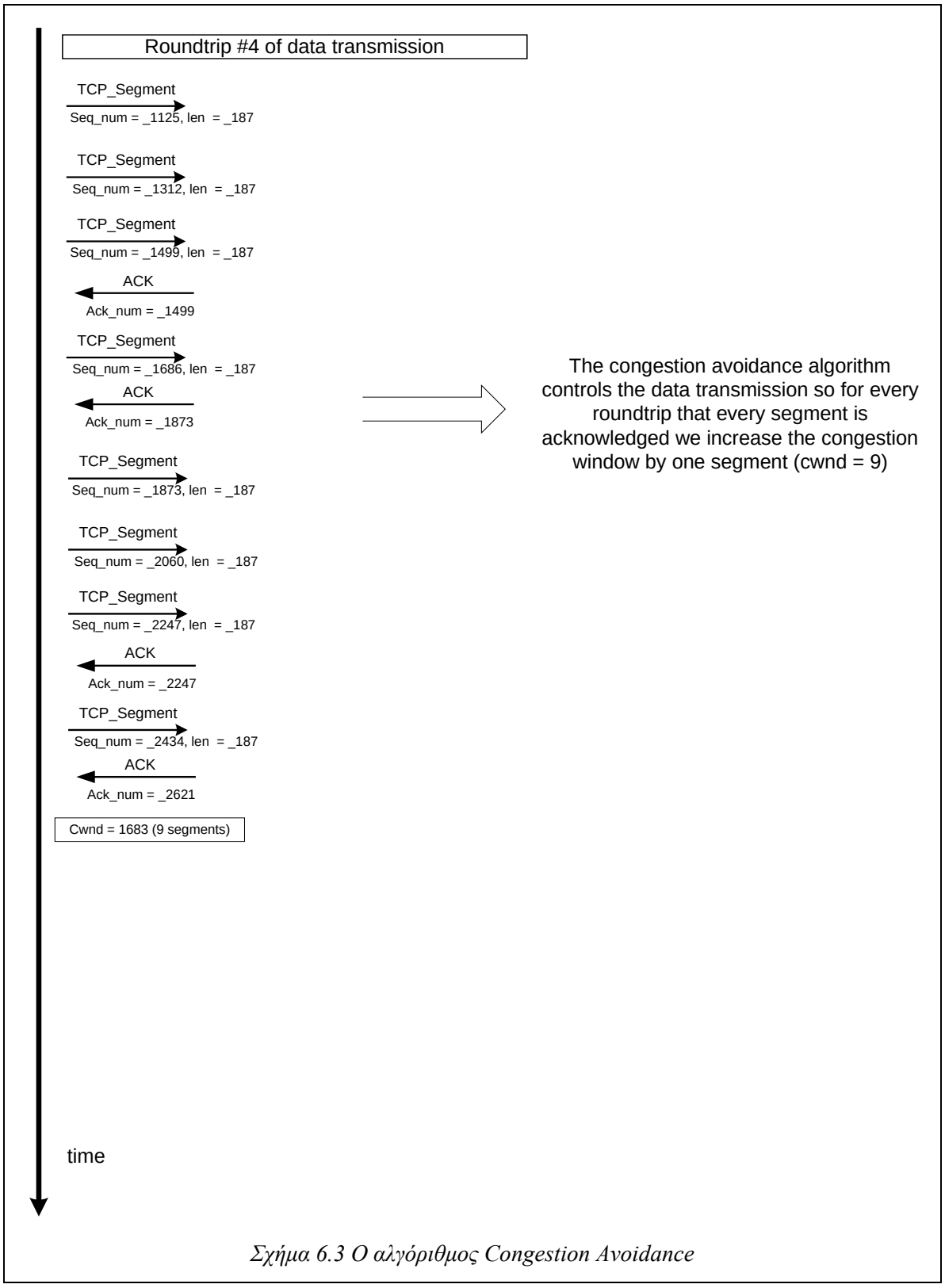
“αποκωδικοποίηση” των κωδίκων τις δεύτερης γενιάς της σχεδίασης και ξέροντας τι αλλαγές έπρεπε να γίνουν έτσι ώστε να μπορέσει η σχεδίαση να υποστηρίξει έλεγχο συμφόρησης είχε γίνει λεπτομερής μελέτη της εργασίας η οποία έπρεπε να γίνει με αποτέλεσμα να μπορούμε να ελέγχουμε την σχεδίαση σε όλα της τα στάδια σε επίπεδο σημάτων, λύνοντας έτσι κατά κάποιο τρόπο τα χέρια μας αφού δεν χρειαζόταν να περιμένουμε την ολοκλήρωσή της για μπορούμε να τρέξουμε τις πρώτες προσομοιώσεις. Πιο αναλυτικά οι προσομοιώσεις οι οποίες έγιναν κάλυψαν τις εξής πτυχές :

- Την σωστή απόκριση της μνήμης η οποία κρατάει τα TCB εφόσον είχε αλλάξει η μορφή της και είχαν προστεθεί νέα δεδομένα τα οποία έπρεπε να κρατάμε για κάθε σύνδεση. Αυτό έγινε τσεκάροντας όλα τα υποσυστήματα τα οποία χρησιμοποιούν πληροφορία από τα TCBs ως προς την σωστή διευθυσιοδότηση της μνήμης και την σωστή αποθήκευση αυτών σε κατάλληλους καταχωρητές για περαιτέρω επεξεργασία.
- Την σωστή λειτουργία των timers, οι οποίοι ήταν εντελώς διαφορετικοί στην τρίτη γενιά της σχεδίασης, ως προς τους τέσσερις τρόπους λειτουργίας, τους οποίους υποστηρίζουν αλλά και ως προς την πληροφορία την οποία πρέπει οι ίδιοι να παρέχουν στο υπόλοιπο υποσύστημα, όπως τον χρόνο που χρειάστηκε ώστε ένα πακέτο να μεταβεί στον δέκτη του και να επιστρέψει η επιβεβαίωσή του, ή το mode με το οποίο δούλευε σε περίπτωση που προκύψει timeout.
- Την σωστή υλοποίηση των αλγορίθμων που υλοποιούν τον έλεγχο συμφόρησης παρατηρώντας κυρίως το πώς διαμορφώνονταν οι τιμές των μεταβλητών congestion window και ssthresh ανάλογα με τις συνθήκες τις οποίες επικρατούσαν. Οι δοκιμές αυτές δεν έγιναν με την χρήση πραγματικών πακέτων αλλά με την ενεργοποίηση των κατάλληλων σημάτων έτσι ώστε το TCP να καταλαβαίνει ότι έχουν συμβεί διάφορα γεγονότα όπως έλευση επιβεβαίωσης ή έλευση διπλότυπης επιβεβαίωσης timeout από κάποιον timer κ.τ.λ. Ακολουθήσαμε αυτήν την μεθοδολογία αφενός λόγω χρόνου και όγκου δουλειάς αλλά επειδή είχαμε ήδη σιγουρέψει ότι η δημιουργία των πακέτων γίνεται με σωστό τρόπο. Έτσι δώσαμε περισσότερο βάρος στην λειτουργία του έλεγχου συμφόρησης ο οποίος βασίζεται στης τιμές των μεταβλητών congestion window και ssthresh.

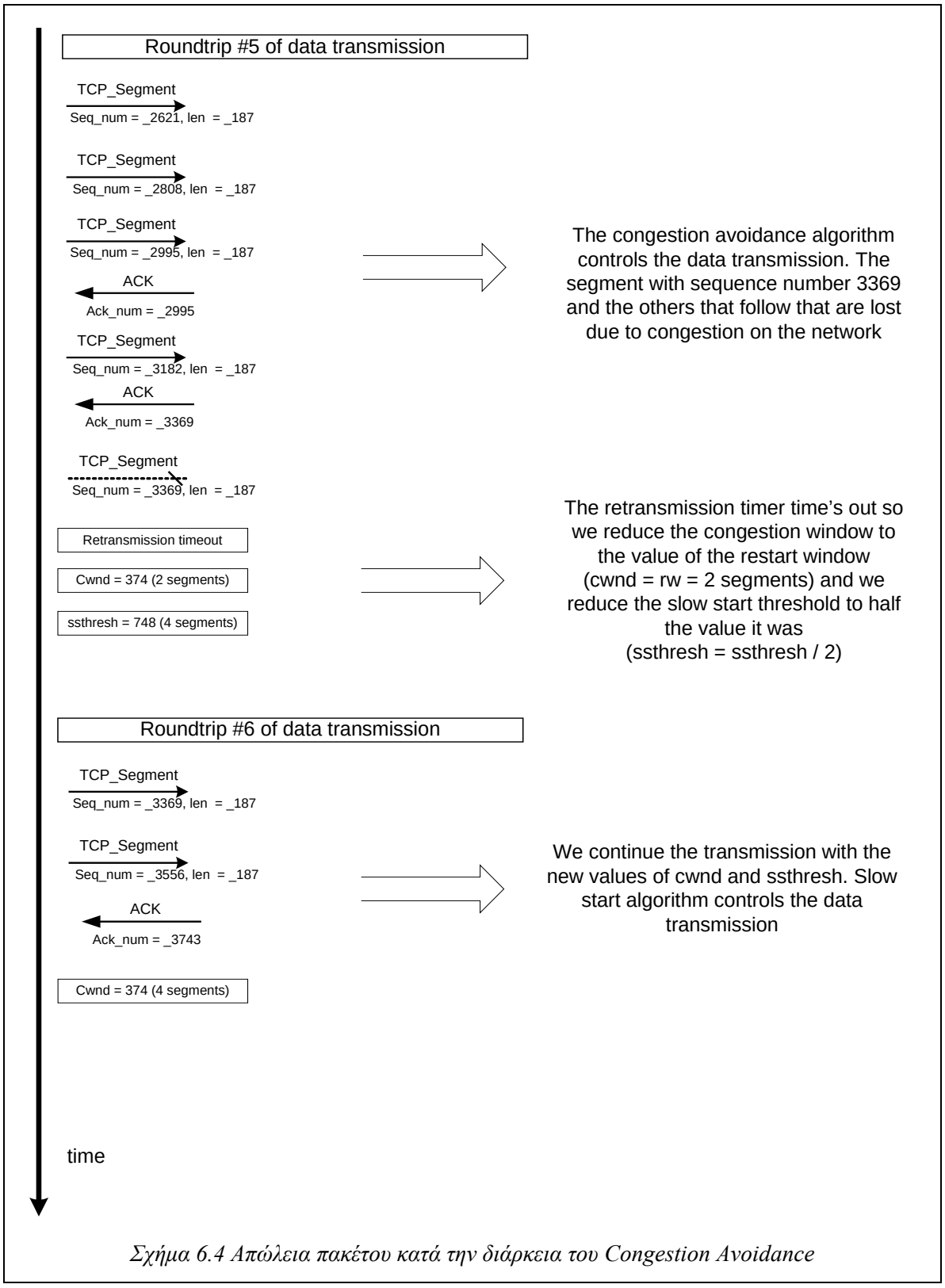
Παρακάτω ακολουθούν κάποια διαγράμματα τα οποία δείχνουν την συμπεριφορά της σχεδίασης σε κάποια σενάρια.



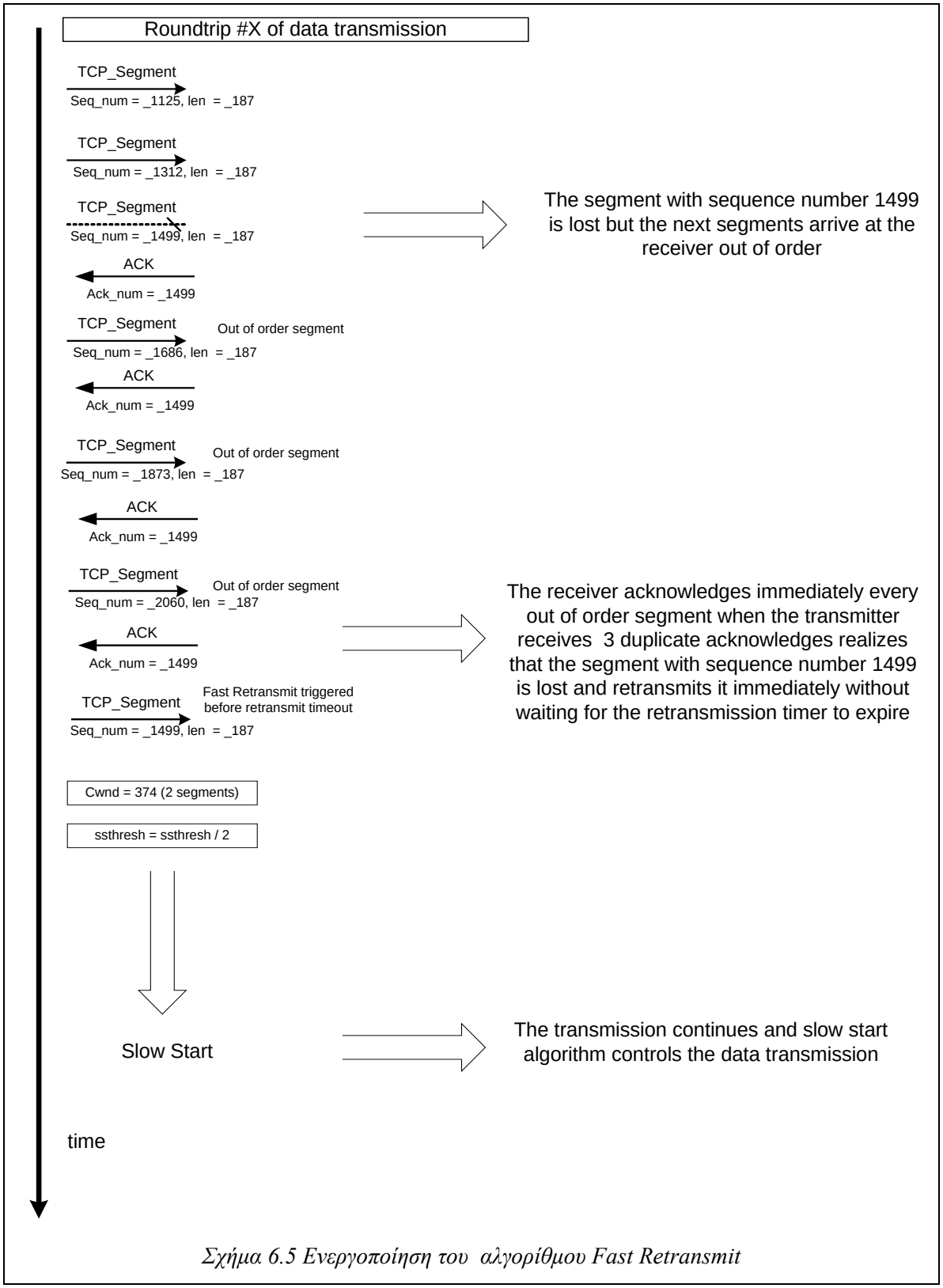
Σχήμα 6.2 Ο αλγόριθμος Slow start



Σχήμα 6.3 Ο αλγόριθμος Congestion Avoidance



Σχήμα 6.4 Απώλεια πακέτου κατά την διάρκεια του Congestion Avoidance



Σχήμα 6.5 Ενεργοποίηση του αλγορίθμου Fast Retransmit

6.3 Προγράμματα για την Ανάλυση Δικτύων

Η δημιουργία των πακέτων για τις δοκιμές είναι επιθυμητό να ανταποκρίνονται όσο γίνεται περισσότερο στα πακέτα των δικτύων που λειτουργούν σε πραγματικό χρόνο. Για το λόγο αυτό χρησιμοποιήθηκε ένα πρόγραμμα ανάλυσης δικτύου το οποίο μπορεί και καταγράφει τα πλαίσια Ethernet λαμβάνονται και αποστέλλονται από τον Ηλεκτρονικό Υπολογιστή στον οποίο είναι εγκατεστημένο. Τα προγράμματα αυτού του είδους μπορούν και φιλτράρουν μόνο πακέτα που ενδιαφέρουν ενώ διαχωρίζουν τα δεδομένα από τις επικεφαλίδες των πρωτοκόλλων και ακόμα περισσότερο διαχωρίζουν τα επιμέρους πεδία των επικεφαλίδων. Για την εργασία αυτή έγινε χρήση του προγράμματος Ethereal [41] το οποίο δίδεται δωρεάν με βάση το μοντέλο GNU General Public License. Με τη χρήση αυτή έγινε κατανοητό πως ακριβώς μεταδίδονται πακέτα, δημιουργούνται συνδέσεις, κλείνουν συνδέσεις, στέλνονται επιβεβαιώσεις ενώ γενικά αποτέλεσε ένα απαραίτητο εργαλείο για την υλοποίηση της εργασίας.

6.4 Υλοποίηση της σχεδίασης για αναδιατασσόμενη λογική

Έχει αναφερθεί ότι η ανάπτυξη του TCP/IP stack πρωτοκόλλων σε IP core στοχεύει στην ευελιξία που προσφέρει η αποτύπωσή του σε αναδιατασσόμενη λογική ανεξάρτητη από τον κατασκευαστή, αλλά και σε τεχνολογία VLSI/ASIC με τη χρήση κατάλληλων μοντέλων για τα στοιχεία μνήμης που υπάρχουν στη σχεδίαση. Για την εργασία αυτή έγινε απεικόνιση σε αναδιατασσόμενη λογική για FPGAs της εταιρείας Xilinx, καθώς υπήρχε η γνώση και η ευχέρεια χρήσης των εργαλείων της παραπάνω εταιρείας. Στην ενότητα αυτή γίνεται λόγος για αυτή την υλοποίηση και παρατίθενται κάποιες λεπτομέρειες για τους πόρους που απαιτεί η σχεδίαση από το εκάστοτε ολοκληρωμένο κύκλωμα αναδιατασσόμενης λογικής καθώς και για τη συχνότητα του ρολογιού του συστήματος που επιτυγχάνεται.

6.4.1 Υλοποίηση για το ολοκληρωμένο κύκλωμα Virtex XCV1000bg560

Η υλοποίηση για το ολοκληρωμένο κύκλωμα αναδιατασσόμενης λογικής XCV1000bg560 έγινε καθώς αυτό υπάρχει πάνω στην πλατφόρμα PLATO του Εργαστηρίου Μικροεπεξεργαστών και Υλικού και η οποία μπορεί να χρησιμοποιηθεί σαν βάση για τη δοκιμή

της σχεδίασης σε πραγματικό δίκτυο Ethernet. Παρακάτω παρατίθεται ο *Πίνακας 6.1* με στοιχεία για τους πόρους που απαιτεί η υλοποίηση από το παραπάνω FPGA Virtex.

Όπως φαίνεται από τον πίνακα αυτό, η συνολική σχεδίαση καταλαμβάνει περίπου το 48% του ολοκληρωμένου κυκλώματος και χρησιμοποιεί 34% των Block RAMs που υπάρχουν διαθέσιμα στο ολοκληρωμένο κύκλωμα Virtex 1000. Η ταχύτητα στην οποία μπορεί να λειτουργήσει η σχεδίαση δίνεται από το περιβάλλον ISE ότι είναι 26.6 MHz και επιτεύχθηκε χωρίς κανένα περιορισμό στους χρονισμούς του κυκλώματος. Η ταχύτητα αυτή που εκτιμάται από το περιβάλλον ISE είναι πολύ συντηρητική και λαμβάνει υπόψη τις χειρότερες περιπτώσεις, ενώ είναι διαπιστωμένο ότι είναι δυνατόν μια σχεδίαση να λειτουργήσει σωστά και σε ταχύτητες μεγαλύτερες από την εκτιμώμενη.

Υποσύστημα	**# slices**	**% της σχεδίασης**	**% του FPGA**	**Ταχύτητα (MHz)**
MII	2	0.1	0.05	338.3
CRC-32	53	1.1	0.55	77.2
RxETHER	138	2.3	1.2	44
TxETHER	63	1.1	0.55	89.8
OUTMUX	34	0.6	0.3	160.8
ARP_RECV_REPLY	219	3.7	1.7	62.9
ARP_REQUEST	85	1.4	0.8	83.1
ETHER_SEND	126	2.1	1.0	74.3
CHECKSUM	40	0.8	0.4	92.7
IP_RECV	57	0.9	0.45	60.2
IP_SEND	99	1.7	0.8	83.9
CONTROL	129	2.3	1.2	89.9
ICMP_RECV_REPLY	111	1.9	0.9	78.3
UDP	506	8.6	4.2	48.6
TCP	4203	70.4	33.9	27.6
TOTAL	**5931**	**100.0**	**48**	**26.6**
Memory (Block RAM)			**11 / 36 Block RAMs (34%)**	

Πίνακας 6.1 Πόροι της σχεδίασης για το FPGA Virtex, XCV1000bg560.

Πρέπει να παρατηρηθεί ότι η ταχύτητα του μεγαλύτερου και πιο πολύπλοκου υποσυστήματος (του TCP) είναι πολύ κοντά στην ταχύτητα της συνολικής σχεδίασης οπότε ουσιαστικά αυτό καθορίζει την ταχύτητα του συστήματος. Το γεγονός ότι δεν παρατηρείται μεγάλη διαφορά ανάμεσα στην ταχύτητα του IP core και σε αυτή του υποσυστήματος TCP, οφείλεται στο γεγονός ότι το υποσύστημα αυτό έχει απόλυτα καθορισμένη διεπαφή με την υπόλοιπη σχεδίαση και η ανταλλαγή πληροφορίας με το πρωτόκολλο του χαμηλότερου επιπέδου

γίνεται με άνεση χρόνου, δηλαδή οι υπολογισμοί μπορούν να γίνουν αρκετούς κύκλους πριν να ζητηθούν τα δεδομένα.

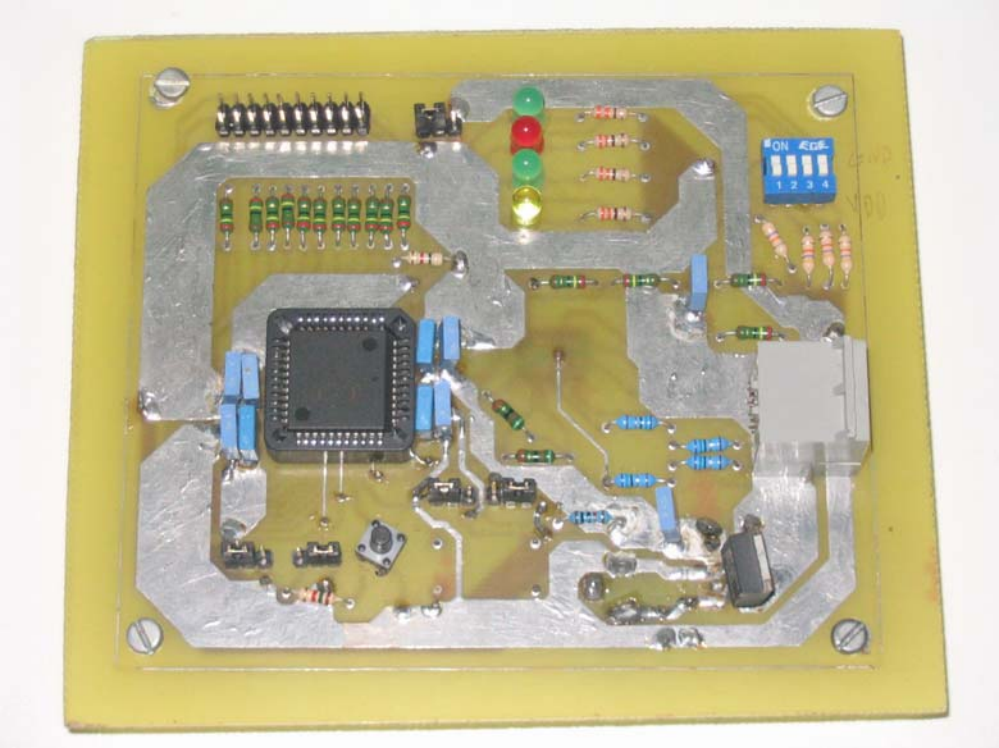
6.5 Υλοποίηση πλακέτας Etherner Transceiver

Στα πλαίσια αυτής της διπλωματικής εργασίας υλοποιήσαμε επίσης και ένα PCB (Printed Circuit Board) το οποίο υλοποιούσε έναν Ethernet transceiver. Το PCB βασιζόταν στο ολοκληρωμένο 80225 10/100 Mbps TX/10BT Ethernet Physical Layer Device (PHY) της εταιρίας LSI LOGIC[42]. Το chip δουλεύει με 25 Mhz ρολόι μπορεί να λειτουργήσει σε δίκτυα και 10 και 100 Mbps με δυνατότητες και half και full duplex μετάδοσης, MII Interface και MI Interface (ένα σειριακό Interface το οποίο μπορεί να χρησιμοποιηθεί για τον έλεγχο του ολοκληρωμένου). Έχει υλοποιημένους σε υλικό autonegotiation αλγορίθμους ώστε να μπορεί να διαπραγματευτεί μόνο του με το εκάστοτε δίκτυο τον τρόπο λειτουργίας του. Κάνει διαμόρφωση κυματομορφών από μόνο του χωρίς την ανάγκη εξωτερικών φίλτρων και έχει και προσαρμοζόμενο αντισταθμιστή.

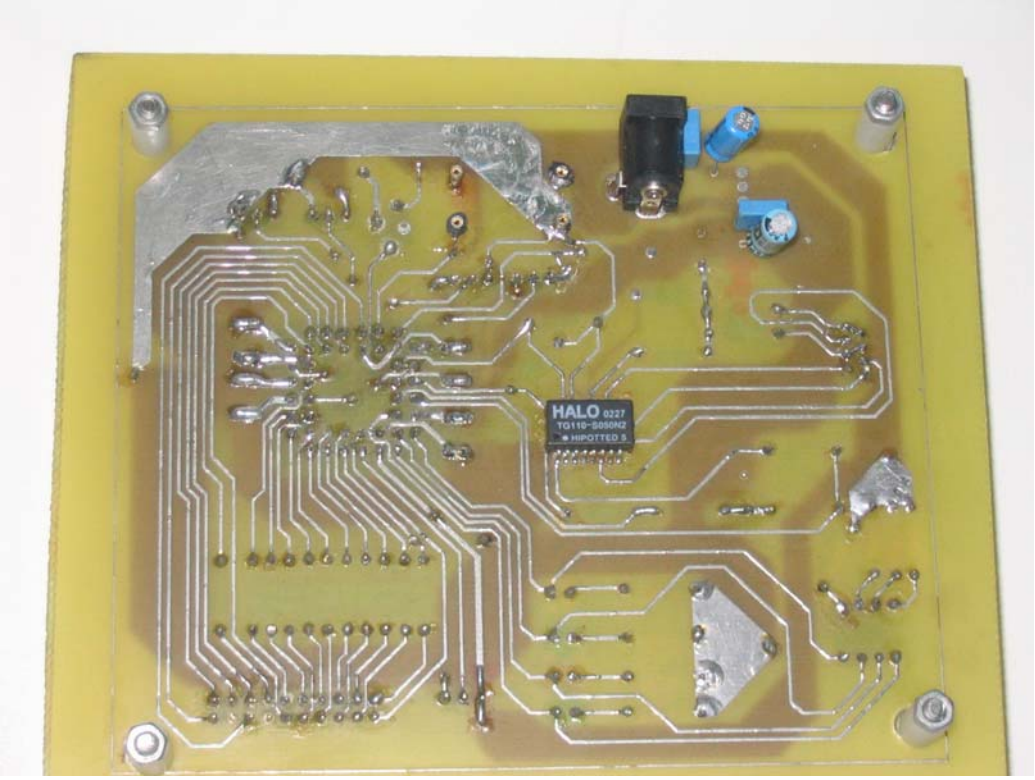
Αποφασίσαμε να χρησιμοποιήσουμε το συγκεκριμένο ολοκληρωμένο μετά από εκτενή έρευνα αγοράς και τα βασικά μας κριτήρια ήταν ότι όλα τα εξαρτήματα τα οποία απαιτούσε για την ορθή του λειτουργία μπορούσαμε να τα βρούμε εύκολα στα πλαίσια της εγχώριας αγοράς και το γεγονός ότι είναι PLCC 44pins κάτι το οποίο τα καθιστά την διαδικασία προσάρτησής του πάνω στην τυπωμένη πλακέτα πολύ εύκολη μέσω μιας πολύ απλής βάσης.

Η πλακέτα που κατασκευάστηκε στα πλαίσια του εργαστηρίου μικροεπεξεργαστών και υλικού τελικά δεν λειτούργησε ποτέ αν και τα πάντα ήταν εντός προδιαγραφών της εταιρίας εκτός της πρότασης για τυπωμένο κύκλωμα τεσσάρων επιπέδων πράγμα αδύνατο για τα δεδομένα του εργαστηρίου μας. Φωτογραφίες της κατασκευασμένης πλακέτας ακολουθούν παρακάτω.

.



Top View



Bottom View

Κεφάλαιο 7ο : Συμπεράσματα - Μελλοντικές επεκτάσεις

7.1 Συμπεράσματα

Στόχος της εργασίας αυτής ήταν η ενσωμάτωση ελέγχου συμφόρησης δικτύου στο ήδη υπάρχων IP core του πρωτοκόλλου TCP/IP. Σε αυτό το κείμενο παρουσιάζονται όλα τα βήματα για την επίτευξη αυτού του στόχου ξεκινώντας από την ιδέα και το πρόβλημα. Στη συνέχεια παρουσιάζεται η μελέτη της Σχετικής Έρευνας και περιγράφονται οι τελικές Προδιαγραφές. Χτίζεται η Αρχιτεκτονική της σχεδίασης και περιγράφεται η Οργάνωση των επιμέρους υποσυστημάτων που απαρτίζουν το συνολικό σύστημα. Τέλος παρουσιάζονται οι προσομοιώσεις που έθεσαν σε δοκιμή το σύστημα όπου διαπιστώθηκε η σωστή λειτουργία του, σύμφωνα πάντα με τις Προδιαγραφές. Ακόμα περιγράφεται η υλοποίηση του συστήματος για ένα ολοκληρωμένο κυκλώμα αναδιατασσόμενης λογικής της εταιρείας Xilinx ενώ διατηρείται η πεποίθηση ότι αυτό το IP core μπορεί να μεταφερθεί και σε αναδιατασσόμενη λογική άλλων εταιρειών (π.χ. Altera, Laticce, Atmel) ή ακόμα και σε τεχνολογία VLSI/ASIC.

Με την ολοκλήρωση αυτής της διπλωματικής εργασίας υπάρχει η δυνατότητα αποτύπωσης του TCP/IP stack πρωτοκόλλων με όλες τις δυνατότητες που αυτά προσφέρουν σε πλήρη λειτουργία, σε καθαρό hardware, το οποίο θεωρείται ότι είναι χρήσιμο. Η χρησιμότητα μπορεί να διαπιστωθεί με την προσθήκη αυτού του IP core σε ενσωματωμένα συστήματα ή σε συστήματα διακίνησης δεδομένων είτε για τη διαχείριση απομακρυσμένων συστημάτων μέσω του διαδικτύου, χωρίς την παρουσία Ηλεκτρονικού Υπολογιστή. Ακόμα μπορεί να χρησιμοποιηθεί για την επιτάχυνση της επεξεργασίας των πρωτοκόλλων με την εκμετάλλευση που προσφέρει η ταχύτητα του ειδικά κατασκευασμένου hardware.

Η εργασία αυτή ακόμη πρόσφερε τεχνογνωσία και εμπειρία σε θέματα τηλεπικοινωνιών και δικτύων, σε θέματα σχεδίασης hardware και σε ζητήματα χρήσης εξελιγμένων εργαλείων CAD.

7.2 Μελλοντικές Επεκτάσεις

Όταν ξεκίνησε η διπλωματική αυτή εργασία ήρθαμε αντιμέτωποι με ένα IP core τεραστίων πραγματικά διαστάσεων και από άποψη πολυπλοκότητας αλλά και από άποψη όγκου κώδικα. Το κακό όμως ήταν ότι δεν υπήρχαν σχόλια στο σύνολο των κωδίκων που περιγράφουν το IP core με αποτέλεσμα να δαπανηθεί πάρα πολύς χρόνος προσπαθώντας να καταλάβουμε και να βάλουμε σχόλια στους ήδη υπάρχοντες κώδικες έτσι ώστε να μπορούμε να καταλάβουμε την λειτουργία τους αλλά και να μπορέσουμε να κάνουμε αλλαγές μέσα σε αυτούς όπου χρειαζόταν. Εκτός λοιπόν του ότι οι κώδικες δεν είχαν σχόλια στο σύνολό τους πράγμα που έκανε εξαιρετικά δύσκολη την κατανόησή τους, το υποσύστημα TCP το οποίο αποτελούνταν από άλλα τέσσερα υποσυστήματα ήταν γραμμένο σε ένα ενιαίο entity με αποτέλεσμα να μην υπάρχουν σωστά καθορισμένες διεπαφές μεταξύ των υποσυστημάτων αυτών δημιουργώντας έτσι ένα υποσύστημα όπου ο κώδικας που το περιγράφει είναι περίπου 3.500 γραμμές (χωρίς σχόλια), καθιστώντας το έτσι εξαιρετικά δυσνόητο και πολύ δύσκολα επεκτάσιμο. Βέβαια όλα αυτά έγιναν αντιληπτά κατά την διάρκεια της προσπάθειας κατανόησης του υπάρχοντος κώδικα και ενώ είχαμε αρχίσει να κάνουμε ήδη αλλαγές έτσι ώστε να μεταβούμε στην τρίτη γενιά της σχεδίασης.

Όπως μπορούμε να καταλάβουμε από όλα τα παραπάνω είναι λογικό εφόσον ακολουθήσαμε την ίδια αρχιτεκτονική κάνοντας αλλαγές μέσα σε αυτήν “ κληρονομήσαμε ” και κάποιους περιορισμούς οι οποίοι προέρχονταν από την αρχιτεκτονική της δεύτερης γενιάς της σχεδίασης. Ο πιο βασικός λοιπόν τέτοιος περιορισμός είναι το γεγονός ότι η σχεδίαση δεν έχει την δυνατότητα να πολυπλέκει πακέτα από διαφορετικές συνδέσεις όταν θέλουν παραπάνω από δύο να αποστείλουν δεδομένα την ίδια στιγμή. Έτσι λοιπόν πρέπει πρώτα κάποια σύνδεση να τελειώσει την αποστολή των δεδομένων τα οποία θέλει να στείλει στο δίκτυο και μετά μπορεί κάποια άλλη σύνδεση να κάνει αποστολή δεδομένων. Καταλαβαίνουμε ότι αυτός ο περιορισμός μειώνει την απόδοση του πρωτοκόλλου αφού πάλι δεν μπορεί να πιάσει το μέγιστο των δυνατοτήτων του. Πάραυτα η απόδοση του πρωτοκόλλου σε σχέση με την δεύτερη γενιά έχει αυξηθεί κατακόρυφα.

Ως πρώτη μελλοντική επέκταση λοιπόν την παρούσας εργασίας, προτείνουμε την αναδιοργάνωση της αρχιτεκτονικής του υποσυστήματος TCP κατά τέτοιο τρόπο έτσι ώστε να εξαλείψουμε όλους τους περιορισμούς των προηγούμενων γενεών αλλά κυρίως να γραφτεί σωστά από την άποψη του πώς είναι σωστό να γράφεται κώδικας έτσι ώστε να είναι ευκολονόητος αλλά κυρίως επαναχρησιμοποιήσιμος και εύκολα αναβαθμίσιμος.

Άλλες αλλαγές που μπορούν να γίνουν στην ίδια τη σχεδίαση είναι οι εξής. Αφαίρεση των υποσυστημάτων που υποστηρίζουν το πρωτόκολλο Ethernet και τοποθέτηση άλλων

υποσυστημάτων στη θέση τους. Έτσι μπορεί να υποστηριχθεί η διακίνηση πακέτων ATM ώστε να δημιουργηθεί ένα σύστημα δικτύωσης IP over ATM. Ακόμα μπορεί να υποστηριχθεί το πρωτόκολλο Bluetooth™, ή το 802.11 πάνω από τα οποία παραμένουν τα πρωτόκολλα IP, UDP και TCP για τη δημιουργία ενός συστήματος ασύρματης δικτύωσης μεταξύ συσκευών.

Επιπλέον μπορεί να αντικατασταθεί το υποσύστημα λήψης και αποστολής πακέτων IP, το οποίο υποστηρίζει την έκδοση 4 (IPv4) με ένα υποσύστημα που μπορεί να υποστηρίζει την έκδοση 6 (IPv6), ώστε να μπορεί να σταθεί η σχεδίαση σε πιθανή ολική μετάβαση στη νεότερη έκδοση.

Είναι δυνατόν επίσης να γίνει μελέτη για την αύξηση της ταχύτητας στην οποία μπορεί να τρέξει το IP core, καθώς και μελέτη για τη αύξηση του εύρους του data bus για να μπορούν να υποστηριχθούν δίκτυα πολύ υψηλών ταχυτήτων. Τέτοιο δίκτυο είναι το Gigabit Ethernet το οποίο στην παρούσα φάση υποστηρίζει μεταδώσεις μέχρι και 10 Gbit με τη χρήση οπτικών ινών.

Τέλος παρουσιάζονται κάποιες εφαρμογές ου μπορούν να αναπτυχθούν και να αξιοποιήσουν τη σχεδίαση σε hardware. Είναι δυνατόν να υλοποιηθούν εφαρμογές για τη συλλογή δεδομένων μέσω του διαδικτύου από απομακρυσμένα συστήματα, όπως είναι τα συστήματα Ανανεώσιμων Πηγών Ενέργειας, αλλά και για τη διαχείριση αυτών των συστημάτων χωρίς την παρουσία Ηλεκτρονικού Υπολογιστή. Μπορεί επίσης να φτιαχτεί κάποια εφαρμογή για τη σύνδεση κάμερας και μικροφώνου-ακουστικών για την υποστήριξη μετάδοσης εικόνας και ήχου μέσω IP.

Η ολοκλήρωση του IP core σε διάφορες συσκευές χειρός όπως κινητά τηλέφωνα, palmtops, μπορεί να προσφέρει σε αυτές υπηρεσίες ubiquitous computing χωρίς κόστος στην επεξεργαστική τους ισχύ. Η ολοκλήρωση του δε με κάποια εφαρμογή που προσφέρει επικοινωνία με Ηλεκτρονικούς Υπολογιστές μέσω PCI/DMA, μπορεί να αξιοποιηθεί για την υλοποίηση γρήγορων web servers που μπορούν να απαντούν άμεσα σε συχνές αιτήσεις σελίδων οι οποίες υπάρχουν αποθηκευμένες στη μνήμη.

Βιβλιογραφία

[1] Α. Αλεξόπουλος, Γ. Λαγογιάννης, *Τηλεπικοινωνίες και Δίκτυα Υπολογιστών*

[2] Jean Walrand, *Δίκτυα Επικοινωνιών, μετάφραση Μ. Αναγνώστου, εκδόσεις Παπασωτηρίου*

[3] Timothy Parker, *Teach yourself TCP/IP in 14 days, published by SAMS*

[4] James F. Kurose, Keith W. Ross, *Computer Networking A Top-Down Approach Featuring the Internet, published by AWL*

[5] Stefan Sjoholm, Lennart Lindh, *VHDL for Designer, published by Prentice Hall*

[6] Peter Ashenden, *The Designer's Guide to VHDL, Morgan Kaufmann Publishers*

[7] Mark Zwolinski, *Digital System Design with VHDL, published by Pearson Education*

[8] Stanley Mazor, Patricia Langstraat, *A Guide To VHDL, Kluwer Academic Publishers*

[9] IEEE Std 802.3, 2000 Edition

[10] A. Dollas, D. Pnevmatikatos, N. Aslanides, S. Kavvadias, E. Sotiriades, S. Zogopoulos, K. Papademetriou, N. Chrysos, K. Harteros, E. Antonidakis, N. Petrakis, *Architecture and Applications of PLATO, a Reconfigurable Active Network Platform*. In Proceedings, 9th International IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM-2001), Rohnert Park, CA, April 30 - May 2, 2001, IEEE.

[11] A. Dollas, D. Pnevmatikatos, N. Aslanides, E. Sotiriades, S. Kavvadias, S. Zogopoulos, Experimental Testing of PLATO, a Reconfigurable Active ATM Network Node. In Proceedings of the 8th Panhellenic Informatics Conference, Cyprus, November, 2001

[12] Wetherall, U. Legedza, and J. Guttag, "Introducing New Internet Services: Why and How", *IEEE Network Magazine*, July/August 1998.

[13] Lehman, S. Garland and D. Tennenhouse, "Active Reliable Multicast", In *IEEE INFOCOM'98*, San Francisco, CA, March 1998.

[14] A. Campell, H. De Meer, M. Kounavis, K. Miki, J. Vicente, and D. Villela, "A Survey of Programmable Networks", In *Proceedings of IEEE Infocom'98,* San Francisco, CA, March 1998

[15] Mirko Benz, Dresden University of Technology, *An Architecture and Prototype Implementation for TCP/IP Hardware Support*. In Proceedings, TERENA Networking Conference 2001

[16] Γ. Ζήσης, "Ανάπτυξη πρωτοκόλλου TCP/IP πολλών θυρών από Application ως Session Layer, βασισμένου σε αναδιατασσόμενη λογική", *Διπλωματική Εργασία*, Πολυτεχνείο Κρήτης

Παγκόσμιος Ιστός

http://www.rfc-editor.org, Σύνδεσμος στα RFC του οργανισμού IETF.

[17] RFC 768 "User Datagram Protocol"

[18] RFC 791 "Internet Protocol"

[19] RFC 792 "Internet Control Message Protocol"

[20] RFC 793 "Transmission Control Protocol"

[21] RFC 826 "Ethernet Address Resolution Protocol"

[22] RFC 894 "Standard for the Transmission of IP Datagrams over Ethernet networks"

[23] RFC 2581 "TCP Congestion Control"

[24] RFC 2988 "Computing TCP'S Retransmission Timer"

[25] RFC 2861 "TCP Congestion Window Validation"

[26] http://www.hynix.com/eng/products/system_ic/sp/down/HMS91C7432.pdf, Σύνδεσμος στο εγχειρίδιο του ολοκληρωμένου κυκλώματος HMS91C7432.

[27] http://www.national.com

[28] http://www.iready.com

[29] http://www.iready.com/EthernetMAXweb.pdf, Σύνδεσμος στο εγχειρίδιο πληροφοριών του project EtherMAX

[30] http://www.edevice.com

[31] http://www.cmx.com

[32] http://www.altera.com/products/devkits/altera/kit-dev_nios_ethernet.html, Σύνδεσμος σε πληροφορίες για τον soft επεξεργαστή Nios και τη δυνατότητα δικτύωσής του.

[33] http://www-ee.eng.hawaii.edu/~msmith/XCoNET/XCoNET.htm, Σύνδεσμος στο project XCoNet.

[34] http://www.itee.uq.edu.au/~peters/xsvboard/stack/stack.htm, Σύνδεσμος στην εργασία του πανεπιστημίου του Queensland.

[35] http://www.xilinx.com/publications/xcellonline/partners/xc_insight41.htm, Σύνδεσμος στην πλατφόρμα VoIP της εταιρείας Insight.

[36] http://www.xilinx.com/xilinxonline/MMT_html/cs_005.htm, Σύνδεσμος στην πλατφόρμα MMT 2000.

[37] http://www.altera.com

[38] http://www.xilinx.com

[39] http://www.celoxica.com

[40] http://www.marconi.com

[41] http://www.ethereal.org

[42] http://www.lsilogic.com

Τμήμα Ηλεκτρονικών Μηχανικών Και Μηχανικών Η/Υ Πολυτεχνείου Κρήτης

Εργαστήριο Μικροεπεξεργαστών και υλικού (ΕΜΥ)

“ Έλεγχος Συμφόρησης σε Επαναχρησιμοποιήσιμο Σχεδιασμό του Πρωτοκόλλου TCP/IP για Αναδιατασσόμενη Λογική”

Κοϊδής Ιωσήφ

Σύντομη Περιγραφή

- Στόχος της Εργασίας
- Σχετική Έρευνα
- Congestion Control
- Αρχιτεκτονική της Σχεδίασης
- Πιστοποίηση της Σχεδίασης
- Συμπεράσματα – Μελλοντικές Επεκτάσεις

Στόχος Της Εργασίας

Σχεδίαση και υλοποίηση έλεγχου συμφόρησης σε Επαναχρησιμοποιήσιμο Σχεδιασμό του Πρωτοκόλλου TCP/IP για αναδιατασσόμενη λογική, FPGA

Αυτό σημαίνει :

- Δυνατότητα μετάδοσης περισσοτέρων του ενός πακέτων τη φορά
- Έλεγχος της διαδικασίας αυτής ώστε να μην υπάρξει συμφόρηση στο δίκτυο

Σχετική Έρευνα (Ι)

- Υλοποίηση του TCP/IP stack για μικροελεγκτές και μικροεπεξεργαστές με Software
  - Ευκολία, ταχύτητα ανάπτυξης εφαρμογών (Γλώσσες υψηλού επιπέδου, C, C^{++})
  - Ευελιξία, εύκολη πρόσθεση / αφαίρεση λειτουργικότητας
  - Χαμηλή απόδοση, εξαιτίας μεγάλου overhead, παρέχονται συνήθως στα 10 Mbps

Σχετική Έρευνα (ΙΙ)

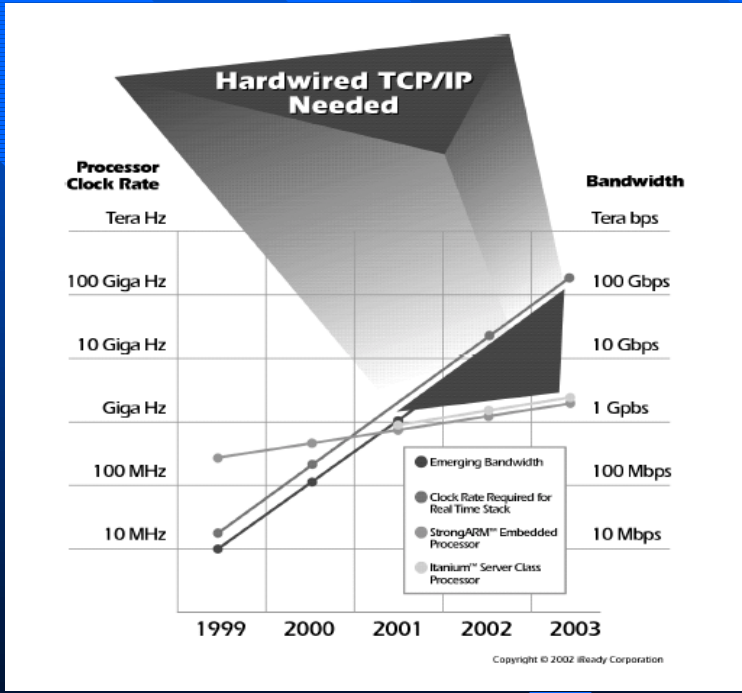
- CMX : Software που υποστηρίζει το πρωτόκολλο TCP/IP για μικροελεγκτές και μικροεπεξεργαστές 8-bit και 16-bit
  - Υποστηρίζει την ανάπτυξη εφαρμογών σε Γλώσσα C παρέχοντας έτοιμες συναρτήσεις
  - Υποστηρίζει μεγάλη γκάμα ολοκληρωμένων κυκλωμάτων όπως ATMEL AVR, Microchip PIC, 80C51, NEC 78K0/K0S, Mitsubishi M16C, STMicroelectronics ST10 …

Σχετική Έρευνα (III)

- Υλοποίηση του TCP/IP stack σε Hardware με τεχνολογία VLSI/ASIC
  - Ταχύτητα και Απόδοση, υποστηρίζουν ρυθμούς μέχρι και 10 Gbps
  - Χαμηλή κατανάλωση, ζητούμενη σε embedded συστήματα
  - Μη ευέλικτη υλοποίηση
  - Υψηλό κόστος κατασκευής, *f* (# τεμαχίων)

Σχετική Έρευνα (IV)

- Project EtherMAX (iReady, National Semiconductors)



Σχετική Έρευνα (V)

- Υλοποίηση του TCP/IP stack σε αναδιατασσόμενη λογική με τη μορφή IP (Intellectual Property) core

  \+ Γρήγορη ανάπτυξη

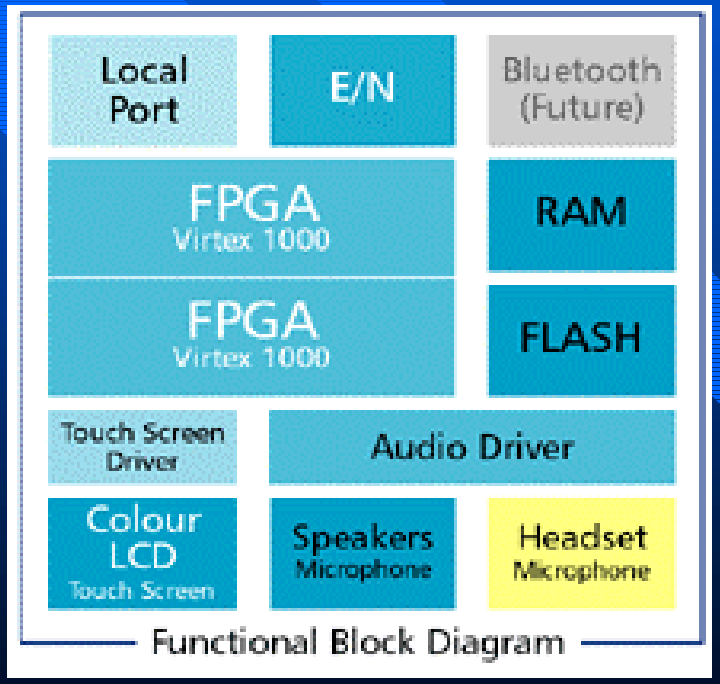
  \+ Ευελιξία

  } Χρήση Γλώσσας υψηλού επιπέδου Περιγραφής Υλικού, όπως η VHDL

  \- Μικρότερη ταχύτητα σε σχέση με την τεχνολογία VLSI/ASIC

Σχετική Έρευνα (VI)

- Πλατφόρμα MMT 2000 (Xilinx, Celoxica, Marconi)
  - Υλοποίηση σε FPGA Virtex 1000
  - Υποστηρίζει 2 συνδέσεις UDP και 1 σύνδεση TCP



Congestion Control (I)

- Για την υλοποίηση του Congestion Control στο TCP προσθέτουμε δύο ακόμα μεταβλητές ανά σύνδεση
  - Congestion window
  - Slow start threshold
- Οι αλγόριθμοι οι οποίοι χρησιμοποιούνται είναι :
  - Slow start
  - Congestion avoidance

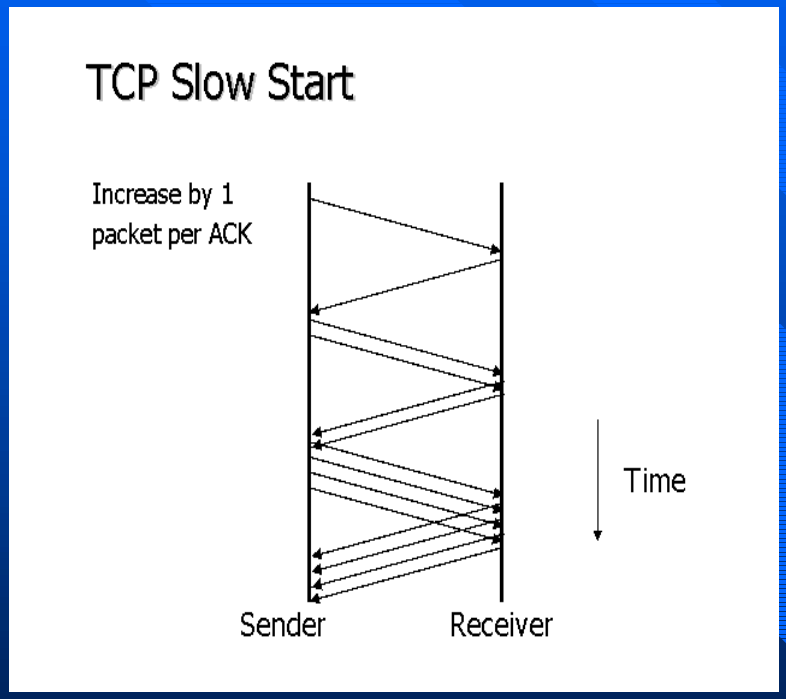
  } Απαραίτητη χρήση
  - Fast Retransmit / Fast Recovery => Προαιρετική χρήση

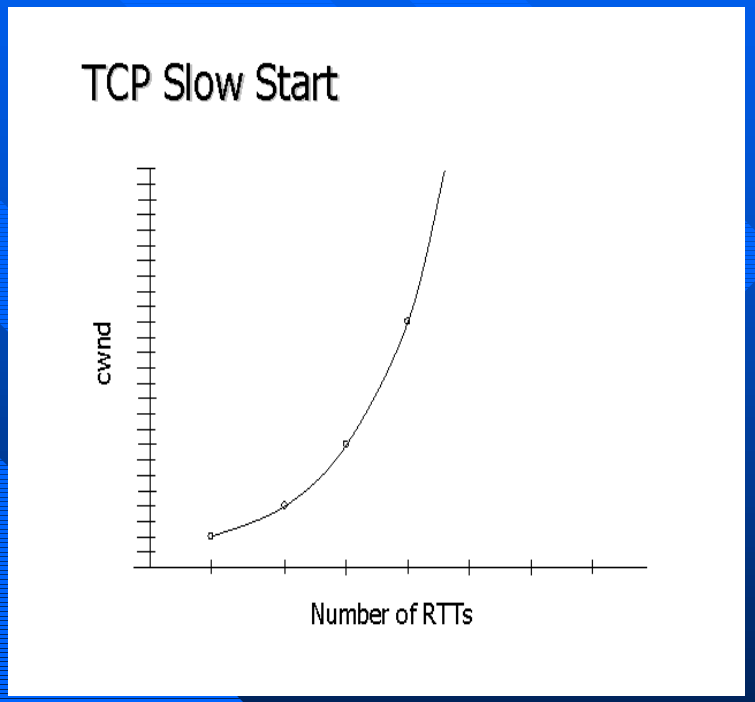
Congestion Control (II)

- Slow Start
  - » Slow Start : εκθετική αύξηση του congestion window αυξανόμενο κατά ένα πακέτο για κάθε πακέτο που επιβεβαιώνεται
  - » Διαρκεί έως ότου :

    congestion window >= slow start threshold

Congestion Control (III)





Congestion Control (IV)

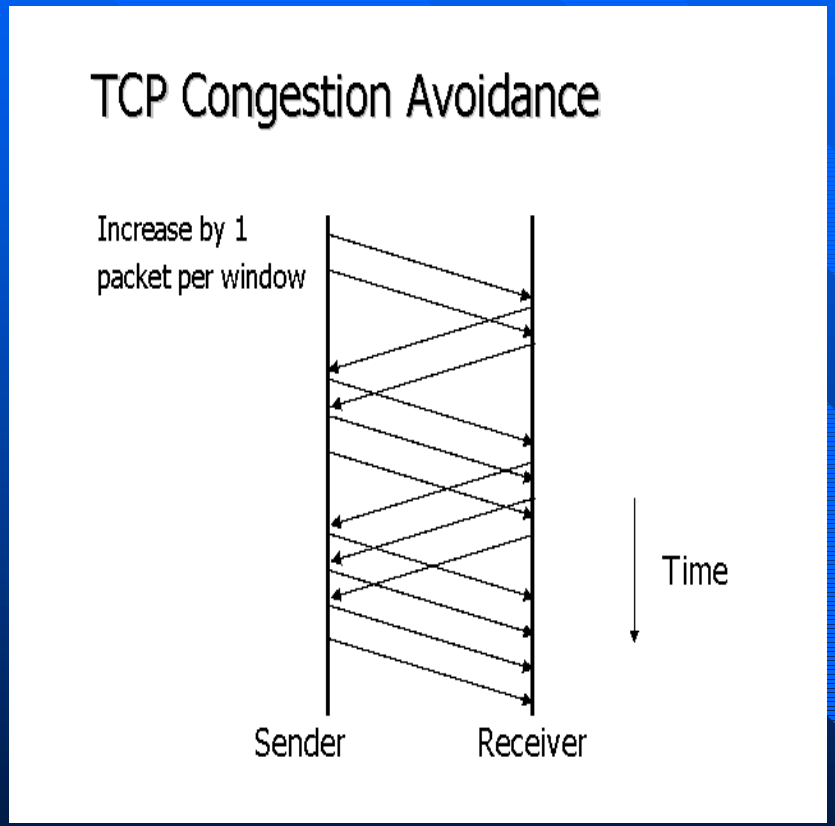
- Congestion Avoidance

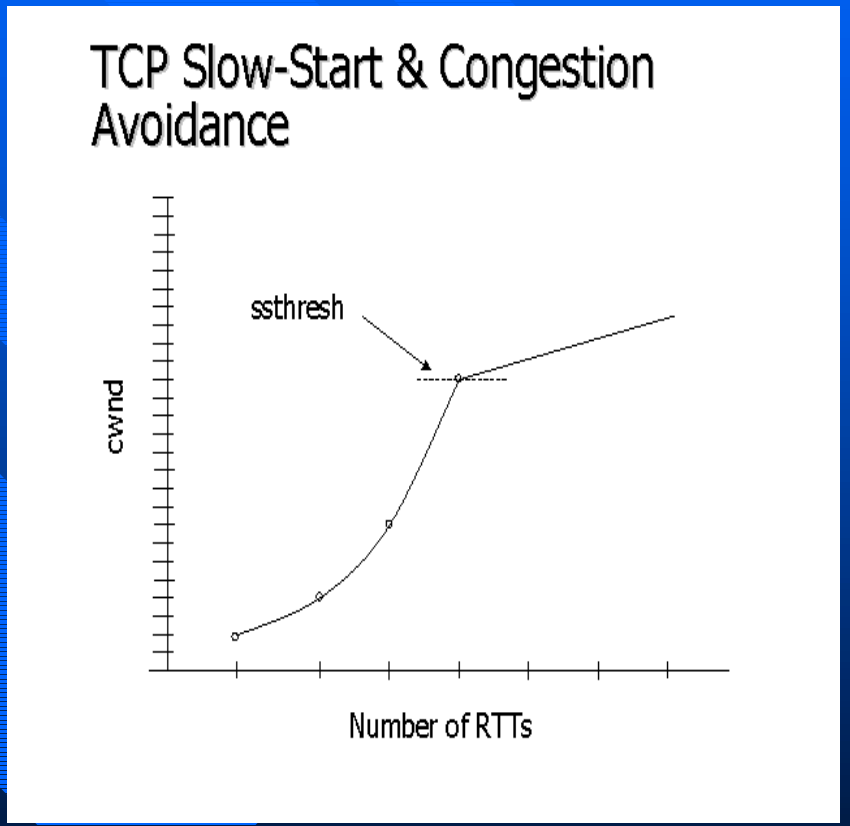
  » congestion window > slow start threshold

  » congestion window = congestion window + 1
  για κάθε παράθυρο δεδομένων που επιβεβαιώνεται

  » Διαρκεί μέχρι το τέλος της μετάδοσης ή μέχρι να έχουμε κάποια απώλεια πακέτου

Congestion Control (V)





Congestion Control (VI)

- Απώλεια πακέτου εκλαμβάνεται ως ένδειξη συμφόρησης στο δίκτυο και γίνονται τα εξής :
  - » slow start threshold = slow start threshold / 2
  - » congestion window = Initial Window
  - » Επαναμετάδοση των χαμένων πακέτων
  - » Ο Slow start ελέγχει την μετάδοση

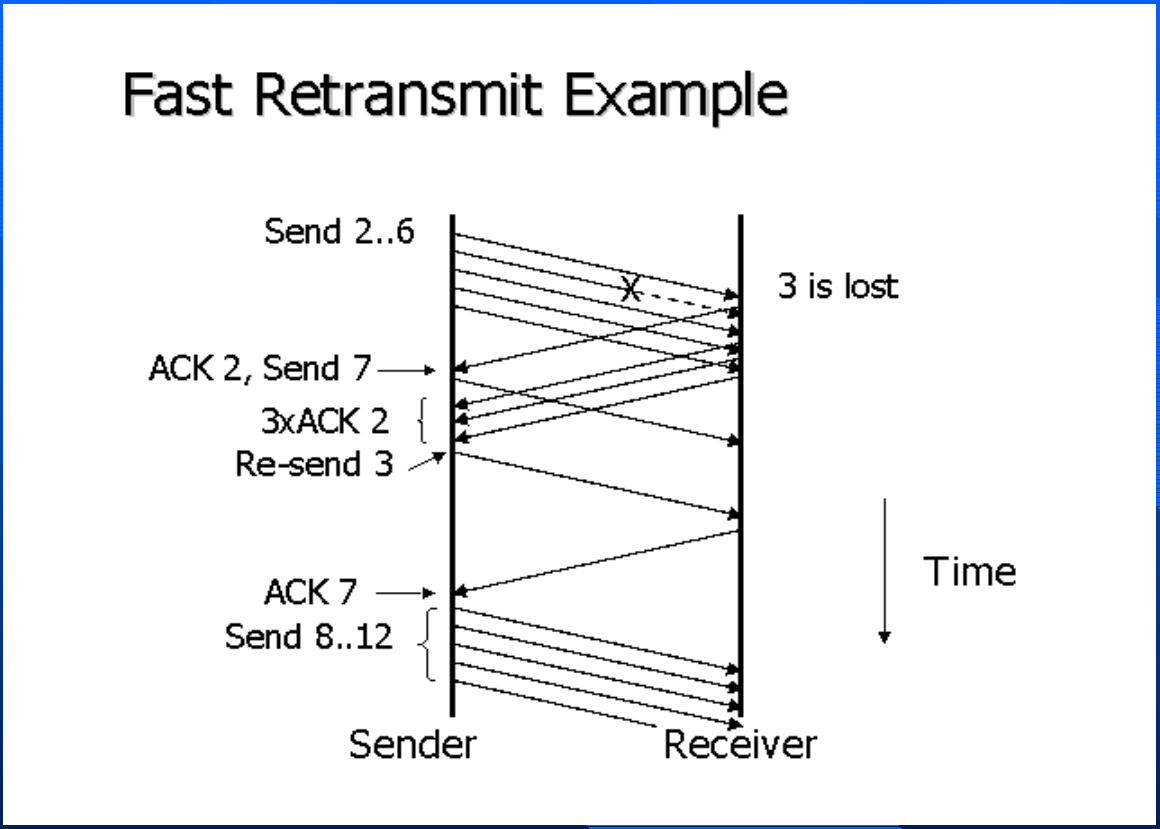
Congestion Control (VII)

- Λήψη πακέτου εκτός σειράς
  - Οφείλεται σε :
    - » Αναδιάταξη πακέτων από το δίκτυο
    - » Δημιουργία πανομοιότυπων πακέτων από το δίκτυο
    - » Απόρριψη κάποιου πακέτου
    - » Συμφόρηση στο δίκτυο
  - Ενέργεια :
    - » Επιβεβαίωση του τελευταίου σωστού πακέτου

Congestion Control (VIII)

- Fast Retransmit / Fast Recovery
  - » Ενεργοποιείται με τρεις διπλότυπες επιβεβαιώσεις
  - » Στοχεύει στην γρηγορότερη αντίληψη κάποιας απώλειας
  - » ssthresh = max(flightsize/2, 2*SMSS)
  - » Τεχνητή διόγκωση του congestion window
  - » Αποδιόγκωση μετά την διόρθωση της απώλειας

Congestion Control (IX)



Congestion Control (X)

- Αν μια σύνδεση μείνει για ένα χρονικό διάστημα ανενεργή το TCP παύει να έχει επίγνωση της κατάστασης του δικτύου

- Το TCP
  - » μειώνει το congestion window στην αρχική του τιμή
  - » ο αλγόριθμος slow start αναλαμβάνει τον έλεγχο της μετάδοσης

Congestion Control (XI)

- TCP και επιβεβαιώσεις
  - Μια επιβεβαίωση μπορεί να καθυστερηθεί αλλά όχι για πιο πολύ από 500ms
  - Πακέτα τα οποία λαμβάνονται εκτός σειράς πρέπει να επιβεβαιώνονται αμέσως
  - Ο δέκτης δεν πρέπει να στέλνει διπλότυπες επιβεβαιώσεις χωρίς λόγο

Η 2η Γενιά Της Σχεδίασης

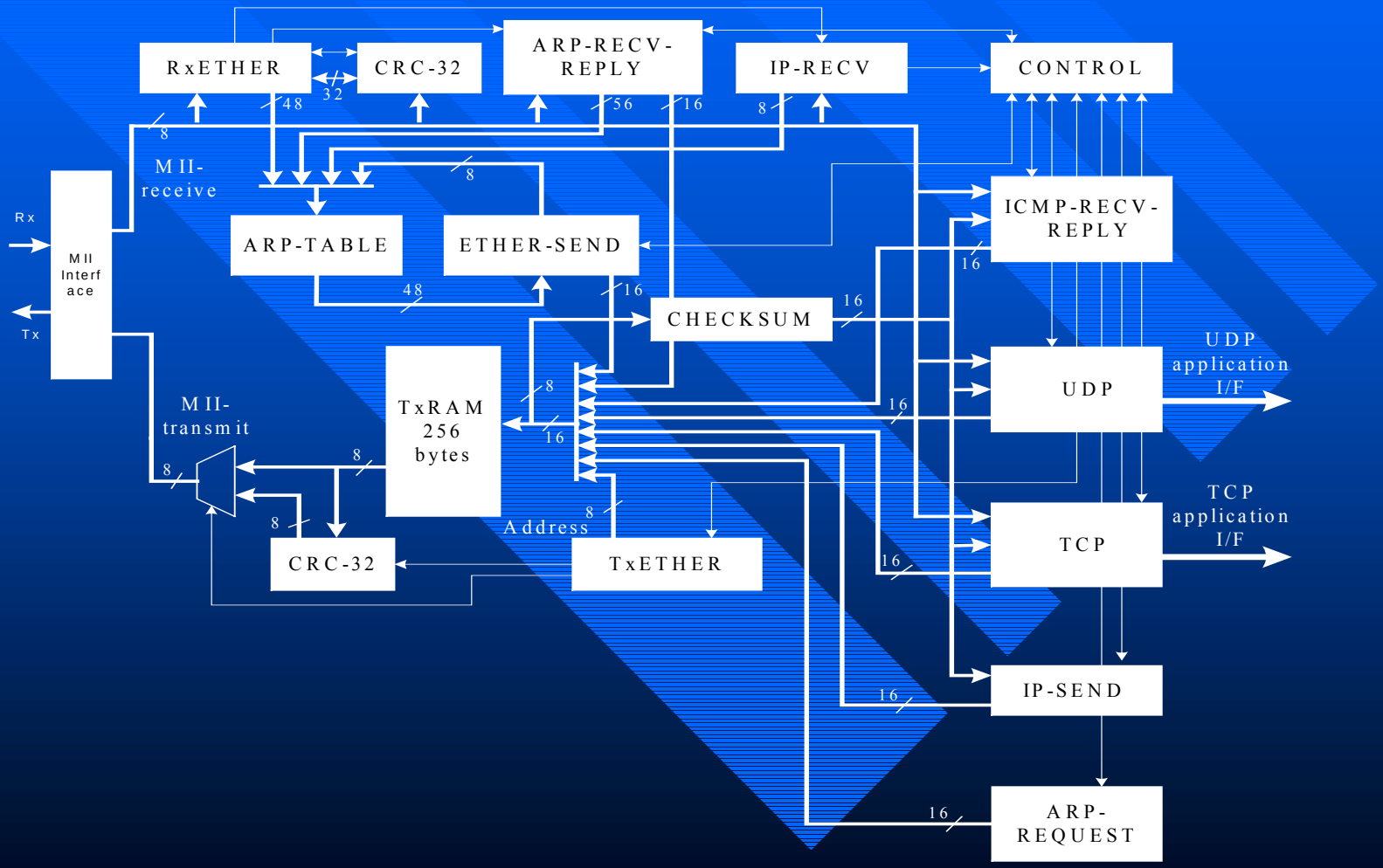
- Υποστήριξη εφαρμογών τόσο τύπου server όσο και εφαρμογών τύπου client και από το UDP και το TCP
- Υποστήριξη πολλών συνδέσεων ή sockets
- Παροχή Interface για επικοινωνία του επιπέδου Εφαρμογών με τη σχεδίαση με ένα μικρό σύνολο εντολών

- Βασικό Μειονέκτημα : Αδυναμία του πρωτοκόλλου TCP να εκμεταλλευτεί πλήρως το διαθέσιμο bandwidth του δικτύου

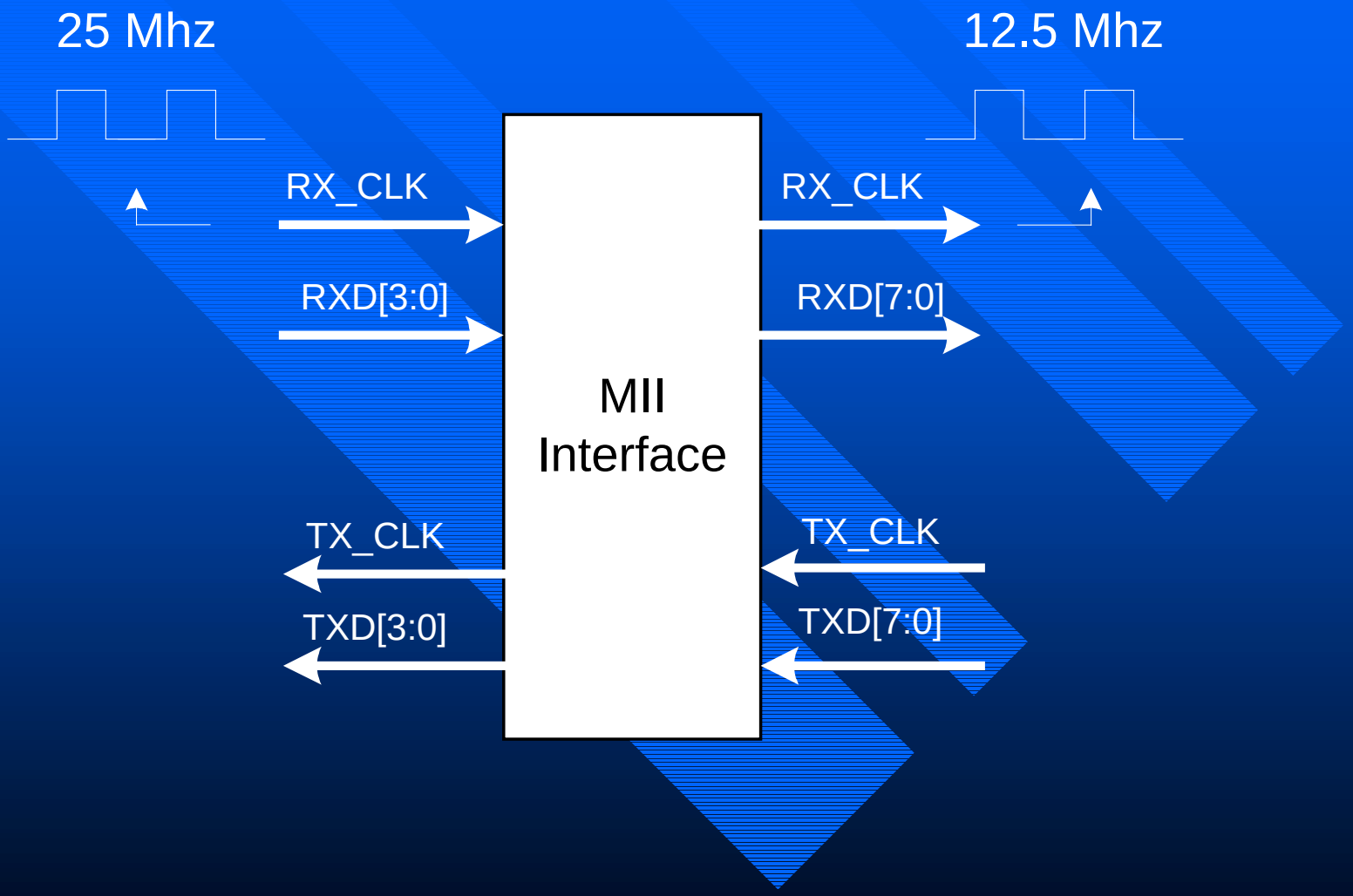
Προσθήκες Και Αλλαγές

- Προσθήκη της διεπαφής MII
- Αλλαγή του πίνακα ARP με έναν μεγαλύτερης χωρητικότητας
- Επανασχεδιασμός του υποσυστήματος TCP
- Περιφερειακές αλλαγές στα άλλα υποσυστήματα

Αρχιτεκτονική Της 3ης Γενιάς Της Σχεδίασης



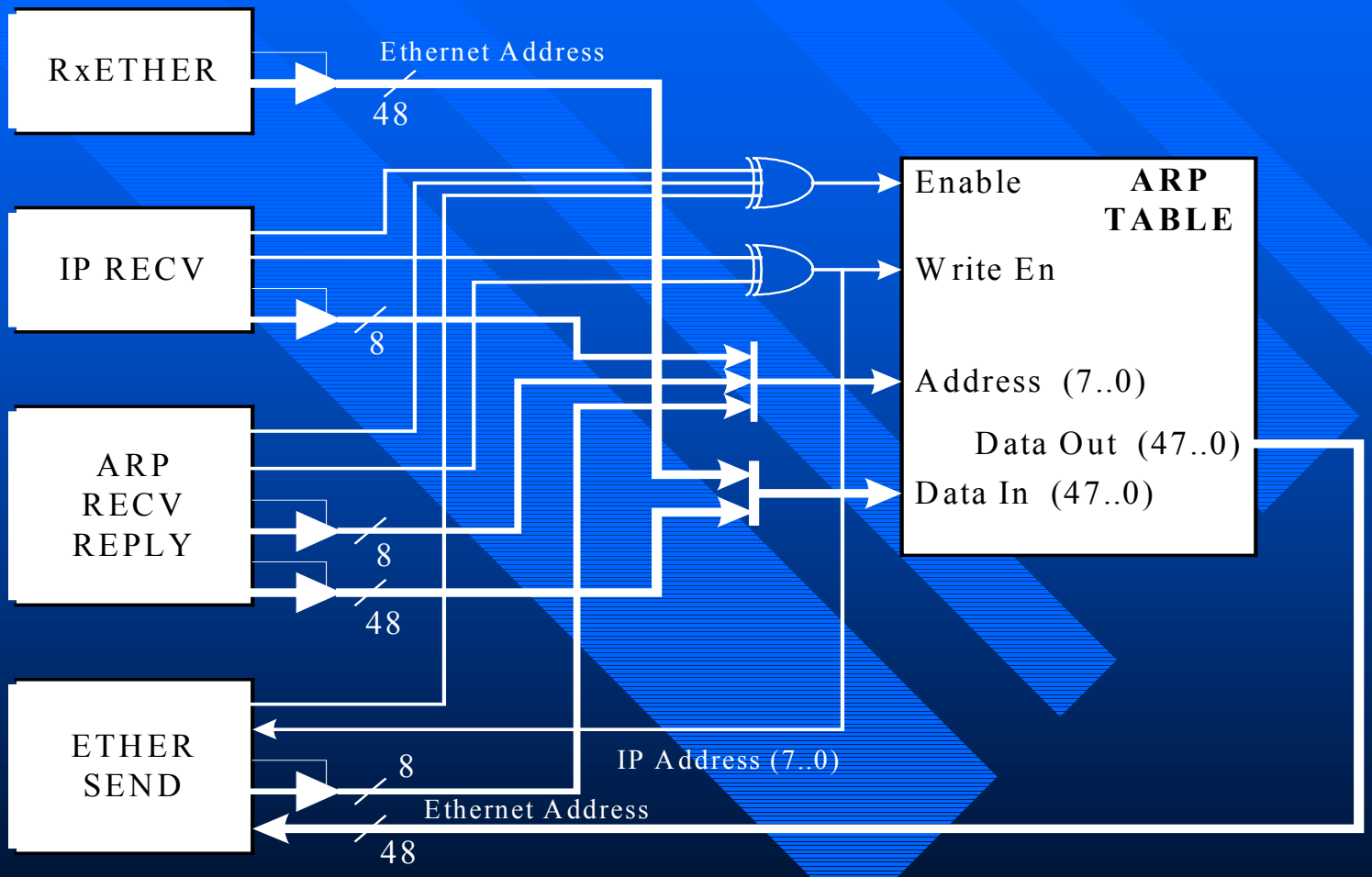
Το Υποσύστημα MII



Ο Πίνακας ARP (Ι)

- Ο πίνακας αντιστοιχίζει διευθύνσεις IP με φυσικές διευθύνσεις
- Είναι μια μονόπορτη στατική μνήμη
- Η διευθυνσιοδότηση γίνεται με τα 8 τελευταία bit της διεύθυνσης IP
  - Μειονέκτημα
    - » Μπορεί να συνυπάρξει στο ίδιο δίκτυο μόνο με 256 άλλα συστήματα

Ο Πίνακας ARP (II)



Το Υποσύστημα Control (I)

- Το υποσύστημα Control ασκεί τον έλεγχο της σχεδίασης
  - Αποπλέκει τα εισερχόμενα δεδομένα
  - Πολυπλέκει τα εξερχόμενα δεδομένα
  - Ασκεί τον κεντρικό έλεγχο στην μνήμη μετάδοσης

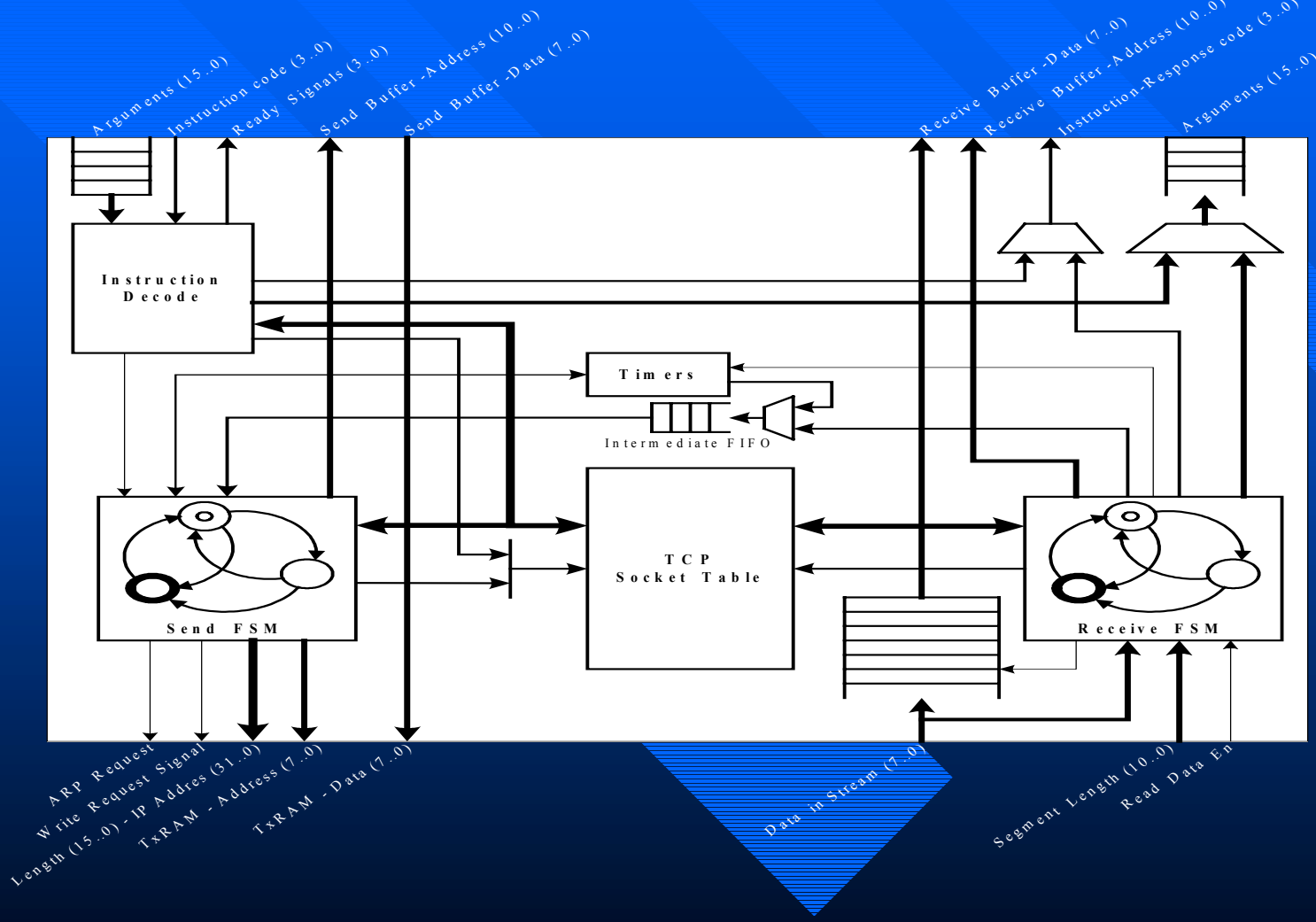
Το Υποσύστημα Control (II)

- Βασίστηκε στην ίδια λογική με την 2η γενιά
- Έγιναν αλλαγές ώστε να υποστηριχθεί το congestion control

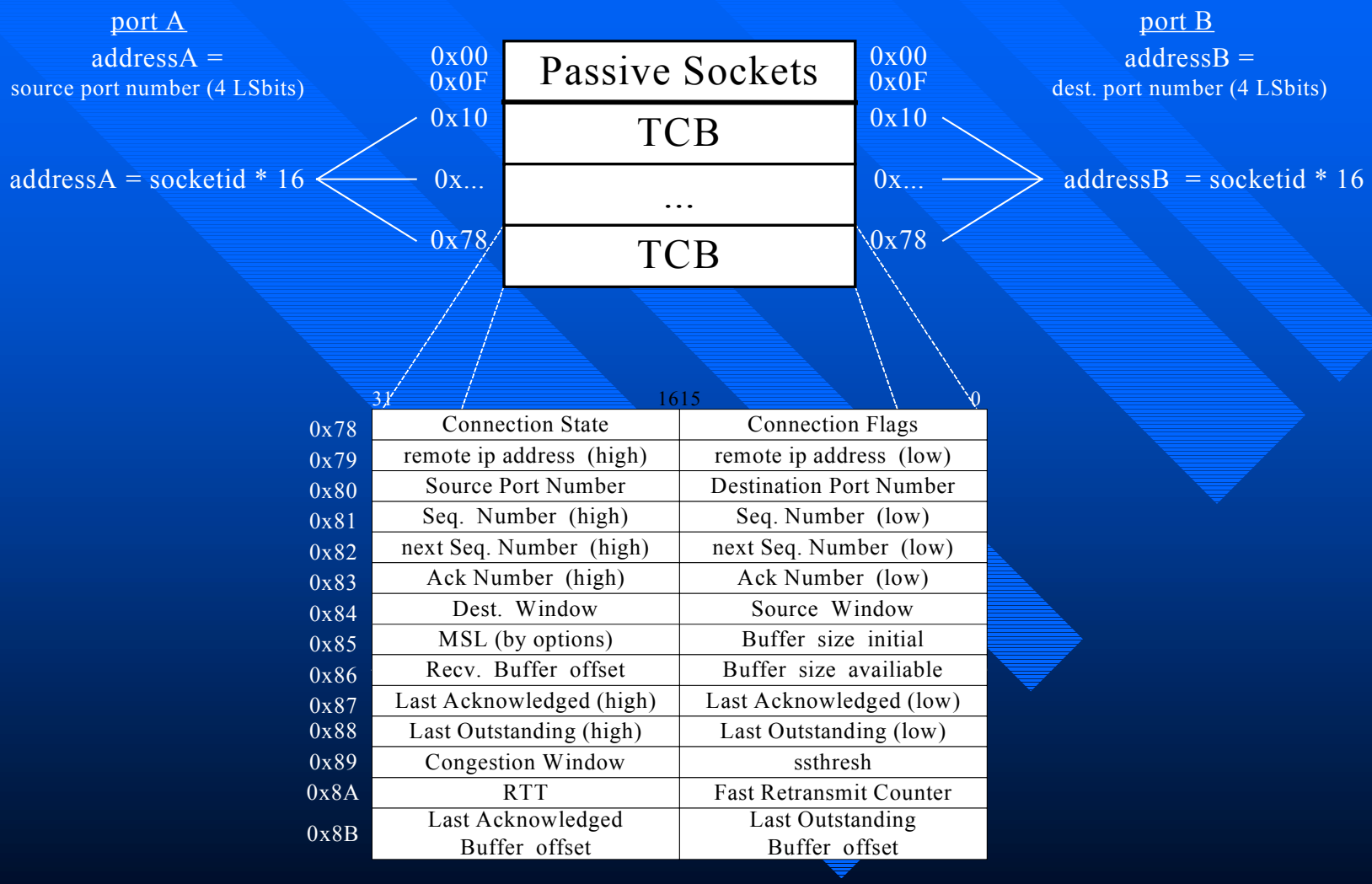
Το Υποσύστημα TCP

- Κρατήθηκε η αρχιτεκτονική της δεύτερης γενιάς
- Άλλαξε εξ'ολοκλήρου σε επίπεδο λειτουργικότητας

Αρχιτεκτονική Του TCP



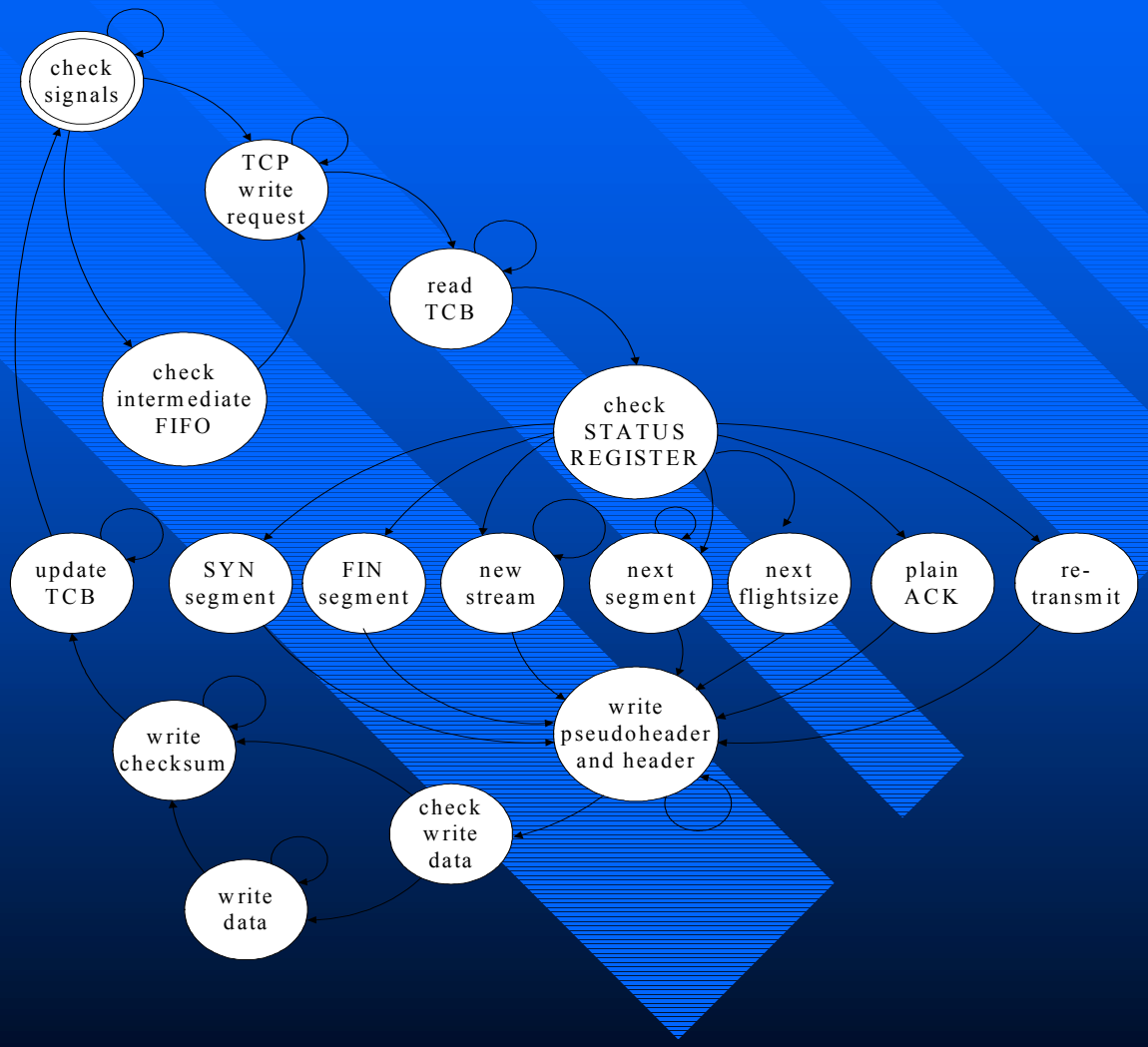
Ο Πίνακας των Sockets



Το Υποσύστημα Instruction Decode

- Αποκωδικοποιεί εντολές από το επίπεδο εφαρμογών
- Προετοιμάζει και δρομολογεί την εκτέλεσή τους
- Επιστρέφει κωδικούς επιτυχίας ή λάθους

Το Υποσύστημα Αποστολής



Timers και ενδιάμεση FIFO

- Μία FIFO πλάτους 7 bit και 16 θέσεων
  - Στέλνει εντολές στο υποσύστημα αποστολής
- Ένας timer για κάθε σύνδεση
  - 20 συνδέσεις αλλά υπάρχει δυνατότητα για 32

Ενδιάμεση FIFO

FIFO data (7..5)	FIFO data (4..0)	Περιγραφή
“100”	Socketid	Timeout με mode 00
“101’	Socketid	Timeout με mode 01
“110”	Socketid	Timeout με mode 10
“111”	Socketid	Timeout με mode 11
“010”	Socketid	Μετάδοση επόμενου Flightsize
“001”	Socketid	Μετάδοση απλής επιβεβαίωσης

TIMER

MODE	FUNCTION
00	Syn timer
01	retransmission timer
10	persistence timer
11	idle timer

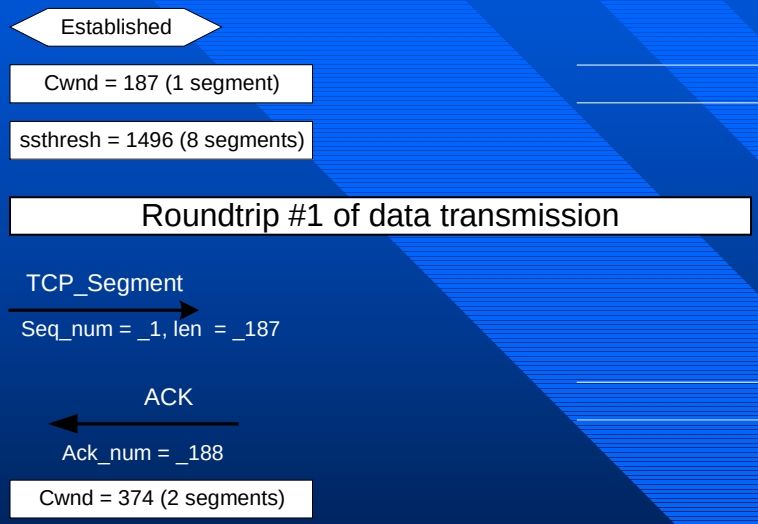
Το Υποσύστημα Λήψης

- Ελέγχει τα εισερχόμενα πλαίσια
- Σταματάει ή επανεκκινεί τους timers
- Οδηγείται από έναν control register

Προσομοίωση της Σχεδίασης

- Η προσομοίωση χωρίστηκε σε δύο στάδια
  - Προσομοίωση των περιφερειακών αλλαγών του TCP
  - Προσομοίωση του TCP μετά την προσθήκη του congestion control

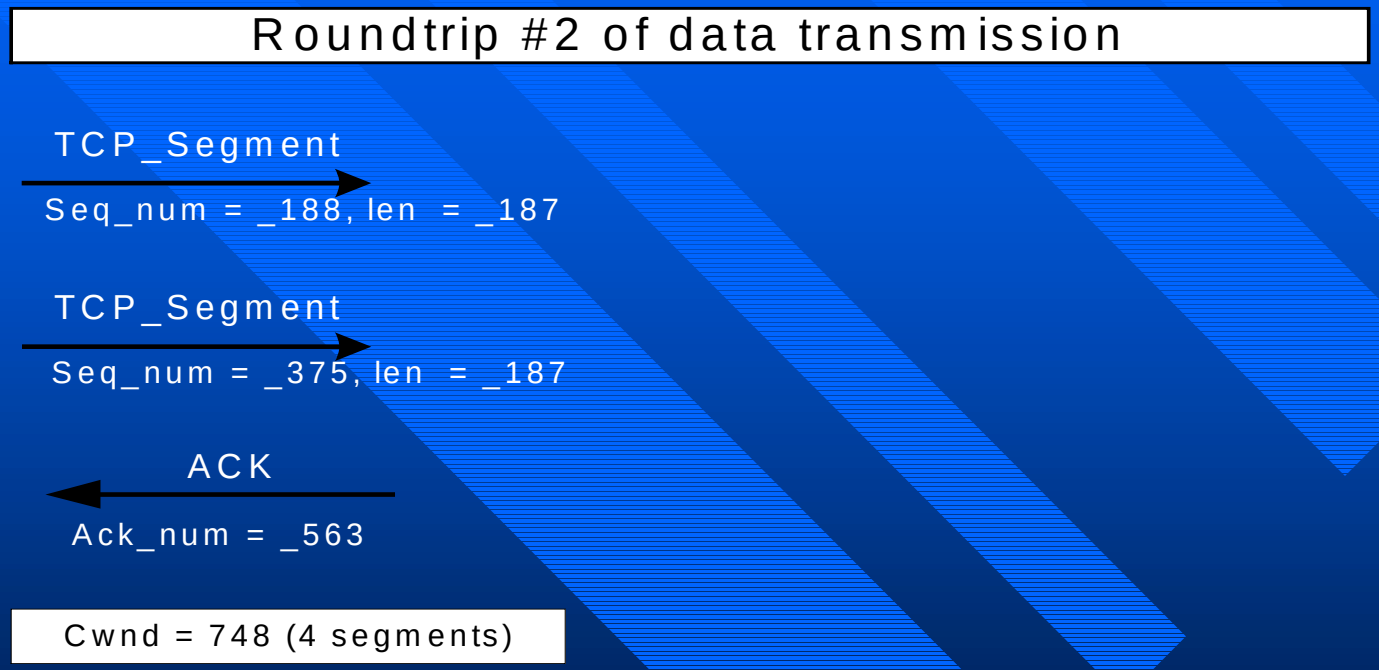
Σενάρια Προσομοιώσεων (Ι)



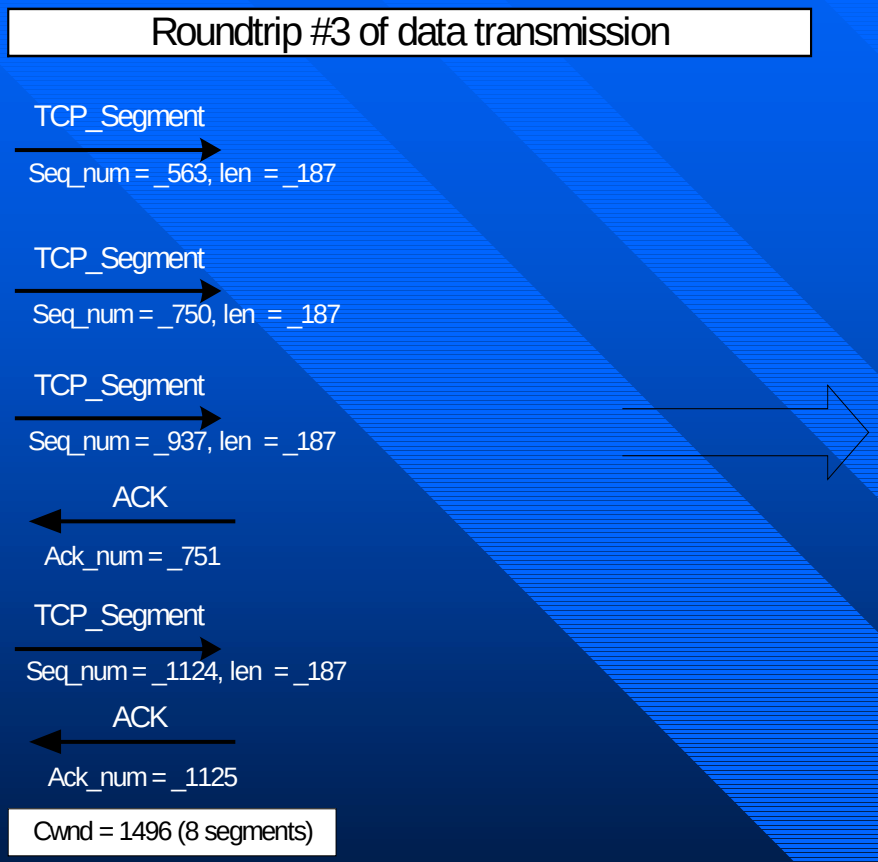
The connection's state is established and the congestion's window value is one segment. The slow start's threshold value is eight segments

The slow start algorithm controls the transmission which means that for every acknowledged segment, we increase the congestion window by one full size segment (cwnd = 2)

Σενάρια Προσομοιώσεων (II)

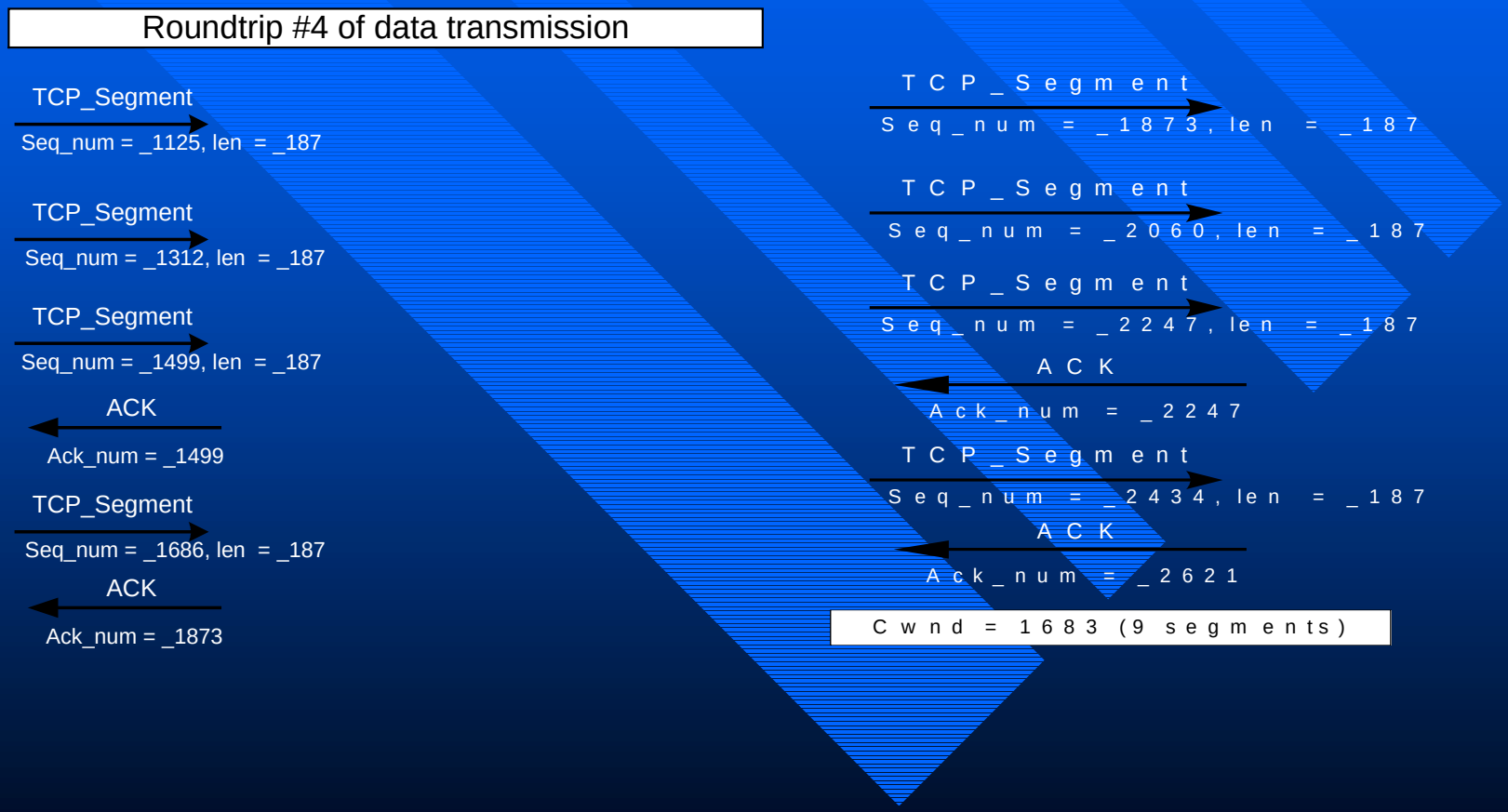


Σενάρια Προσομοιώσεων (III)

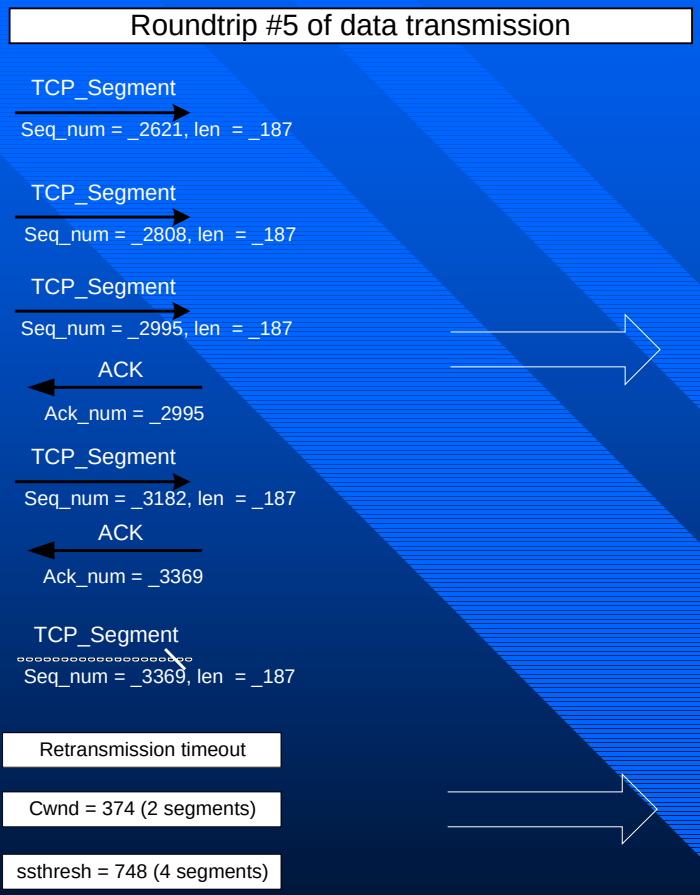


The value of the congestion window has become equal to the slow start threshold (cwnd = ssthresh). So at the next transmission the congestion avoidance algorithm takes over the transmission control

Σενάρια Προσομοιώσεων (IV)



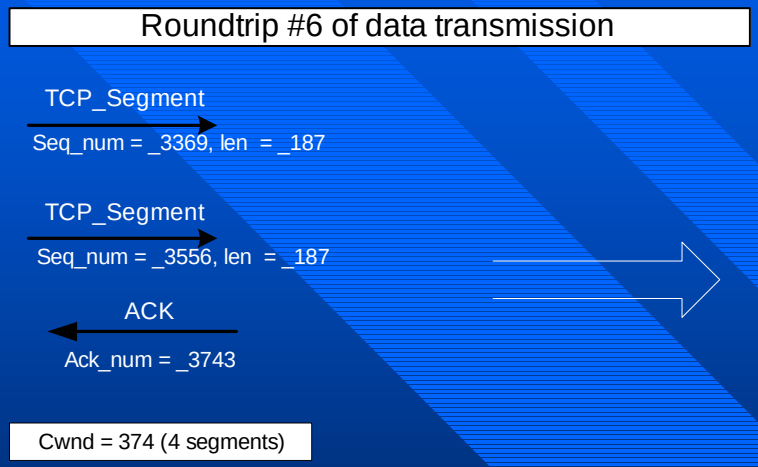
Σενάρια Προσομοιώσεων (V)



The congestion avoidance algorithm controls the data transmission. The segment with sequence number 3369 and the others that follow that are lost due to congestion on the network

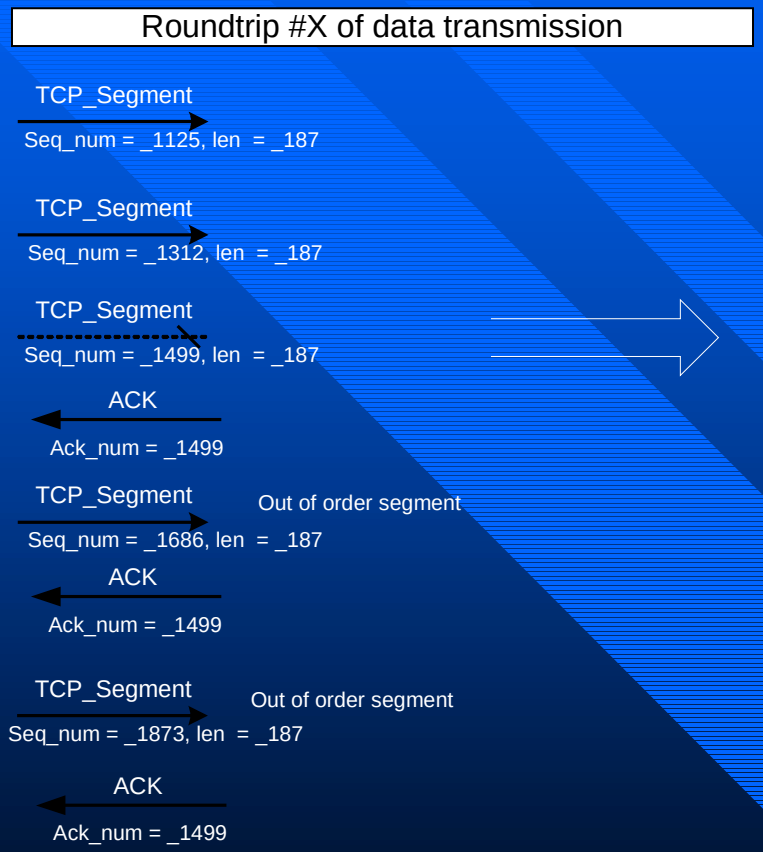
The retransmission timer time's out so we reduce the congestion window to the value of the restart window (cwnd = rw = 2 segments) and we reduce the slow start threshold to half the value it was (ssthresh = ssthresh / 2)

Σενάρια Προσομοιώσεων (VI)



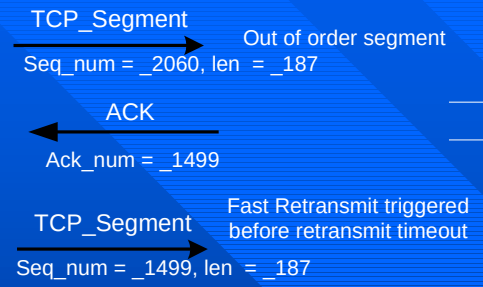
We continue the transmission with the new values of cwnd and ssthresh. Slow start algorithm controls the data transmission

Σενάρια Προσομοιώσεων (VII)



The segment with sequence number 1499 is lost but the next segments arrive at the receiver out of order

Σενάρια Προσομοιώσεων (VIII)



Cwnd = 374 (2 segments)

ssthresh = ssthresh / 2

Slow Start

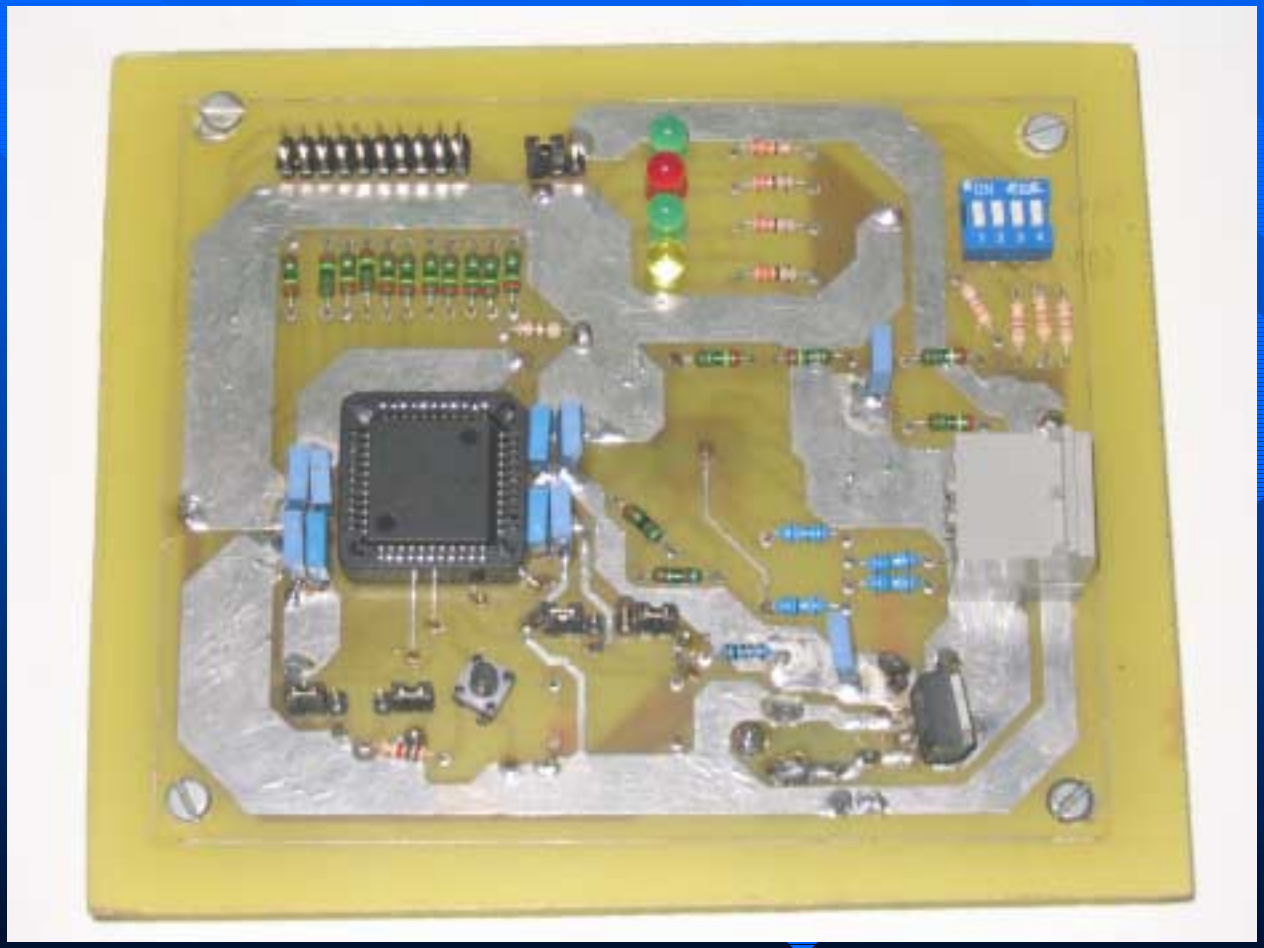
The receiver acknowledges immediately every out of order segment when the transmitter receives 3 duplicate acknowledges realizes that the segment with sequence number 1499 is lost and retransmits it immediately without waiting for the retransmission timer to expire

The transmission continues and slow start algorithm controls the data transmission

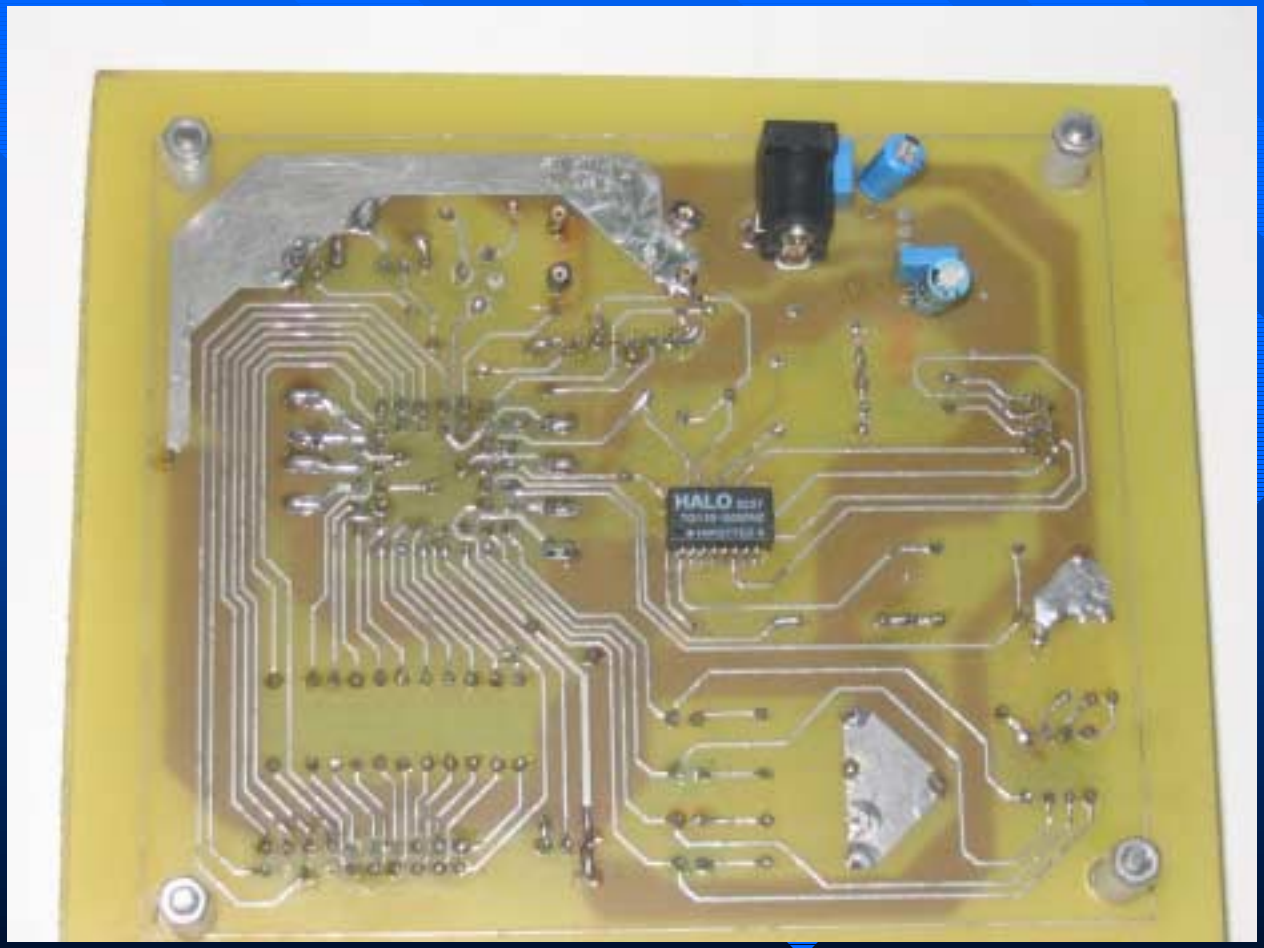
Υλοποίηση για το FPGA Virtex XCV1000bg560

Υποσύστημα	# slices	%	% του FPGA	Ταχύτητα (MHz)
MII	2	0.1	0.05	338.3
CRC-32	53	1.1	0.55	77.2
RxETHER	138	2.3	1.2	44
TxETHER	63	1.1	0.55	89.8
OUTMUX	34	0.6	0.3	160.8
ARP_RECV_REPLY	219	3.7	1.7	62.9
ARP_REQUEST	85	1.4	0.8	83.1
ETHER_SEND	126	2.1	1.0	74.3
CHECKSUM	40	0.8	0.4	92.7
IP_RECV	57	0.9	0.45	60.2
IP_SEND	99	1.7	0.8	83.9
CONTROL	129	2.3	1.2	89.9
ICMP_RECV_REPLY	111	1.9	0.9	78.3
UDP	506	8.6	4.2	48.6
TCP	4203	70.4	33.9	27.6
TOTAL	**5931**	**100**	**48**	**26.6**
Memory (Block RAM)			**11 / 36 Block RAMs (34%)**	

Ethernet Transceiver (Top View)



Ethernet Transceiver (Bottom View)



Συμπεράσματα

- Έχουμε πλέον ένα πλήρες IP core του TCP το οποίο μπορεί να εκμεταλλευτεί πλήρως το bandwidth του δικτύου
- Αυτή η σχεδίαση είναι χρήσιμη για να τοποθετηθεί είτε embedded συστήματα, είτε για να προσφέρει επιτάχυνση στην επεξεργασία του πρωτοκόλλου σε μεγαλύτερα συστήματα
- Πρόσφερε τεχνογνωσία και εμπειρία σε θέματα Τηλεπικοινωνιών και Δικτύων, σε θέματα σχεδίασης Hardware και σε ζητήματα χρήσης εργαλείων CAD

Μελλοντικές Επεκτάσεις

- Αλλαγή του πρωτοκόλλου Ethernet με κάποιο άλλο που προσφέρει διαφορετικές υπηρεσίες, όπως ATM, Bluetooth™
- Προσπάθεια για τη βελτίωση της απόδοσης του πρωτοκόλλου, και προσπάθεια για την αύξηση της ταχύτητας για να υποστηριχθούν πολύ γρήγορα δίκτυα
- Ανάπτυξη διαφόρων εφαρμογών

ΤΕΛΟΣ