ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ Η/Υ

ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΩΝ & ΥΛΙΚΟΥ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη μοντέλου ανοιχτού κώδικα για επαναχρησιμοποιήσιμο υποσύστημα πρωτοκόλλου TCP/IP

Ερμής Ιωάννης

Επιβλέπων Καθηγητής: Καθηγητής Δόλλας Απόστολος
Εξεταστική Επιτροπή: Καθηγητής Δόλλας Απόστολος
Καθηγητής Πατεράκης Μιχαήλ
Αν. Καθηγητής Πνευματικάτος Διονύσιος

Χανιά, Σεπτέμβριος 2004

Στους γονείς μου...

Ευχαριστίες

Στο σημείο αυτό, θα ήθελα να ευχαριστήσω τον Καθηγητή κ. Απόστολο Δόλλα τόσο για την ανάθεση αυτής της πραγματικά ενδιαφέρουσας διπλωματικής εργασίας,όσο και για την υποστήριξή σε όλη την διάρκεια της εργασίας αυτής. Επιπλέον, θα ήθελα να ευχαριστήσω τους καθηγητές Διονύσιο Πνευματικάτο και Μιχαήλ Πατεράκη για την συμμετοχή τους στην εξεταστική επιτροπή.

Ακόμα, θα ήθελα να απευθύνω ένα μεγάλο ευχαριστώ στον κ. Ιωσήφ Κοϊδή, χωρίς την βοήθεια του οποίου ο χρόνος ολοκλήρωσης της εργασίας αυτής θα επεκτείνονταν πολύ περισσότερο.

Ένα μεγάλο ευχαριστώ, τέλος, στον υπεύθυνο λειτουργίας του Εργαστηρίου ΕΜΥ, κ. Μάρκο Κιμιωνή και σε όλα τα υπόλοιπα μέλη του εργαστηρίου, μεταπτυχιακούς και προπτυχιακούς, για την τεράστια συμπαράστασή τους καθ'όλη την διάρκεια αυτής της διπλωματικής εργασίας....

Περιεχόμενα

Κεφάλαιο 1: Εισαγωγή

Στις μέρες μας, τα δίκτυα υπολογιστών είναι ένας από τους πιο συναρπαστικούς και σημαντικούς τεχνολογικούς τομείς. Το internet αλληλοσυνδέει εκατομμύρια – και ίσως δισεκατομμύρια – υπολογιστών παρέχοντας, έτσι, ένα παγκόσμιο δίκτυο επικοινωνίας, αποθήκευσης και υπολογιστικού εξοπλισμού. Επιπλέον, το internet σήμερα ενσωματώνει και την ασύρματη τεχνολογία αποκτώντας έτσι, ακόμα περισσότερα πλεονεκτήματα και νέες εφαρμογές.

Γίνεται, λοιπόν, αντιληπτό, ότι από την δεκαετία του 1960 που εμφανίστηκε για πρώτη φορά ο πρόγονος του internet, μέχρι και σήμερα η ανάπτυξη συνεχίζει με ραγδαίους ρυθμούς και φυσικά δεν πρόκειται να σταματήσει εδώ, καθώς οι νέες γενιές των μηχανικών, θα οδηγήσουν τις εξελίξεις σε εντελώς νέα και ανεξερεύνητα επίπεδα.

Η ραγδαία ανάπτυξη, όμως, του internet και γενικότερα των δικτύων των υπολογιστών, για να αποδώσει στο μέγιστο και παράλληλα να μπορεί να θεωρείται αξιόπιστη, απαιτεί την ύπαρξη ενός συνόλου κανόνων επικοινωνίας. Αυτόν τον ρόλο έχουν τα πρωτόκολλα επικοινωνίας: την εξασφάλιση των σωστών συνθηκών για την ομαλή, αδιάκοπη και αξιόπιστη λειτουργία των δικτύων.

Αξίζει να σημειωθεί, ότι οι πρώτες γενεές των πρωτοκόλλων ήταν προσαρτημένες στα τότε λειτουργικά συστήματα των υπολογιστών κάτι που με την εξέλιξη της τεχνολογίας τόσο σε επίπεδο λογισμικού, όσο και σε επίπεδο υλικού προσπεράστηκε, με αποτέλεσμα πολύ σύντομα η υλοποίηση τους να στραφεί και σε άλλες κατευθύνσεις. Σήμερα τα πρωτόκολλα επικοινωνίας μπορούν να υλοποιηθούν ακολουθώντας πιο ευέλικτες μεθόδους, όπως την χρήση μικροεπεξεργαστών ή μικροελεγκτών, την χρήση κυκλωμάτων ASIC (Application Specific Integration Circuits) ή κυκλωμάτων αναδιατασσόμενης λογικής (Field Programmable Gate Arrays – FPGAs).

Τα κυκλώματα αναδιατασσόμενης λογικής προγραμματίζονται με την τα IP Cores (Intellectual Property Cores), τα οποία αναπτύσσονται σε κάποια γλώσσα περιγραφής υλικού (Hardware Description Language – HDL). Το IP Core αποτελεί ένα συγκεκριμένο και δοκιμασμένο σύστημα ή υποσύστημα λογισμικού, το οποίο χρησιμοποιείται για την άμεση δημιουργία των λειτουργιών που επιθυμούμε να υλοποιήσουμε σε μία FPGA. Ένα IP Core δημιουργείται με την βοήθεια εργαλείων CAD και μπορεί να περιλαμβάνει οτιδήποτε, από αλγορίθμους ψηφιακής επεξεργασίας σημάτων, ελεγκτές μνήμης (Memory Controllers) μέχρι διεπαφές διαύλων και επεξεργαστές. Η αξία των IP Cores γίνεται ακόμα πιο αντιληπτή, αν λάβουμε υπόψιν μας τον χρόνο και το κόστος που θα έπρεπε να διαθέσουμε επιπλέον προκειμένου να δημιουργήσουμε τις παραπάνω πολύπλοκες λειτουργίες, εάν χρησιμοποιούσαμε τον κλασικό τρόπο. Τέλος, ας μην ξεχνάμε ότι στις μέρες μας, ο χρόνος είναι πολλές φορές σημαντικότερος παράγοντας από ότι το χρηματικό κόστος, καθώς στην βιομηχανία, συχνά, εκείνο που παίζει καθοριστική σημασία είναι ο χρόνος που βγαίνει ένα προϊόν στην αγορά.

1.1 Συνεισφορά της παρούσας διπλωματικής - Το "Open TCP/IP core"

Αξιοποιώντας όλα τα πλεονεκτήματα των IP Cores που αναφέρθηκαν παραπάνω, το Εργαστήριο Μικροεπεξεργαστών και Υλικού (ΕΜΥ) του Πολυτεχνείου Κρήτης, τα τελευταία χρόνια αναπτύσσει το πρωτόκολλο TCP/IP σε IP Core. Δεδομένης της πολυπλοκότητας, αλλά και της σημασίας του πρωτοκόλλου αυτού, η προσπάθεια συνεχίζεται και στην διπλωματική αυτή, βασιζόμενη στην εργασία που έγινε κατά την διάρκεια της διπλωματικής εργασίας των κ.κ. Κάχρη, Ζήση και Κοϊδή.

Τα πρωτόκολλα που υποστηρίζονται και σε αυτήν την εργασία είναι το πρωτόκολλο του επιπέδου MAC (Media Account Layer), το πρωτόκολλο ARP (Address Resolution Protocol), το ICMP (Internet Control Message Protocol), το IP (Internet Protocol), το UDP (User Datagram Protocol) και το TCP (Transmission Control Protocol). Τα προαναφερθέντα υποστηριζόμενα πρωτόκολλα σε σχέση με τα επίπεδα OSI (OSI layers – Open Systems Interconnection Reference Model layers) φαίνονται στον πίνακα 1.

Applications *User Application*	
Presentation *Upper Layer Protocols*	
Session *Sockets*	
Transport *UDP, TCP*	***IP Core***
Network *ARP, IP, ICMP*	
Data Link *LLC, MAC*	
Physical *MII, PMI, MMD, PMD*	***Transceiver***

Πίνακας 1: Τα επίπεδα OSI.

Το γενικό interface του TCP/IP Core παρέμεινε αμετάβλητο, καθώς οι αλλαγές επικεντρώθηκαν εσωτερικά στο υποσύστημα TCP. Το υποσύστημα αυτό, το οποίο ήταν το αντικείμενο και της προηγούμενης διπλωματικής εργασίας, αν και σε σχέση με τις προηγούμενες εκδόσεις του δούλευε καλύτερα, καθώς είχε εμπλουτιστεί με περισσότερες δυνατότητες όπως ο έλεγχος συμφόρησης (Congestion Control), η ταυτόχρονη λήψη και αποστολή πολλαπλών τμημάτων, ο μεγαλύτερος αριθμός συνδέσεων και η διεπαφή με το MII, λόγω της πολυπλοκότητάς του, έπρεπε να διασπαστεί σε επιμέρους υποσυστήματα ώστε να επιτυγχάνεται παραλληλισμός των λειτουργιών και κατά συνέπεια, καλύτερη απόδοση.

Στην νέα γενιά του TCP/IP Core (το οποίο παρακάτω θα αναφέρουμε με την ονομασία “Open TCP/IP Core”) το υποσύστημα του TCP διασπάστηκε σε πολλά επιμέρους υποσυστήματα, με αποτέλεσμα την σαφώς αποδοτικότερη λειτουργία του και επιπλέον, την υποστήριξη κάποιων νέων σημαντικών χαρακτηριστικών όπως:

- Την αποκωδικοποίηση πολλών εντολών από το επίπεδο των εφαρμογών και την αποθήκευσή τους σε προσωρινή μνήμη κατά την διάρκεια που το σύστημα απασχολείται με την αποστολή ή λήψη κάποιας ροής ή την εκτέλεση μίας άλλης εντολής.
- Την υποστήριξη παράλληλης αποστολής/λήψης πολλών ροών.
- Την υλοποίηση επιπλέον χαρακτηριστικών του Congestion Control όπως ο persistence και ο idle timer.
- Την υποστήριξη επιπλέον εσωτερικών εντολών που προκύπτουν από ελέγχους, όπως εκείνον της συμφόρησης, και την σημαντική

βελτιστοποίηση των ήδη υποστηριζόμενων από την προηγούμενη έκδοση του υποσυστήματος, όπως οι Send Probe, Send Plain Ack, Initialize Congestion Window κλπ.

- Τον ακριβή υπολογισμό του Round Trip Time για κάθε νέο flight size.

Η διάσπαση του υποσυστήματος αποστολής/λήψης πλαισίων TCP υλοποιήθηκε σε βάθος μέχρι και τριών επιπέδων, ενώ, παράλληλα προστέθηκαν και αρκετά νέα υποσυστήματα προκειμένου να βελτιωθεί η απόδοση, να υποστηριχθούν νέες δυνατότητες και να βελτιωθεί η δομή. Η διάσπαση κρίθηκε αναγκαία και για λόγους σωστής συντήρησης του κώδικα, καθώς τώρα, δίνεται η δυνατότητα να επέμβουμε ξεχωριστά σε κάθε αυτόνομο υποσύστημα.

Τέλος, σημαντικό βάρος δόθηκε στον έλεγχο του υποσυστήματος. Έτσι, δημιουργήθηκε test bench, το οποίο επιτρέπει τον έλεγχο σε πολύ μεγαλύτερο εύρος χρόνου και επομένως σε πλήθος δυνατών περιπτώσεων. Όπως αναφέρεται και στο κεφάλαιο της δοκιμής – πιστοποίησης, το test bench αποτελείται από ένα τμήμα κώδικα που έχει την δυνατότητα να δίνει σήματα σε εισόδους και να ελέγχει τις εξόδους καταγράφοντας ή όχι τα αποτελέσματα σε αρχεία. Η λειτουργία ενός test bench εμπλουτίζεται με την χρήση ροών εισόδου/εξόδου, που παρέχουν οι βασικές βιβλιοθήκες της vhdl.

Έτσι, χρησιμοποιώντας test bench τα εισερχόμενα ή εξερχόμενα πακέτα, πλέον, μπορούν να αποθηκευτούν σε αρχεία και ο έλεγχος τους να είναι πληρέστερος και ακριβέστερος. Επιπλέον, το test bench, παρέχει ειδικά μηνύματα για σήματα τα οποία προέρχονται από το υποσύστημα TCP προς τα βοηθητικά επιμέρους συστήματα, όπως η μνήμη αποστολής, η μνήμη λήψης κλπ.

1.2 Η χρησιμότητα του Open TCP/IP Core

Η χρησιμότητα της υλοποίησης του TCP/IP πρωτοκόλλου 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 και μάλιστα, με εύκολη προσαρμοστικότητα. Παράλληλα, μπορεί να παρέχει την ταχύτητα που προσφέρουν οι σχεδιάσεις σε hardware, δεδομένου ότι σκοπός της

σχεδίασης είναι το πρωτόκολλο αυτό καθαυτό, χωρίς άλλες «παράπλευρες» απώλειες σε επεξεργαστική ισχύ, φαινόμενο που λαμβάνει χώρα στις περιπτώσεις που το πρωτόκολλο τρέχει σε επεξεργαστές κοινής χρήσης, όπως για παράδειγμα αυτούς των Ηλεκτρονικών Υπολογιστών.

Το TCP/IP stack μπορεί να αξιοποιηθεί άμεσα από τις εφαρμογές για home networking. Οι εφαρμογές home networking προσφέρουν την δυνατότητα του ελέγχου ή της παρακολούθησης από απομακρυσμένα σημεία και λειτουργούν πάντα σε σύνδεση με το internet. Άλλη μία εφαρμογή είναι η χρήση του Core σε μία συσκευή για content based routing, η οποία δρομολογεί τα πακέτα με βάση ο περιεχόμενό τους, χωρίς την παρέμβαση κάποιου προγράμματος,

1.3 Η οργάνωση της αναφοράς

Στα επόμενα κεφάλαια της εργασίας αυτής θα δούμε με μεγάλη λεπτομέρεια τις διαφοροποιήσεις που προέκυψαν στο Open TCP/IP Core κατά την μετάβασή του από την τρίτη στην τέταρτη γενεά.

Πιο αναλυτικά, στο επόμενο κεφάλαιο, το Κεφάλαιο 2, γίνεται μια σύντομη αλλά ουσιαστική αναφορά στην σχετική έρευνα που γίνεται στον τομέα υλοποίησης του TCP/IP stack. Δίνονται περισσότερες λεπτομέρειες για την εργασία που έγινε στα πλαίσια των τριών προηγούμενων διπλωματικών εργασιών και τέλος, δίνεται έμφαση στα νέα, βελτιωμένα, χαρακτηριστικά του Open TCP/IP Core.

Στο Κεφάλαιο 3, γίνεται εκτενής αναφορά στην αρχιτεκτονική του Open TCP/IP Core, με στόχο την κατανόηση των νέων λειτουργιών και την αλλαγή που επιτελέστηκε στο υποσύστημα της αποστολής/λήψης πλαισίων TCP με χρήση block diagrams, ώστε να γίνει όσο το δυνατόν πιο ξεκάθαρος ο λόγος της τμηματοποιήσης του υποσυστήματος.

Στο Κεφάλαιο 4, θα εισχωρήσουμε με ακόμη μεγαλύτερη λεπτομέρεια στο υποσύστημα αποστολής/λήψης πλαισίων, για να δούμε με μεγάλη ακρίβεια τα επιμέρους υποσυστήματα που δημιουργήθηκαν, όπως επίσης και τον τρόπο λειτουργίας και συνεργασίας τους.

Στο Κεφάλαιο 5, θα ασχοληθούμε με τα test benches που υλοποιήθηκαν με στόχο την αναλυτική δοκιμή του υποσυστήματος αποστολής/λήψης πλαισίων για την

ορθή λειτουργία του. Το τμήμα της δοκιμής του υποσυστήματος αποτελεί θέμα ιδιαίτερης σημασίας, καθώς, σε όλη την διαδικασία της προετοιμασίας αυτής της διπλωματικής εργασίας, απέσπασε σημαντικότατο χρονικό διάστημα.

Στο Κεφάλαιο 6, θα γίνει αναφορά στα Open Cores και τον λόγο για τον οποίο το Open TCP/IP Core θα συμπεριληφθεί σε αυτήν την τεράστιας σημασίας βιβλιοθήκη γνώσεων.

Στο Κεφάλαιο 8, θα αναλύσουμε τα συμπεράσματα που προέκυψαν κατά την διάρκεια της εργασίας αυτής, καθώς και για τις μελλοντικές προσθήκες που μπορεί να υποστεί το σύστημα, για ακόμη πιο εύρυθμη και αποδοτική λειτουργία.

Τέλος, στο παράρτημα αναφέρονται περιληπτικά κάποιες λεπτομέρειες σχετικά με τα πρωτόκολλα που συναντώνται στην παρούσα αναφορά με στόχο την καλύτερη κατανόηση της εργασίας.

Κεφάλαιο 2 : Το Open TCP/IP Core

Τα τελευταία χρόνια πολλές εταιρίες και αρκετά εκπαιδευτικά ιδρύματα ανά τον κόσμο δραστηριοποιούνται στην μεταφορά των πιο γνωστών πρωτοκόλλων σε χαμηλότερο επίπεδο. Μέχρι πριν από λίγα χρόνια, τα πρωτόκολλα υλοποιόντουσαν σε κάποιο πρόγραμμα, το οποίο έτρεχε σε επεξεργαστή γενικού σκοπού. Αυτό, όπως είδαμε στο προηγούμενο κεφάλαιο είχε ως συνέπεια την κατανάλωση επεξεργαστικής ισχύος - κάτι που ας μην ξεχνάμε πριν χρόνια αποτελούσε ακανθώδες πρόβλημα, καθώς οι επεξεργαστές εκείνης της εποχής δεν διέθεταν την ταχύτητα που διαθέτουν οι σημερινοί - η οποία συχνά ήταν εντελώς απαραίτητη για να μπορούν να τρέξουν οι υπόλοιπες εφαρμογές που διαμοιράζονταν τον επεξεργαστή.

Στις μέρες μας είναι γενικώς αποδεκτό, ότι τα επόμενα χρόνια όλο και περισσότερες συσκευές με ανάγκη δικτύωσης θα είναι ενσωματωμένα (embedded) συστήματα και όχι εξαρτώμενα από pc. Σύμφωνα με την πιο δημοφιλή πρόβλεψη, μάλιστα, μέχρι το 2010 το 95% των συστημάτων θα είναι ενσωματωμένα συστήματα και όχι standalone computers!

Μια ευρέως αποδεκτή και συνάμα απλή λύση, είναι η χρήση 8-bit μικροελεγκτών, όπως ο Rabbit 2000, οι AVR, οι PIC κλπ σε συνδυασμό με Ethernet MAC [18]. Η παραπάνω υλοποίηση, όμως, παρουσιάζει τρωτά σημεία, όπως τα μειωμένα resources που διαθέτει ένας 8-bit μικροελεγκτής, τα οποία μειώνουν την λειτουργικότητά του. Ένας άλλος παράγοντας που δρα αρνητικά στην υλοποίηση αυτή είναι οι επιθέσεις denial of service (DOS attacks), οι οποίες είναι εύκολο να θέσουν εκτός λειτουργίας έναν τέτοιου είδους μικροελεγκτή.

Αναπτύσσοντας περισσότερο την συζήτησή μας και τις σκέψεις μας γύρω από μειονεκτήματα και τα πλεονεκτήματα των hardware υλοποιήσεων, πρέπει να αναφέρουμε ότι η χρήση μικροελεγκτή ή μικροεπεξεργαστή προσφέρει μία φθηνή λύση και μία ευκολία ανάπτυξης, καθώς μπορούν να χρησιμοποιηθούν γενικότερες δομές από τις υλοποιήσεις που υπάρχουν για τους επεξεργαστές γενικού σκοπού,

λόγω της ομοιότητάς τους με τους μικροελεγκτές ή τους μικροεπεξεργαστές. Οι ομοιότητες αυτές, όμως, εξαντλούνται μετά από ένα σημείο, δεδομένου ότι ένας μικροελεγκτής ή μικροεπεξεργαστής παρουσιάζει αρκετές ιδιαιτερότητες, με αποτέλεσμα η μεταφορά ενός προγράμματος να απαιτεί προσαρμογή πριν την μετακίνησή του από μικροελεγκτή σε μικροελεγκτή, συχνά ακόμη και της ίδιας εταιρίας! Τέλος, δεν θα πρέπει να παραλείψουμε, ότι η απόδοση συνήθως δεν είναι ικανοποιητική, λόγω της εκτέλεσης ενός μεγάλου όγκου εντολών, προκειμένου να εξασφαλιστεί η επιθυμητή λειτουργικότητα και κατά δεύτερο λόγο, εξαιτίας της χαμηλής ταχύτητας.

Η υλοποίηση του TCP/IP stack σε ASIC πραγματοποιείται κυρίως σε περιπτώσεις που το ζητούμενο είναι η ταχύτητα, η απόδοση και η μικρή κατανάλωση. Τέτοιου είδους υλοποιήσεις, προσφέρουν διεπαφές ικανές να συνδέσουν το ολοκληρωμένο κύκλωμα είτε με μικροελεγκτή είτε με υπολογιστή, αυξάνοντας, έτσι, την συνολική απόδοση και ευελιξία. Στην περίπτωση, όμως, που χρειαστεί να γίνουν αλλαγές για την βελτίωση, ας υποθέσουμε, της απόδοσης τότε τα πράγματα είναι δύσκολα, καθώς η υλοποίηση αυτή δεν υποστηρίζει κανενός είδους αλλαγή μετά την κατασκευή. Τέλος, αξίζει να σημειωθεί, ότι η υλοποίηση αυτή για μικρό όγκο παραγωγής είναι οικονομικά ασύμφορη.

Μία τρίτη λύση είναι η υλοποίηση του TCP/IP με τη μορφή IP Core χρησιμοποιώντας κάποια γλώσσα περιγραφής υλικού, όπως η VHDL. Άμεσο πλεονέκτημα από την χρήση μίας γλώσσας περιγραφής υλικού είναι η αδέσμευτη σχεδίαση, δεδομένου ότι κάθε γλώσσα περιγραφής υλικού δεσμεύεται από διεθνή πρότυπα, πρότυπα που καλούνται να τηρήσουν και οι κατασκευαστές των προγραμματιζόμενων συσκευών. Πέρα από τα σχεδόν ανύπαρκτα προβλήματα ασυμβατότητας μεταξύ των επαναπρογραμματιζόμενων συσκευών, άλλο ένα πολύ σημαντικό πλεονέκτημα, έναντι των προηγούμενων δύο υλοποιήσεων, είναι η δυνατότητα δημιουργίας μίας αρχιτεκτονικής που υλοποιεί τις λειτουργίες του πρωτοκόλλου με υποσυστήματα που έχουν συγκεκριμένη λειτουργία και παράλληλα, η διατήρηση υψηλού επιπέδου που συνεπάγεται η χρήση μίας τέτοιας γλώσσας. Η διατήρηση υψηλού επιπέδου προσφέρει με την σειρά της την δυνατότητα της προσαρμογής τους συστήματος, σύμφωνα με τις εκάστοτε ανάγκες, αφού ο επαναπρογραμματισμός των συσκευών είναι άμεσος, μεταφέροντας όλες τις αλλαγές στην σχεδίαση. Όσον αφορά στην ταχύτητα, παράγοντας ο οποίος τις περισσότερες φορές είναι κρίσιμος και δεσμευτικός για μία σχεδίαση, αυτή διατηρείται σε

ικανοποιητικά επίπεδα. Η κατανάλωση που αποτελεί, επίσης, κρίσιμο παράγοντα μπορεί να παραμείνει σε χαμηλά επίπεδα, κάνοντας την κατάλληλη μελέτη κατά την σχεδίαση. Ένα ακόμα σημαντικότατο πλεονέκτημα είναι η αυτούσια χρήση και η ολοκλήρωση ενός IP Core σε μεγαλύτερα συστήματα, με σαφώς περισσότερες δυνατότητες, που βρίσκονται στην ίδια συσκευή.

Στις επόμενες παραγράφους θα πραγματοποιήσουμε μία σύντομη έρευνα στις υλοποιήσεις κάποιων εταιριών ή εκπαιδευτικών ιδρυμάτων, πάνω στο θέμα των TCP/IP stacks, προκειμένου να γίνει ακόμα πιο αντιληπτό το ενδιαφέρον των τελευταίων ετών, για τα embedded συστήματα.

2.1 CMX Systems: CMX-TCPIP ™

Η CMX [19] αποτελεί μία από τις εταιρίες που δραστηριοποιούνται στο χώρο των embedded συστημάτων και προσφέρει το TCP/IP stack για διάφορους μικροελεγκτές και μικροεπεξεργαστές όπως οι: 80x86, ARM, Infeon 80C16x, Motorola 683xx, Motorola Coldfire, PowerPc, Renesas (Hitachi) h8/300H, Texas Instuments DSP C54x και άλλους.

Τα πρωτόκολλα και οι λειτουργίες που υποστηρίζονται είναι τα IP, ICMP/Raw IP, TCP, UDP, ARP, SNTP, DNS, ενώ επιπλέον σαν πρόσθετα προσφέρονται τα: BOOTB, TFTP, DHCP, FTP, IMAP4, NAT, POP3, PPP, SLIP, PPPoE, SMTP, SNTP, TELNET κ.α.

Η μεταφορά του κώδικα στον εκάστοτε μικροελεγκτή ή μικροεπεξεργαστή, γίνεται με την χρήση compiler για την μετάφραση σε γλώσσα μηχανής, ενώ, η επικοινωνία με το επίπεδο εφαρμογών επιτυγχάνεται με την κλήση συναρτήσεων που είναι υλοποιημένες σε C.

Η δημιουργία των sockets γίνεται σύμφωνα με τους διαθέσιμους πόρους. Το CMX-TCPIP ™ τρέχει κάνοντας χρήση του πυρήνα λειτουργικού συστήματος πραγματικού χρόνου CMX-RTX, ενώ, δίνεται η δυνατότητα λειτουργίας και χωρίς την χρήση του παραπάνω πυρήνα.

2.2 KADAK: KwikNet ™ TCP/IP Stack

Η εταιρία KADAK [20] προσφέρει την δική της λύση με το KwikNet ™ TCP/IP Stack, υποσχόμενη στον σχεδιαστή σημαντική μείωση σε κόστος και χρόνο σχεδίασης. Και στην περίπτωση αυτή, υποστηρίζεται μία μεγάλη γκάμα επεξεργαστών από τις οικογένειες 80x86/88 (Intel: 80186/88,386, 486 AMD: Am186, Am188, NEC: V53, V35, V25 κ.α.), Motorola 68000, Motorola ColdFire, PowerPC, ARM (Arm Ltd, NEC, Atmel, Cirrus Logic, Samsung, Intel), MIPS, BlackFin (Analog Devices).

Για κάθε επεξεργαστή προσφέρεται στον σχεδιαστή διαφορετική σειρά εργαλείων ανάπτυξης, ενώ, στην ιστοσελίδα δίνονται πληροφορίες για την χρήση της μνήμης ανά επεξεργαστή που διευκολύνουν κατά πολύ την επιλογή του κατάλληλου επεξεργαστή.

Τα υποστηριζόμενα πρωτόκολλα είναι τα TCP, UDP, IP, ICMP, ARP, DHCP, DHCP, DNS. Προαιρετικά παρέχονται τα PPP, TFTP, FTP, HTTP, TELNET, SMTP και SNTP.

2.3 Microchip: FREE TCP/IP Stack

Η Microcrochip [21] κάνει μία πρόταση η οποία, έχει μηδαμινό κόστος για τον σχεδιαστή, καθώς το stack που προτείνει διανέμεται δωρεάν για download από το website της. Αν και δεν αναφέρεται ξεκάθαρα, ο κώδικας που διατίθεται δωρεάν απευθύνεται μόνο για την σειρά μικροελεγκτών της εταιρίας.

Το stack υποστηρίζει τα πρωτόκολλα IP, ARP, ICMP, TCP, UDP, HTTP, FTP, DHCP, και MPFS. Παράλληλα, προσφέρεται δωρεάν υποστήριξη σε περίπτωση χρήσης του μικροελεγκτή C18 της ίδιας εταιρίας. Αξίζει να σημειωθεί ότι η συγκεκριμένη υλοποίηση δεν στηρίζεται σε πυρήνα λειτουργικού συστήματος πραγματικού χρόνου (RTOS independent). Τέλος, σύμφωνα πάντα με τα specification που δίνονται από την εταιρία, το stack υλοποιεί το πρωτόκολλο TCP με την μορφή ενός state machine, ενώ, ο κώδικας είναι κατασκευασμένος με αρθρωτό τρόπο.

2.4 MicroDigital: SmxNet ™

Η MicroDigital [22] προσφέρει ως λύση το SmxNet™ . Το συγκεκριμένο stack έχει την προσφέρει την δυνατότητα της ρύθμισης από τον σχεδιαστή για την ελάχιστη απαιτούμενη μνήμη που κάνει χρήση, προκειμένου να λειτουργεί χωρίς προβλήματα. Επίσης, παρέχει λεπτομερείς αναφορές (logs) λειτουργίας και σφαλμάτων.

Τα πρωτόκολλα που υποστηρίζονται είναι τα ARP, RARP, BOOTP, DNS, ICMP, IGMP, RIP, SLIP και CSLIP, ενώ, προαιρετικά παρέχεται υποστήριξη και για τα DHCP, FTP, TFTP, IMAP, NAT, NFS, POP3, PPP, PPPoE, SMB, SMTP, SNTP, SSL, TELNET και 802.11b (Wireless).

Στην ιστοσελίδα της εταιρίας παρέχονται με πολύ αναλυτικό τρόπο οι εντολές προς το stack, ενώ διευκρινίζεται ότι ανάμεσα στις εντολές αυτές υπάρχουν εννέα εντολές που υποστηρίζουν τα πρωτόκολλα TCP και UDP, οι οποίες, με την σειρά τους, υποστηρίζουν την ανταλλαγή μηνυμάτων από επίπεδο σε επίπεδο με ακαριαίο τρόπο, δεδομένου, ότι τα μηνύματα αυτά περνάνε άμεσα στο άλλο επίπεδο χωρίς την χρήση ενδιάμεσων buffers. Οι επεξεργαστές που υποστηρίζονται είναι οι x86, ARM, ColdFire, PowerPC και SH.

2.5 Team F1: NetF1™

Το Stack NetF1 [23] είναι ένα network stack, που υποστηρίζει τα πρωτόκολλα TCP, UDP, IP, ICMP και IGMP. Το ενδιαφέρον στην πρόταση της TeamF1 είναι ότι υποστηρίζεται η επόμενη έκδοση του IP, το IPv6. Το stack μπορεί να λειτουργήσει και ως IPv4 router και ως IPv6 router. Παρέχει, επίσης, και ένα εικονικό routing framework, το οποίο επιτρέπει την δημιουργία ξεχωριστών εικονικών δρομολογητών στο ίδιο φυσικό σύστημα. Στο σύνολό του το stack είναι με τέτοιο τρόπο δομημένο, ώστε να κάνει σωστή χρήση των πόρων, με απώτερο στόχο την σωστή λειτουργία και σε συστήματα με λίγους διαθέσιμους πόρους. Τονίζεται ότι η απόδοσή του δεν επηρεάζεται είτε λειτουργεί ως τερματικό είτε ως δρομολογητής.

Δεδομένου ότι η μετάβαση από το IPv4 στο IPv6 δεν έχει ακόμη ολοκληρωθεί το stack NetF1 υποστηρίζει το γνωστό tunneling έτσι ώστε τα IPv6 πακέτα να

ενθυλακώνονται σε IPv4 πακέτα, ώστε να μπορούν να δρομολογηθούν και από δρομολογητές που υποστηρίζουν μόνο το IPv4.

Καλή υποστήριξη παρέχεται, επίσης, στον τομέα του debugging. Έτσι, το stack διαθέτει ειδικές ρουτίνες οι οποίες επιβλέπουν την λειτουργία των υποστηριζόμενων πρωτοκόλλων και κρατάνε ειδικά logs, τα οποία αποθηκεύονται σε buffers για να αναλύσει ο σχεδιαστής αργότερα.

Ανάμεσα στις αρχιτεκτονικές που υποστηρίζονται και είναι οι x86, MIPS, ARM, Xscale, PPC.

2.6 Lantronix: Micro 100™

Η εταιρία Lantronix [24] προτείνει σε επίπεδο board το Micro 100™ και απευθύνεται σε κατασκευαστές που θέλουν να προσθέσουν δικτυακή συνδεσιμότητα στα προϊόντα τους, χωρίς να αναλώσουν ιδιαίτερο χρόνο στο θέμα αυτό.

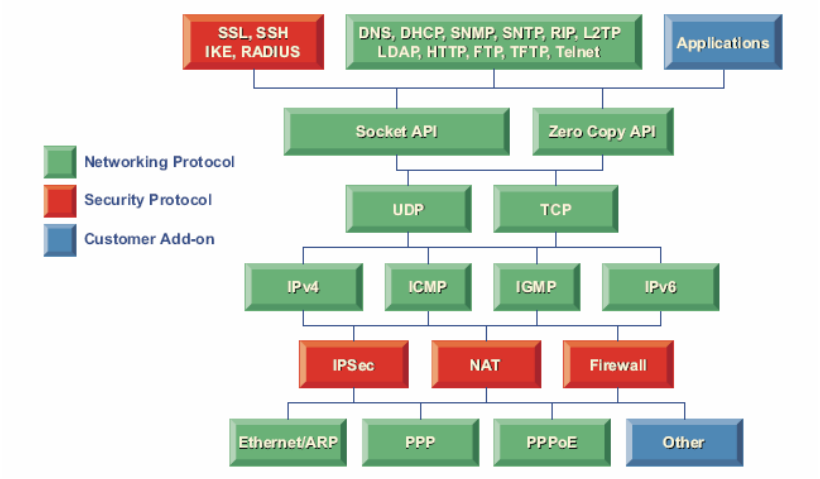
Διαθέτει υποδοχή RJ45 για σύνδεση με το δίκτυο, ενώ η τάση τροφοδοσίας είναι 5 VDC. Για τα updates έχει εξοπλιστεί με flash ROM και για την παρακολούθηση της ορθής λειτουργίας υπάρχουν τρία διαγνωστικά leds.

Σύμφωνα με τον κατασκευαστή, η διαμεταγωγή κυμαίνεται από 300 bps έως 230 kbps . Τα υποστηριζόμενα πρωτόκολλα είναι τα ARP, UDP, TCP/IP, Telnet, ICMP, SNMP, DHCP, TFTP, και HTTP.

2.7 Interpeak: IPNET

Η εταιρία Interpeak [25], που ειδικεύεται στον χώρο των embedded συστημάτων, παρουσίασε πριν ένα χρόνο περίπου το IP Stack με εμπορικό όνομα IPNET. Σύμφωνα με τις προδιαγραφές που παρουσιάζονται στην ιστοσελίδα της εταιρίας, πρόκειται για ένα διπλό stack με υποστήριξη τόσο στην παλιά έκδοση του IP (v4) όσο και στην νέα (v6).

Τα πρωτόκολλα που υποστηρίζονται είναι τα IPv4, IPv6, IPsec, PPP, TCP, UDP, NAT, ARP, ETHERNET, ICMP, ICMPv6/MLD/MDP και IGMP. Ένα ιδιαίτερα σημαντικό χαρακτηριστικό που αξίζει να τονισθεί, είναι το filtering, σύμφωνα με το οποίο παρέχεται φιλτράρισμα της εισερχόμενης δικτυακής κίνησης βασισμένο σε πολλούς παράγοντες. Το χαρακτηριστικό αυτό, επιτρέπει την υλοποίηση επιπλέον εφαρμογών όπως για παράδειγμα firewall κλπ.



Εικόνα 1: Η δομή του IPNET

Σημαντικό βάρος έχει δοθεί και στην ρύθμιση του stack, σύμφωνα με το οποίο ο χρήστης δύναται να ρυθμίσει το σύστημα σύμφωνα με τις δικές του ανάγκες, ακόμη και να απενεργοποιήσει την υποστήριξη πρωτοκόλλων που δεν χρειάζεται στην υλοποίηση της εφαρμογής του, με το ανάλογο κέρδος σε μνήμη. Οι κατηγορίες των υποστηριζόμενων αρχιτεκτονικών είναι πολλές. Ενδεικτικά αναφέρουμε τις CISC, RISC και DSP.

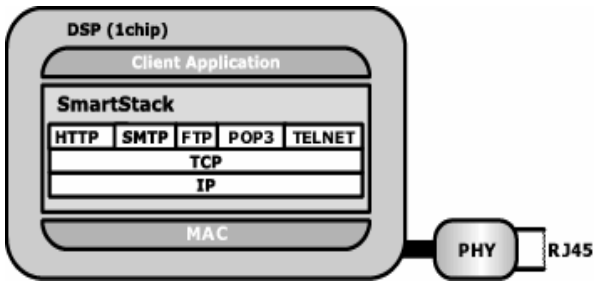
2.8 eDevice: eDbox-200

Στις προηγούμενες αναφορές των παλαιότερων εκδόσεων του Open TCP/IP Core στα πλαίσια της σχετικής έρευνας παρουσιάστηκε το SmartStack™ από την εταιρία eDevice [26].

Η ίδια εταιρία πριν 2 χρόνια περίπου παρουσίασε eDbox, προϊόν που ουσιαστικά, δεν έχει να δείξει κάτι νεότερο καθώς βασίζεται εξ'ολοκλήρου στο SmartStack™ και αποτελεί την hardware υλοποίηση του.

Το eDbox φέρει μία θύρα επικοινωνίας RS-232 και μία υποδοχή RJ-45 για την σύνδεση με το δίκτυο. Όπως συμβαίνει και με όλες τις προηγούμενες εταιρίες, η συσκευή αυτή σκοπό έχει την παροχή εύκολης δικτύωσης των προϊόντων του σχεδιαστή, με σκοπό τον απομακρυσμένο έλεγχό τους.

Τα χαρακτηριστικά της συσκευής παραμένουν τα ίδια. Έτσι, τα υποστηριζόμενα πρωτόκολλα είναι τα ARP, IP(v4), UDP, TCP, HTTP, SMTP, POP3, FTP και DHCP. Η υλοποίηση του SmartStack™ συνεχίζει να βασίζεται σε έναν 16-bit επεξεργαστή DSP από την Analog Devices από το επίπεδο MAC μέχρι και το επίπεδο των εφαρμογών.



Εικόνα 2: Το διάγραμμα του SmartStack.

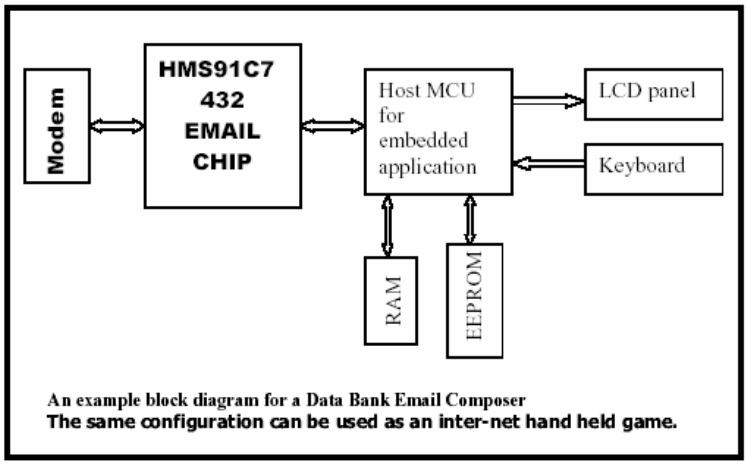
2.9 Hynix Semiconductors: HMS91C7432

Η εταιρία Hynix [27] 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.

Τα πρωτόκολλα που υποστηρίζονται είναι τα: TCP, IP, SMTP, POP3, PPP και DNS.



Εικόνα 3: Διάγραμμα μιας τυπικής εφαρμογής με το HMS91C743

2.10 Altera: Nios embedded processor

Η εταιρία Altera [28] 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 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.

- 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.: Ethernet, ARP, IP, ICMP, UDP και 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.

2.11 Insight Electronics

Η εταιρία Insight Electronics [29] 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 [?]. Επιπλέον έχει φτιαχτεί μια εφαρμογή που ολοκληρώνει τη σχεδίαση και παρέχει τη δυνατότητα για μετάδοση φωνής μέσω IP (Voice over IP - 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, 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.

2.12 Λοιπά TCP/IP Stacks

Κατά την διάρκεια της σύντομης έρευνας για τα διαθέσιμα TCP/IP Stacks σταθήκαμε και σε πολλές άλλες προτάσεις εταιριών ή εκπαιδευτικών ιδρυμάτων, κάποιες από τις οποίες αξίζει να αναφέρουμε ονομαστικά όπως:

- Η εταιρία Digi [30], η οποία παρέχει πλατφόρμες που υλοποιούν τα πιο γνωστά πρωτόκολλα και επιπλέον υποστήριξη του 802.11b. Βασικά χαρακτηριστικά των προϊόντων αποτελούν η μεγάλες flash και RAM μνήμες και ο 32-bit επεξεργαστής ARM που φιλοξενούνται κατά κύριο λόγο σε PCB σημαντικά μικρών διαστάσεων.
- Η εταιρία Connect One [31], η οποία διαθέτει το iChip σε εφαρμογές ασύρματες ή dial up. Το iChip χρησιμοποιείται κατά κόρον σε κινητά τηλέφωνα καθώς υποστηρίζει τα AMPS, CDMA, CDPD, GPRS, GSM, Iden και TDMA. Τα πρωτόκολλα που συναντάμε στο internet και υποστηρίζονται από την εν λόγω υλοποίηση είναι τα ARP, IP, ICMP, UDP, TCP, DNS, DHCP, SMTP, POP3, MIME και HTTP.
- Η εταιρία WIZnet [32], η οποία έχει παρουσιάσει το chip W3100A, chip το οποίο διαθέτει ένα ολοκληρωμένο TCP/IP Stack σε συνδυασμό με το MAC layer. Υποστηριζόμενα πρωτόκολλα: CP, IPv4, UDP, ICMP, ARP, DLC και MAC (DHCP, HTTP, SMTP και PING).
- Το project lwIP [33], το οποίο αναπτύχθηκε από τον Adam Dunkels στο εργαστήριο Αρχιτεκτονικής Υπολογιστών και Δικτύων, του Σουηδικού Ινστιτούτου Επιστήμης Υπολογιστών. Σκοπός του project ήταν η υλοποίηση του TCP/IP Stack με πλήρη λειτουργικότητα σε συνδυασμό με την μειωμένη χρήση διαθέσιμων πόρων. Τα υποστηριζόμενα πρωτόκολλα είναι το IP, ICMP, UDP, TCP, DHCP, PPP και ARP. Το project συνεχίζει να εξελίσσεται και να βελτιώνεται με τελευταία έκδοση την 0.7.2 (18/4/2004).
- Το FlexGate TCP/IP Stack 2.0 [34], που έχει αναπτυχθεί για την οικογένεια των μικροεπεξεργαστών 8051 και υπόσχεται χαμηλές απαιτήσεις από άποψης hardware (λιγότερο από 1 KB μνήμης και 12 KB κώδικα). Η στοίβα ανήκει στην κατηγορία του «ανοιχτού λογισμικού» και υποστηρίζει δημοφιλή πρωτόκολλα όπως το ICMP,

το ARP, το PING, το UDP και το TCP. Το εν λόγω stack χρησιμοποιήθηκε με απόλυτη επιτυχία στο MicroWeb Server που δημοσιεύτηκε στο περιοδικό Ελέκτορ (7-8/2004), για την μεταφορά μετρήσεων υγρασίας, θερμοκρασίας, ταχύτητας ανέμου ή άλλες εφαρμογές όπως τον απομακρυσμένο έλεγχο (έλεγχος πρόσβασης, τηλεχειρισμός και παρακολούθηση συσκευών κ.α.). Η κατασκευή στο σύνολό της αποτελείται από δύο κάρτες, μία εκ των οποίων είναι η κάρτα δικτύου που τρέχει το FlexGate TCP/IP και η άλλη που είναι υπεύθυνη για τον απομακρυσμένο έλεγχο και ολοκληρώνεται γύρω από το CS8900 της Texas Instruments.

2.13 Το Open TCP/IP Core στο Πολυτεχνείο Κρήτης

Από την παραπάνω σχετική έρευνα που πραγματοποιήθηκε στα πλαίσια της διπλωματικής εργασίας αυτής, αλλά και κατά τις προηγούμενες διπλωματικές εργασίες των κ. Κάχρη, Ζήση και Κοϊδή γίνεται αντιληπτή η σημασία των IP Cores και ειδικότερα, των TCP/IP Stacks. Το Εργαστήριο Μικροεπεξεργαστών και Υλικού του Πολυτεχνείου Κρήτης αντιλαμβανόμενο τις ανάγκες αλλά και τις εξελίξεις των τελευταίων χρόνων στον χώρο των embedded συστημάτων προχώρησε στην ανάπτυξη του δικού του TCP/IP Stack με την ονομασία Open TCP/IP Core.

Η ανάπτυξη του Open TCP/IP Core ξεκίνησε το 2000 και συνεχίζεται έως και σήμερα. Έχουν ήδη παρουσιαστεί τρεις εκδόσεις του συστήματος, στα πλαίσια προηγούμενων διπλωματικών εργασιών, ενώ, στην διπλωματική εργασία παρουσιάζεται η τέταρτη κατά σειρά έκδοση. Κάθε νέα έκδοση βασίζεται στην προηγούμενη και έχει ως στόχο την αύξηση της λειτουργικότητας και της ευελιξίας του συστήματος καθώς και την διόρθωση μικρών αστοχιών. Παρακάτω θα ρίξουμε μία σύντομη ματιά στις προηγούμενες εκδόσεις του Open TCP/IP συμπεριλαμβανομένης και της νεότερης, έτσι ώστε να συνειδητοποιήσουμε την όλη πορεία του project.

2.13.1 Η διπλωματική εργασία του κ. Κάχρη

Στα πλαίσια της διπλωματικής του κ. Κάχρη [9] παρουσιάστηκε η πρώτη έκδοση του Open TCP/IP Core. Όπώς είναι λογικό, δεδομένου ότι το σύστημα δημιουργήθηκε για πρώτη φορά τότε, παρουσίαζε σχετικά μειωμένη λειτουργικότητα, ενώ, σε κάποια σημεία η πολυπλοκότητα ήταν αυξημένη, γεγονός που είχε ως συνέπεια την μείωση της ευελιξίας, της αυτονομίας και του σωστού ελέγχου στα ελάχιστα επιμέρους διαθέσιμα υποσυστήματα. Παρόλα αυτά, η δουλειά του κ. Κάχρη ήταν σημαντικότατη, καθώς, έθεσε τις πρώτες «γερές» βάσεις και τα πρώτα σημεία προβληματισμού για την ομαλή εξέλιξη των επόμενων εκδόσεων.

Πιο αναλυτικά, στην πρώτη έκδοση το υποσύστημα λήψης πλαισίων Ethernet πραγματοποιούσε αρκετές λειτουργίες, κάποιες από τις οποίες δεν ανήκαν στην δικαιοδοσία λόγω της απουσίας του κεντρικού ελέγχου από κάποιο ανεξάρτητο υποσύστημα (Control). Έτσι, το εν λόγω υποσύστημα έπαιζε τρεις πολύ βασικούς ρόλους: τον έλεγχο του εισερχόμενου πλαισίου, την διαχείριση όλου του συστήματος προκειμένου το πακέτο να ανέβει στα ανώτερα επίπεδα, την δημιουργία επικεφαλίδας Ethernet και την ενεργοποίηση του συστήματος μετάδοσης.

Στην πρώτη έκδοση του Core, το σύστημα υποστήριζε μόνο την λήψη αίτησης για σύνδεση από απομακρυσμένο σταθμό, αναμένοντας σε κάποιο port, είτε του TCP, είτε του UDP, χωρίς να έχει το ίδιο την δυνατότητα την εκκίνηση σύνδεσης. Επίσης, το υποσύστημα αποστολής/λήψης πακέτων IP ήταν σε θέση να υποστηρίξει μία μόνο σύνδεση με απομακρυσμένο σταθμό.

2.13.2 Η διπλωματική εργασία του κ. Ζήση

Στα πλαίσια της διπλωματικής του κ. Ζήση [13], η λειτουργικότητα και η συνολική δομή του συστήματος, βελτιώθηκαν σημαντικά. Στην έκδοση αυτή το σύστημα λήψης πλαισίων είχε πλέον περιορισμένη λειτουργικότητα: τον έλεγχο των πακέτων, την δημιουργία επικεφαλίδας και την παροχή δεδομένων από και προς τον πίνακα του ARP. Προκειμένου να επιτύχει αυτήν την πιο σωστή κατανομή λειτουργιών και να απαλλάξει το υποσύστημα του Ethernet από τον ρόλο του γενικού συντονιστή, ο κ. Ζήσης δημιούργησε ένα υποσύστημα Control. Το υποσύστημα αυτό δεν επέφερε μόνο την σωστή κατανομή των λειτουργιών του συστήματος, αλλά και τον παραλληλισμό της λειτουργίας, γεγονός το οποίο είχε άμεσο αντίκτυπο στην καλύτερη απόδοση.

Το πρόβλημα της υποστήριξης πολλών συνδέσεων ξεπεράστηκε με τον επανασχεδιασμό των υποσυστημάτων αποστολής/λήψης πλαισίων TCP και UDP σε συνδυασμό με την χρήση μίας δίπορτης μνήμης, η οποία ονομάζεται socket table και σκοπός της είναι η φύλαξη των στοιχείων κάθε σύνδεσης. Ο επανασχεδιασμός των παραπάνω συστημάτων κρίθηκε απαραίτητος καθώς η προσθήκη του socket table αυτομάτως έθετε εκτός λειτουργίας αρκετούς καταχωρητές οι οποίοι ήταν συνυφασμένοι με τα συστήματα αυτά.

Η αδυναμία της εκκίνησης μίας σύνδεσης από την πλευρά της σχεδίασης ξεπεράστηκε με τον σχεδιασμό δύο νέων συστημάτων, με στόχο την αποστολή αίτησης ARP και την λήψη της απάντησης αντίστοιχα. Με αυτόν τον τρόπο το σύστημα ήταν πλέον σε θέση να γνωρίζει την διεύθυνση Ethernet του απομακρυσμένου σταθμού.

Βέβαια, η προσθήκη του socket table μπορεί να έγινε η αιτία να ξεπεραστεί ο σκόπελος της μία μόνο σύνδεσης, όμως, ο μέγιστός αριθμός συνδέσεων παρέμεινε σε χαμηλά επίπεδα (16 συνδέσεις). Επίσης, η όλη σχεδίαση παρουσίαζε κάποια ασυμβατότητα με τα RFCs που αναφέρονται στο πρωτόκολλο TCP, και αυτό γιατί απουσίαζε πλήρως ο έλεγχος συμφόρησης από το αντίστοιχο υποσύστημα με αποτέλεσμα όχι μόνο να χάνεται το βασικότερο χαρακτηριστικό του πρωτοκόλλου TCP που είναι η αξιόπιστη αποστολή και λήψη, αλλά και η κατακόρυφη πτώση της απόδοσης. Οι παραπάνω αδυναμίες έγιναν η αιτία για το πέρασμα στην τρίτη έκδοση του συστήματος.

2.13.3 Η διπλωματική εργασία του κ. Κοϊδή

Η διπλωματική εργασία του κ. Κοιδή [14] ήρθε για να καλύψει τις ασυμβατότητες που αναφέρθηκαν παραπάνω και παράλληλα, να δώσει μεγαλύτερη ευελιξία στο σύστημα με την αύξηση του αριθμού των υποστηριζόμενων συνδέσεων από 16 σε 32.

Πλέον, με την προσθήκη του ελέγχου συμφόρησης, το σύστημα ήταν σε θέση να υποστηρίξει την αξιόπιστη μεταφορά δεδομένων μέσω του πρωτοκόλλου TCP κάνοντας πολύ πιο αποδοτική χρήση του δικτύου. Το σύστημα στην έκδοση αυτή ήταν σε θέση να στέλνει και να λαμβάνει πολλαπλά πλαίσια (και όχι ένα κάθε φορά) υπό την μορφή παραθύρου. Παράλληλα, ο μεγαλύτερος αριθμός μέγιστων συνδέσεων έκανε την σχεδίαση ικανή να βρίσκεται μεταξύ 256 υπολογιστών.

Ένα νέο εξίσου σημαντικό χαρακτηριστικό που προστέθηκε στην έκδοση αυτή, την τρίτη κατά σειρά, ήταν και προσθήκη της διεπαφής MII ώστε να μπορεί η σχεδίαση να συνδεθεί στο δίκτυο μέσω ενός Ethernet Transceiver.

2.13.4 Η παρούσα διπλωματική εργασία

Όντας, ολοκληρωμένη και η τρίτη έκδοση της σχεδίασης, το IP Core βρίσκονταν σε ένα ικανοποιητικό επίπεδο από άποψη λειτουργικότητας καθώς τα πρωτόκολλα υποστηρίζονταν σχεδόν πλήρως.

Παρόλα αυτά, και επειδή η σχεδίαση αναπτύσσονταν σταδιακά βασιζόμενη στις προηγούμενες εκδόσεις, ορισμένα υποσυστήματα είχαν αρχίσει να παρουσιάζουν πρόβλημα στην δομή τους και κατ' επέκταση κάποιες φορές στην λειτουργία ή την απόδοσή τους. Το πρόβλημα εντοπίζονταν κυρίως στο υποσύστημα αποστολής/λήψης πλαισίων TCP, το οποίο ξεκίνησε ως ένα μία αυτόνομη οντότητα από την πρώτη κιόλας έκδοση και συνέχισε να παραμένει ενοποιημένη μέχρι και την τρίτη έκδοση της σχεδίασης.

Μετά την ολοκλήρωση ,όμως, της τρίτης έκδοσης της σχεδίασης και την προσθήκη του έλεγχου συμφόρησης, ένα χαρακτηριστικό το οποίο αναμφισβήτητα βάρυνε την σχεδίαση πολύ λόγω της έντονης πολυπλοκότητάς του, το υποσύστημα αποστολής/λήψης TCP, όχι μόνο απέκτησε λειτουργικότητες που θα έπρεπε να ρυθμίζονται εξωτερικά, αλλά έλαβε και τεράστιες διαστάσεις από άποψη γραμμών κώδικα. Τα παραπάνω χαρακτηριστικά είχαν αρνητικές επιπτώσεις τόσο στην καλή συντήρηση του κώδικα, όσο και στην αποδοτική λειτουργία του.

Μετά, λοιπόν, από την προσθήκη του έλεγχου συμφόρησης, άρα και την κατακόρυφη βελτίωση της απόδοσης σε σχέση με τις προηγούμενες εκδόσεις, η τμηματοποίηση του υποσυστήματος TCP έγινε επιτακτική ανάγκη, προκειμένου να είμαστε σε θέση να κερδίσουμε ακόμα πιο πολύ σε απόδοση αξιοποιώντας καλύτερα τις αλλαγές που είχαν ήδη γίνει. Επιπλέον, η έλλειψη σωστής δομής στο ίδιο υποσύστημα καθιστούσε αδύνατη την παράλληλη εκτέλεση κάποιων λειτουργιών, όπως για παράδειγμα, την εξυπηρέτηση περισσότερων από μία ροές παράλληλα επειδή το ενσωματωμένο υποσύστημα αποκωδικοποίησης των εντολών από το επίπεδο εφαρμογών παρέμενε δεσμευμένο μέχρι την πλήρη αποστολή ή λήψη κάποιας ροής.

Λαμβάνοντας υπόψιν ότι μετάβαση από την δεύτερη έκδοση στην τρίτη δεν ήταν καθόλου εύκολη υπόθεση, καθώς ο έλεγχος συμφόρησης είναι από μόνος του αρκετά πολύπλοκος και απαιτητικός από πλευράς hardware και χρόνου σχεδίασης, στην έκδοση αυτή είχαν παραμείνει κάποια χαρακτηριστικά ανολοκλήρωτα. Για παράδειγμα οι timers που αποτελούν ακρογωνιαίο λίθο για την ομαλή λειτουργία του έλεγχου συμφόρησης υπήρχαν στην σχεδίαση ως 16 instances και όχι ως 31. Με λίγα λόγια υπήρχαν timers σε 16 μόνο sockets. Παράλληλα, ο timer ως υποσύστημα δεν παρείχε υποστήριξη για την ορθή λειτουργία του ως Persistence ή ως Idle timer, με αποτέλεσμα το congestion control να αδυνατεί να αποδώσει στο μέγιστο βαθμό.

Στα πλαίσια αυτής της διπλωματικής και με γνώμονα την απόδοση, την λειτουργικότητα, την ευελιξία, άλλα και την σωστή, αρθρωτή δομή, το υποσύστημα αποστολής/λήψης πλαισίων TCP δομικά άλλαξε τελείως. Οι λειτουργίες διασπάστηκαν και μοιράστηκαν σε ξεχωριστά υποσυστήματα, τα οποία είναι σε θέση να παρακολουθούν συνεχώς τις αλλαγές των άλλων υποσυστημάτων και να δρουν αυτόνομα όποτε χρειαστεί. Συνάμα, για να μπορεί να επιτευχθεί ο επιθυμητός παραλληλισμός, προστέθηκαν μνήμες και FIFOs οι οποίες κρατάνε δεδομένα για όσο χρόνο χρειαστεί, μέχρι ένα υποσύστημα να γίνει διαθέσιμο. Με αυτόν τον τρόπο, κάθε υποσύστημα μπορεί να δουλεύει ανεξάρτητα και ασταμάτητα ,καθώς πλέον, δεν χρειάζεται στις περισσότερες των περιπτώσεων, να περιμένει άλλα υποσυστήματα.

Επίσης, σε αυτήν την έκδοση της σχεδίασης, βελτιώθηκε η λειτουργία του ελέγχου συμφόρησης, καθώς δημιουργήθηκαν ξεχωριστά υποσυστήματα διαχείρισης των timeouts, υπολογισμού, αλλαγής και αποθήκευσης του μέσου χρόνου αναμονής μέχρι το timeout (Round Trip Time – RTT) και νέες εντολές για την σωστότερη διαχείριση του όλου συστήματος TCP από το Congestion Control. Αξίζει να σημειωθεί, ότι και στην τρίτη έκδοση υπήρχαν εσωτερικές εντολές για την διαχείριση του υποσυστήματος, οι οποίες προέρχονταν από το ίδιο το υποσύστημα. Αυτές οι εντολές σαφώς και συμπεριλήφθηκαν και σε αυτήν την έκδοση, έχοντας, όμως, υποστεί σημαντική βελτίωση.

Μιλώντας με αριθμούς, το υποσύστημα TCP διασπάστηκε σε δεκατέσσερα επιμέρους υποσυστήματα, το καθένα από τα οποία έχει βάθος μέχρι και τέσσερα ή πέντε επίπεδα, μα χαμηλότερο επίπεδο εκείνο που φέρει απλά στοιχεία όπως πολυπλέκτες, flip-flops κλπ.

Τέλος, θα πρέπει να τονίσουμε, ότι μέσα στους στόχους της διπλωματικής εργασίας αυτής ήταν και ο σωστός, τεκμηριωμένος έλεγχος του συστήματος καθώς η

πολυπλοκότητα της σχεδίασης έχει αυξηθεί κατά πολύ σε σχέση με τις προηγούμενες υλοποιήσεις. Για τον λόγο αυτό, ένα πολύ σημαντικό χρονικό διάστημα που είχαμε στην διάθεσή μας, αφιερώθηκε, αποκλειστικά και μόνο, στα test benches που υλοποιήθηκαν εξ αρχής, προκειμένου να διορθωθούν όσο το δυνατόν περισσότερες αστοχίες.

Στα κεφάλαια που ακολουθούν θα μελετήσουμε με αναλυτικό τρόπο τόσο την νέα δομή του συστήματος, όσο και τα επιμέρους υποσυστήματα που υπέστησαν αλλαγή η προστέθηκαν στην υλοποίηση μας.

Κεφάλαιο 3: Η αρχιτεκτονική του Open TCP/IP Core

Στο κεφάλαιο αυτό θα εισέλθουμε με ακόμη μεγαλύτερη λεπτομέρεια στο Open TCP/IP Core, για να μελετήσουμε την αρχιτεκτονική του συστήματος ρίχνοντας ιδιαίτερο βάρος στα υποσυστήματα που σχεδιάστηκαν για πρώτη φορά αλλά και σε αυτά που επανασχεδιάστηκαν προκειμένου να είναι σε θέση να λειτουργήσουν αρμονικά με τα υπόλοιπα. Επιπλέον, θα αναλύσουμε τις προδιαγραφές της σχεδίασης τονίζοντας και πάλι, τις προδιαγραφές που άλλαξαν σε σχέση με την προηγούμενη έκδοση.

Η προσέγγιση του θέματος θα γίνει ιεραρχικά σύμφωνα με το πρότυπο OSI, ενώ παράλληλα, θα παρουσιαστούν διαγράμματα με στόχο να γίνει καλύτερα κατανοητή η νέα αρχιτεκτονική, οι νέες λειτουργίες καθώς και τα νέα ή βελτιωμένα χαρακτηριστικά της έκδοσης αυτής.

Σημειώνεται ότι στο κεφάλαιο αυτό, αλλά και σε αυτά που ακολουθούν, θα γίνονται πολύ περιληπτικές αναφορές στα στοιχεία των δικτύων και γενικότερα των πρωτοκόλλων. Ο λόγος για τον οποίο ακολουθείται αυτή η τεχνική είναι ότι τα στοιχεία σχετικά με την θεωρία πάνω στα δίκτυα ή στα πρωτόκολλα έχουν παρουσιαστεί εκτενώς σε όλες τις προηγούμενες διπλωματικές εργασίες. Σκοπός αυτής της αναφοράς είναι η παρουσίαση της νέας έκδοσης του συστήματος και όχι η εισαγωγή σε στοιχεία θεωρίας που θεωρούνται ήδη γνωστά.

Παρόλ'αυτά στο Παράρτημα παρουσιάζονται περιληπτικά κάποια στοιχεία θεωρίας που αφορούν στην εργασία αυτή, προκειμένου να είναι εύκολη η παρακολούθηση της αναφοράς. Για περισσότερες λεπτομέρειες, όμως, προτείνεται αναφορά στις προηγούμενες εργασίες ή σε εγχειρίδια σχετικά με την θεωρία των δικτύων.

3.1 Το Φυσικό Επίπεδο και το Επίπεδο Ζεύξης Δεδομένων

Ακολουθώντας την γραμμή που προαναφέραμε στην εισαγωγή του κεφαλαίου αυτού, ξεκινάμε την πρώτη αναλυτική μας προσέγγιση από το χαμηλότερο, από άποψης ταξινόμησης στο πρότυπο OSI, επίπεδο: το φυσικό επίπεδο και το επίπεδο ζεύξης.

Στην εργασία αυτή, όπως και σε όλες εκείνες που προηγήθηκαν, θεωρούμε ότι το φυσικό επίπεδο υλοποιείται από ένα ολοκληρωμένο κύκλωμα (Transceiver). Το κύκλωμα αυτό, που υπάρχει στο εμπόριο, συνδέεται μέσω κατάλληλου κυκλώματος στο δίκτυο χρησιμοποιώντας καλώδιο UTP. Τα σήματα προς το πρωτόκολλο MAC, που παρέχονται από το προαναφερθέν κύκλωμα, είναι εκείνα που προορίζονται από την διεπαφή MII (Media Independent Interface). Η διεπαφή MII είναι συμβατή με το πρότυπο IEE-802.3u.

Η διεπαφή MII παρέχει σήματα που δείχνουν την είσοδο ενός πλαισίου στο σύστημα, σήματα που ειδοποιούν για την κατάσταση στο κανάλι, όπως την ύπαρξη σύγκρουσης και την ύπαρξη φέροντος, ρολόι μετάδοσης και λήψης το οποίο είναι ήδη συγχρονισμένο στα δεδομένα αυτά και μπορεί να αντεπεξέρχεται σε 10BASE-T ή 100BASE-T μεταδόσεις σε half ή full duplex.

Στο Επίπεδο Ζεύξης υπάρχει ένα υποσύστημα με ρόλο διαμεσολαβητή μεταξύ των εξόδων που περιγράφτηκαν και των εισόδων του συστήματός μας. Η λειτουργία του είναι η ομαδοποίηση των 4-bit δεδομένων εισόδου και εξόδου από την διεπαφή MII σε οκτάδες από bits. Στο ίδιο επίπεδο υπάρχουν τα υποσυστήματα λήψης, αποστολής και δημιουργίας πλαισίων.

Το υποσύστημα λήψης πλαισίων πραγματοποιεί έλεγχο του εισερχόμενου πλαισίου. Ο έλεγχος του εισερχόμενου πλαισίου στηρίζεται στην ανίχνευση της διεύθυνσης προορισμού. Επιπρόσθετα, κατά την φάση ελέγχου αναγνωρίζεται και η ζητούμενη υπηρεσία (ARP request/reply, IP).

Το υποσύστημα αποστολής πλαισίων υποστηρίζει το επίπεδο MAC (Multiple Access Control), το πρωτόκολλο του οποίου είναι υπεύθυνο για την σωστή μετάδοση των δεδομένων από την μνήμη μετάδοσης (Transmit RAM – TxRAM). Για να επιτευχθεί σωστή μετάδοση απαιτείται μία σειρά ελέγχων, όπως η ανίχνευση ύπαρξης φέροντος, ύπαρξης σύγκρουσης κλπ. Σημειώνεται ότι η εκκίνηση του υποσυστήματος αποστολής προκαλεί και την ενεργοποίηση του υποσυστήματος CRC. Το υποσύστημα CRC υπολογίζει το checksum το οποίο και προσκολλάται στο τέλος του

προς αποστολή πακέτου. Στον πίνακα 2 φαίνονται τα δεδομένα που αποστέλλονται στην TxRAM συναρτήσει των αντίστοιχων διευθύνσεων.

	Διεύθυνση	**Δεδομένα**
	0x00	Συνολικό Μήκος
ETHERNET	*0x01 – 0x08*	Preamble
	0x09 – 0x0E	Προορισμός Ethernet
	0x0f – 0x14	Πηγή Ethernet
	0x15 – 0x16	Τύπος
IP	*0x17*	Version/IHL
	0x18	Τύπος Υπηρεσίας
	0x19 – 0x1A	Μήκος πλαισίου IP
	0x1B – 0x1C	Identification #
	0x1D – 0x1E	Flags/Fragment Offset
	0x1F	TTL
	0x20	Πρωτόκολλο
	0x21 – 0x22	CheckSum
	0x23 – 0x26	IP πηγής
	0x27 – 0x2A	IP προορισμού
UDP	*0x2B – 0x2C*	Port # πηγής
	0x2D – 0x2E	Port # προορισμού
	0x2F – 0x30	Μήκος πακέτου UDP
	0x31 – 0x32	CheckSum
	0x33 – 0xFF	Δεδομένα UDP
TCP	*0x2B – 0x2C*	Port # πηγής
	0x2D – 0x2E	Port # προορισμού
	0x2F – 0x32	Sequence #
	0x33 – 0x36	Acknowledge #
	0x37	Data Offset
	0x38	Flags
	0x39 – 0x3A	Window
	0x3B – 0x3C	CheckSum
	0x3D – 0x3E	Urgent Pointer
	0x3F – 0xFF	Δεδομένα TCP

Πίνακας 2: Χώρος διευθύνσεων μνήμης TxRAM.

Το υποσύστημα δημιουργίας πλαισίων προς αποστολή συμπληρώνει τα πακέτα IP με επικεφαλίδες Ethernet, που ανάμεσα στα άλλα πεδία περιέχουν την διεύθυνση προορισμού και την διεύθυνση πηγής. Η διεύθυνση προορισμού Ethernet λαμβάνεται από τον πίνακα ARP. Ο πίνακας αυτός περιέχει αντιστοιχίες διευθύνσεων IP με τις αντίστοιχες του Ethernet. Σημειώνεται ότι το υποσύστημα αυτό σε

συνδυασμό με το υποσύστημα αποστολής υλοποιούν το πρωτόκολλο LLC (Logical Link Control).

3.2 Το Επίπεδο Δικτύου

Ο ρόλος του πρωτοκόλλου που συμπεριλαμβάνεται στο επίπεδο δικτύου σε γενικές γραμμές είναι η αποστολή και η λήψη πακέτων μεταξύ απομακρυσμένων σταθμών, οι οποίοι είναι συνδεδεμένοι σε δίκτυα μεταγωγής πακέτων. Είναι ανάγκη να τονισθεί, ότι το πρωτόκολλο που «τρέχει» στο επίπεδο αυτό δεν εξασφαλίζει αξιόπιστη μεταφορά δεδομένων, ούτε είναι σε θέση να διακρίνει για ποια εφαρμογή αναφέρονται τα δεδομένα που λαμβάνει ή στέλνει, καθώς τα χαρακτηριστικά αυτά εξασφαλίζονται από πρωτόκολλα υψηλότερων επιπέδων.

Η λειτουργικότητα του επιπέδου δικτύου παρέχεται από το πρωτόκολλο IP και η υλοποίησή του στηρίζεται στη χρήση διευθύνσεων σταθερού μήκους για την πηγή και τον προορισμό κάθε πακέτου. Ένα άλλο, επιπλέον, χαρακτηριστικό του πρωτοκόλλου IP είναι ο κατακερματισμός των πακέτων πολλά μικρότερα, έτσι ώστε να μπορούν να μεταδοθούν από πρωτόκολλα χαμηλότερου επιπέδου. Το Open TCP/IP δεν υποστηρίζει την δυνατότητα κατακερματισμού. Αντιθέτως, θέτει τον περιορισμό για κάθε πακέτο, να έχει μέγεθος μέχρι και 1500 bytes. Ο αριθμός αυτός είναι το μέγιστο μέγεθος δεδομένων που μπορεί να μεταφέρει ένα πλαίσιο Ethernet.

Το πρωτόκολλο IP παρέχει μεθόδους κλήσης του, τόσο για πρωτόκολλα χαμηλότερων, όσο και για πρωτόκολλα υψηλότερων επιπέδων. Κατ'αναλογία το IP είναι σε θέση να μπορεί να καλέσει πρωτόκολλα, προκειμένου να ολοκληρωθεί η διαδικασία λήψης ή αποστολής με επιτυχία. Στην αυτήν την διαδικασία αποστολής λήψης, η διεύθυνση IP του απομακρυσμένου σταθμού παίζει σημαντικότατο ρόλο. Έτσι, όταν ένα πρωτόκολλο υψηλότερου επιπέδου προσπαθεί να στείλει ένα πακέτο, τότε παρέχει στο IP την διεύθυνση αυτή. Το IP, με την σειρά του, καλεί το επίπεδο Ζεύξης δίνοντας του και πάλι την διεύθυνση. Τέλος, το επίπεδο Ζεύξης, με βάση την IP διεύθυνση, βρίσκει την φυσική διεύθυνση και δημιουργεί ένα πακέτο έτοιμο για αποστολή. Για την λήψη ενός πακέτου ακολουθείται η ακριβώς αντίστροφη διαδικασία.

Πριν κλείσουμε την παράγραφο αυτή, θα πρέπει να αναφέρουμε ότι στο επίπεδο Δικτύου «τρέχει» και το πρωτόκολλο ICMP (Internet Control Message Protocol). Το πρωτόκολλο ICMP έχει ως στόχο την ανταλλαγή πληροφοριών μεταξύ των σταθμών. Οι πληροφορίες αυτές αφορούν στην λειτουργία ή την κατάσταση του δικτύου. Ένα από τα πιο χαρακτηριστικά μηνύματα είναι το ICMP Echo Request και ICMP Echo Reply. Η σχεδίασή μας, υποστηρίζει στο πρωτόκολλο αυτό τα Echo Request & Reply.

3.2.1 Το πρωτόκολλο IP

Ο σχεδιασμός του πρωτοκόλλου IP σε hardware επιτυγχάνεται με επεξεργασία της επικεφαλίδας του κάθε πακέτου από συγκεκριμένα υποσυστήματα. Στην εικόνα 4 φαίνεται η επικεφαλίδα ενός πακέτου IP. Παρακάτω θα αναφέρουμε εκείνα τα πεδία που υποστηρίζονται εν μέρει ή καθόλου στο Open TCP/IP. Στο παράρτημα περιγράφονται περιληπτικά όλα τα πεδία.

<table>
<tr><td>00</td><td>01</td><td>02</td><td>03</td><td>04</td><td>05</td><td>06</td><td>07</td><td>08</td><td>09</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
<tr><td colspan="4">Version</td><td colspan="4">IHL</td><td colspan="8">TOS</td><td colspan="16">Total length</td></tr>
<tr><td colspan="16">Identification</td><td colspan="3">Flags</td><td colspan="13">Fragment offset</td></tr>
<tr><td colspan="8">TTL</td><td colspan="8">Protocol</td><td colspan="16">Header checksum</td></tr>
<tr><td colspan="32">Source IP address</td></tr>
<tr><td colspan="32">Destination IP address</td></tr>
<tr><td colspan="32">Options</td></tr>
</table>

Εικόνα 4: Το IP header.

- Version: Περιγράφει την έκδοση του IP. Η σχεδίασή μας υποστηρίζει την έκδοση 4 (IPv4), καθώς δεν έχει ολοκληρωθεί ακόμα η μετάβαση από το IPv4 στο IPv6. Με σχετικά, όμως, λίγες αλλαγές στην σχεδίαση μπορεί να υποστηριχθεί η νέα έκδοση (IPv6).
- TOS (Type of Service): Περιγράφει παραμέτρους σχετικές με την ποιότητα της υπηρεσίας όπως η προτεραιότητα, η καθυστέρηση κλπ. Το συγκεκριμένο πεδίο δεν υποστηρίζεται από την σχεδίαση μας, δεδομένου ότι αυτή χειρίζεται όλα τα πακέτα ισότιμα.
- Identification/Flags/Fragment Offset: Τα πεδία αυτά έχουν να κάνουν με τον κατακερματισμό των πακέτων και δεν υποστηρίζονται στην σχεδίασή μας.

Όπως έχει ήδη αναφερθεί, στο θέμα αυτό υπάρχει περιορισμός, σύμφωνα με τον οποίο δεν μπορεί ένα πακέτο να έχει μέγεθος μεγαλύτερο από εκείνο των δεδομένων που μπορεί να φέρει ένα πλαίσιο Ethernet, δηλαδή 1500 bytes. Κατά την δημιουργία ενός IP πακέτου τα πεδία «Flags» και «Fragment Offset» συμπληρώνονται με την τιμή μηδέν, προκειμένου να υπάρχει η πληροφορία ότι το πακέτο είναι αυτοτελές και δεν έχει υποστεί κατακερματισμό.

- TTL (Time To Live): Περιγράφει τον χρόνο παραμονής ενός πακέτου στο δίκτυο. Από κόμβο σε κόμβο η τιμή μειώνεται. Όταν μηδενιστεί και το πακέτο συνεχίζει να βρίσκεται στο δίκτυο, τότε αυτό απορρίπτεται. Η σχεδίαση θέτει μία σταθερή τιμή στο πεδίο αυτό, η οποία, όμως, δεν μπορεί να ελεγχθεί με κάποια εξωτερική εντολή.
- Protocol: Περιγράφει το πρωτόκολλο του επόμενου επιπέδου για το οποίο προορίζονται τα ενθυλακωμένα δεδομένα. Η σχεδίαση μας, όπως έχουμε ήδη αναφέρει, υποστηρίζει τα πρωτόκολλα ICMP, TCP και UDP με δεκαεξαδικές τιμές 0x01, 0x06 και 0x11, αντίστοιχα.
- Options: Το πεδίο αυτό χρησιμοποιείται για την μετάδοση κάποιων παραμέτρων που αφορούν στην ασφάλεια, στην δρομολόγηση, σε περιορισμούς διαχείρισης κλπ. Το πεδίο αυτό δεν ελέγχεται και ούτε υποστηρίζεται από την σχεδίαση, επειδή η χρήση του περιορίζεται σε ειδικές περιπτώσεις και συνάμα, ακόμα και σήμερα, δεν είναι πλήρως καθορισμένο.

3.2.2 Το πρωτόκολλο ICMP

Τα διάφορα επιμέρους δίκτυα συνδέονται μεταξύ τους μέσω συσκευών που ονομάζονται Gateways (πύλες). Για να προσπελάσει ένας σταθμός μία πύλη πρέπει να στείλει κάποιο πακέτο IP στην πύλη του δικτύου του. Το πρωτόκολλο ICMP σχεδιάστηκε με σκοπό να επιτρέπει την ανταλλαγή πληροφοριών μεταξύ των πυλών και των σταθμών του δικτύου. Οι πληροφορίες έχουν να κάνουν με την αναζήτηση ενός σταθμού μέσα στο δίκτυο (εφαρμογή Ping), την αναφορά σφαλμάτων κλπ.

Η σχεδίασή μας υποστηρίζει μόνο την εφαρμογή Ping: είναι σε θέση να στείλει και να λάβει μηνύματα Ping. Στην εικόνα 5 φαίνεται η δομή ενός πακέτου

ICMP, ενώ παρακάτω αναφέρονται με τα πεδία που υποστηρίζονται εν μέρει ή καθόλου από την σχεδίαση, όπως έγινε και στην προηγούμενη υποπαράγραφο.

00 01 02 03 04 05 06 07	08 09 10 11 12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Type	Code	ICMP header checksum
Identifier		Sequence Number
Data		

Εικόνα 5: Η δομή του πακέτου ICMP.

- Type: Περιγράφει την υπηρεσία που μεταφέρει το πακέτο ICMP. Η σχεδίαση υποστηρίζει μόνο τις Echo Request και Echo Reply. Οι κωδικοί που αναφέρονται στις υπηρεσίες αυτές είναι οι 0x08 και 0x00, αντίστοιχα. Σημειώνεται ότι το σύστημα απαντά αυτόματα σε κάθε Echo Request.
- Code: Χρησιμοποιείται για να δηλωθούν οι κωδικοί Echo Request & Reply. Την σχεδίασή μας αφορά μόνο η τιμή 0x00.
- Sequence Number: Ένας αριθμός που επιλέγεται από το σύστημα που εκτελεί την αίτηση Echo Request. Ο αριθμός αυτός πρέπει να επιστραφεί από την συσκευή που απαντά με Echo Reply. Η σχεδίασή μας, προκειμένου να μην ξοδέψει μνήμη στην αποθήκευση τέτοιου είδους δεδομένων, απαντά πάντα με μία σταθερή ακολουθία αριθμών, που έχει παρατηρηθεί ότι αποστέλλεται από την πλειονότητα των σταθμών, που στέλνουν Echo Request (0x61 … 0x77, 0x61 … 0x69).

3.3 Το Επίπεδο Διακίνησης

Το επίπεδο διακίνησης προσφέρει την υπηρεσία μεταφοράς μηνυμάτων που δημιουργούν οι εφαρμογές του υψηλότερου επιπέδου από ένα σταθμό σε έναν άλλον. Τα πρωτόκολλα που χρησιμοποιούνται στο διαδίκτυο για την μεταφορά μηνυμάτων είναι το UDP (User Datagram Protocol) και TCP (Transmission Control Protocol). Τα δύο αυτά πρωτόκολλα ακολουθούν το μοντέλο πελάτη – εξυπηρετητή.

Η διαφορά των πρωτοκόλλων που προαναφέρθηκαν έγκειται στο γεγονός ότι το μεν UDP δεν δημιουργεί συνδέσεις μεταξύ εφαρμογών και δεν παρέχει καμία

απολύτως εγγύηση για την αξιόπιστη μεταφορά δεδομένων, ενώ το δε, TCP, δημιουργεί συνδέσεις μεταξύ εφαρμογών και ταυτόχρονα παρέχει εγγύηση για την αξιόπιστη μεταφορά δεδομένων, σε συνδυασμό με μηχανισμούς ελέγχου της συμφόρησης (Congestion Control).

3.3.1 Το πρωτόκολλο UDP

Το πρωτόκολλο UDP αποτελεί, κατά κάποιο τρόπο, την «συνέχεια» του IP, με την επιπλέον δυνατότητα μεταφοράς δεδομένων σε επίπεδο διεργασιών. Η επιπλέον πληροφορία που φέρει ένα πακέτο UDP (source port & destination port number) χρησιμοποιείται για την πολυπλεξία και αποπολυπλεξία των μηνυμάτων, που στέλνονται και λαμβάνονται αντίστοιχα. Στην εικόνα 6 παρουσιάζεται η δομή ενός πακέτου UDP. Τα πεδία τα οποία υποστηρίζονται πλήρως από την σχεδίασή μας, επεξηγούνται στο Παράρτημα.

00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Source Port	Destination Port
Length	Checksum
Data	

Εικόνα 6: Η δομή του πακέτου UDP.

3.3.2 Το πρωτόκολλο TCP

Το πρωτόκολλο TCP μοιάζει με το UDP σε ελάχιστα σημεία. Τα σημεία αυτά είναι η πολυπλεξία/αποπολυπλεξία και η δημιουργία του Checksum. Κατά τ'άλλα υπάρχουν θεμελιώδεις διαφορές μεταξύ των δύο πρωτοκόλλων. Όπως έχουμε ήδη δει, το πρωτόκολλο TCP εξασφαλίζει συνδέσεις μεταξύ εφαρμογών, που τρέχουν σε απομακρυσμένους σταθμούς. Επιπλέον, το βασικότερο χαρακτηριστικό του, είναι ότι εγγυάται την αξιόπιστη μεταφορά δεδομένων. Λέγοντας αξιόπιστη, εννοούμε την μεταφορά χωρίς απώλειες.

Θεμελιώδη ρόλο στην αξιόπιστη μεταφορά και γενικότερα, στην μεταφορά δεδομένων χωρίς προβλήματα, παίζει το σύστημα του ελέγχου συμφόρησης (Congestion Control) που διαθέτει. Η τέταρτη έκδοση της σχεδίασής μας, υποστηρίζει τον έλεγχο συμφόρησης, με ακόμη μεγαλύτερη πληρότητα. Στις

επόμενες παραγράφους θα έχουμε την δυνατότητα να δούμε με πολύ μεγαλύτερη λεπτομέρεια το σύστημα αυτό.

Πριν την ανάλυση του ελέγχου συμφόρησης από την σκοπιά των προδιαγραφών, ας δούμε την δομή ενός πακέτου TCP, ακολουθώντας την ίδια «γραμμή» με τις προηγούμενες παραγράφους, αναφέροντας λεπτομερώς, δηλαδή, μόνο τα «κρίσιμα» για την σχεδίασή μας πεδία.

00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Source Port	Destination Port
Sequence Number	
Acknowledgment Number	
Data Offset / Unused / Flags	Receive Window
Checksum	Urgent Pointer
Options and padding	
Data	

Εικόνα 7: Η δομή του πακέτου TCP.

- Flags: Τα πεδία αυτά παρέχουν πληροφορίες σχετικά με την εγκυρότητα κάποιων από τα υπόλοιπα πεδία της επικεφαλίδας ή το αν συμμετέχουν σε διαδικασία χειραψίας. Τα flags αυτά είναι:
    - URG: Το πεδίο Urgent Pointer είναι έγκυρο.
    - ACK: Το πεδίο Acknowledge number είναι έγκυρο.
    - PSH: Τα δεδομένα παραδίδονται άμεσα στην εφαρμογή.
    - RST: Αρχικοποίηση της σύνδεσης.
    - SYN: Ένδειξη της διαδικασίας καθιέρωσης της σύνδεσης.
    - FIN: Ένδειξη της διαδικασίας διακοπής (κλεισίματος) της σύνδεσης.

  Η τέταρτη έκδοση της σχεδίασης μας, υποστηρίζει πλήρως όλα τα Flags, πλην του URG. Ειδικά, στην περίπτωση του PSH Flag, επιστρέφει άμεσα μήνυμα στο επίπεδο εφαρμογών, που ειδοποιεί για επιτυχή λήψη, ενημερώνοντας έτσι την εφαρμογή ότι έφτασε το πακέτο. Σε διαφορετική περίπτωση η ενημέρωση αυτή δεν γίνεται άμεσα μετά από κάθε λήψη, αλλά σε συγκεκριμένες χρονικές στιγμές, που ορίζονται από παράγοντες όπως το buffer λήψης, το Flight size, την ολοκλήρωση της λήψης της ροής κλπ.

- Receive Window: Στο πεδίο αυτό φυλάσσεται η πληροφορία σχετικά με το μέγεθος δεδομένων που μπορεί να δεχτεί ο παραλήπτης. Η χρήση του σκοπό έχει να περιορίσει την ροή, που αποστέλλεται σύμφωνα με τις ανάγκες του δέκτη και παράλληλα να περιορίσει την συμφόρηση στο δίκτυο από πακέτα που αποστέλλονται και δεν γίνονται δεκτά λόγω της αδυναμίας του παραλήπτη να τα απορροφήσει. Η σχεδίασή μας υποστηρίζει το πεδίο αυτό, χωρίς όμως να είναι σε θέση να το συμπληρώσει, ελέγχοντας παράγοντες όπως η συμφόρηση στο δίκτυο κλπ. Η αδυναμία αυτή θεωρείται σχεδόν αμελητέα με δεδομένη την βελτίωση του Congestion Control στην νέα έκδοσή της.
- Urgent Pointer: Όπως είδαμε προηγουμένως, το URG Flag δεν αξιοποιείται από την σχεδίασή μας. Συνεπώς, το πεδίο αυτό, επίσης, αγνοείται. Ο pointer αυτός χρησιμοποιείται για να δείξει μέχρι ποιο byte από τα εισερχόμενα δεδομένα βρίσκονται τα επείγοντα δεδομένα, που πρέπει να παραδοθούν άμεσα.
- Options: Στο πεδίο αυτό φιλοξενούνται κάποιες επιπλέον πληροφορίες, όπως το μέγιστο μέγεθος τμήματος (maximum segment size – MSS), το Echo Reply, το Timestamp κλπ. Η μόνη πληροφορία που δύναται να αναγνωρίσει η σχεδίαση μας είναι το maximum segment size. Σημειώνεται ότι, οι πληροφορίες στο πεδίο αυτό αναγνωρίζονται ελέγχοντας μία σειρά από bytes. Το πρώτο byte ενημερώνει τον δέκτη για τις πληροφορίες που θα ακολουθήσουν. Στην περίπτωσή μας η σειρά τον bytes φαίνονται στον πίνακα 3.

Byte #	**Πληροφορία**	**Τιμή**
0	Κωδικός παραμέτρου	*0x02*
1	Μέγεθος παραμέτρου	*0x04*
2	Most Significant Bit (MSB)	*MSB*
3	Less Significant Bit (LSB)	*LSB*

Πίνακας 3: Η υποστηριζόμενη από την σχεδίαση μας, παράμετρος MSS

Πριν προχωρήσουμε στο επόμενο επίπεδο, εκείνο της συνόδου, θα πρέπει να σταθούμε λίγο περισσότερο στο βασικό χαρακτηριστικό του εξεταζόμενου πρωτοκόλλου, τον έλεγχο συμφόρησης (Congestion Control). Αυτό επιβάλλεται,

κυρίως, επειδή, η υποστήριξή του από την τρίτη και τέταρτη έκδοση της σχεδίασης στάθηκε αιτία για την αναθεώρηση και τον επανασχεδιασμό ολόκληρου του μηχανισμού του υποσυστήματος αποστολής/λήψης πλαισίων TCP.

Ο έλεγχος συμφόρησης στηρίζεται σε τέσσερις αλγορίθμους, τους Slow Start (αργής εκκίνησης), Congestion Avoidance (Αποφυγής Συμφόρησης), Fast Retransmit (Γρήγορης Επαναμετάδοσης) και Fast Recovery (Γρήγορης Ανακάμψης – σε ελεύθερη μετάφραση). Στο κεφάλαιο αυτό θα περιοριστούμε σε εκείνα να χαρακτηριστικά των αλγορίθμων, που έθεσαν τις προδιαγραφές στην σχεδίαση στις δύο τελευταίες εκδόσεις της.

Οι δύο πρώτοι αλγόριθμοι, οι Slow Start και Congestion Avoidance, χρησιμοποιούν δύο μεταβλητές, τις Congestion Window (CongWin) και Threshold (sThres). Οι μεταβλητές αυτές είναι απαραίτητες και στους δύο εμπλεκόμενους απομακρυσμένους σταθμούς, που επιχειρούν να μεταδώσουν με το πρωτόκολλο TCP. Για το λόγο αυτό, η σχεδίασή μας διαθέτει ειδικές θέσεις, σε μνήμη που κρατά τα στοιχεία κάθε σύνδεσης. Και οι δύο αλγόριθμοι υποστηρίζονται πλήρως από την τέταρτη έκδοση της σχεδίασης.

Οι αλγόριθμοι Fast Retransmit και Fast Recovery, όπως περιγράφεται και στο παράρτημα, έχουν να κάνουν με την επαναμετάδοση χαμένων πακέτων. Η κατάσταση αυτή γίνεται αντιληπτή, είτε με την λήψη τριπλότυπης[1] αναφοράς παράδοσης (Ack – Acknowledge), είτε με το timeout κάποιου χρονομετρητή (timer), που διαθέτει το πρωτόκολλο TCP. Εμβαθύνοντας λίγο περισσότερο, ο Fast Retransmit ενεργοποιείται στην περίπτωση στην περίπτωση λήψης τριπλότυπης ack και εν συνεχεία αναλαμβάνει δράση ο Fast Recovery. Ο τελευταίος ρυθμίζει το Threshold και το Congestion Window, προκειμένου να επιτευχθεί γρήγορη ανάκαμψη του ρυθμού αποστολής της σύνδεσης.

Η σχεδίασή μας, στην τέταρτη έκδοσή της, υποστηρίζει μόνο τον αλγόριθμο Fast Retransmit. Στην θέση του Fast Recovery υπάρχει ο Slow Start. Η αντικατάσταση αυτή έγινε για λόγους απόδοσης σε περίπτωση πολλαπλών απωλειών πακέτων. Εν γένει, έχει αποδειχτεί ότι ο Fast Recovery δεν αποδίδει καλά σε περιπτώσεις πολλαπλών απωλειών και αυτό γιατί διογκώνει για μικρό χρονικό διάστημα το congestion window, προκειμένου να παραδοθούν τα ήδη απεσταλμένα

[1] Η λέξεις «τριπλότυπη» και «διπλότυπη» προέρχονται από την μετάφραση των αγγλικών όρων «triple» και «double» αντίστοιχα και χρησιμοποιούνται συνήθως αντί των λέξεων «τριπλή» και «διπλή» σε μεταφρασμένα συγγράμματα που αναφέρονται στο πρωτόκολλο TCP και κατ' επέκταση στα δίκτυα υπολογιστών.

πακέτα. Στην σχεδίαση δόθηκε μεγαλύτερο βάρος στην δραματική μείωση των απωλειών και λιγότερο, στην τάχιστη ανάκαμψη του συστήματος, όσον αφορά σε άποψη ταχύτητας. Για τον παραπάνω λόγο, επιλέχθηκε ο αλγόριθμος Slow Start στην θέση του Fast Recovery.

Όταν μιλάμε για απώλειες και συνεπώς, για επαναμεταδόσεις, θα πρέπει να ξέρουμε ότι στην πλειοψηφία των περιπτώσεων, εκείνος που αποφασίζει για επαναμετάδοση ή κάποιου άλλου είδους αντίδραση από πλευράς του TCP, είναι ένας χρονομετρητής (timer). Η τέταρτη έκδοση της σχεδίασης του Open TCP/IP Core παρουσιάζει πληρότητα από άποψης υποστήριξης των timers καθώς συμπεριλαμβάνει με βελτιωμένη δομή τους ήδη υπάρχοντες από την τρίτη έκδοση timers, ενώ, παράλληλα, διαθέτει επιπλέον άλλους δύο.

Οι timers που «τρέχουν» για την υλοποίηση του congestion control είναι συνολικά πεντε: ο Syn Timer, ο Retransmission Timer, ο Persistence Timer, ο Idle Timer και ο Fast Retransmit Timer. Κάθε ένας από αυτούς παίζει θεμελιώδη ρόλο στην άρτια λειτουργία του ελέγχου συμφόρησης. Χωρίς να μπαίνουμε, σε αυτήν την φάση σε λεπτομέρειες που αφορούν στον σχεδιασμό του υποσυστήματος των timers, ας δούμε την λειτουργία καθενός εξ'αυτών, προκειμένου να ορίσουμε τις προδιαγραφές με πιο παραστατικό τρόπο.

- Ο Syn Timer είναι υπεύθυνος για την παρακολούθηση της φάσης χειραψίας κατά την εκκίνηση μίας σύνδεσης. Όταν το σύστημα αποστείλει SYN για εκκίνηση σύνδεσης, τότε αυτός ενεργοποιείται. Σε περίπτωση που αυτός κάνει time out, τότε επιστρέφεται μήνυμα λάθους στην εφαρμογή που ζητά την σύνδεση.
- Ο Retransmission Timer είναι υπεύθυνος για τις επαναμεταδόσεις, πλην εκείνων που προκαλούνται από τριπλότυπες επιβεβαιώσεις. Ενεργοποιείται με την αποστολή ενός πακέτου και απενεργοποιείται με την λήψη του τελευταίου πακέτου σε κάθε Flight Size. Σε περίπτωση time out, δημιουργεί μία εσωτερική (ως προς το υποσύστημα) εντολή, την εντολή "Retransmit Segment".
- Ο Persistence Timer ανήκει στην κατηγορία των timers που επιβλέπουν το υποσύστημα TCP, ενεργοποιούνται, όμως, πιο σπάνια και σε περιπτώσεις που θα μπορούσαν να παραλύσουν το σύστημα, όπως για παράδειγμα ένα deadlock. Το deadlock, που καλείται να

επιλύσει με την παρουσία του ο εν λόγω timer, είναι εκείνο που προκαλείται σε περίπτωση που Receive Window έχει πάρει την τιμή μηδέν και το πακέτο που θα ανανέωνε την τιμή αυτή έχει χαθεί. Ο Persistence Timer ελέγχει την τιμή του παραθύρου και ενεργοποιείται όταν διαπιστώσει ότι αυτή είναι μηδέν. Σε περίπτωση time out δημιουργεί την εσωτερική εντολή "Send Probe". Η εντολή αυτή δημιουργεί ένα κενό πακέτο που παίζει το ρόλο του «δειγματολήπτη» και αναγκάζει τον δέκτη να στείλει ack, το οποίο φέρει την σωστή τιμή του receive window.

- Ο Idle Timer επιβλέπει την σύνδεση από πλευράς φορτίου. Όταν αυτή μείνει ανενεργή για μεγάλο χρονικό διάστημα, τότε είναι πολύ πιθανόν η κατάσταση του δικτύου να έχει αλλάξει. Για να αποφευχθούν απώλειες, σε περίπτωση που το φορτίο του δικτύου είναι μεγαλύτερο απ'ότι ήταν όταν η σύνδεση σταμάτησε να στέλνει, ο timer παρεμβαίνει και αλλάζει την τιμή του congestion window μέγεθος δύο πακέτων. Η εκκίνηση του timer γίνεται με το τέλος της αποστολής μίας ροής και το timeout του προκαλεί την εσωτερική εντολή "Initialize Congestion Window". Σε περίπτωση που η αποστολή της νέας ροής ξεκινήσει πριν γίνει time out, ο timer απενεργοποιείται.
- Ο Fast Retransmit Timer, ο οποίος ουσιαστικά είναι counter και συμπεριλήφθηκε εδώ για λόγους οργάνωσης, μετράει τον αριθμό των επιβεβαιώσεων και σε περίπτωση που αυτός γίνει ίσος με τρία, τότε προκαλεί την άμεση αποστολή των απολεσθέντων πακέτων (εντολή "Retransmit Segments After Last Ack").

Από τους πέντε timers που αναφέρθηκαν οι Syn, Fast Retransmit και Retransmission timers υπήρχαν και στην τρίτη έκδοση της σχεδίασης, ενώ οι Persistence και Idle απουσίαζαν, αν και υπήρχε μια σχετική μελέτη για την εισαγωγή τους. Όλοι οι timers, πλην του Fast Retransmit, επανασχεδιάστηκαν και άλλαξε η επίδρασή τους στις περιπτώσεις των timeouts.

Ειδικότερα, στις περιπτώσεις των timeouts, η δομή, όπως φαίνεται στην ενότητα, άλλαξε εντελώς. Αυτό έγινε, διότι, στην προηγούμενη έκδοση ήταν αδύνατη η εξυπηρέτηση πολλαπλών παράλληλων (ταυτόχρονων) timeouts, γεγονός που προσέδιδε στην σχεδίαση αστάθεια και πολλές πιθανότητες αστοχίας. Στην παρούσα

έκδοση το πρόβλημα αυτό επιλύθηκε με αντίτιμο ένα αμελητέο κόστος καθυστέρησης.

Άλλο ένα ζήτημα που προκαλούσε ασυμβατότητα με τα RFCs, χωρίς, ωστόσο, να προκαλεί μεγάλα προβλήματα ήταν ο χρόνος timeout ή πιο σωστά το Round Trip Time (RTT). Στην προηγούμενη έκδοση, για λόγους απλότητας, ο χρόνος αυτός δεν ακολουθούσε τα πρότυπα του αντίστοιχου RFC, όσον αφορά στην επιτυχημένη μετάδοση: ήταν ίσος πάντοτε με τον χρόνο της προηγούμενης επιτυχούς μετάδοσης. Στην έκδοση αυτή, το χαρακτηριστικό αυτό βελτιώθηκε και πλέον ισούται, σύμφωνα με τον κινούμενο εκθετικό μέσο όρο βάρους (exponential weighted moving average), με το 90% της παλαιάς τιμής συν το 10% του χρόνου της τελευταίας επιτυχούς μετάβασης (μιλώντας με πράξεις: Estimated_RTT = (1-x)*Estimated_RTT + x*Sample_RTT, όπου x τυπικά 0,1). Για τον υπολογισμό του χρόνου αυτού χρησιμοποιήθηκε ένα επιπλέον υποσύστημα που, αν και χρονοβόρο, λόγω του παραλληλισμού που επιτυγχάνεται με την νέα σχεδίαση, δεν καθυστερεί το συνολικό υποσύστημα TCP.

Τέλος, πριν κλείσουμε την μελέτη, του υποσυστήματος αυτού θα πρέπει να αναφέρουμε κάποια, επιπλέον ευέλικτα χαρακτηριστικά που προστέθηκαν στην νέα σχεδίαση. Αυτά είναι:

- Ο παραλληλισμός στο υποσύστημα αποκωδικοποίησης. Το θέμα αναφέρεται με λεπτομέρεια στην επόμενη παράγραφο, επειδή το υποσύστημα αυτό εμπλέκεται με τις εντολές που δέχεται το Επίπεδο Συνόδου.
- Η εξυπηρέτηση πολλών ροών προς αποστολή παράλληλα. Στις προηγούμενες εκδόσεις οι διεργασίες[2] αποστολής πλαισίων δεσμεύονταν πλήρως μέχρι την πλήρη αποστολή κάθε ροής. Επειδή, όμως, οι περιπτώσεις, κατά τις οποίες το εικονικό υποσύστημα αποστολής πλαισίων περίμενε κάποια επιβεβαίωση και ήταν σε αδράνεια, την ίδια στιγμή που περιπτώσεις ύπαρξης και άλλων ροών προς αποστολή από άλλες συνδέσεις δε σπάνιζαν, έπρεπε να αλλάξει η δομή, προκειμένου η σχεδίαση να γίνει αποδοτικότερη. Έτσι στην νέα έκδοση, όταν το υποσύστημα αποστολής πλαισίων TCP πέφτει σε

[2] Ο όρος «διεργασία» αναφέρεται στον όρο process. Στην προηγούμενη έκδοση του TCP/IP Core ο κώδικας του υποσυστήματος αποστολής/λήψης πλαισίων TCP, διέθετε πολλά processes που χώριζαν το σύστημα σε πολλά επιμέρους «νοητά» υποσυστήματα. Οι εν λόγω διεργασίες αποτέλεσαν το πρώτο βήμα για την διάσπαση του υποσυστήματος TCP.

προσωρινή αδράνεια και υπάρχει άλλη διαθέσιμη ροή προς αποστολή, τότε αυτή εξυπηρετείται μέχρι να ληφθεί το αναμενόμενο ack. Το υποσύστημα είναι με τέτοιο τρόπο σχεδιασμένο, ώστε να μην χάνονται τα δεδομένα της αποστολής (το offset κλπ). Τη λειτουργία αυτή θα την δούμε διεξοδικά στο κεφάλαιο της σχεδίασης του Open TCP/IP.

3.4 Το Επίπεδο Συνόδου

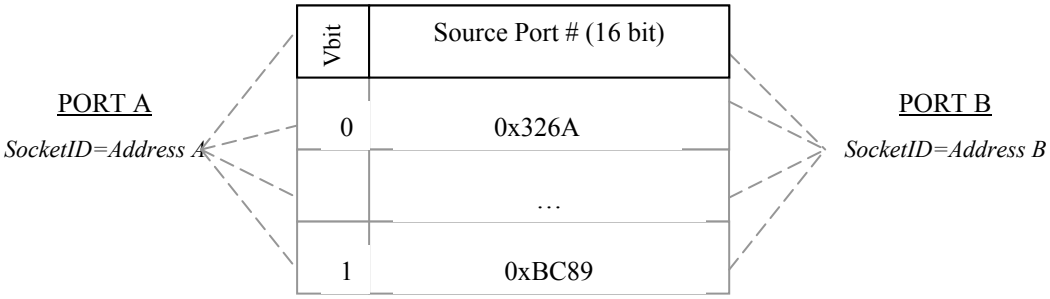
Στο Επίπεδο Συνόδου το μοντέλο προγραμματισμού προσφέρει στους χρήστες που θέλουν να αναπτύξουν εφαρμογές, οι οποίες θα επικοινωνούν μεταξύ τους την έννοια των Sockets. Η δημιουργία ενός socket, προκαλεί την δημιουργία μίας δομής η οποία είναι ικανή να φυλάσσει τις μεταβλητές που είναι απαραίτητες για την ύπαρξη του socket και την δημιουργία σύνδεσης από το πρωτόκολλο του κατώτερου επίπέδου. Η δομή αυτή ονομάζεται TCB (Transmission Control Block). Σημειώνεται ότι η καταστροφή ενός socket και το κλείσιμο της σύνδεσης προκαλεί, κατ'αντιστοιχία με όσα προαναφέρθηκαν, την διαγραφή της δομής αυτής. Στις υποπαραγράφους που ακολουθούν, θα μελετήσουμε τις δομές αυτές, καθώς και τις εντολές που παρέχονται στα παραπάνω επίπεδα για την διαχείριση των sockets.

3.4.1 Τα Sockets και οι Εντολές στο πρωτόκολλο UDP

Στο πρωτόκολλο UDP μπορούμε να διακρίνουμε δύο τύπους Sockets. Στον πρώτο τύπο ανήκουν τα Sockets τα οποία ανοίγουν για να εξυπηρετήσουν εφαρμογές που δουλεύουν σαν εξυπηρετητές. Στον δεύτερο τύπο ανήκουν τα Sockets που ανοίγονται από εφαρμογές που δουλεύουν σαν πελάτες. Τα πρώτα ανοίγουν περιμένοντας δεδομένα, ενώ, τα δεύτερα ανοίγουν για να στείλουν δεδομένα σε κάποια εφαρμογή εξυπηρετητή, που τρέχει σε απομακρυσμένο μηχάνημα.

Από άποψης hardware, πριν ανοίξει ένα socket γίνεται έλεγχος προκειμένου να διαπιστωθεί εάν αυτό είναι ήδη ανοιχτό. Στη περίπτωση που είναι ανοιχτό επιστρέφεται ένα μήνυμα σφάλματος, διαφορετικά ένα μήνυμα επιτυχίας, το οποίο ανάμεσα στις άλλες πληροφορίες, φέρει και τον αριθμό του socket, το SocketID.

Ο πίνακας των UDP Sockets, που υλοποιείται με μία δίπορτη μνήμη, φαίνεται στην εικόνα 7 της επόμενης σελίδας.



Εικόνα 8: Ο πίνακας των UDP Sockets.

	Όνομα Εντολής	Κωδικός	Ορίσματα	Περιγραφή
Εντολές από το επίπεδο εφαρμογής	Open_Server_Socket	001	Source Port #	Ανοίγει Sockets για εφαρμογές εξυπηρετητή.
	Open_Client_Socket	010	1. Source Port # 2. Remote IP	Ανοίγει Sockets για εφαρμογή πελάτη.
	Send_Datagram	011	1. SocketID 2. Remote IP 3. Offset 4. Byte Count 5. Destination port #	Αποστέλλει δεδομένα από την μνήμη αποστολής.
	Close Socket	100	Socket ID	Κλείνει ένα Socket.
Απαντήσεις προς το επίπεδο εφαρμογής	Suc_Open_Server_Socket	001	SocketID	Το Socket ανοίχθηκε επιτυχώς.
	Err_Open_Server_Socket	110	Source Port #	Σφάλμα στο άνοιγμα του Socket.
	Suc_Open_Client_Socket	010	SocketID	Το Socket ανοίχθηκε επιτυχώς.
	Err_Open_Client_Socket	101	Source Port #	Σφάλμα στο άνοιγμα του Socket.
	Suc_Send	011	SocketID	Η αποστολή ολοκληρώθηκε επιτυχώς.
	Err_Send	100	SocketID	Σφάλμα στην αποστολή
	Delivered_Data	111	1. SocketID 2. Remote IP 3. Offset 4. Byte Count 5. Source Port #	Η λήψη δεδομένων ολοκληρώθηκε και τα δεδομένα έχουν φθάσει στην μνήμη λήψης.

Πίνακας 4: Οι εντολές και τα αντίστοιχα ορίσματα του UDP

Δεδομένου, ότι η διαχείριση των socket πρέπει να είναι μία διαφανής εργασία στο επίπεδο των εφαρμογών, παρέχονται από την σχεδίαση κάποιες εντολές που συνοδεύονται από ορίσματα. Τα ορίσματα κατά το πέρασμά τους στο επίπεδο Συνόδου φυλάσσονται σε μία FIFO εύρους δεκαέξι bits. Όσον αφορά στις εντολές, αυτές παρέχονται με δυαδική μορφή. Σε κάθε επιτυχή ολοκλήρωση εντολής επιστρέφεται ως απάντηση η ίδια (δυαδική) εντολή, διαφορετικά το συμπλήρωμά της. Σημειώνεται ότι το SocketID που επιστρέφεται ως απάντηση, αποτελείται από το 4 less significant bits του local port number. Οι εντολές από και προς το επίπεδο εφαρμογής συνοψίζονται στον πίνακα 4.

3.4.2 Sockets και Εντολές στο πρωτόκολλο TCP

Αν και τα Sockets του πρωτοκόλλου TCP έχουν κάποια ομοιότητα με τα αντίστοιχα του πρωτοκόλλου UDP, παρουσιάζονται και σημαντικές διαφορές. Το γεγονός αυτό απορρέει από την ανάγκη φύλαξης πολύ περισσότερων πληροφοριών σε σχέση με το UDP. Η ομοιότητα εμφανίζεται στον τομέα των δύο παρεχόμενων τύπων Sockets, εκείνων που εξυπηρετούν εφαρμογές τύπου πελάτη και εκείνων που εξυπηρετούν εφαρμογές τύπου εξυπηρετητή.

Κάθε σύνδεση είναι μοναδική στο δίκτυο, αφού αποτελείται από ένα μοναδικό ζεύγος sockets. Κάθε socket με την σειρά του περιλαμβάνει την κρίσιμη πληροφορία του source port number της τοπικής εφαρμογής, του destination port number της απομακρυσμένης εφαρμογής και της διεύθυνσης IP του απομακρυσμένου σταθμού. Οι παραπάνω πληροφορίες, σε συνδυασμό με μία δυαδική τιμή που ορίζει την κατάσταση της σύνδεσης, αποτελούν σημαντικότατες πληροφορίες και φυλάσσονται στο TCB μαζί με άλλες που αφορούν στον έλεγχο συμφόρησης κλπ.

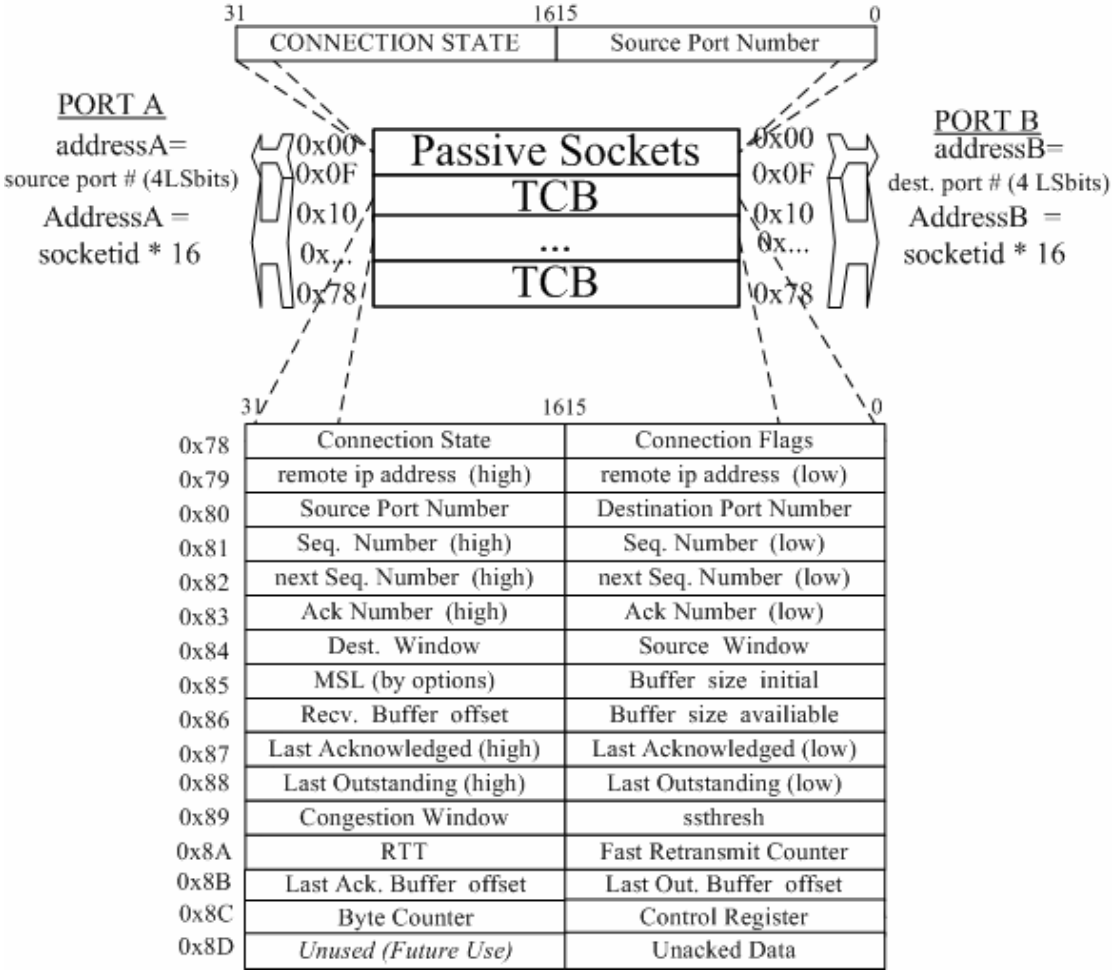
Το TCB για το πρωτόκολλο TCP, υλοποιείται με μία δίπορτη σύγχρονη μνήμη. Η προσπέλασή του γίνεται ανά δεκάξι θέσεις με την βοήθεια του αντίστοιχου περιγραφέα, του SocketID, που είδαμε και στο UDP. Το SocketID για τα sockets τύπου Client (active sockets) και για τα Sockets τύπου Server (passive sockets) σχηματίζονται ως εξής:

- Active SocketID: Remote IP[0] & Destination Port #[1…0] & Source Port #[1…0].
- Passive SocketID:
    - Port A: ‘0’ & Source Port # [3…0].
    - Port B: ‘0’ & Destination Port #[3…0] .

Χρησιμοποιώντας την παραπάνω οργάνωση, μπορούμε να υποστηρίξουμε μέχρι τριανταένα συνδέσεις. Στην προηγούμενη έκδοση αν και οι προδιαγραφές ήταν για το ίδιο αριθμό συνδέσεων, εντούτοις, είχαν υλοποιηθεί μόνο είκοσι για λόγους απλότητας. Η συγκεκριμένη έκδοση υποστηρίζει πλήρως και τις τριανταένα συνδέσεις με το ανάλογο κόστος, βέβαια, σε υλικό.

Όσον αφορά στα passive sockets, που ανοίγουν για εφαρμογές τύπου εξυπηρετητή, αυτά υλοποιούνται στις χαμηλότερες δεκαέξι θέσεις του TCB. Οι εγγραφές που είναι απαραίτητες για τα sockets αυτά, είναι η κατάσταση της σύνδεσης και το source port number της τοπικής εφαρμογής που άνοιξε το socket. Σε κάθε αίτηση για σύνδεση εξετάζεται εάν η πληροφορία που περικλείει για το destination port number είναι ίδια με το source port number της αντίστοιχης εγγραφής στο TCB και εάν η κατάσταση της σύνδεσης είναι LISTEN.

Η δομή του πίνακα του sockets παρουσιάζεται στην εικόνα 8.



Εικόνα 9: Το Socket Table του TCP

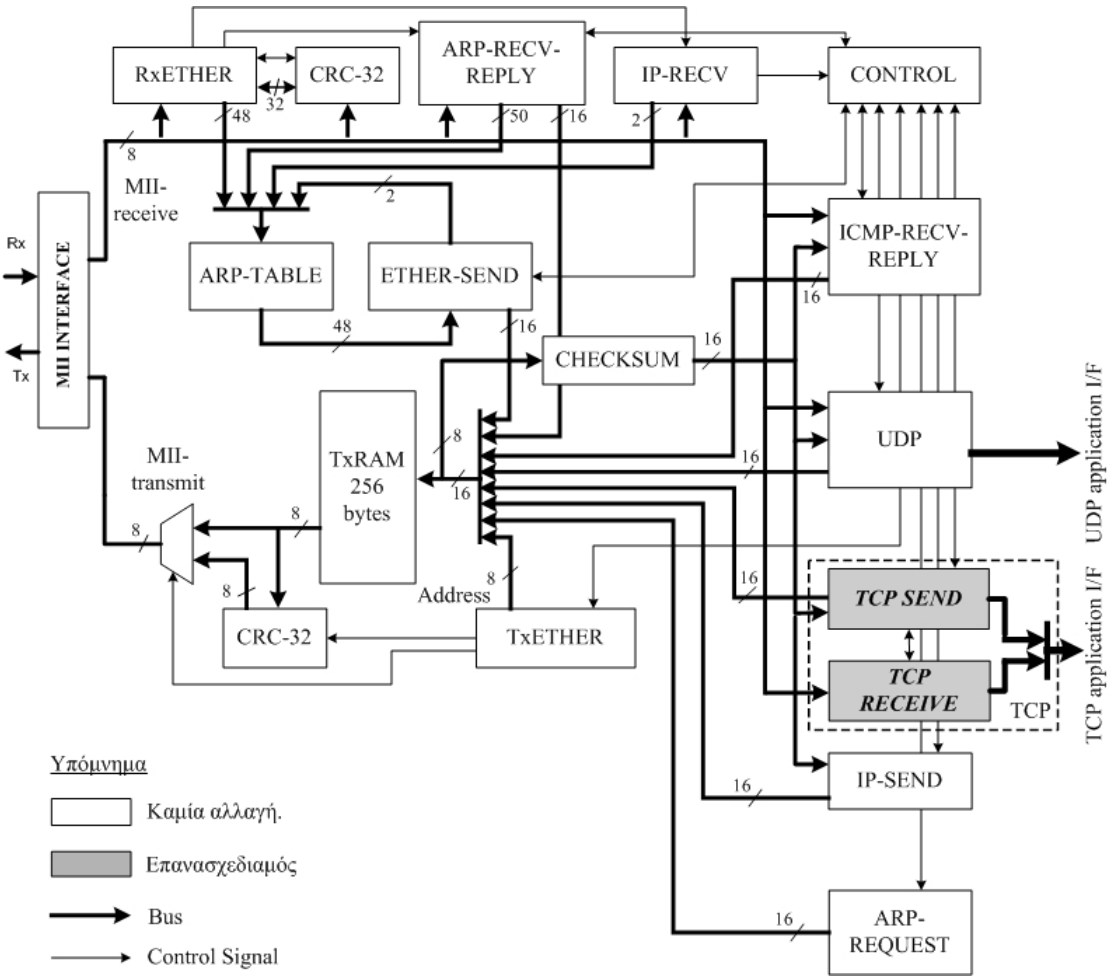
Οι εντολές που υποστηρίζει η σχεδίαση και παρέχουν πρόσβαση στο πρωτόκολλο TCP, είναι παρόμοιες με αυτές που παρέχει το μοντέλο προγραμματισμού. Όπως και με τις αντίστοιχες εντολές που υποστηρίζει η σχεδίαση για το UDP, κάθε εντολή αντιπροσωπεύεται με ένα δυαδικό κωδικό. Τα ορίσματα διακινούνται με την βοήθεια FIFOs εύρους 16 bits. Σε περίπτωση σφάλματος επιστρέφεται το συμπλήρωμα του κωδικού της εκάστοτε εντολής, αντίθετα, σε περίπτωση επιτυχίας, ο ίδιος κωδικός. Οι εντολές αυτές περιγράφονται στον πίνακα 5.

	Όνομα	Κωδικός	Ορίσματα	Περιγραφή
ΕΝΤΟΛΕΣ ΑΠΟ ΤΟ ΕΠΙΠΕΔΟ ΤΗΣ ΕΦΑΡΜΟΓΗΣ ΠΡΟΣ ΤΟ TCP	Close_Passive_Socket	0001	Source Port #	Άνοιγμα socket για εφαρμογές εξυπηρετητή και αναμονή για αιτήσεις σύνδεσης
	Open_Active_Socket	0010	Remote IP Destin. Port # Source Port # Cong. Win. Threshold	Άνοιγμα socket για εφαρμογές πελάτη. Αποστολή αίτησης για σύνδεση.
	Send_Segment	0011	SocketID Offset Byte Count	Αποστολή δεδομένων από συγκεκριμένη θέση της μνήμης αποστολής.
	Receive_Segment	0100	SocketID Offset Byte Count	Ενεργοποιεί την δυνατότητα λήψης δεδομένων σε μία συγκεκριμένη θέση της μνήμης λήψης με χώρο ίσο με "byte count".
	Bind	0101	Remote IP Dest. Port # Source Port # Cong. Win. Threshold	Αποδοχή αίτησης σύνδεσης από απομακρυσμένο σταθμό, δημιουργία του αντίστοιχου TCB.
	Close_Active_Socket	0111	SocketID	Κλείσιμο του socket τύπου πελάτη (active socket), αποστολή FIN και διαγραφή του TCB.
	Close_Passive_Socket	1111	SocketID	Κλείσιμο socket τύπου εξυπηρετητή και διαγραφή του TCB.
ΕΝΤΟΛΕΣ ΑΠΟ ΤΟ TCP ΣΤΟ ΕΠΙΠΕΔΟ ΕΦΑΡΜΟΓΗΣ	Suc_Open_Server_Socket	0001	SocketID	Επιτυχία δημιουργίας του passive socket.
	Err_Open_Server_Socket	1110	Source Port #	Σφάλμα δημιουργίας του passive socket.
	Suc_Open_Client_Socket	0010	SocketID	Επιτυχία δημιουργίας του active socket.
	Err_Open_Clien_Socket	1101	Remote IP Destin. Port # Source Port #	Σφάλμα δημιουργίας του active socket.
	Suc_Send	0011	SocketID	Επιτυχία αποστολής.
	Err_Send	1100	SocketID	Σφάλμα στην αποστολή.
	Suc_Receive	0100	SocketID Offset Byte Count	Η λήψη ενός μηνύματος μεγέθους "byte count" στην θέση "offset" της μνήμης λήψης έχει ολοκληρωθεί.
	Err_Receive	1011	SocketID	Σφάλμα λήψης.
	Connection_Request	1111	SocketID (LISTEN) Remote IP Destin. Port #	Ανακοίνωση ερώτησης απομακρυσμένου σταθμού για επικύρωση σύνδεσης.
	Close Wait	1001	SocketID	Ένδειξη ότι κάποια σύνδεση στο άλλο άκρο έχει κλείσει.

Πίνακας 5: Οι εντολές για το πρωτόκολλο TCP.

3.5 Η αρχιτεκτονική του Open TCP/IP Core

Στο σημείο αυτό, κρίνεται αναγκαία μία συνολική ματιά στην αρχιτεκτονική της τέταρτης έκδοσης της σχεδίασης. Στην εικόνα 9 φαίνεται η αρχιτεκτονική της παρούσας σχεδίασης. Ρίχνοντας μία σύντομη ματιά στην αρχιτεκτονική, παρατηρούμε ότι οι αλλαγές που έχουν γίνει ως προς την δομή είναι ελάχιστες και αφορούν μόνο το υποσύστημα αποστολής λήψης πλαισίων, το οποίο στην έκδοση αυτή έχει διασπαστεί σε δύο υποσυστήματα, ένα αποστολής και ένα λήψης πλαισίων.



Εικόνα 10: Η αρχιτεκτονική του Open TCP/IP Core.

Το παραπάνω συμπέρασμα είναι εν μέρει σωστό διότι, το μόνο υποσύστημα που υπέστη αλλαγές είναι το υποσύστημα αποστολής/λήψης πλαισίων TCP. Όμως, οι αλλαγές που έγιναν στο υποσύστημα αυτό, προκάλεσαν με αλυσιδωτό τρόπο μεγάλες

αλλαγές, οι οποίες με την σειρά τους οδήγησαν στον πλήρη επανασχεδιασμό σχεδών όλων των υποσυστημάτων (του TCP) και συγχρόνως στην εισαγωγή νέων, τα οποία απουσίαζαν εντελώς από τις προηγούμενες σχεδιάσεις. Επειδή στην εικόνα 9 παρουσιάζεται η συνολική αρχιτεκτονική του συστήματος, η παρουσίαση κάθε υποσυστήματος με λεπτομέρεια είναι αδύνατη. Αυτό θα γίνει στο επόμενο κεφάλαιο, στο οποίο κάθε υποσύστημα θα αποτελεί και μία ξεχωριστή παράγραφο, που θα αναλύει δομή, σχεδίαση και λειτουργία. Εδώ θα περιοριστούμε στην γενικότερη λειτουργία του συστήματος και πολύ περιληπτικά, στην λειτουργία κάθε υποσυστήματος.

Τα δεδομένα εισέρχονται στο υποσύστημα MII Interface ανά ομάδες των τεσσάρων bits και εξέρχονται προς το κύριο σύστημα ανά ομάδες των οκτώ bits. Η διαδικασία αυτή εξελίσσεται αντίστροφά κατά την αποστολή κάποιου πακέτου.

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 για το σύνολο του εισερχόμενου πλαισίου ένα πολυώνυμο. Από την τιμή του πολυωνύμου 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 ενημερώνει τον πίνακα ARP (ARP-TABLE), ενώ παράλληλα, ενημερώνει και 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) το εισερχόμενο πλαίσιο.

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 την γράψει στην κατάλληλη θέση της μνήμης TxRAM.

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 του πρωτοκόλλου Ethernet. Παράλληλα, συμβουλεύεται τον πίνακα ARP για να λάβει μια διεύθυνση Ethernet, που να αντιστοιχίζεται στη διεύθυνση IP του πακέτου που έχει γραφτεί στη μνήμη TxRAM.

Το υποσύστημα ARP-REQUEST ενεργοποιείται όταν, είτε το UDP, είτε το TCP ανοίγει καινούρια σύνδεση για λογαριασμό μιας εφαρμογής πελάτη από το επίπεδο εφαρμογών. Δημιουργεί 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, το οποίο υπολογίζει ένα πολυώνυμο για προσάρτηση στο τέλος του πλαισίου που μεταδίδεται, μέσω του πολυπλέκτη που υπάρχει στην έξοδο του συστήματος. Το πλαίσιο καταλήγει στο δίκτυο, αφού περάσει από το υποσύστημα MII Interface.

3.5.1 Το υποσύστημα MII

Το υποσύστημα MII 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. Το πρόβλημα που καλείται να λύσει το υποσύστημα αυτό, είναι ότι ο transceiver δέχεται και στέλνει δεδομένα κατά "τετράδες" bit ενώ η σχεδίαση μας δουλεύει με bytes. Ρόλος του υποσυστήματος MII είναι η γεφύρωση αυτού του χάσματος μεταξύ των δύο συστημάτων. Το υποσύστημα MII παρέμεινε ως έχει και στην έκδοση αυτή.

3.5.2 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 αναφοράς του πλαισίου. Στον έλεγχο αυτό διαπιστώνεται αν το πλαίσιο αναφέρεται στο IP και ενεργοποιείται το αντίστοιχο υποσύστημα για να αρχίσει η λήψη του.

4. Έλεγχο του CRC που υπάρχει στο τέλος του πλαισίου για να διαπιστωθεί ότι αυτό λήφθηκε ακέραιο και χωρίς σφάλματα.

5. Παροχή των δεδομένων που πρέπει να αποθηκευθούν στον πίνακα ARP, ώστε να παραμένει ενημερωμένος.

Το συγκεκριμένο υποσύστημα παρέμεινε ως έχει και σε αυτήν την σχεδίαση.

3.5.3 Το υποσύστημα αποστολής πλαισίων Ethernet: ETHER-SEND

Το υποσύστημα ETHER-SEND, το οποίο δεν μεταβλήθηκε, είναι υπεύθυνο για την προσθήκη της επικεφαλίδας του πρωτοκόλλου Ethernet, ώστε να υπάρχει ένα πλήρες πλαίσιο στη μνήμη μετάδοσης, έτοιμο για αποστολή. 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 γραφτεί στη μνήμη.

3.5.5 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.

Όταν το κανάλι γίνει διαθέσιμο, διαβάζει το μήκος του πλαισίου από την πρώτη θέση της μνήμης και στη συνέχεια, ξεκινάει τη μετάδοση στέλνοντας δεδομένα στη διεπαφή MII. Ταυτόχρονα ενεργοποιεί το υποσύστημα υπολογισμού CRC-32, η έξοδος του οποίου προσαρτάται στο τέλος του πλαισίου.

Καθ' όλη τη διάρκεια μετάδοσης, ελέγχεται η περίπτωση σύγκρουσης (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 ως έχει.

3.5.6 Το υποσύστημα ARP-RECV-REPLY

Το υποσύστημα *ARP-RECV-REPLY* 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. Στην περίπτωση αυτή, αρχίζει το διάβασμα στην επικεφαλίδα του και αν διαπιστωθεί ότι πρόκειται για ARP request, αναλαμβάνει να ετοιμάσει ένα πακέτο ARP response και να το στείλει στο σύστημα που έκανε την αίτηση, ενημερώνοντάς το για την τρέχουσα αντιστοίχηση των διευθύνσεων Ethernet και IP.

Αν το πακέτο που εισέρχεται στο σύστημα περιέχει ARP response, τότε ελέγχεται αν η απάντηση αυτή αναφέρεται στο τοπικό σύστημα και ενημερώνεται ο πίνακας ARP. Το υποσύστημα αυτό δεν άλλαξε.

3.5.7 Το υποσύστημα 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 προς μια διεύθυνση IP και να ενημερώσει το υποσύστημα, το οποίο έκανε αίτηση για ARP, όταν έρθει η απάντηση. Το συγκεκριμένο υποσύστημα παρέμεινε ως έχει.

3.5.8 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 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. Αυτό συμβαίνει 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. 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 IP-RECV δεν υπέστη αλλαγή.

3.5.9 Ο πίνακας ARP

Ο πίνακας ARP είναι μια μονόπορτη[3], 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 τέταρτη έκδοση της σχεδίασης, ακριβώς όπως και στις προηγούμενες. Έχει, δηλαδή, ακριβώς την ίδια λειτουργικότητα και τις ίδιες διασυνδέσεις με τα υπόλοιπα υποσυστήματα τα οποία τον χρησιμοποιούν. Ρόλος του πίνακα ARP είναι η αντιστοίχηση των διευθύνσεων IP με τις αντίστοιχες Ethernet.

3.5.10 Το υποσύστημα αποστολής πακέτων IP: IP-SEND

Το υποσύστημα αυτό αναλαμβάνει να προσθέσει μια επικεφαλίδα IP σε δεδομένα που έχουν, ήδη, γραφτεί από άλλο πρωτόκολλο στη μνήμη TxRAM. Το υποσύστημα του πρωτοκόλλου IP δεν είναι προσπελάσιμο απευθείας από κάποια

[3] Ο όρος «μονόπορτη» αποτελεί ελεύθερη μετάφραση του αντίστοιχου αγγλικού «single port».

εφαρμογή, αλλά ο χειρισμός του εναπόκειται 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 εσωκλείει ένα πακέτο IP μπορεί να είναι είτε ένα πακέτο UDP, είτε ένα τμήμα TCP, είτε ένα πακέτο ICMP echo reply. Το σύστημα IP-SEND δεν άλλαξε.

3.5.11 Το υποσύστημα ICMP-RECV-REPLY

Η σχεδίαση, από την δεύτερη κιόλας έκδοση, 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.

3.5.12 Το υποσύστημα CONTROL

Όπως είδαμε στο δεύτερο κεφάλαιο, το υποσύστημα CONTROL σχεδιάστηκε στην δεύτερη έκδοση του συστήματος, προκειμένου να καθορίζει πλήρως τη ροή 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 πολύπλεξης 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 δεν υπέστη καμία ουσιαστική τροποποίηση στην έκδοση αυτή, δεδομένου ότι είχε ήδη υποστεί αλλαγές στην προηγούμενη για την υποστήριξη του Congestion Control.

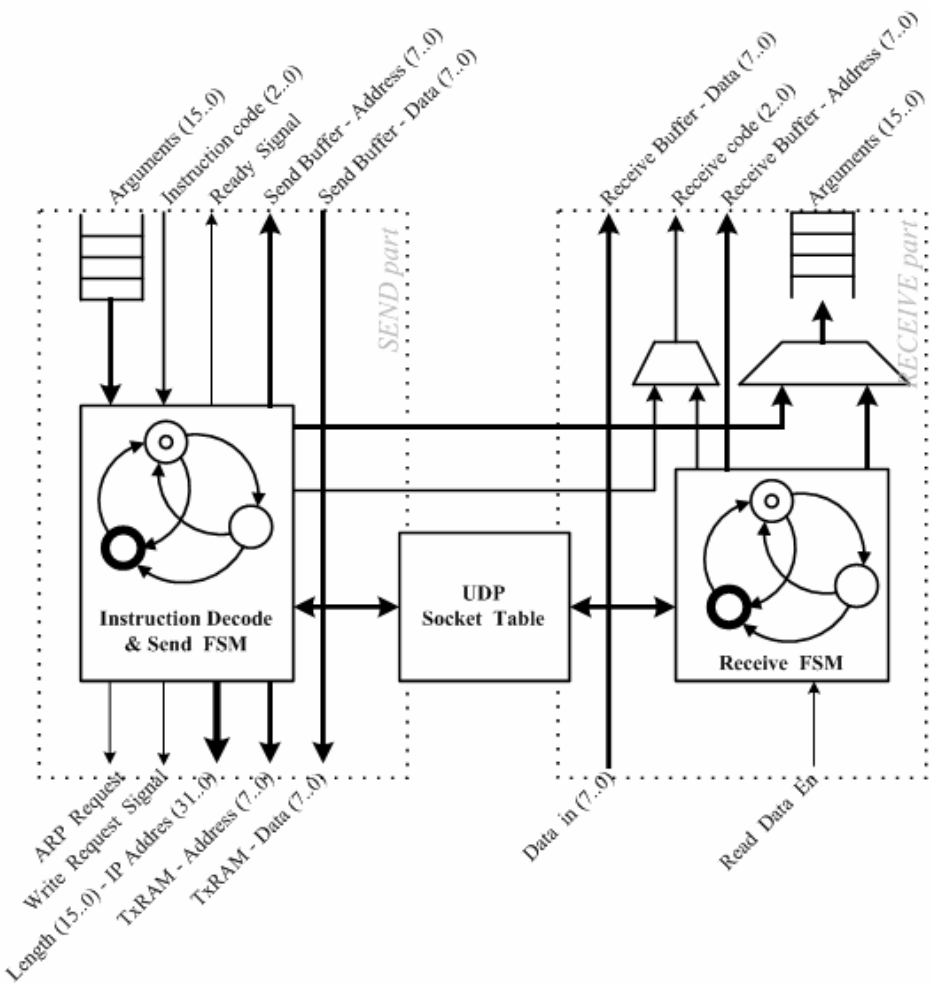
3.5.13 Το υποσύστημα λήψης και αποστολής UDP

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.

Στην εικόνα 10 φαίνεται 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 το υποσύστημα TCP.

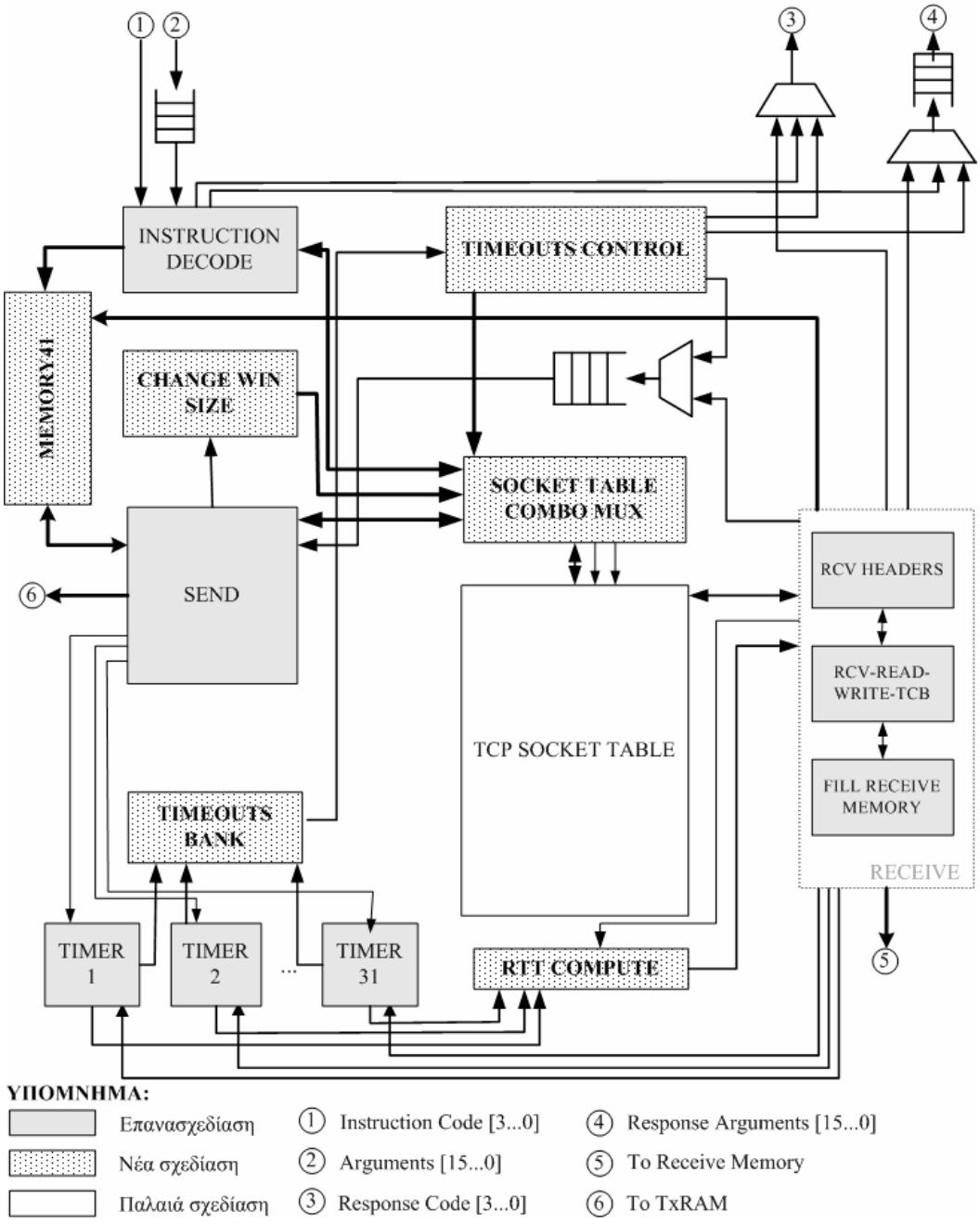


Εικόνα 11: Η δομή του υποσυστήματος UDP.

3.5.14 Το υποσύστημα TCP

Το υποσύστημα TCP ήταν ο βασικός στόχος αυτής της διπλωματικής εργασίας. Όπως αναφέρθηκε στα προηγούμενα κεφάλαια, λόγω της εξαιρετικά μεγάλης διάστασης που είχε λάβει και συνάμα, της μεγάλης πολυπλοκότητας που είχε αποκτήσει, επιβάλλονταν διάσπαση και επανασχεδίαση, έτσι ώστε η δομή να γίνει πιο σωστή.

Στην έκδοση αυτή της σχεδίασης, το υποσύστημα έχει αλλάξει εντελώς από άποψης δομής. Έγιναν πολλές διασπάσεις σε επιμέρους συστήματα, ενώ, παράλληλα εισήχθησαν και νέα υποσυστήματα που ολοκληρώνουν την ορθή λειτουργία του. Από την παλαιά έκδοση, το μόνο χαρακτηριστικό που διατηρήθηκε ήταν, ως ένα βαθμό, κάποια states στα Finite State Machines και η χρήση των επιμέρους βοηθητικών στοιχείων όπως οι FIFOs των ορισμάτων, το Socket Table κλπ.



Εικόνα 12: Η δομή του υποσυστήματος TCP

Όσον αφορά στο Socket Table, πρέπει να σημειωθεί ότι στην έκδοση αυτή, διατηρήθηκε η ίδια δομή με την μόνη διαφορά, την προσθήκη κάποιων επιπλέον πεδίων στις θέσεις των active sockets. Κατά τ'άλλα, η δίπορτη, στατική μνήμη που το υλοποιεί παρέμεινε η ίδια (καθώς η προηγούμενη σχεδίαση είχε μεριμνήσει για μελλοντικές επεκτάσεις αφήνοντας χώρο για την προσθήκη ενός μικρού αριθμού νέων πεδίων).

Στην εικόνα 11 παρουσιάζεται η βασική δομή του υποσυστήματος. Στα επόμενα κεφάλαια το υποσύστημα TCP θα μελετηθεί ενδελεχώς με τη παρουσίαση κάθε υποσυστήματος ξεχωριστά.

Κεφάλαιο 4: Η σχεδίαση του Open TCP/IP

Στο προηγούμενο κεφάλαιο ασχοληθήκαμε με την γενική αρχιτεκτονική της τέταρτης έκδοσης της σχεδίασης και παράλληλα, αναφέραμε κάποια από τα βασικά στοιχεία λειτουργίας κάθε υποσυστήματος. Στο κεφάλαιο αυτό θα εισχωρήσουμε βαθύτερα στην σχεδίαση, προκειμένου να αναλύσουμε με κάθε λεπτομέρεια κάποια από τα υποσυστήματα της σχεδίασης.

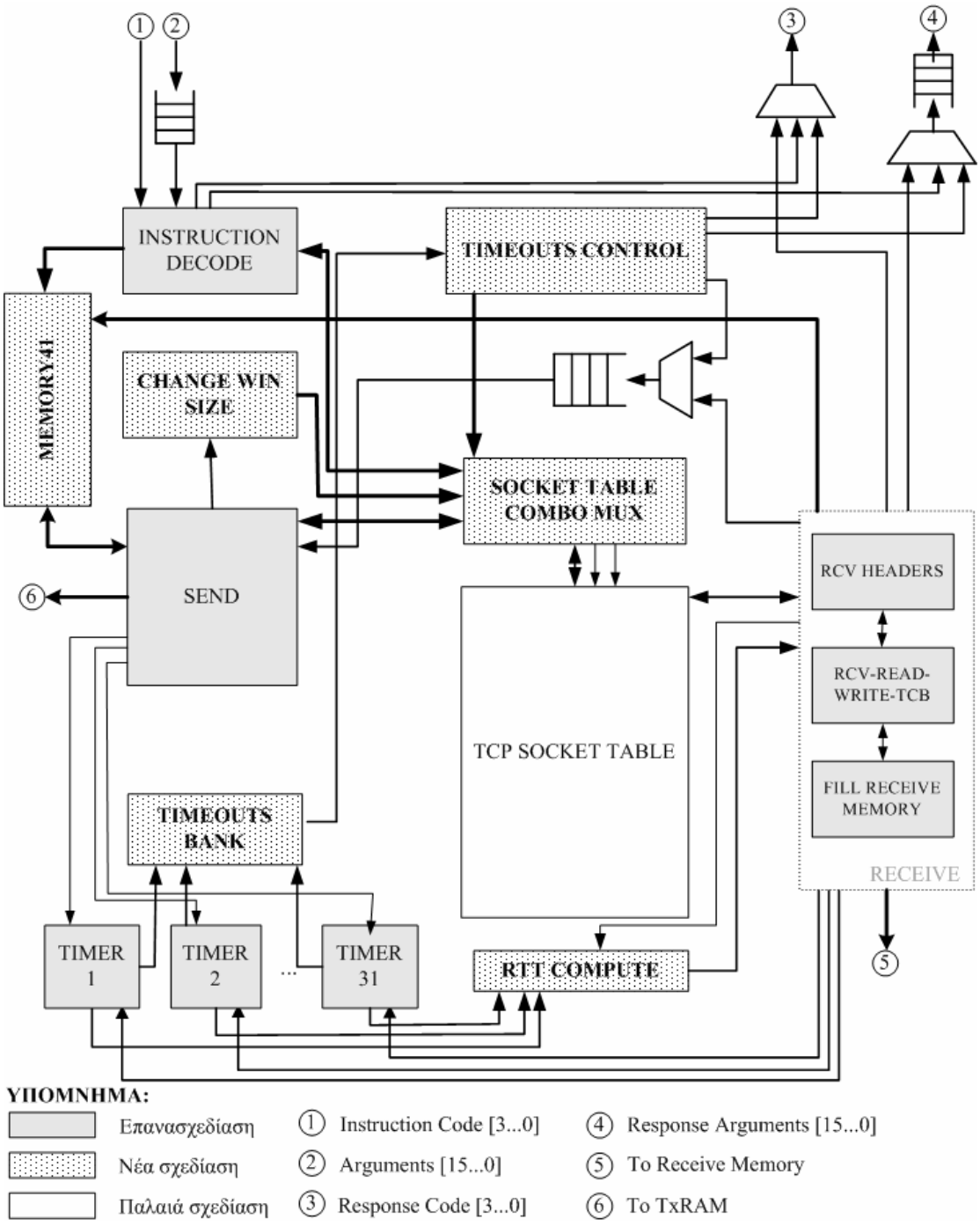
Όπως έχει γίνει ήδη αντιληπτό, η τέταρτη έκδοση της σχεδίασης, επομένως και η διπλωματική εργασία αυτή, επικεντρώθηκε στο υποσύστημα αποστολής/λήψης πλαισίων TCP. Τα υπόλοιπα υποσυστήματα διατηρήθηκαν ως είχαν. Για το λόγο αυτό και επειδή, όσον αφορά στα συστήματα αυτά, δεν υπάρχει κάτι νέο, στο κεφάλαιο αυτό θα εστιάσουμε την προσοχή μας μόνο στο υποσύστημα TCP. Τα υπόλοιπα υποσυστήματα αναφέρονται με κάθε λεπτομέρεια στα αντίστοιχα κεφάλαια των διπλωματικών εργασιών των κ. Κάχρη, Ζήση και Κοϊδή.

4.1 Τα επιμέρους υποσυστήματα του TCP

Πριν ξεκινήσουμε την ανάλυση του υποσυστήματος αποστολής/λήψης πλαισίων TCP καλό θα ήταν να δούμε ονομαστικά τα επιμέρους υποσυστήματα από τα οποία απαρτίζεται. Για μεγαλύτερη ευκολία επαναλαμβάνουμε την εικόνα της δομής του υποσυστήματος (εικόνα 12).

Όπως φαίνεται στην εικόνα 12, το υποσύστημα TCP απαρτίζεται από τα παρακάτω επιμέρους υποσυστήματα:

1. Instruction Decode: Αναλαμβάνει την αποκωδικοποίηση των εντολών που έρχονται από το επίπεδο εφαρμογών.
2. Memory41: Αναλαμβάνει την αποθήκευση των αποκωδικοποιημένων εντολών που έρχονται από το Instruction Decode μέχρι να εκτελεστούν.



Εικόνα 13: Η δομή του υποσυστήματος TCP

3. Socket Table: Αποθηκεύει τα στοιχεία κάθε σύνδεσης.
4. Send: Αναλαμβάνει την αποστολή πλαισίων TCP.
5. Receive: Αναλαμβάνει τον έλεγχο και την λήψη των εισερχόμενων πλαισίων TCP.
6. Timer 1, Timer 2, … , Timer 31: Αναλαμβάνουν την χρονομέτρηση για λογαριασμό του Congestion Control.

7. Timeouts Bank: Αποθηκεύει τα βασικά στοιχεία της σύνδεσης στην οποία συνέβη το timeout.
8. Timeouts Control: Αναλαμβάνει δράση σε περίπτωση timeout.
9. Socket Table Combo Mux: Ο ρόλος του είναι ανάλογος με εκείνον ενός πολυπλέκτη. Δίνει πρόσβαση στο socket table σε όποιο επιμέρους υποσύστημα το χρειαστεί.
10. RTT Compute: Υπολογίζει το Round Trip Time για λογαριασμό του Receive.
11. Change Win Size: Αλλάζει το CongWin στο socket table επαναφέροντάς το στην τιμή των δύο πακέτων.
12. Intermediate FIFO: Αποθηκεύει τις ενδιάμεσες εντολές που έρχονται από τα επιμέρους υποσυστήματα του TCP.

4.2 Το διάγραμμα καταστάσεων σύμφωνα με το RFC 793

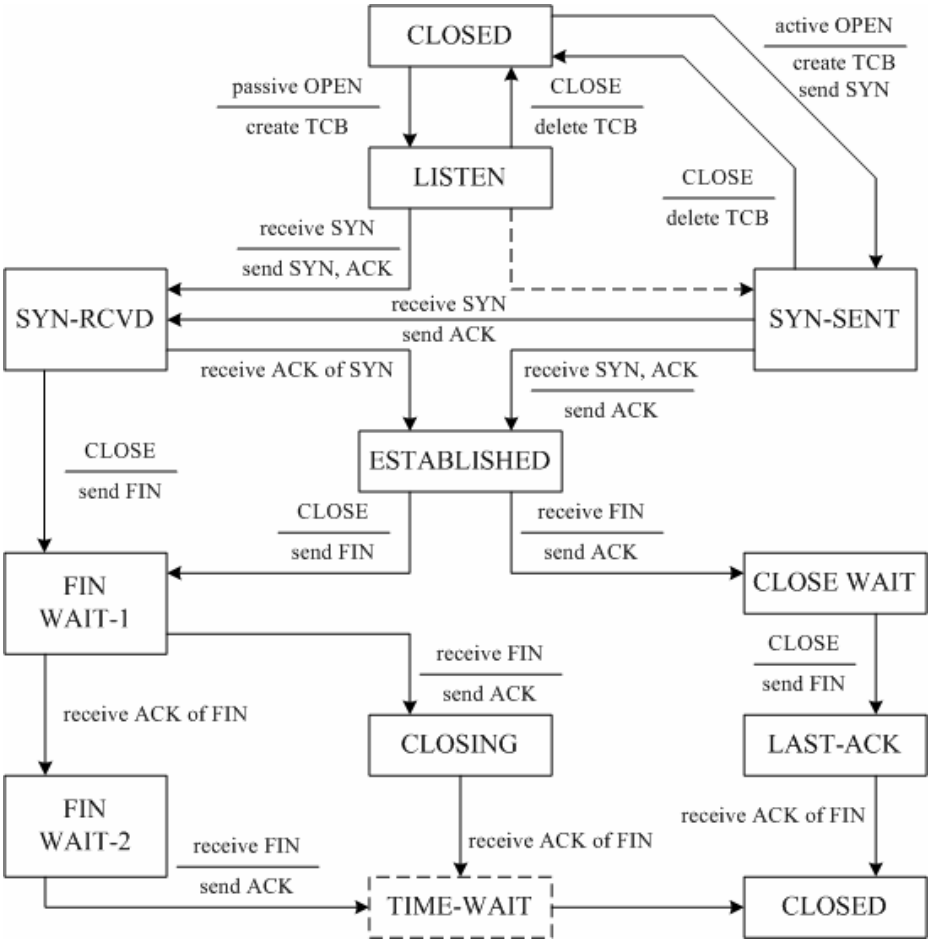
Τα περισσότερα από τα παραπάνω επιμέρους υποσυστήματα είναι σε θέση να αναγνωρίζουν σε ποια κατάσταση βρίσκεται το σύστημα κάθε φορά. Το υποσύστημα αποστολής πλαισίων TCP μπορεί να περάσει από συγκεκριμένες καταστάσεις, οι οποίες ορίζονται στο RFC 793 και παρουσιάζονται στην εικόνα 13.

Το συγκεκριμένο διάγραμμα δεν έχει υλοποιηθεί ως μία αυτόνομη μηχανή πεπερασμένων καταστάσεων (FSM). Κάθε κατάσταση περιγράφεται από μία συγκεκριμένη δυαδική τιμή σε κωδικοποίηση one hot. Η τιμή κάθε κατάστασης, για κάθε σύνδεση, φυλάσσεται στο TCB. Ο τρόπος αυτός επιτρέπει στην σχεδίαση να μπορεί να υποστηρίξει περισσότερες από μία συνδέσεις.

Έχοντας υπόψιν τα όσα αναφέρθηκαν μέχρι τώρα, μπορούμε να φθάσουμε στο συμπέρασμα, ότι η δράση του συστήματος αποστολής/λήψης πλαισίων είναι, πλέον, συνάρτηση τόσο των εντολών που δέχεται από το επίπεδο εφαρμογών και των εισερχόμενων πακέτων, όσο και της κατάστασης στην οποία βρίσκεται κάθε σύνδεση.

Όπως και στις προηγούμενες εκδόσεις, έτσι και σε αυτήν δεν θεωρείται χρήσιμη η μετατροπή μίας σύνδεσης από τύπου server σε τύπο client. Για τον λόγο αυτό, δεν υποστηρίζεται η μετάβαση από την κατάσταση LISTEN στην κατάσταση

SYN – SENT. Κατά το κλείσιμο μίας σύνδεσης, υπάρχει η κατάσταση TIME-WAIT, η οποία προκαλεί σκόπιμα καθυστέρηση. Ο λόγος ύπαρξης της καθυστέρησης αυτής, είναι η απορρόφηση από το δίκτυο των τυχόν πλαισίων που έχουν παραμείνει και αφορούν στην, προς κλείσιμο, σύνδεση. Η κατάσταση TIME-WAIT συνεχίζει να μην υποστηρίζεται και στην έκδοση αυτή. Αντ'αυτού τίθεται ο περιορισμός να μην ανοίγονται από τις εφαρμογές συνδέσεις, οι οποίες έκλεισαν πριν πολύ μικρό χρονικό διάστημα.



Εικόνα 14: Το διάγραμμα καταστάσεων μία σύνδεσης.

ΚΑΤΑΣΤΑΣΗ	**ΚΩΔΙΚΟΣ**
CLOSED	0000000000000001
LISTEN	0000000000000010
SYN-SENT	0000000000000100
SYN-RECVD	0000000000001000
ESTABLISHED	0000000000010000
FIN WAIT-1	0000000000100000
FIN WAIT-2	0000000001000000
CLOSING	0000000010000000
CLOSE WAIT	0000001000000000
LAST-ACK	0000010000000000

Πίνακας 6: Οι καταστάσεις μία σύνδεσης και οι αντίστοιχοι δυαδικοί κωδικοί.

Στον πίνακα 6 παρουσιάζονται οι υποστηριζόμενες καταστάσεις συναρτήσει των αντίστοιχων δυαδικών τιμών αναπαράστασης τους στην σχεδίαση. Όπως προαναφέρθηκε, η κωδικοποίηση που ακολουθείται είναι τύπου one hot.

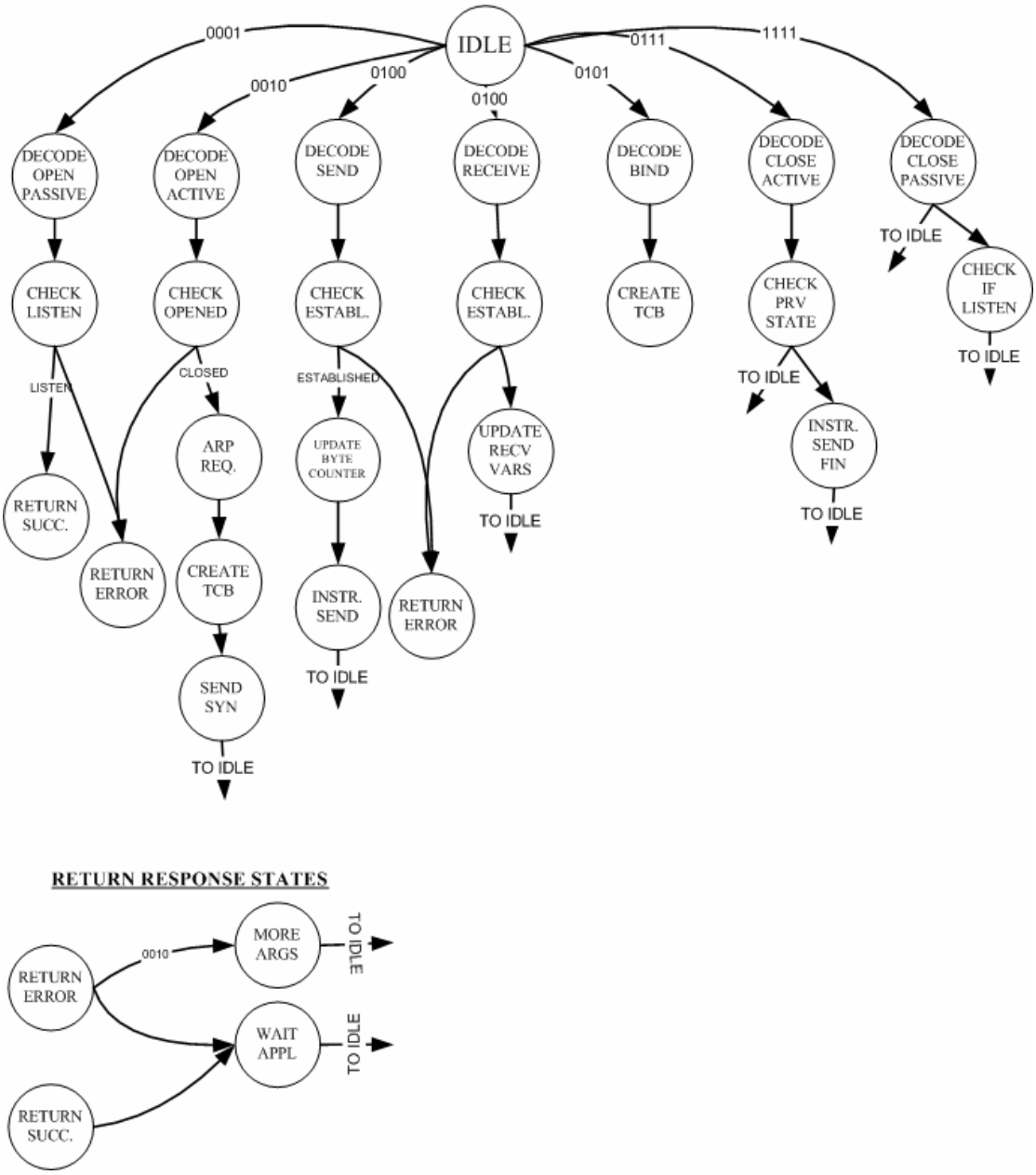
4.3 Το "Instruction Decode"

Το υποσύστημα Instruction Decode αποτελεί, κατά κάποιο τρόπο, τον «διαμεσολαβητή» μεταξύ των εντολών, που έρχονται από το επίπεδο εφαρμογών και των υποσυστημάτων Send και Receive, που αναλαμβάνουν την εκτέλεση τους. Πιο αναλυτικά, σκοπός του υποσυστήματος Instruction Decode είναι η αποκωδικοποίηση των εντολών από το επίπεδο εφαρμογών, η ενημέρωση του socket table και η επιστροφή μηνυμάτων σφάλματος η επιτυχίας. Η λειτουργία του Instruction Decode βασίζεται σε ένα FSM, το οποίο παρουσιάζεται στην εικόνα 14.

Αν και η δομή του FSM σε σχέση με την προηγούμενη σχεδίαση φαίνεται να έχει αλλάξει ελάχιστα, το υποσύστημα στην παρούσα σχεδίαση έχει αποκτήσει πολύ περισσότερη αυτονομία. Σε συνεργασία με το επιμέρους υποσύστημα "Memory 41", που θα δούμε παρακάτω, είναι σε θέση να αποκωδικοποιεί άμεσα κάθε εντολή που καταφθάνει, χωρίς να περιμένει την πλήρη εκτέλεση της προηγούμενης. Η παρούσα αυτονομία για να επιφέρει καλύτερη απόδοση συνδυάζεται και με την νέα δυνατότητα του επιμέρους υποσυστήματος αποστολής πλαισίων TCP ("Send") να εξυπηρετεί σχεδόν ταυτόχρονα περισσότερες από μία ροές. Έτσι, όταν, για παράδειγμα, μία ροή περιμένει επιβεβαίωση προκειμένου να συνεχίσει η αποστολή της, τότε το "Send", μέχρι την λήψη της επιβεβαίωσης αυτής, εξυπηρετεί άλλη εντολή η οποία είχε αποκωδικοποιηθεί νωρίτερα από το υποσύστημα Instruction Decode.

Ο μόνος περιορισμός στην αυτονομία λειτουργίας του επιμέρους υποσυστήματος "Instruction Decode" εισέρχεται από τον παράγοντα της διαθέσιμης μνήμης του υποσυστήματος "Memory 41" που φυλάσσει τις αποκωδικοποιημένες εντολές. Βέβαια, ο παραπάνω περιορισμός δεν αποτελεί κάτι ανησυχητικό και αυτό γιατί, ο κύριος στόχος αυτής της παράλληλης λειτουργίας δεν είναι η άμεση αποκωδικοποίηση κάθε εισερχόμενης εντολής, αλλά η ύπαρξη τουλάχιστον μίας

διαθέσιμης αποκωδικοποιημένης εντολής, σε περιπτώσεις που η εκτέλεση της προηγούμενης παρουσιάσει κάποια κολλήματα, όπως για παράδειγμα η αναμονή για την λήψη μίας επιβεβαίωσης.



Εικόνα 15: Διάγραμμα καταστάσεων του Instruction Decode

Όπως φαίνεται και στο διάγραμμα καταστάσεων, η σχεδίαση εξέρχεται από την κατάσταση idle και μεταβαίνει σε μία άλλη ανάλογα με τον κωδικό της εισερχόμενης εντολής. Κάθε κατάσταση στην οποία γίνεται μετάβαση, παρέχει

κάποιους επιπλέον έλεγχους, πριν αλλάξει κάτι στο socket table ή επιστραφεί κάποιο μήνυμα.

Έτσι:

- Στην περίπτωση της εντολής "Open Passive Socket" γίνεται έλεγχος της κατάστασης της σύνδεσης. Αν αυτή είναι σε κατάσταση LISTEN τότε επιστρέφεται μήνυμα επιτυχίας, διαφορετικά σφάλματος.
- Στην περίπτωση της εντολής "Open Active Socket" ελέγχεται εάν η σύνδεση είναι "CLOSED". Αν όντως είναι "CLOSED", τότε ζητείται η διεύθυνση Ethernet του απομακρυσμένου σταθμού με ένα ARP Request, δημιουργείται η αντίστοιχη εγγραφή στο Socket Table και τέλος, στέλνεται εντολή για αποστολή SYN από το υποσύστημα αποστολής πλαισίων TCP. Αν η σύνδεση δεν είναι σε κατάσταση "CLOSED" τότε επιστρέφεται μήνυμα σφάλματος.
- Στην περίπτωση της εντολής "Send Segment" ελέγχεται αν η σύνδεση είναι "ESTABLISHED" ή "LAST ACK". Αν είναι σε μία από αυτές τις καταστάσεις, τότε αφού γίνει ανανέωση μίας μεταβλητής, που δηλώνει πόσα bytes πρέπει να αποσταλούν, (byte_counter), στέλνεται εντολή για την εκκίνηση της διαδικασίας αποστολής από το υποσύστημα αποστολής πλαισίων.
- Στην περίπτωση της εντολής "Bind" δεν υπάρχουν επιπλέον έλεγχοι. Δημιουργείται η εγγραφή της σύνδεσης στο socket table χρησιμοποιώντας για αρχικοποίηση τα ορίσματα που έρχονται μαζί με την εντολή και παράλληλα κάποιες σταθερές (fixed) τιμές που διαθέτει η σχεδίαση.
- Στην περίπτωση της εντολής "Receive Segment" ελέγχεται αν η σύνδεση είναι "ESTABLISHED" ή "FIN WAIT 2". Σε περίπτωση που βρίσκεται στις δύο παραπάνω καταστάσεις τότε η σχεδίαση προχωρά στην ανανέωση κάποιων τιμών της σύνδεσης στο socket table, μεταξύ αυτών και του Receive Validy Bit που είναι υπεύθυνο για την δυνατότητα λήψης της σύνδεσης. Σε αντίθετη περίπτωση η σχεδίαση επιστρέφει στην κατάσταση idle.

- Στην περίπτωση της εντολής "Close Active" ελέγχεται και πάλι, η κατάσταση της σύνδεσης και ανάλογα με αυτήν, η σύνδεση μεταβαίνει στην κατάσταση που περιγράφεται από το διάγραμμα καταστάσεων. Παράλληλα, στέλνεται εντολή στο σύστημα αποστολής πλαισίων για αποστολή FIN πλαισίου.
- Τέλος, στην περίπτωση της εντολής "Close Passive" γίνεται έλεγχος προκειμένου να διαπιστωθεί αν η σύνδεση είναι σε κατάσταση "LISTEN". Αν είναι στην κατάσταση αυτή, μεταβαίνει στην κατάσταση "CLOSED" και η FSM στην "Idle".

4.4 Το "Memory 41"

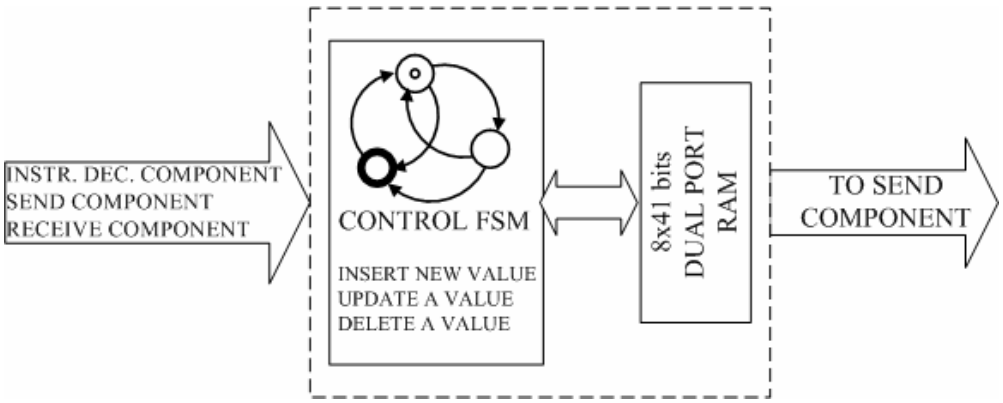
Όπως αναφέρθηκε και στην προηγούμενη παράγραφο, το επιμέρους υποσύστημα "Memory 41" αποτελεί στην ουσία ένα είδος buffer για την προσωρινή φύλαξη των αποκωδικοποιημένων εντολών. Σε αυτό το σημείο θα πρέπει να τονιστεί ότι η ύπαρξη του υποσυστήματος αυτού δεν είναι πάντα απαραίτητη για την αυτόνομη λειτουργία του Instruction Decode.

Το Memory 41 φυλάσσει τις παρακάτω πληροφορίες:

1. Το Buffer Offset
2. Το Buffer Size
3. Το SocketID και
4. κάποιες επιπλέον που έχουν να κάνουν με το είδος της εντολής που προκάλεσε την εκάστοτε εγγραφή.

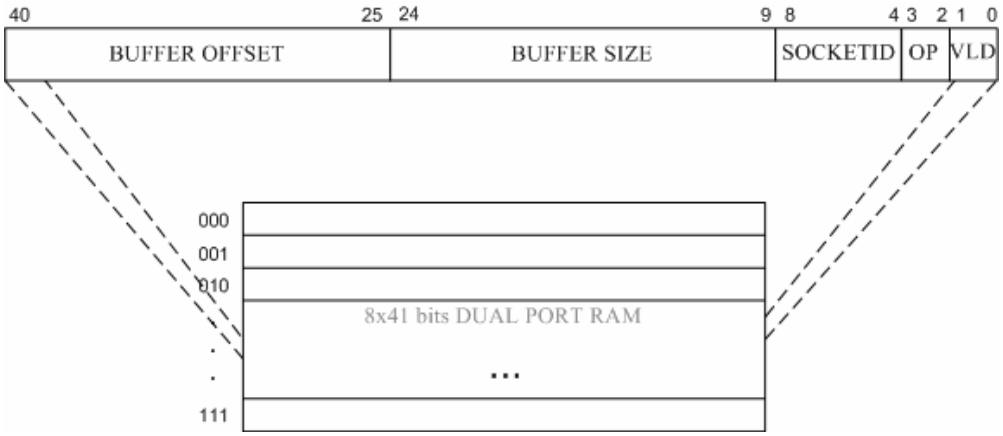
Μελετώντας πιο προσεκτικά τις πληροφορίες που φυλάσσονται στο παρόν υποσύστημα, συμπεραίνουμε ότι η ύπαρξη του είναι απαραίτητη μόνο στις εντολές "Send", "Open Active" και "Close Active". Οι υπόλοιπες αποκωδικοποιούνται, ενημερώνοντας το socket table, όπου χρειάζεται, χωρίς να απασχολούν καθόλου το υποσύστημα αυτό.

Το Memory 41 αποτελείται από δύο βασικά τμήματα: μία σύγχρονη δίπορτη μνήμη και ένα FSM που φροντίζει για την τροποποίηση της μνήμης (εγγραφή, ενημέρωση, διαγραφή). Στην εικόνα 15 φαίνεται παραστατικά η δομή αυτή.



Εικόνα 16: Η δομή του υποσυστήματος "Memory 41".

Θα πρέπει να τονισθεί ότι το υποσύστημα αυτό δέχεται εισόδους από τα υποσυστήματα Instruction Decode, Send και Receive ενώ η έξοδός του οδηγείται στο υποσύστημα Send. Κανένα από τα υποσυστήματα που ζητάνε πρόσβαση στο Memory 41 δεν είναι σε θέση να γνωρίζει διεύθυνση για την μνήμη που ενθυλακώνεται στο ίδιο υποσύστημα. Την διαδικασία του χειρισμού των διευθύνσεων και της τροποποίησης της μνήμης την αναλαμβάνει το FSM.



Εικόνα 17: Η μνήμη του υποσυστήματος Memory 41

Πιο αναλυτικά, όπως φαίνεται από την εικόνα 16, κάθε εγγραφή στην μνήμη αποτελεί και μία αποκωδικοποιημένη εντολή που αφορά μία σύνδεση. Σε κάθε εγγραφή αποθηκεύεται επίσης το SocketID, πληροφορία «κλειδί» για την

αναγνώριση της σύνδεσης στην οποία αναφέρεται η αποκωδικοποιημένη εντολή και άλλα τέσσερα bits ελέγχου τα οποία περιγράφονται στον πίνακα 7.

	BITS	ΣΗΜΑΣΙΑ
VLD	00	Άκυρη εγγραφή (ελεύθερη θέση)
	01	Έγκυρη εγγραφή (αναμονή ack)
	10	Έγκυρη εγγραφή – έτοιμη για εκτέλεση
OP	00	Αποστολή SYN
	01	Αποστολή FIN
	10	Αποστολή DATA

Πίνακας 7: Τα bits ελέγχου εγγραφής του Memory 41.

Όταν το υποσύστημα Instruction Decode αποκωδικοποιεί κάποια εντολή που απαιτεί εγγραφή στο Memory 41, τότε αυτό απευθύνεται στο FSM, το οποίο με την σειρά του είναι υπεύθυνο για την εύρεση κενής θέσης (θέσης με VLD bits = “00”). Αντίστροφα, όταν το υποσύστημα αποστολής πλαισίων ζητά από το FSM νέα εντολή, τότε αυτό διατρέχει την μνήμη μέχρι να βρει έγκυρη εγγραφή, την οποία διαβάζει προκειμένου να ενημερώσει το αρχικό υποσύστημα για το Socket ID, το buffer size, το buffer offset, αν πρόκειται για μετάδοση data ή για την αποστολή SYN ή FIN πακέτου.

Τέλος, σε περιπτώσεις αναμονής για λήψη επιβεβαίωσης (περιπτώσεις δηλαδή που πρέπει να αλλάξουν τα validy bits σε “01”), σε περιπτώσεις αποστολής SYN, FIN (που απαιτούν διαγραφή μετά την αποστολή) κλπ, οι εγγραφές πρέπει να τροποποιούνται. Την τροποποίηση και πάλι αναλαμβάνει το FSM, το οποίο με «κλειδί» το socket id που του δίνεται εξωτερικά, ψάχνει να βρει την προς τροποποίηση εγγραφή. Σημειώνεται ότι το μόνο που χρειάζεται για την διαγραφή μίας εγγραφής είναι η αλλαγή των validy bits από “01” ή “10” σε “00”.

4.5 Το “Socket Table”

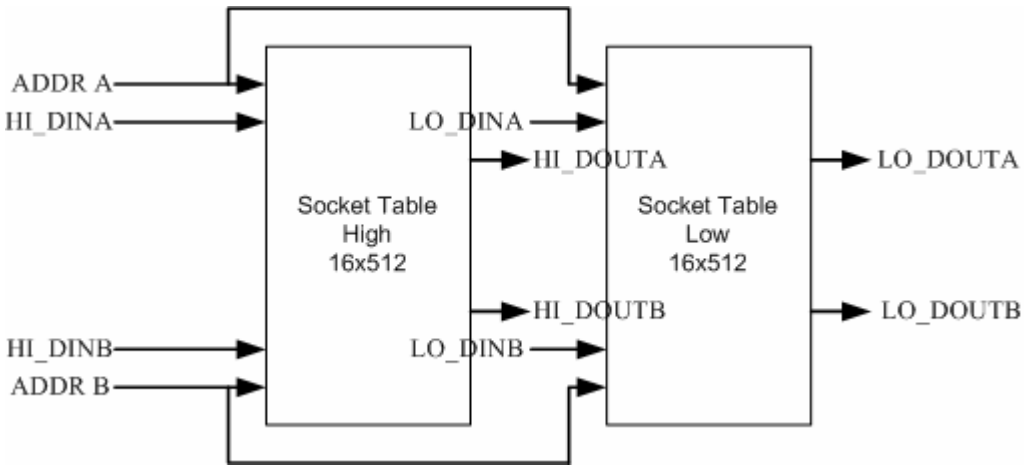
Όπως έχει ήδη ειπωθεί, το Socket Table είναι μία μνήμη, στην οποία φυλάσσονται τα στοιχεία κάθε σύνδεσης. Οι πρώτες δεκαέξι θέσεις προορίζονται μόνο για passive sockets, ενώ οι υπόλοιπες μόνο για active sockets. Κάθε passive

socket απαιτεί μία θέση μνήμης, σε αντίθεση με το active που απαιτεί δεκαέξι θέσεις. Η διευθυνσιοδότηση των active και των passive sockets γίνεται με την χρήση του περιγραφέα SocketID. Το SocketID των passive και των active sockets σχηματίζεται διαφορετικά.

Μέχρι το σημείο αυτό, η περιγραφή ήταν γνωστή και από τα προηγούμενα κεφάλαια. Αναλύοντας περισσότερο την δομή του Socket Table (εικόνα 13), πρέπει να αναφέρουμε, ότι υλοποιείται από δύο δίπορτες σύγχρονες μνήμες 512x16 bits. Τα Address Busses των εν λόγω μνημών είναι κοινά. Έτσι, το Address Bus του Port A της μίας μνήμης είναι κοινό με το Address Bus του Port A της άλλης μνήμης. Ανάλογη είναι η συνδεσμολογία των Address Busses και για το δεύτερο port.

Η διάταξη αυτή, δημιουργεί περιορισμό, όσον αφορά στον τρόπο γραψίματος και διαβάσματος κάθε μνήμης. Έτσι, δεν μπορούμε να διαβάσουμε ή να γράψουμε χρησιμοποιώντας το ίδιο port (ή το A, ή το B) σε άλλη διεύθυνση στην μία μνήμη και σε άλλη στην άλλη. Η παραπάνω ιδιαιτερότητα, δεν δημιουργεί πρόβλημα, αντιθέτως μειώνει την πολυπλοκότητα της σχεδίασης. Αυτό συμβαίνει, διότι, συνήθως γράφουμε ή διαβάζουμε τον πίνακα ανά γραμμή.

Η οργάνωση του πίνακα των sockets, γίνεται κατά τέτοιο τρόπο ώστε ο μισός πίνακας να φυλάσσεται στην μία μνήμη (την high) και ο άλλος μισός στην άλλη (την low). Μιλώντας με bits, στην μνήμη high φυλάσσονται όλα τα bits από 31 έως 16, ενώ στην low τα 15 έως 0. Η οργάνωση της πληροφορίας ανά socket φαίνεται στην εικόνα 8 του προηγούμενου κεφαλαίου. Σχηματικά η σύνδεση των δύο μνημών που υλοποιούν το socket table φαίνεται στην εικόνα 17.



Εικόνα 18: Η δομή του Socket Table

Τέλος, σημειώνεται ότι το port B χρησιμοποιείται μόνο από το υποσύστημα Receive ενώ το port A από όλα τα υπόλοιπα.

4.6 Το "Send"

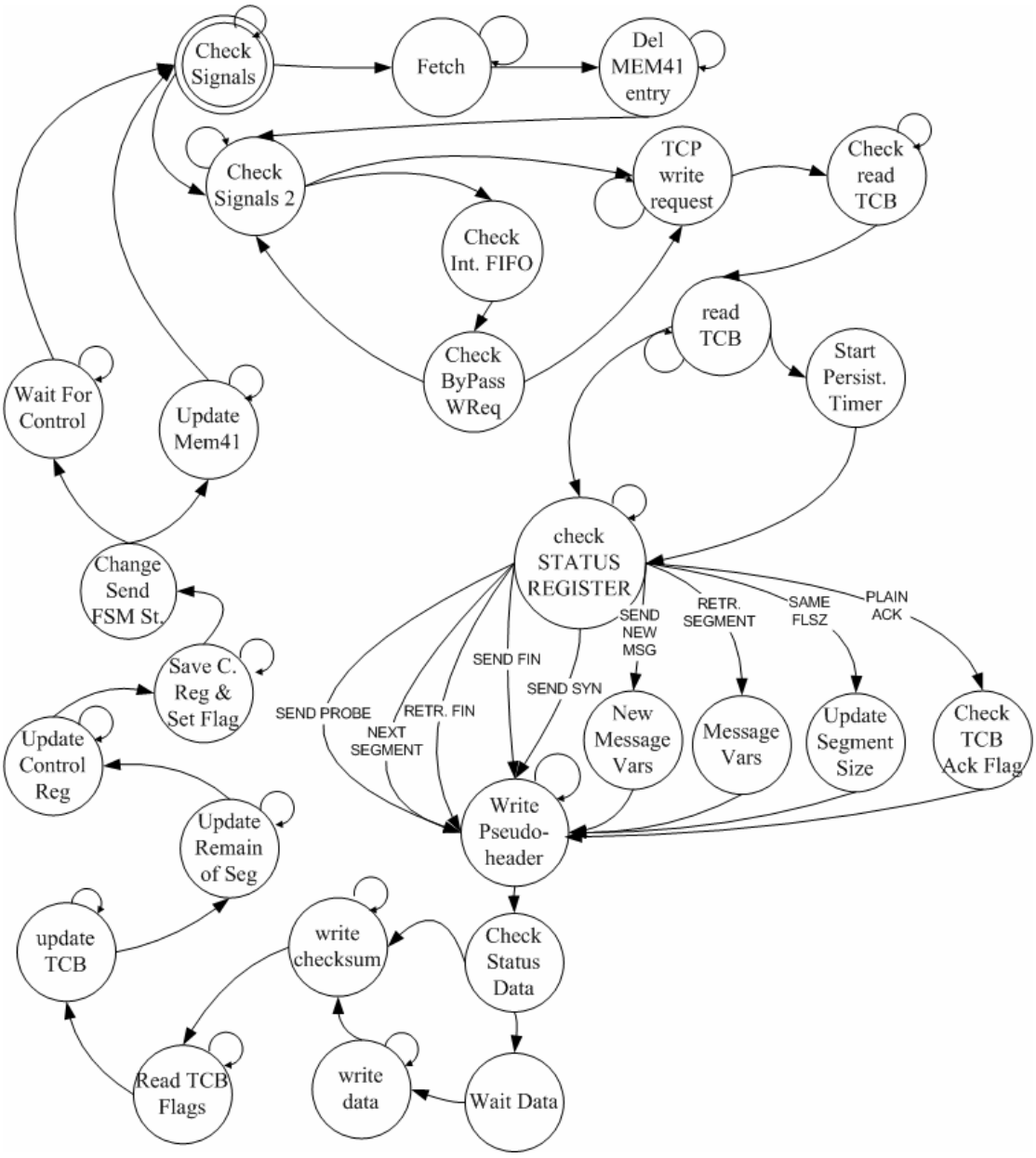
Το υποσύστημα αποστολής πλαισίων TCP αποτελείται από ένα FSM, το οποίο υλοποιεί την ετοιμασία και αποστολή πλαισίων είτε για την δημιουργία σύνδεσης (SYN), είτε για το κλείσιμο της σύνδεσης (FIN), είτε για απλή επιβεβαίωση, είτε για μεταφορά δεδομένων. Ο τύπος του πλαισίου που δημιουργείται εξαρτάται από τις εντολές που δέχεται το υποσύστημα. Οι εντολές αυτές προέρχονται από δύο υποσυστήματα:

1. Το "Memory 41", που, όπως είδαμε, παρέχει τις αποκωδικοποιημένες από το "Instruction Decode" εντολές.
2. Την "Intermediate FIFO", που παρέχει κάποιες εντολές οι οποίες δημιουργήθηκαν εσωτερικά του υποσυστήματος, κάτω από συγκεκριμένες περιπτώσεις.

Στην εικόνα 18 παρουσιάζεται το διάγραμμα καταστάσεων του υποσυστήματος. Πριν προχωρήσουμε στην ανάλυση του διαγράμματος, θα πρέπει να σημειώσουμε ότι, αν και δομικά το FSM μοιάζει με εκείνο της προηγούμενης έκδοσης, υπάρχουν βασικές διαφορές. Μία από αυτές συνιστάται στο γεγονός, ότι στην σχεδίαση, η FSM δεν "κοιτάζει" το Instruction Decode για εντολή αλλά αρχικά την "Memory 41" και έπειτα την "Intermediate FIFO", διαδικασία που πραγματοποιείται πάντα εναλλάξ. Μία άλλη διαφορά, είναι η αντίδραση του FSM σε περίπτωση τέλους του Flight size (και αναμονής για επιβεβαίωση): στην παλαιά σχεδίαση περίμενε ολόκληρο το υποσύστημα μέχρι την λήψη της επιβεβαίωσης ενώ στην νέα, το υποσύστημα προχωρά στην εξυπηρέτηση της αμέσως επόμενης εντολής, αφού πρωτύτερα δώσει οδηγίες στο "Memory 41", για να γίνει η τροποποίηση της προηγούμενης εντολής από ενεργή σε προσωρινά ανενεργή ("Waiting Ack" στα validy bits).

Η μηχανή πεπερασμένων καταστάσεων που υλοποιεί το υποσύστημα, ξεκινά από την κατάσταση "CHECK SIGNALS" και περνώντας εναλλάξ από τις καταστάσεις "FETCH" και "CHECK SIGNALS 2", ελέγχει την "Memory 41" αλλά και την "Intermediate FIFO" για εντολή.

Ας υποθέσουμε ότι η "Memory 41" διαθέτει αποθηκευμένη κάποια αποκωδικοποιημένη εντολή. Τότε, το σύστημα θα κάνει μετάβαση στην "DEL MEM 41 ENTRY", όπου και θα σβήσει την εντολή από το υποσύστημα "Memory 41", μόνο στην περίπτωση που αυτή αφορά στην μετάδοση πλαισίου SYN ή FIN. Στην κατάσταση, επίσης, "CHECK SIGNALS 2" δημιουργείται μία τιμή που αποθηκεύεται σε καταχωρητή και παίζει σημαντικό ρόλο στην μετέπειτα πορεία των μεταβάσεων. Ο καταχωρητής ονομάζεται "Send FSM Status" ή πιο απλά, "Status Register".



Εικόνα 19: Διάγραμμα καταστάσεων του Send.

Η τιμή του Send FSM Status, όπως είδαμε, έχει ήδη δημιουργηθεί από την κατάσταση "CHECK SIGNALS 2". Πριν, όμως, χρησιμοποιηθεί για έλεγχο μετάβασης, το FSM περνά από τις καταστάσεις "CHECK READ TCB" και "READ TCB", προκειμένου να ενεργοποιήσει και να διαβάσει το Socket Table. Το διάβασμα του Socket Table είναι αναπόφευκτο, καθώς εκεί βρίσκονται τα στοιχεία που είναι απαραίτητα για την χρήση οποιασδήποτε σύνδεσης από το υποσύστημα.

Πριν προχωρήσει η μετάβαση στην κατάσταση ελέγχου του Send FSM Status, "CHECK STATUS REGISTER", γίνεται ένας πιο πρώιμος έλεγχος, στον οποίον γίνεται απόπειρα να διαπιστωθεί εάν η σύνδεση έχει πέσει σε deadlock ώστε να ενεργοποιηθεί ο Persistence Timer. Μετά τον έλεγχο αυτό, το σύστημα μεταβαίνει στην προαναφερθείσα κατάσταση. Εκεί, ανάλογα με την τιμή του Send FSM Status, το σύστημα διακρίνει εάν πρόκειται για δημιουργία πλαισίου SYN, πλαισίου FIN, πλαισίου δεδομένων κλπ. Πρέπει να σημειωθεί, ότι ο Send FSM Status δεν μας δίνει μόνο πληροφορίες για τον τύπο του πλαισίου, αλλά και για τον τύπο της μετάδοσης: αν, δηλαδή, πρόκειται για επαναμετάδοση, για αποστολή νέας ροής, για μετάδοση νέου παραθύρου κλπ.

Οι περισσότερες περιπτώσεις καταλήγουν αμέσως στην κατάσταση "WRITE PSEUDO - HEADER". Οι υπόλοιπες απαιτούν την μετάβαση από ενδιάμεσες καταστάσεις πριν καταλήξουν και αυτές στην "WRITE PSEUDO - HEADER". Αυτό συμβαίνει, διότι, χρειάζεται να υπολογισθούν κάποιες παράμετροι μετάδοσης, όπως η ποσότητα της πληροφορίας που δεν έχει μεταδοθεί ακόμα (remain of segment), το μέγεθος του προς αποστολή πλαισίου κλπ.

Στην κατάσταση "CHECK STATUS DATA" διαπιστώνεται εάν το πακέτο προς μετάδοση θα φέρει ή όχι δεδομένα. Στην περίπτωση που φέρει δεδομένα, τότε η FSM μεταβαίνει στις καταστάσεις "WAIT DATA", "WRITE DATA" και "WRITE CHECKSUM", διαφορετικά κατευθείαν στην "WRITE CHECKSUM". Στην κατάσταση αυτή γράφεται το checksum στην TxRAM και η μετάβαση συνεχίζει στις "READ TCB FLAGS" και "UPDATE TCB" καταστάσεις στις οποίες διαβάζονται ή τροποποιούνται τα flags και ενημερώνεται το socket table. Επόμενα βήματα που γίνονται πριν την ολοκλήρωση της αποστολής, είναι η αποθήκευση της τιμής remain of segment στο socket table, ο υπολογισμός της τιμής ενός καταχωρητή που ονομάζεται Control Register και ο υπολογισμός ενός άλλου flag το οποίο θα χρειαστεί στις επόμενες καταστάσεις. Οι παραπάνω κινήσεις γίνονται στις

καταστάσεις “UPDATE REMAIN OF SEGMENT”, “UPDATE CONTROL REGISTER” και “SAVE CONTROL REGISTER AND SET FLAG”.

Σχετικά με τον Control Register θα πρέπει να σημειωθεί, ότι ο ρόλος του είναι η επικοινωνία του υποσυστήματος αποστολής με εκείνο της λήψης. Τα πρώτα τέσσερα bits χρησιμοποιούνται ως ID της σύνδεσης ενώ το πέμπτο (MSB) χρησιμοποιείται προκειμένου να γίνεται γνωστό αν η μετάδοση της ροής έχει τελειώσει ή απομένουν και άλλα πλαίσια μέχρι το τέλος αυτής. Με αυτόν τον τρόπο γίνεται εύκολη η αποστολή piggy back επιβεβαιώσεων, καθώς το υποσύστημα λήψης μπορεί να διαπιστώσει εάν η σύνδεση στέλνει ακόμα.

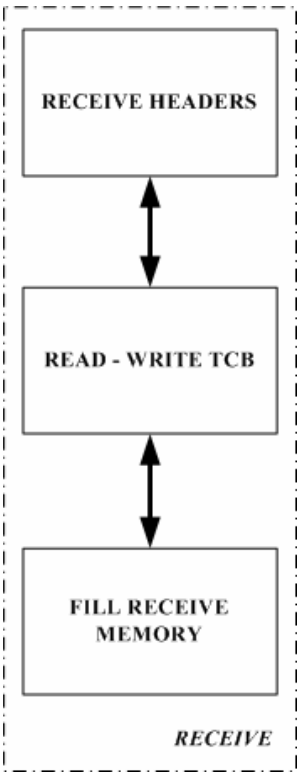
Τέλος, το σύστημα μεταβαίνει στην κατάσταση “CHANGE SEND FSM STATUS”. Εκεί ενεργοποιείται ο κατάλληλος timer και ελέγχεται εάν το flight size έχει τελειώσει ή ακολουθούν και άλλα πλαίσια ακόμα. Στην πρώτη περίπτωση το σύστημα μεταβαίνει στην “UPDATE MEM41”, προκειμένου να αλλάξει το validy της εντολής σε “wait ack”, ενώ στην δεύτερη το σύστημα μεταβαίνει στην “WAIT FOR CONTROL”, όπου γίνεται αίτηση στο control για επανεγγραφή της μνήμης αποστολής. Και στις δύο παραπάνω περιπτώσεις το σύστημα ξαναγυρίζει στην αρχική κατάσταση, την κατάσταση “CHECK SIGNALS”.

4.7 Το “Receive”

Το υποσύστημα λήψης πλαισίων TCP, όπως είδαμε στην μελέτη της γενικότερης αρχιτεκτονικής του συστήματος, απαρτίζεται από τρία επιμέρους υποσυστήματα (εικόνα 19). Σκοπός του συστήματος είναι ο έλεγχος και η λήψη των εισερχόμενων πακέτων. Συμπερασματικά, μπορούμε να πούμε ότι σκοπός του υποσυστήματος λήψης δεν είναι μόνο η απλή λήψη των πακέτων, αλλά και ένας μεγάλος αριθμός από ελέγχους και κινήσεις προκειμένου να εξελιχθεί σωστά, τόσο η διαδικασία λήψης, όσο και αποστολής η οποία βασίζεται στις επιβεβαιώσεις που λαμβάνονται από το υπό εξέταση υποσύστημα.

Στις επόμενες υποπαραγράφους θα αναλύσουμε ένα-ένα τα επιμέρους υποσυστήματα που αποτελούν το Receive, με στόχο τόσο την κατανόηση της

λειτουργίας του υποσυστήματος, όσο και την κατανόηση των λόγων που οδήγησαν στην διάσπαση του υποσυστήματος Receive.



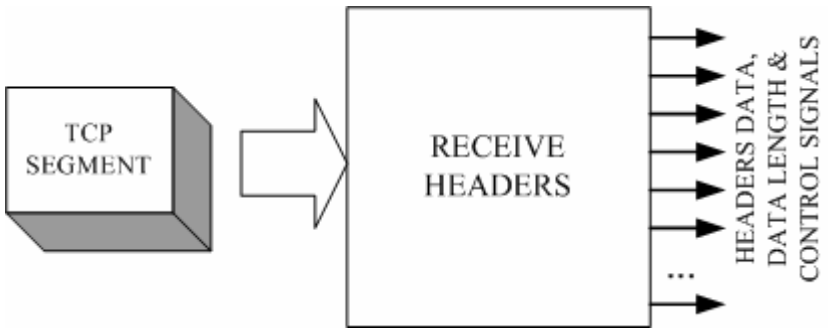
Εικόνα 20: Τα επιμέρους υποσυστήματα του "Receive".

4.7.1 Receive: Το υποσύστημα "Receive Headers"

Το υποσύστημα Receive Headers είναι υπεύθυνο για την λήψη των πεδίων της επικεφαλίδας του εισερχόμενου πακέτου και του μήκους των καθαρών δεδομένων που εσωκλείονται στο πακέτο. Πιο αναλυτικά, το Receive Headers:

1. Διαβάζει την επικεφαλίδα του εισερχόμενου πακέτου και παρέχει τις πληροφορίες της στις εξόδους του.
2. Υπολογίζει το μήκος των «καθαρών» δεδομένων, με βάση μία επιπλέον πληροφορία που παίρνει από το IP και αφορά στο μήκος του πακέτου TCP.
3. Ξεχωρίζει την περίπτωση "Maximum Segment Size" στα options του εισερχόμενου πεδίου και βγάζει την συγκεκριμένη πληροφορία στην έξοδό του.
4. Ενεργοποιεί την FIFO απόζευξης για την αποθήκευση των δεδομένων.

Στην εικόνα 20 παρουσιάζεται, με παραστατικό τρόπο, η κατανομή εισόδων – εξόδων του υποσυστήματος "Receive Headers" καθώς και η λειτουργία του.



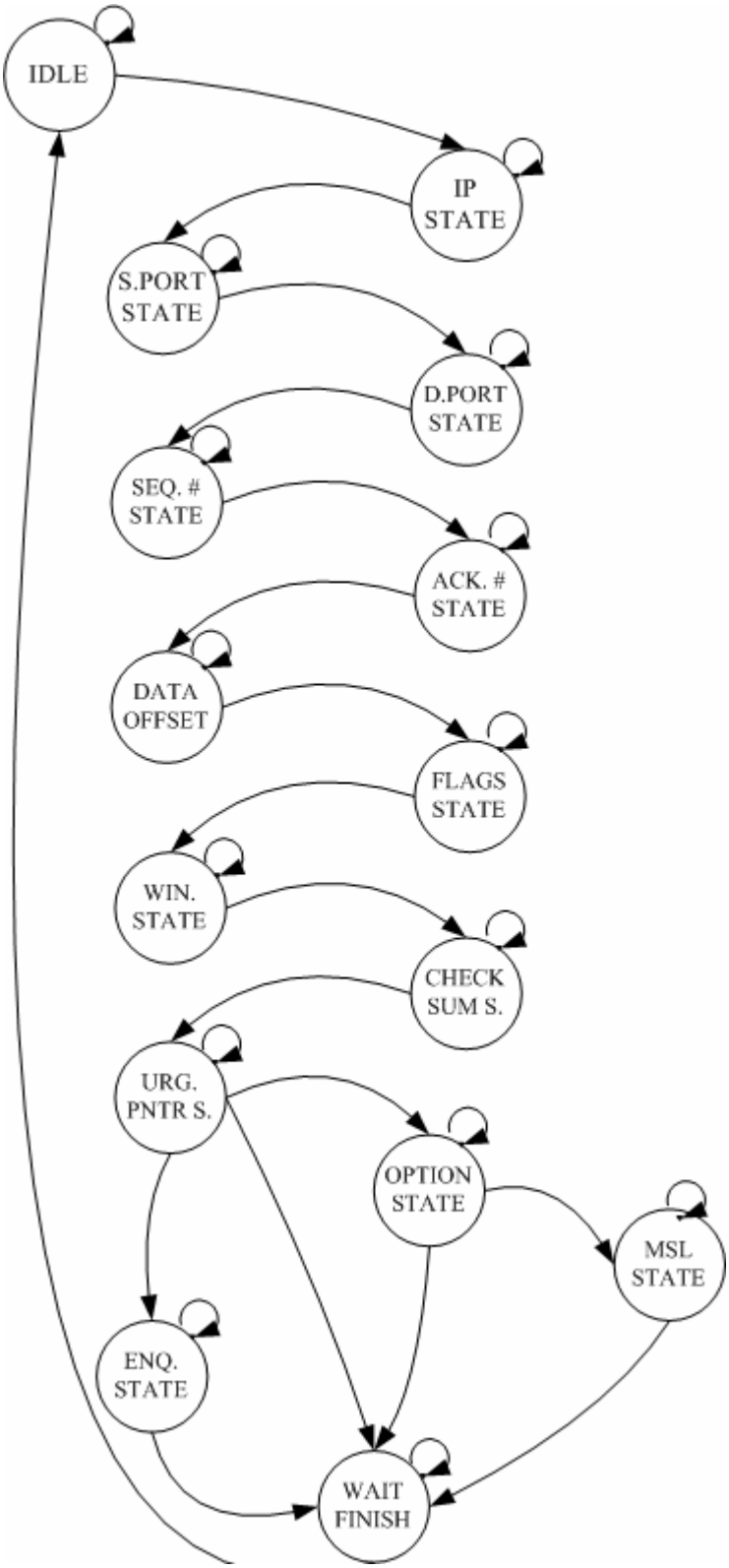
Εικόνα 21: Ο ρόλος του "Receive Headers".

Η λειτουργία του "Receive Headers" βασίζεται σε μία μηχανή πεπερασμένων καταστάσεων (FSM). Το διάγραμμα καταστάσεων της FSM αυτής παρουσιάζεται στην εικόνα 21.

Όπως, γίνεται αμέσως αντιληπτό, η λειτουργία του FSM δεν παρουσιάζει κάποια ιδιαίτερη πολυπλοκότητα καθώς οι μεταβάσεις που γίνονται από την αρχική κατάσταση ("IDLE") μέχρι και την "URGENT POINTER" γίνονται «σειριακά» χωρίς ελέγχους. Κατά την διάρκεια των μεταβάσεων αυτών διαβάζονται πεδία της επικεφαλίδας, όπως το IP address, το source port number, το destination port number κ.α.

Στην κατάσταση "URGENT POINTER" γίνεται έλεγχος του μήκους του πακέτου, προκειμένου να διαπιστωθεί αν αυτό φέρει options. Αν πράγματι φέρει, τότε μεταβαίνουμε στην "MSL OPTION STATE". Στην κατάσταση αυτήν διαβάζουμε το πεδίο options. Όπως αναφέραμε και στα χαρακτηριστικά του συστήματος, η σχεδίαση είναι σε θέση να αναγνωρίσει μόνο το option maximum segment size. Μετά την "MSL OPTION STATE", το σύστημα μεταβαίνει στην "WAIT FINISH" όπου και περιμένει μέχρι να πέσει το σήμα που εκκίνησε την όλη διαδικασία διαβάσματος της επικεφαλίδας.

Αν δεν περιέχονται options τότε το FSM μεταβαίνει στις καταστάσεις ENQUEUE STATE" ή "WAIT FINISH", αν στο εισερχόμενο πλαίσιο περιέχονται ή όχι δεδομένα, αντίστοιχα. Στην κατάσταση "ENQUEUE STATE" το σύστημα παραμένει μέχρι να ολοκληρωθεί το διάβασμα όλων των δεδομένων και έπειτα μεταβαίνει στην αρχική κατάσταση.



Εικόνα 22: Το διάγραμμα καταστάσεων του "Receive Headers".

4.7.2 Receive: Το υποσύστημα Read-Write TCB

Το υποσύστημα Read-Write TCB αποτελεί το πιο πολύπλοκο υποσύστημα σε όλη την σχεδίαση. Με διαστάσεις μεγαλύτερες από όλα τα υποσυστήματα, αρμοδιότητά του είναι, κατά κάποιο τρόπο, το control του Receive.

Πιο αναλυτικά, κάποιες από τις πιο βασικές λειτουργίες του Read – Write TCB είναι:

- Ο έλεγχος των εισερχόμενων πακέτων.
- Η διαχείριση των επιβεβαιώσεων και η δρομολόγηση των εντολών για την αποστολή τους.
- Ο έλεγχος των timers και γενικότερα, του Congestion Control.
- Ο έλεγχος της κατάστασης της σύνδεσης και η μετάβασή της σε άλλη κατάσταση, ανάλογα με τα εισερχόμενα πακέτα.
- Η ανανέωση κάποιων πεδίων στο socket table.
- Η αποστολή στο επίπεδο των εφαρμογών, απαντήσεων επιτυχούς λειτουργίας ή σφάλματος.
- Η εγγραφή των εισερχόμενων δεδομένων στην μνήμη λήψης ή το άδειασμα της FIFO απόζευξης εάν το πακέτο απορριφθεί.

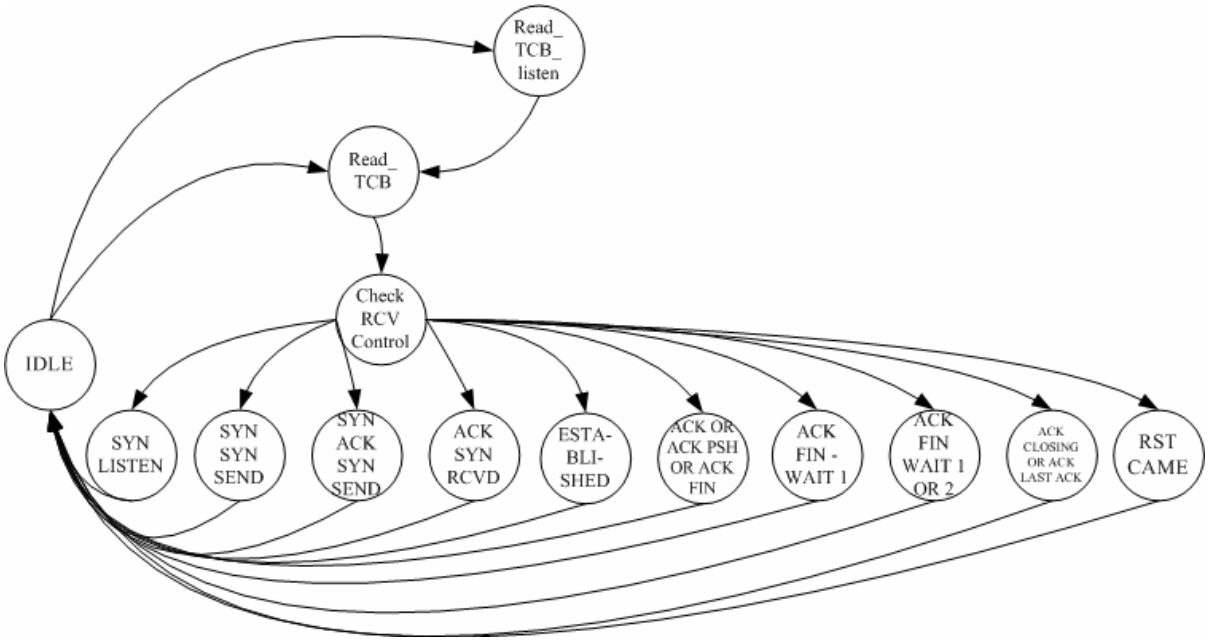
Η FSM η οποία υλοποιεί το υποσύστημα αυτό, παρουσιάζεται στην εικόνα 22. Σημειώνεται, ότι το διάγραμμα είναι αρκετά απλουστευμένο προκειμένου να γίνει πιο εύκολα αντιληπτός ο τρόπος λειτουργίας της FSM. Έτσι, πολλές «ομοειδείς» καταστάσεις έχουν συγχωνευτεί σε μία.

Πριν προχωρήσουμε στην ανάλυση του διαγράμματος των καταστάσεων που ακολουθεί η FSM, θα πρέπει να αναφέρουμε ότι στην μετάβαση από κατάσταση σε κατάσταση, βασικό ρόλο παίζει ο Receive Control Register. Ο Receive Control Register είναι ένας καταχωρητής ο οποίος παίρνει τιμές που εξαρτώνται α) από την κατάσταση της σύνδεσης και β) από τα Flags του εισερχόμενου πακέτου. Υπενθυμίζεται ότι τα flags κάθε εισερχόμενου πακέτου διαβάζονται από το υποσύστημα Receive Headers, το οποίο τα βγάζει, με την σειρά του, στην έξοδο σε ανεξάρτητο bus.

Στον πίνακα 8 παρουσιάζονται όλες οι πιθανές τιμές του Receive Control Register συναρτήσει της κατάστασης και των flags του εισερχόμενου πακέτου.

ΚΩΔΙΚΟΣ	ΚΑΤΑΣΤΑΣΗ ΣΥΝΔΕΣΗΣ	FLAGS IN	FSM STATE
00000000000001	Passive Socket = **LISTEN** Active Sockey = **CLOSED**	**SYN**	**SYN LISTEN**
00000000000010	**SYN - SEND**	**SYN**	**SYN SYN SENT**
00000000000100	**SYN – SEND**	**SYN-ACK**	**SYN ACK SYN SEND**
00000000001000	**SYN – RCVD**	**ACK**	**ACK SYN RCVD**
00000000010000	**ESTABLISHED** **FIN –WAIT-2**	**ACK**	**ACK OR ACK PSH OR ACK FIN**
00000000100000	**ESTABLISHED** **FIN –WAIT-2**	**ACK-PSH**	**ACK OR ACK PSH OR ACK FIN**
00000001000000	**ESTABLISHED**		**ESTABLISHED**
00000010000000	**ESTABLISHED**	**FIN-ACK**	**ACK OR ACK PSH OR ACK FIN**
00000100000000	**FIN-WAIT-1**	**ACK**	**ACK FIN WAIT 1**
00001000000000	**FIN-WAIT-1**	**FIN-ACK**	**ACK FIN WAIT 1 OR 2**
00010000000000	**FIN-WAIT-2**	**FIN-ACK**	**ACK FIN WAIT 1 OR 2**
00100000000000	**CLOSING**	**ACK**	**ACK CLOSING OR LAST ACK**
01000000000000	**LAST ACK**	**ACK**	**ACK CLOSING OR LAST ACK**
10000000000000		**RST**	**RST CAME**

Πίνακας 8: Οι τιμές του Receive Control Register και οι αντίστοιχες καταστάσεις μετάβασης της Read-Write TCB FSM.



Εικόνα 23: Το διάγραμμα καταστάσεων του "Read-Write TCB".

Όπως φαίνεται στην εικόνα 22 το FSM ξεκινά από την κατάσταση “IDLE”. Από την κατάσταση “IDLE” μπορεί να φθάσει στην “READ TCB” είτε άμεσα, είτε μέσω της “READ_TCB_LISTEN”. Από την ενδιάμεση κατάσταση περνάμε μόνο στην περίπτωση που το syn flag του εισερχόμενου πακέτου είναι set. Στην κατάσταση

αυτή δημιουργείται το address για πρόσβαση στα passive sockets του socket table, ώστε να γίνει ο έλεγχος για το αν η σύνδεση τύπου εξυπηρετητή είναι "LISTEN" ή "CLOSED".

Στο σημείο αυτό, θα πρέπει να υπενθυμίσουμε ότι για να γίνει ένα επιτυχές άνοιγμα σε passive socket θα πρέπει η κατάσταση στην αντίστοιχη θέση του socket table για τα passive sockets να είναι "LISTEN" και η κατάσταση στο active μέρος του πίνακα να είναι "CLOSED". Αντίθετα για να γίνει ένα επιτυχές άνοιγμα σε active socket, απαιτείται έλεγχος μόνο στο active τμήμα του socket table προκειμένου να διαπιστωθεί ότι η κατάσταση είναι "CLOSED".

Ανεξάρτητα από την πορεία που θα ακολουθήσει η μετάβαση, κάποια στιγμή φθάνει στην κατάσταση "READ TCB". Στην κατάσταση αυτή, το Read-Write TCB διαβάζει τα στοιχεία της σύνδεσης από την αντίστοιχη θέση του socket table και έπειτα μετακινείται στην κατάσταση "CHECK RCV CONTROL REGISTER". Στην κατάσταση αυτή, γίνεται ο πιο ουσιαστικός έλεγχος όλης της FSM. Από τον έλεγχο του Receive Control Register – του οποίου η τιμή διαμορφώθηκε από την τρέχουσα κατάσταση της σύνδεσης και τον τύπο του εισερχόμενου πακέτου, όπως είδαμε στον πίνακα 8 – το σύστημα θα αποφασίσει προς ποια κατάσταση θα κινηθεί.

Οι καταστάσεις μετά την "CHECK RCV CONTROL REGISTER" δεν ανταποκρίνονται στην πραγματική δομή της FSM, καθώς κάτι τέτοιο θα εισήγαγε στο σχήμα πολυπλοκότητα που μόνο μπέρδεμα θα προκαλούσε. Αντίθετα, οι καταστάσεις που παρουσιάζονται αποτελούν μία «περίληψη» των καταστάσεων που ακολουθούνται ανά περίπτωση. Παρακάτω θα εξετάσουμε μία-μία περίπτωση ξεχωριστά.

Στην κατάσταση "SYN LISTEN" μεταφερόμαστε όταν το εισερχόμενο πακέτο έχει το SYN flag set. Στην περίπτωση αυτή, ελέγχουμε το destination port number ώστε να σιγουρευτούμε ότι το εισερχόμενο πακέτο έχει φθάσει στον σωστό προορισμό, ενημερώνουμε το socket table και επιστρέφουμε το κατάλληλο μήνυμα στο επίπεδο των εφαρμογών.

Στην κατάσταση "SYN SYN SENT" μεταφερόμαστε όταν το σύστημα λάβει SYN ενώ αυτό βρίσκεται σε κατάσταση SYN-SEND. Σύμφωνα με το διάγραμμα καταστάσεων της εικόνας 13, το σύστημα θα πρέπει να μεταβεί στην κατάσταση SYN-RCVD αφού στείλει πιο πρώτα ack. Στην παρούσα κατάσταση, γίνεται ενημέρωση των βασικών παραμέτρων στο TCB και στέλνεται εντολή στην ενδιάμεση FIFO για αποστολή απλής επιβεβαίωσης (plain ack). Από τις πιο βασικές τιμές που

ενημερώνονται στο TCB είναι το Connection State, το οποίο πλέον «δείχνει» στην κατάσταση SYN-RCVD και τα Connection Flags τα οποία δείχνουν ότι πρέπει να σταλεί ack αλλά και ότι έγινε μετάβαση από το SYN-SEND στο SYN-RCVD.

Στην κατάσταση SYN ACK SYN SEND μεταφερόμαστε όταν το σύστημα λάβει SYN-ACK, ενώ αυτό βρίσκεται στην κατάσταση SYN-SEND. ΄Συμφωνα με το διάγραμμα καταστάσεων της εικόνας 13, το σύστημα θα πρέπει να μεταβεί στην κατάσταση "ESTABLISHED". Πριν την μετάβαση αυτή, όπως έγινε και στην προηγούμενη περίπτωση, διορθώνονται τα Connection Flags ώστε να μπορεί να σταλεί επιβεβαίωση, ενημερώνεται στο socket table και αποστέλλεται η εσωτερική εντολή στην ενδιάμεση FIFO "Send Plain Ack". Τέλος, αποστέλλεται μήνυμα επιτυχούς ανοίγματος σύνδεσης στο επίπεδο των εφαρμογών.

Στην κατάσταση "ACK SYN RECEIVED" μεταφερόμαστε στην περίπτωση που το σύστημα βρίσκεται στην κατάσταση "SYN-RCVD" και λάβει ack. Σύμφωνα με το διάγραμμα καταστάσεων, το σύστημα μεταβαίνει στην κατάσταση "ESTABLISHED". Προηγουμένως, όμως, έχει ενημερωθεί το socket table. Τέλος αποστέλλεται μήνυμα επιτυχούς ανοίγματος σύνδεσης στο επίπεδο των εφαρμογών.

Στην κατάσταση "ESTABLISHED" μεταφερόμαστε σε περίπτωση που το σύστημα βρίσκεται στην "ESTABLISHED" και το εισερχόμενο πακέτο δεν εσωκλείει επιβεβαίωση, RST (reset) ή PSH (push) flag. Στην περίπτωση αυτή, το FSM προχωρά σε έλεγχο του εισερχόμενου πακέτου, προκειμένου να διαπιστώσει, εάν αυτό διαθέτει δεδομένα, εάν είναι εντός σειράς κλπ. Μετά από μία μικρή αναμονή για να ολοκληρωθεί το dequeue από την FIFO απόζευξης, το σύστημα αποστέλλει μήνυμα επιτυχούς λήψης στο επίπεδο εφαρμογών, ξεκινά τον αντίστοιχο idle timer και αποστέλλει εντολή "Send Plain Ack" στην ενδιάμεση FIFO.

Στην κατάσταση "ACK OR ACK PSH OR ACK FIN" μεταφερόμαστε στις περιπτώσεις που το σύστημα:

I. Βρίσκεται στην κατάσταση "ESTABLISHED" ή "FIN-WAIT-2" και έγινε λήψη ACK,
II. Βρίσκεται στην κατάσταση "ESTABLISHED" ή "FIN-WAIT-2" και έγινε λήψη ACK & PSH,
III. Βρίσκεται στην κατάσταση "ESTABLISHED" και έγινε λήψη FIN & ACK.

Στην εν λόγω κατάσταση, πρέπει, να σημειωθεί, ότι γίνεται ο μεγαλύτερος αριθμός ελέγχων. Έτσι, αρχικά γίνεται έλεγχος για το αν το εισερχόμενο πακέτο διαθέτει δεδομένα και έπειτα λαμβάνει χώρα μία σειρά πιο αναλυτικών ελέγχων,

όπως, το αν το πακέτο είναι εντός σειράς, αν αποτελεί το τελευταίο πακέτο του flight size, αν φέρει σωστό acknowledge number κλπ.

Οι παραπάνω έλεγχοι προωθούν την μετάβαση σε άλλες καταστάσεις, στις οποίες ενημερώνεται η μνήμη "Memory 41" (σε περίπτωση λήψης ack που επιβεβαιώνει ένα απεσταλθέν flight size), δίνεται εντολή για επανέναρξη της αποστολής από το τελευταίο επιβεβαιωμένο πακέτο (με χρήση της εντολής "Send Packet After Last Ack"), επιστρέφεται μήνυμα επιτυχούς αποστολής στο επίπεδο εφαρμογών (σε περίπτωση λήψης σωστής ack για το τελευταίο απεσταλθέν πακέτο), ελέγχεται η δυνατότητα αποστολής ack με piggy back (αν η ίδια σύνδεση στέλνει την στιγμή λήψης), ελέγχονται οι timers (σταματά ή επανεκκινείται ο Retransmission Timer), επιστρέφεται μήνυμα επιτυχούς λήψης στο επίπεδο των εφαρμογών (στην περίπτωση που ο χώρος της μνήμης λήψης ελαττωθεί επικίνδυνα ή το PSH flag του εισερχόμενου πακέτου είναι set) κλπ. Επίσης, ένας από τους τελευταίους ελέγχους που γίνονται, αφορά στο αν το εισερχόμενο πακέτο είναι FIN, οπότε η σύνδεση μεταβαίνει στην κατάσταση "CLOSE WAIT", ενημερώνεται το TCB και στέλνεται εντολή στην ενδιάμεση FIFO για αποστολή επιβεβαίωσης. Στις προηγούμενες περιπτώσεις που προαναφέρθηκαν, πριν η FSM μεταφερθεί στην κατάσταση "IDLE", λαμβάνει χώρα η ενημέρωση του socket table με τους αριθμούς sequence, acknowledgement κλπ. Επιπλέον, στην ενημέρωση αυτή χρησιμοποιούμε και την νέα τιμή του round trip time που έρχεται από το υποσύστημα "Compute RTT".

Το υποσύστημα "Compute RTT" είχε δεχτεί εντολή από το υποσύστημα αυτό και συγκεκριμένα από την κατάσταση που αναλύουμε, αρκετούς κύκλους πριν. Ο λόγος για τον οποίο ενεργοποιούμε το υποσύστημα υπολογισμού του RTT νωρίτερα είναι ότι για την ολοκλήρωση της διαδικασίας απαιτούνται είκοσι δύο ολόκληροι κύκλοι και το αποτέλεσμα πρέπει να είναι έτοιμο κατά την διάρκεια ενημέρωσης του socket table.

Στην κατάσταση "ACK FIN WAIT 1" μεταφερόμαστε στην περίπτωση που η σύνδεση είναι σε "FIN WAIT 1" και ληφθεί ack. Στην περίπτωση αυτή, γίνεται έλεγχος για το αν το πακέτο περιέχει δεδομένα, έλεγχος σειράς, επιστροφή μηνύματος επιτυχούς λήψης (αν ο διαθέσιμος χώρος που απομένει στην μνήμη λήψης είναι μικρός). Επίσης εκκινείται ο κατάλληλος idle timer, ενημερώνεται το socket table, η κατάσταση της σύνδεσης μεταβαίνει στην "FIN WAIT 2" και δίνεται εντολή για αποστολή απλής επιβεβαίωσης (εσωτερική εντολή "Send Plain Ack).

Στην κατάσταση "ACK FIN WAIT 1 OR 2" μεταφερόμαστε στην περίπτωση που η σύνδεση ήταν σε "FIN WAIT 1" ή "FIN WAIT 2" και ελήφθη FIN-ACK. Όμοια με τις προηγούμενες περιπτώσεις, και σε αυτήν γίνεται έλεγχος του εισερχόμενου πακέτου. Εφόσον είναι εντός σειράς, διαπιστώνεται εάν φέρει δεδομένα. Εάν φέρει δεδομένα, αυτά γράφονται στην μνήμη λήψης. Πριν η σύνδεση οδηγηθεί στην κατάσταση "CLOSED" (σύμφωνα, πάντα, με το διάγραμμα καταστάσεων της εικόνας 13), αποστέλλεται στην εσωτερική FIFO η εντολή "Send Plain Ack" και ενημερώνεται το socket table.

Στην κατάσταση "ACK CLOSING OR ACK LAST ACK" μεταφερόμαστε εάν η σύνδεση βρίσκεται σε "LAST ACK" ή "CLOSING" και ληφθεί ACK. Στην περίπτωση αυτή, γίνεται έλεγχος σειράς για το εισερχόμενο πακέτο και η σύνδεση μεταβαίνει στην κατάσταση "CLOSED".

Στην κατάσταση "RST CAME" μεταφερόμαστε κάθε φορά που λαμβάνουμε πακέτο με το RST flag set. Στην περίπτωση αυτή, ελέγχεται αν η σύνδεση που έλαβε το reset στέλνει ακόμα (και είναι σε κατάσταση "ESTABLISHED"). Αν ναι, τότε επιστρέφεται μήνυμα λάθους στο επίπεδο εφαρμογών. Στις υπόλοιπες περιπτώσεις η σύνδεση μεταβαίνει στην κατάσταση "CLOSED" και ενημερώνεται το socket table.

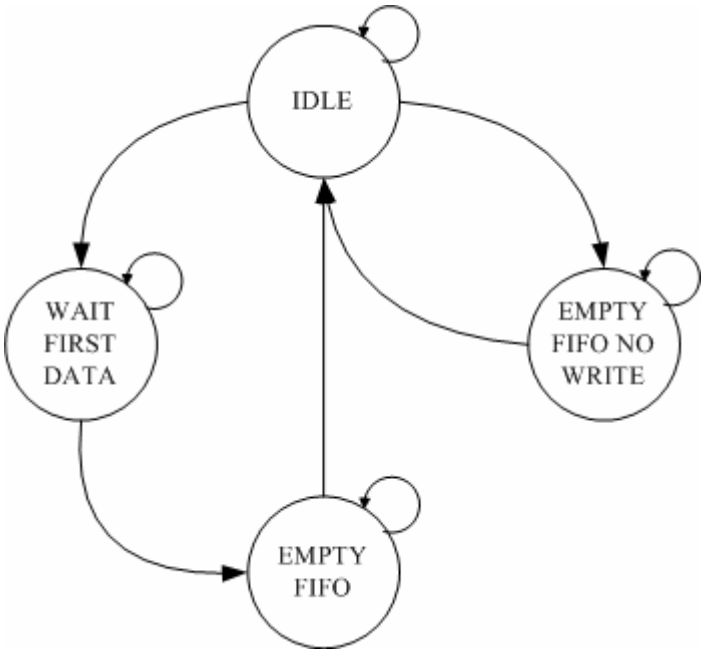
4.7.3 Receive: Το υποσύστημα "Fill Receive Memory"

Το υποσύστημα "Fill Receive Memory" είναι το πιο απλό υποσύστημα του Receive. Όπως υποδηλώνει και το όνομά του, βασικός στόχος του υποσυστήματος είναι η μεταφορά των δεδομένων από την FIFO απόζευξης (Decouple FIFO) στην μνήμη λήψης. Βέβαια, το υποσύστημα αυτό δεν δρα αυτόνομα, αλλά «οδηγείται» από το "Receive Read Write TCB", που μελετήθηκε αναλυτικά στην προηγούμενη υποπαράγραφο. Στην εικόνα 23 φαίνεται παραστατικά ο ρόλος του υπό συζήτηση υποσυστήματος.

Το υποσύστημα "Fill Receive Memory" υλοποιείται με χρήση μίας μηχανής πεπερασμένων καταστάσεων (FSM). Οι καταστάσεις από τις οποίες διέρχεται το σύστημα προκειμένου να ολοκληρώσει την λειτουργία του φαίνονται στην εικόνα 24.

Εικόνα 24: Η λειτουργία του υποσυστήματος "Fill Receive Memory"

Το σύστημα ξεκινά από την κατάσταση “IDLE”, στην οποία παραμένει μέχρι να λάβει σήμα από το “Receive Read Write TCB” για εκκίνηση μεταφοράς των δεδομένων από την decouple FIFO στην μνήμη λήψης ή για άδειασμα της decouple FIFO (περίπτωση απόρριψης πακέτου). Στην πρώτη περίπτωση, το σύστημα μεταβαίνει στην “WAIT FIRST DATA”, για να υπολογισθεί η διεύθυνση από την οποία θα ξεκινήσει η εγγραφή της μνήμης λήψης και αμέσως μετά, στην “EMPTY FIFO”. Στην κατάσταση “EMPTY FIFO” παραμένει μέχρι να αδειάσει τελείως η FIFO απόζευξης και τα δεδομένα να περάσουν στην μνήμη λήψης. Τέλος, στην δεύτερη περίπτωση το σύστημα μεταβαίνει στην κατάσταση “EMPTY FIFO NO WRITE” και παραμένει εκεί μέχρι να αδειάσει η FIFO τελείως. Σημειώνεται ότι, η μνήμη λήψης στην κατάσταση αυτή είναι απενεργοποιημένη, διότι, δεν είναι επιθυμητά τα δεδομένα που περικλείει το πακέτο.



Εικόνα 25: Διάγραμμα καταστάσεων του "Fill Receive Memory".

4.8 Οι Timers

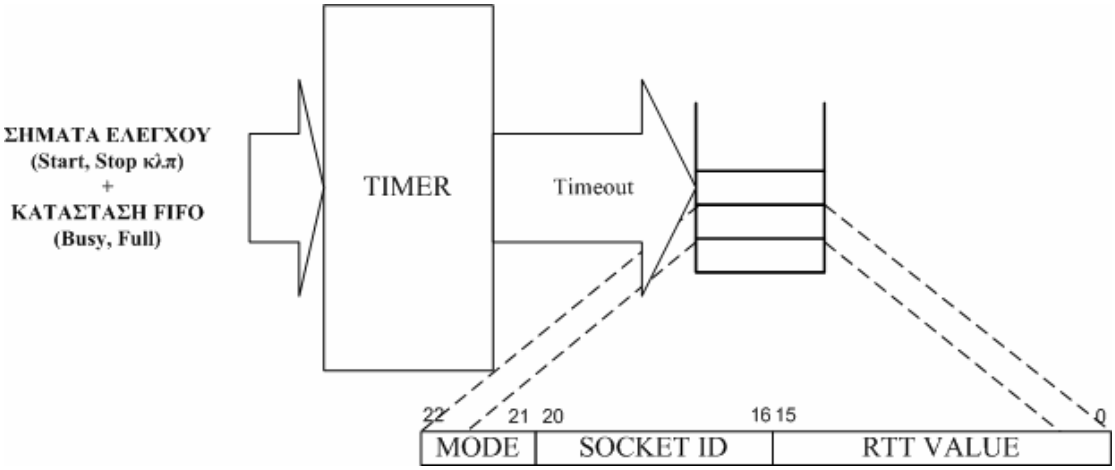
Όπως αναφέρθηκε αρκετές φορές σε αυτό, αλλά και στο προηγούμενο κεφάλαιο, ένα από τα πιο βασικά υποσυστήματα του TCP αποτελεί ο timer. Κάθε σύνδεση διαθέτει τον δικό της timer. Μάλιστα, το υποσύστημα αυτό έχει μία ιδιαιτερότητα στην λειτουργία του, καθώς είναι σε θέση να χρονομετρά με τέσσερις διαφορετικούς τρόπους, υλοποιώντας τους syn, retransmission, persistence και idle timer που έχουμε δει.

Η πολλαπλή λειτουργικότητα του timer, βέβαια, δεν σταματά εδώ, καθώς, σε αυτή την έκδοση της σχεδίασης εξοπλίστηκε με τέτοιο τρόπο, ώστε κάθε φορά που κάνει time out, να τοποθετεί τα στοιχεία του time out σε μία FIFO, την "Timeouts Bank". Η λειτουργικότητα που μόλις περιγράψαμε παρουσιάζεται παραστατικά στην εικόνα 25.

Αναλύοντας περισσότερο το υποσύστημα του timer, θα πρέπει να αναφέρουμε ότι διαθέτει τέσσερα modes λειτουργίας και τρία modes busy. Τα modes αυτά παρουσιάζονται στον πίνακα 9. Τονίζεται, ότι ο timer δεν διαθέτει καμία απολύτως ικανότητα στο να μπορεί να διακρίνει πιο mode έχει την μεγαλύτερη προτεραιότητα. Έτσι, τα υποσυστήματα που συνδέονται με τον timer ασκούν ολοκληρωτικό έλεγχο, δεδομένου ότι αυτός δεν μπορεί κανονίσει την προτεραιότητα των αιτήσεων. Το χαρακτηριστικό αυτό δεν αποτελεί πρόβλημα, καθώς, για τον λόγο αυτό υπάρχουν τα busy modes, τα οποία λαμβάνουν υπόψιν όλα τα υποσυστήματα που ασκούν έλεγχο στον timer.

Τα σήματα ελέγχου που διαθέτει ο timer είναι τα start, stop, clear και restart, που ξεκινούν, σταματούν, σβήνουν τον χρόνο timeout και επανεκκινούν, αντίστοιχα τον timer. Παράλληλα, τα σήματα busy και full της FIFO συνδέονται σε κάθε timer. Με τον τρόπο αυτό, ο timer είναι σε θέση να ανιχνεύσει αν η FIFO είναι απασχολημένη ή γεμάτη.

Τέλος, θα πρέπει να συμπληρώσουμε ότι, εκτός από τις εξόδους για την FIFO, κάθε timer διαθέτει άλλες δύο προκειμένου να περνά τις τιμές του Round Trip Time (την παλαιά που φόρτωσε ως χρόνο timeout και την νέα που είναι ο χρόνος που μέτρησε μέχρι να δεχτεί το σήμα stop) στο υποσύστημα υπολογισμού του RTT (Compute RTT) που θα δούμε στην παράγραφο 4.12.



Εικόνα 26: Ο timer του TCP.

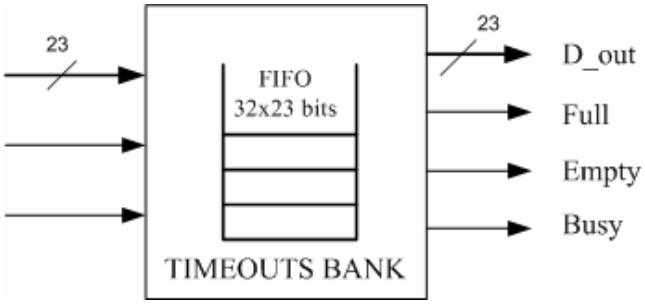
FUNCTION	OPERATION MODE	BUSY MODE
Syn Timer	00	01
Retransmission Timer	01	01
Persistence Timer	10	10
Idle Timer	11	

Πίνακας 9: Τα modes του Timer.

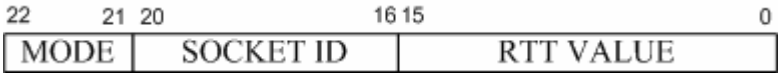
4.9 Το "Timeouts Bank"

Το υποσύστημα "Timeouts Bank" δεν παρουσιάζει σχεδόν καμία πολυπλοκότητα. Όπως φαίνεται και στην εικόνα 26, αποτελείται από μία FIFO 32 θέσεων εύρους 23 bits. Η μόνη λογική που υπάρχει στο υποσύστημα είναι αυτή που δημιουργεί το σήμα busy.

Ο ρόλος του υποσυστήματος "Timeouts Bank" είναι η φύλαξη των timeouts ώστε να μπορούν να εξυπηρετηθούν από το υποσύστημα "Timeouts Control". Ο σχεδιασμός αυτός, μας προφυλάσσει από την αστάθεια που παρουσίαζε η προηγούμενη έκδοση της σχεδίασης, όταν δέχονταν παράλληλα timeouts. Στην παρούσα έκδοση μπορούν να εξυπηρετηθούν ακόμα και 31 παράλληλα timeouts (η χειρότερη περίπτωση)!



Εικόνα 27: Το υποσύστημα "Timeouts Bank".



Εικόνα 28: Η δομή ενός αποθηκευμένου timeout.

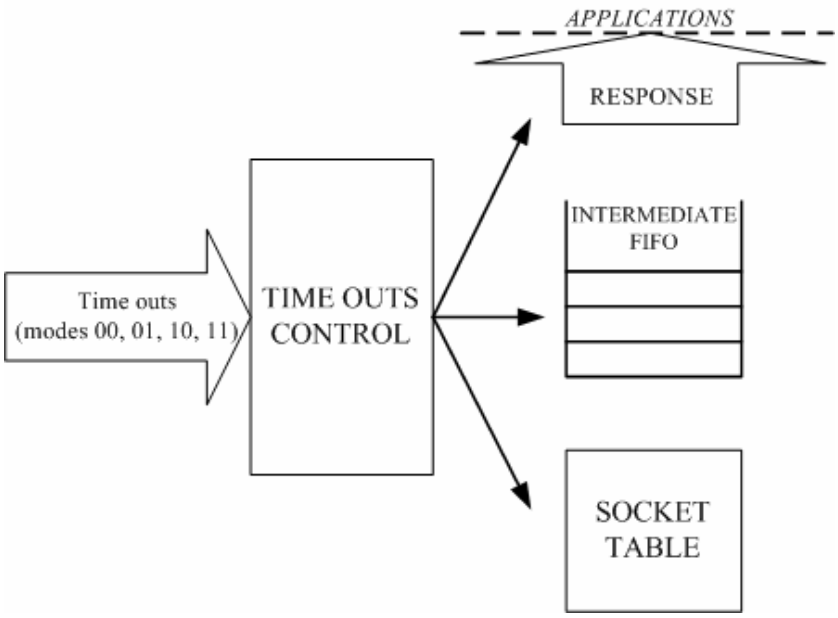
Στην εικόνα 27 παρουσιάζεται η δομή ενός αποθηκευμένου στην FIFO time out. Τα δύο MSB (πεδίο "MODE") χρησιμοποιούνται για την αποθήκευση του timeout mode (syn time out, retransmission time out, persistence time out, idle time out), τα επόμενα πέντε για το Socket ID και τέλος, τα τελευταία 16 lsb για την τιμή time out πεδίο "RTT VALUE"). Η τιμή time out κρίνεται απαραίτητη γιατί σε περίπτωση time out το νέο RTT ισούται με το παλιό πολλαπλασιασμένο επί δύο. Ο υπολογισμός του νέου RTT γίνεται από το "Timeouts Control" που θα δούμε στην επόμενη παράγραφο.

4.10 Το "Timeouts Control"

Το υποσύστημα "Timeouts Control", όπως φανερώνει και η ονομασία του, είναι υπεύθυνο για την διαχείριση των time outs των timer. Η διαχείριση των time outs, βέβαια, θα μπορούσε να γίνει και από τους ίδιους τους timers, προτιμήθηκε, όμως, η πρώτη λύση, προκειμένου να υπάρχει καλύτερη ιεραρχία σε θέματα αρμοδιότητας των υποσυστημάτων.

Στην εικόνα 28 παρουσιάζεται παραστατικά ο ρόλος του "Timeouts Control". Όπως φαίνεται στην εικόνα, το υποσύστημα δέχεται τα time outs και ανάλογα με τον τύπο τους, αποφασίζει πως θα δράσει. Αν, δηλαδή, θα στείλει κάποιο μήνυμα λάθους στο επίπεδο των εφαρμογών, αν θα εισάγει στην ενδιάμεση FIFO κάποια εντολή ή αν

θα ενημερώσει κάποια πεδία από το socket table. Βέβαια, πρέπει να αναφερθεί, ότι η φράση «το υποσύστημα δέχεται τα time outs» χρησιμοποιείται καταχρηστικά, γιατί, το υποσύστημα στην πραγματικότητα, δεν δέχεται άμεσα τα time outs, αλλά έμμεσα. Πιο συγκεκριμένα, κάθε time out οδηγείται στο "Timeouts Bank", το οποίο μετά «σαρώνεται» από το "Timeouts Control". Στον πίνακα 10 παρουσιάζεται η δράση του "Timeouts Control" ανά timer mode.

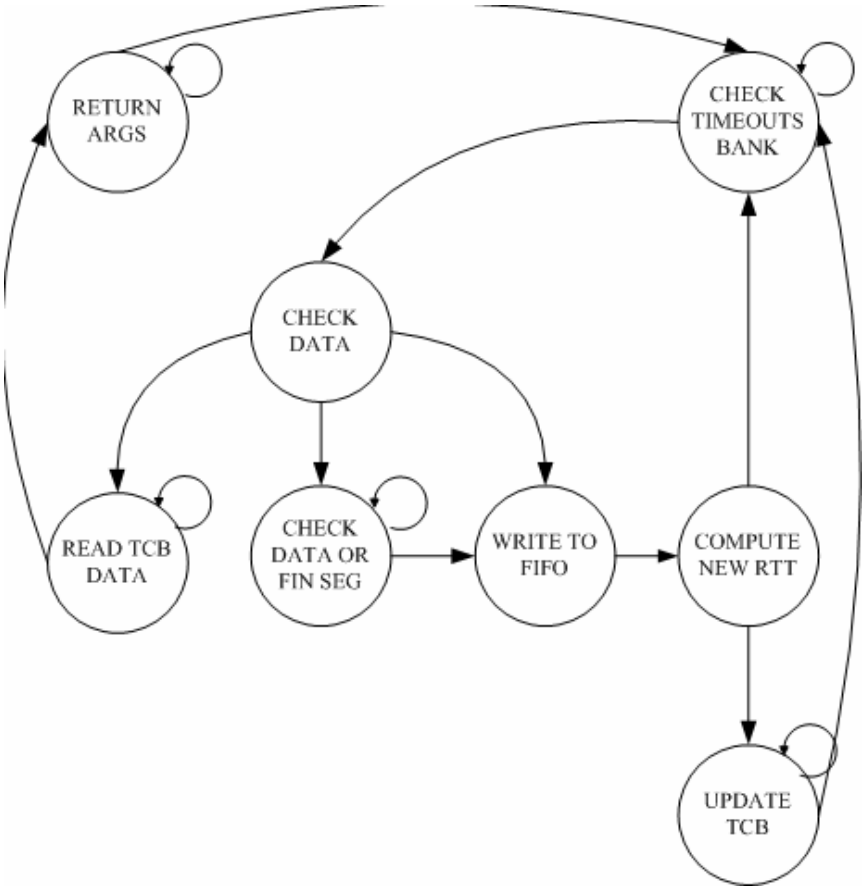


Εικόνα 29: Ο ρόλος του "Timeouts Control".

TIMEOUT MODE	**ΔΡΑΣΗ ΥΠΟΣΥΣΤΗΜΑΤΟΣ "TIMEOUTS CONTROL"**
00 (Syn timeout)	Αποστολή μηνύματος "*Err_Open_Client_Socket*" και διαγραφή του TCB.
01 (Retransmission Timeout)	Αποστολή εντολής "*Retransmit Data*" στην ενδιάμεση FIFO.
10 (Persistence Timeout)	Αποστολή εντολής "*Send Probe*" στην ενδίαμεση FIFO.
11 (Idle Timeout)	Αποστολή εντολής "*Initialize Congestion Window*" στην ενδιάμεση FIFO.

Πίνακας 10: Η αντίδραση του υποσυστήματος "Timeouts Control" ανά timeout mode.

Έχοντας ολοκληρώσει την διαδικασία περιγραφής της λειτουργίας του υποσυστήματος, θα εισέλθουμε στα ενδότερά του,προκειμένου να αναλύσουμε λίγο περισσότερο την δομή αλλά και την υλοποίησή του. Το υποσύστημα "Timeouts Control" υλοποιείται με μία μηχανή πεπερασμένων καταστάσεων (FSM), η δομή της οποίας είναι, σχετικά, απλή και φαίνεται στην εικόνα 29.



Εικόνα 30: Το διάγραμμα καταστάσεων του "Timeouts Control".

Το υποσύστημα βρίσκεται, συνήθως, στην κατάσταση “CHECK TIMEOUTS BANK”. Στην κατάσταση αυτή ελέγχεται διαρκώς η FIFO που βρίσκεται στο υποσύστημα “TIMEOUTS BANK”. Όσο αυτή είναι κενή, το υποσύστημα παραμένει σε κατάσταση ετοιμότητας. Όταν, όμως, στην FIFO γραφτεί έστω και ένα timeout από έναν από τους τριανταένα timers, τότε, αυτομάτως, το υποσύστημα μεταβαίνει στην κατάσταση “CHECK DATA”. Στην κατάσταση “CHECK DATA” διαβάζεται η εγγραφή της FIFO και αποθηκεύονται σε μεταβλητές τα επιμέρους πεδία της. Ανάλογα με το mode του timeout η FSM μεταβαίνει σε μία από τις καταστάσεις “READ TCB DATA”, “CHECK DATA OR FIN SEG” και “WRITE TO FIFO”.

Το σύστημα μεταβαίνει στην κατάσταση “READ TCB DATA” στην περίπτωση syn timeout, προκειμένου να διαβάσει τα στοιχεία της σύνδεσης που αφορούν το remote IP address, destination port number και source port number. Οι πληροφορίες αυτές χρησιμοποιούνται, ως ορίσματα, στην επόμενη κατάσταση, την “RETURN ARGS”, κατά την διάρκεια της οποία το σύστημα αποστέλλει μήνυμα σφάλματος στο επίπεδο εφαρμογών (“Err_Open_Clent_Socket”).

Η μετάβαση στην κατάσταση "CHECK DATA OR FIN SEG" συμβαίνει στην περίπτωση retransmission timeout. Κατά την διάρκεια της κατάστασης αυτής το υποσύστημα προσπελαύνει το socket table και ελέγχει εάν το πακέτο για το οποίο σημειώθηκε timeout ήταν fin ή data. Ανάλογα με το είδος του πακέτου δημιουργείται η κατάλληλη εντολή και εγγράφεται στην ενδιάμεση FIFO, διαδικασία η οποία λαμβάνει χώρα στην επόμενη κατάσταση, την "WRITE TO FIFO" .

Το σύστημα μεταβαίνει στην κατάσταση "WRITE TO FIFO", είτε μετά από την "CHECK DATA OR FIN SEG", είτε μετά την "CHECK DATA" υπό την συνθήκη ότι έχει συμβεί idle ή persistence timeout. Στην παρούσα κατάσταση, λαμβάνει χώρα η εγγραφή της εντολής στην ενδιάμεση FIFO.

Μετά την κατάσταση "WRITE TO FIFO" το σύστημα μεταβαίνει στην "COMPUTE NEW RTT", στην οποία γίνεται έλεγχος του είδους του time out. Σε περίπτωση που προκύψει ότι αυτό είναι retransmission time out τότε υπολογίζεται η νέα τιμή RTT και δίνεται εντολή για μετάβαση στην κατάσταση "UPDATE TCB". Διαφορετικά, το σύστημα μεταβαίνει στην αρχική κατάσταση ("CHECK TIMEOUTS BANK").

Τέλος, στην κατάσταση "UPDATE TCB" λαμβάνει χώρα η ενημέρωση του TCB με την νέα τιμή του Round Trip Time και το σύστημα οδηγείται στην αρχική "CHECK TIMEOUTS BANK".

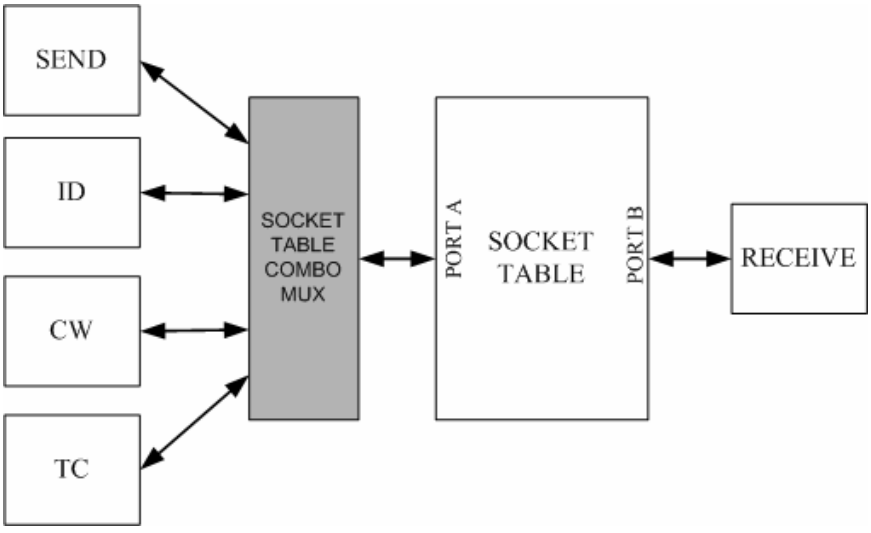
Σημειώνεται, ότι για τις εντολές της ενδιάμεσης FIFO που είδαμε μέχρι στιγμής ονομαστικά, θα δοθεί αναλυτική περιγραφή στην παράγραφο 4.14 που θα μιλήσουμε για το υποσύστημα αυτό.

4.11 Το "Socket Table Combo Mux"

Η λειτουργία του υποσυστήματος "Socket Table Mux" έχει πολλές ομοιότητες με την λειτουργία ενός πολυπλέκτη. Η μόνη διαφορά μεταξύ ενός πολυπλέκτη και του "Socket Table Compo Mux", έγκειται στο γεγονός ότι, στον μεν πολυπλέκτη έχουμε να κάνουμε με πολλές εισόδους και μία έξοδο, ενώ, στο υπό εξέταση υποσύστημα υπάρχουν περισσότερες από μία έξοδοι. Γενικεύοντας την συζήτησή μας, θα μπορούσαμε να πούμε ότι το "Socket Table Combo Mux" μοιάζει με ένα

υποσύστημα δομημένο με πολλούς πολυπλέκτες. Αξίζει να σημειωθεί, επίσης, ότι το συγκεκριμένο υποσύστημα είναι ασύγχρονο - δεν φέρει ρολόι - για δύο βασικούς λόγους. Κατά πρώτον, γιατί η παρουσία του παλμού του ρολογιού θα εισήγαγε μία «νόμιμη» καθυστέρηση ενός κύκλου, υπό την έννοια ότι κάθε υφιστάμενη αλλαγή των εισόδων θα γίνονταν αντιληπτή στην έξοδο στον επόμενο κύκλο, και κατά δεύτερον, διότι, τα σήματα εισόδου είναι ήδη συγχρονισμένα από τα υπόλοιπα υποσυστήματα, γεγονός που σε συνδυασμό με την απλότητα της δομής του υποσυστήματος "Socket Table Combo Mux", επέτρεψε παράληψη του ρολογιού.

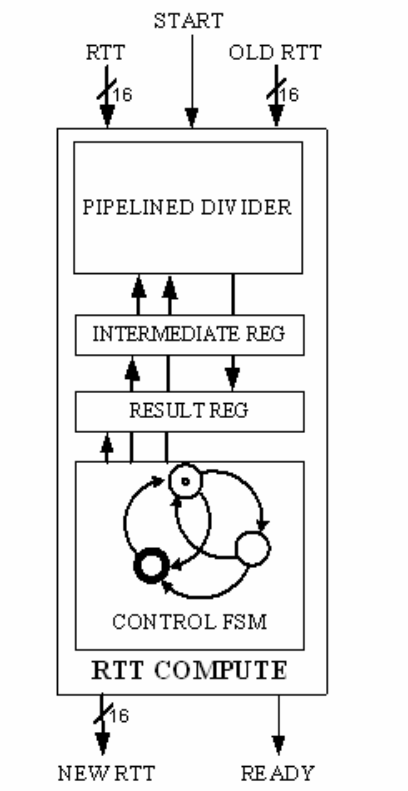
Ο ρόλος του υποσυστήματος "Socket Table Combo Mux" στην σχεδίαση μας μοιάζει με εκείνον του «τροχονόμου». Όπως είδαμε, νωρίτερα, στην παρούσα αναφορά, πολλά από τα υποσυστήματα ζητάνε πρόσβαση στο socket table. Από όλα τα υποσυστήματα που έχουν πρόσβαση στο socket table, το μόνο υποσύστημα που δεν έχει απολύτως κανένα πρόβλημα είναι το Receive. Αυτό συμβαίνει διότι το port B των δύο μνημών που υλοποιούν το socket table προορίζεται μόνο για το υποσύστημα λήψης. Όλα τα υπόλοιπα υποσυστήματα (Send, Change Window, Timeouts Control, Instruction Decode) αναγκάζονται να «διαμοιραστούν» το άλλο port (το port A). Αν και τα υποσυστήματα αυτά γνωρίζουν πότε και ποιο πραγματοποιεί πρόσβαση στο socket table, με αποτέλεσμα να υπάρχει η επιθυμητή αναμονή πριν την δική τους πρόσβαση, χρειάζονταν κάποιο υποσύστημα που να είναι σε θέση να «διασταυρώσει» τα σήματα από τα παραπάνω υποσυστήματα από και προς το socket table. Το υποσύστημα αυτό είναι το "Socket Table Compo Mux" και ο ρόλος του περιγράφεται σχηματικά στην εικόνα 30.



Εικόνα 31: Ο ρόλος του "Socket Table Combo Mux".

4.12 Το "RTT Compute"

Το υποσύστημα "RTT Compute" χρησιμοποιείται για τον υπολογισμό της νέας τιμής Round Trip Time μετά από κάθε επιτυχημένη μετάδοση των flight sizes. Η εντολή έναρξης του υπολογισμού έρχεται από το υποσύστημα λήψης πλαισίων, ενώ οι τιμές των δύο RTTs έρχονται από τον timer.

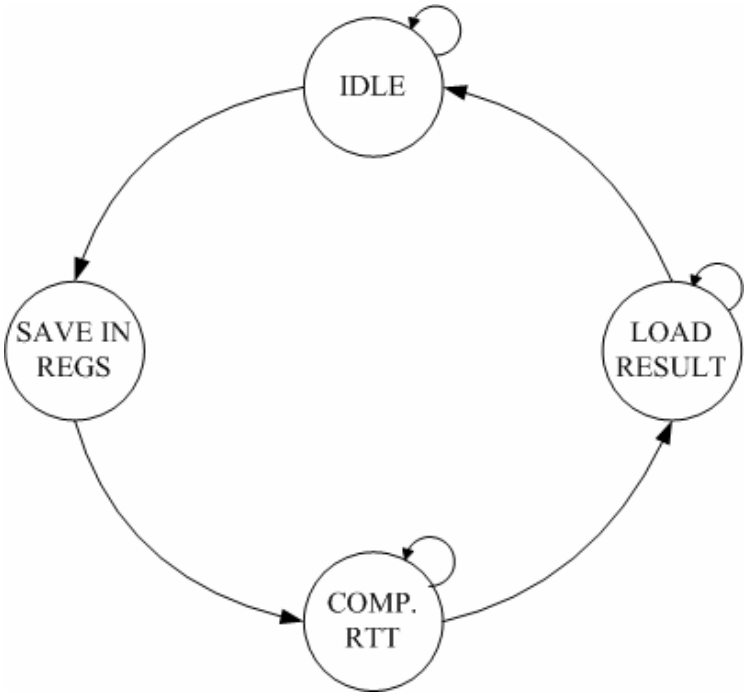


Εικόνα 32: Η δομή του "RTT Compute".

Αναλύοντας το υποσύστημα ως προς την δομή του, όπως φαίνεται και στην εικόνα 31, αυτό αποτελείται από ένα pipelined divider, δύο καταχωρητές που κρατάνε την τιμή του διαιρετέου και το αποτέλεσμα από τον divider αντίστοιχα και μία μηχανή πεπερασμένων καταστάσεων, που αναλαμβάνει, κατά κάποιο, τρόπο «χρέη» control για το υποσύστημα.

Η μηχανή πεπερασμένων καταστάσεων φαίνεται στην εικόνα 32. Το σύστημα ξεκινά από την κατάσταση "IDLE" και παραμένει εκεί, σε κατάσταση ετοιμότητας, μέχρι να λάβει σήμα από το υποσύστημα Receive για έναρξη του υπολογισμού. Στην περίπτωση που δεχθεί σήμα για έναρξη υπολογισμού, τότε η FSM μεταβαίνει στην κατάσταση "SAVE IN REGS". Σε αυτήν την κατάσταση, υπολογίζεται και φυλάσσεται σε καταχωρητή ο διαιρετέος, ενώ παράλληλα, δίνεται εντολή για μετάβαση στην επόμενη κατάσταση, την "COMPUTE RTT". Στην κατάσταση αυτή,

ξεκινά η πράξη της διαίρεσης, η οποία διαρκεί δεκαεννέα κύκλους. Μετά το πέρας του υπολογισμού, το σύστημα μεταβαίνει στην "LOAD RESULT", προκειμένου το αποτέλεσμα της διαίρεσης να φορτωθεί στον τελικό καταχωρητή και να βγει στην έξοδο.



Εικόνα 33: Διάγραμμα καταστάσεων του υποσυστήματος "Compute RTT".

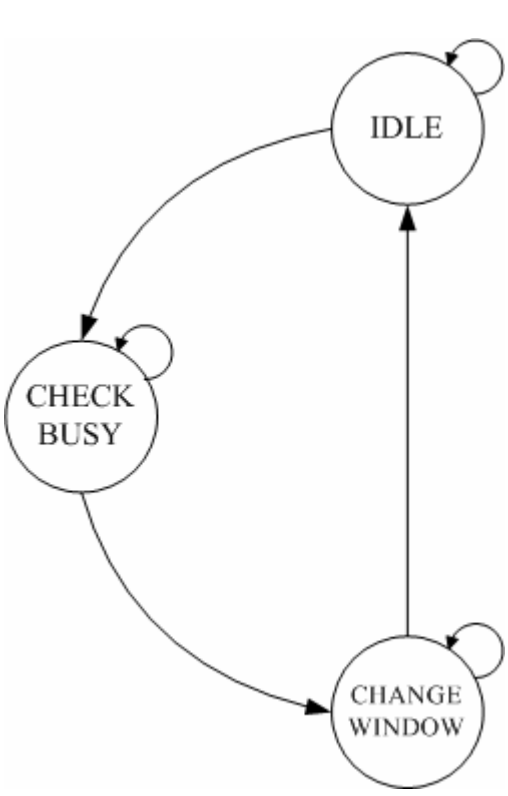
4.13 Το "Change Window Size"

Το υποσύστημα "Change Window Size" είναι υπεύθυνο για την αρχικοποίηση του παραθύρου συμφόρησης σε περίπτωση timeout από τον idle timer. Η τεχνική αυτή είναι πολύ χρήσιμη, διότι όταν μία σύνδεση παραμένει ανενεργή για μεγάλο χρονικό διάστημα, τότε οι μεταβολές που συμβαίνουν στο δίκτυο περνάνε απαρατήρητες, με αποτέλεσμα, αρκετά συχνά, όταν έρθει η ώρα για μετάδοση, το μεγάλο παράθυρο συμφόρησης σε συνδυασμό με την μεγάλη κινητικότητα στο δίκτυο, να οδηγούν σε απώλειες και εκ των υστέρων, σε αρχικοποίηση του παραθύρου.

Η εντολή που δραστηριοποιεί το υποσύστημα, είναι η εντολή "Change Window Size" και ανήκει στο σύνολο των εντολών της ενδιάμεσης FIFO, που θα

δούμε στην επόμενη παράγραφο. Παρόλ' αυτά το σύστημα δεν έχει πρόσβαση στην ενδιάμεση FIFO. Αντίθετα, ενεργοποιείται με βάση σήματα που έρχονται από το υποσύστημα αποστολής. Το ίδιο υποσύστημα διαβάζει την εντολή από την ενδιάμεση FIFO και παρέχει στο "Change Window Size" την κατάλληλη τιμή του SocketID για την δημιουργία έγκυρης διεύθυνσης για πρόσβαση στο socket table.

Το υποσύστημα υλοποιείται με μία απλή μηχανή πεπερασμένων καταστάσεων που παρουσιάζεται στην εικόνα 33. Στην κατάσταση "IDLE" παραμένει σε ετοιμότητα μέχρι να ενεργοποιηθεί από το υποσύστημα αποστολής. Στην κατάσταση "CHECK BUSY" γίνεται έλεγχος, προκειμένου να διαπιστωθεί ότι κανένα άλλο υποσύστημα δεν χρησιμοποιεί το socket table (από το port A), ενώ, στην κατάσταση "CHANGE WINDOW" το υποσύστημα προσπελαύνει το socket table, στην διεύθυνση που έχει σχηματίσει από το SocketID και αλλάζει την τιμή του RTT.



Εικόνα 34: Το διάγραμμα καταστάσεων του συστήματος "Change Window Size".

4.14 Η "Ενδιάμεση FIFO" (Intermediate FIFO)

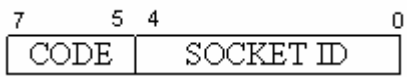
Το υποσύστημα της ενδιάμεσης FIFO «δομικά» δεν παρουσιάζει ιδιαίτερο ενδιαφέρον, καθώς, αποτελείται από μία FIFO πλάτους οκτώ και βάθους δεκαέξι θέσεων. Η μόνη επιπλέον λογική που εφαρμόστηκε είναι κάποιες πύλες για την δημιουργία του σήματος "busy".

Ωστόσο, από λειτουργική άποψη, παρουσιάζει ιδιαίτερο ενδιαφέρον, διότι σε αυτήν αποθηκεύεται ένας σχετικά μεγάλος αριθμός εντολών που δημιουργούνται εσωτερικά του συστήματος TCP. Οι εντολές, αυτές είναι:

- Retransmit Fin Segment: Δημιουργείται μετά από time out στον retransmission timer και προκαλεί την επαναμετάδοση ενός πακέτου FIN.
- Retransmit Data: Αντίστοιχα με τη προηγούμενη, δημιουργείται μετά από time out του retransmission timer και προκαλεί την επαναμετάδοση πακέτων DATA.
- Initialize Congestion Window: Δημιουργείται μετά από time out στον Idle timer και προκαλεί την δράση του υποσυστήματος "Change Window Size" για αρχικοποίηση του Congestion Window μίας σύνδεσης.
- Send Probe: Δημιουργείται μετά από time out στον persistence timer και προκαλεί την αποστολή ενός probe πακέτου. Το πακέτο probe είναι ένα πακέτο που δεν φέρει δεδομένα και η λήψη του προκαλεί κάποιο ack, από όπου ο αποστολέας ενημερώνεται για το μέγεθος του παραθύρου του δέκτη.
- Send Next Flight Size: Δημιουργείται από το υποσύστημα λήψης προκειμένου να ενημερώσει το σύστημα αποστολής ότι οι αναμενόμενες επιβεβαιώσεις έχουν ληφθεί και μπορεί να σταλεί το επόμενο «παράθυρο» πακέτων.
- Send Plain Ack: Δημιουργείται από το υποσύστημα λήψης προκειμένου να ενημερώσει το υποσύστημα αποστολής για αποστολή απλής επιβεβαίωσης.
- Send Packets After Last Ack: Ίσως η πιο «επώδυνη» εντολή καθώς ενημερώνει το υποσύστημα αποστολής για απώλεια πακέτων και λήψη τριπλότυπης επιβεβαίωσης. Πηγή της εντολής αυτής, φυσικά, είναι το υποσύστημα λήψης, ενώ, το υποσύστημα αποστολής που θα λάβει την

εντολή αυτή, είναι υποχρεωμένο να ξεκινήσει την επαναμετάδοση όλων των πακέτων από την τριπλότυπη επιβεβαίωση και μετά, άμεσα.

Μία εντολή για την ενδιάμεση FIFO σχηματίζεται σύμφωνα με την εικόνα 34: τα πρώτα τρία most significant bits αποτελούν τον κωδικό της εντολής (πεδίο “Code”), ενώ τα πέντε τελευταία αποτελούν το SocketID της σύνδεσης στην οποία αναφέρεται η συγκεκριμένη εντολή. Στον πίνακα 11 συνοψίζονται όλες οι εντολές σε συνάρτηση με την κωδικοποίησή τους.



Εικόνα 35: Η δομή μίας "ενδιάμεσης εντολής".

ΚΩΔΙΚΟΣ	**ΣΗΜΑΣΙΑ**
100	Retransmit FIN
101	Retransmit DATA
110	Initialize Congestion Window
111	Send Probe Segment
010	Send Next Flight Size
001	Send Plain Ack
011	Send Packets After Last Ack

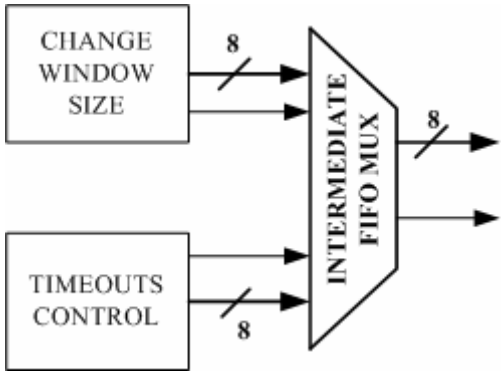
Πίνακας 11: Οι κωδικοί των "ενδιάμεσων εντολών".

4.15 Λοιπά επιμέρους υποσυστήματα

Στις προηγούμενες παραγράφους παρουσιάστηκαν τα πιο βασικά επιμέρους υποσυστήματα που απαρτίζουν το υποσύστημα TCP. Ωστόσο, στην σχεδίαση χρησιμοποιήθηκαν και κάποια πολύ μικρότερα, η λειτουργία των οποίων είναι απλή και δεν παίζει θεμελιώδη ρόλο στην συνολική λειτουργία του υποσυστήματος. Τα «μικρά» αυτά υποσυστήματα αναφέρονται σε αυτήν την παράγραφο περιληπτικά.

4.15.1 Ο "πολυπλέκτης" της "ενδιάμεσης FIFO"

Η λέξη πολυπλέκτης αναφέρεται εντός εισαγωγικών για τους ίδιους λόγους που το υποσύστημα "Socket Table Combo Mux" παρουσιάστηκε «ξεχωριστά» και δεν ονομάστηκε πολυπλέκτης. Το εν λόγω υποσύστημα είναι ασύγχρονο και ελέγχει την πρόσβαση στην ενδιάμεση FIFO «διασταυρώνοντας» τα σήματα Data In και Enqueue που προέρχονται από τα υποσυστήματα "Timeouts Control" και "Change Window". Η λειτουργία του φαίνεται παραστατικά στην εικόνα 35.



Εικόνα 36: Η λειτουργία του "Intermediate FIFO Mux".

4.15.2 Οι "Send Arguments List"& "Receive Arguments List"

Τόσο η "Send Arguments List" όσο και η "Receive Arguments List" υλοποιούνται με μία FIFO εύρους δεκαέξι bits και βάθους δεκαέξι θέσεων. Χρησιμοποιούνται για την φύλαξη των ορισμάτων των εντολών ή των απαντήσεων της σχεδίασης. Το "Enqueue" της "Send Arguments List" και αντίστροφα, το "Dequeue" της "Receive Arguments List" ελέγχονται εξωτερικά του υποσυστήματος TCP.

4.15.3 Η "Decouple FIFO"

Η "Decouple FIFO" (FIFO απόζευξης) χρησιμοποιείται από το υποσύστημα λήψης πλαισίων TCP ως «προσωρινή μνήμη» για την αποθήκευση των δεδομένων του εισερχόμενου πακέτου, μέχρι να γίνουν οι απαραίτητοι έλεγχοι και ενέργειες που πριν την αποδοχή του. Έχει εύρος οκτώ bits και βάθος τριάντα δύο θέσεων. Το "Enqueue" ελέγχεται από το control του υποσυστήματος "Receive", το "Read Write TCB", ενώ το "Dequeue" από το "Fill Receive Memory".

Κεφάλαιο 5: Δοκιμή και Πιστοποίηση του "Open TCP/IP Core"

Βασικό και αναπόσπαστο τμήμα της παραγωγής κάθε προϊόντος, είτε αυτό ανήκει στην κατηγορία του software, είτε στην κατηγορία του hardware αποτελεί η Δοκιμή και η Πιστοποίηση. Στο στάδιο αυτό, λαμβάνει χώρα ο λεπτομερής έλεγχος της εφαρμογής με κριτήρια, όχι μόνο της σωστής λειτουργίας αλλά και άλλων, όπως το κόστος της λειτουργίας κλπ.

Μεταφέροντας τον συλλογισμό μας, στον πραγματικό χώρο των επιχειρήσεων, η σωστή δοκιμή και πιστοποίηση είναι έννοια που πάντα απασχολούσε και συνεχίζει να απασχολεί τόσο τους σχεδιαστές, όσο και τις ίδιες τις εταιρίες, οι οποίες από την πλευρά τους ζητάνε το μέγιστο κέρδος από κάθε νέα εφαρμογή. Η σωστή δοκιμή και πιστοποίηση για μία εταιρία, η οποία περιμένει το νέο της προϊόν να βρει απήχηση στους χρήστες, εξασφαλίζει κατά κύριο λόγο, σωστή, αδιάκοπη και με ελάχιστες αστοχίες λειτουργία. Επιπλέον, σημαίνει λιγότερο κόστος στην μετέπειτα υποστήριξη του προϊόντος και περισσότερος χρόνος παραμονής στον χώρο της αγοράς. Συμπερασματικά, η Δοκιμή και η Πιστοποίηση είναι εκείνο το στάδιο παραγωγής, που δίνει την σωστή ώθηση σε κάθε προϊόν λίγο πριν την έξοδό του στον χώρο της αγοράς.

Επιστρέφοντας την συζήτησή και πάλι στην σχεδίασή μας, η έννοια "Δοκιμή και Πιστοποίηση" σημαίνει σωστή και λεπτομερής προσομοίωση. Η σχεδίαση, που αναλύθηκε στα προηγούμενα κεφάλαια, πέρασε από διαδικασίες "Synthesis" και "Place And Route" για αποτύπωση στο ολοκληρωμένο κύκλωμα αναδιατασσόμενης λογικής FPGA, Virtex 2 XC2V8000-4FF1517, της εταιρίας Xilinx [38] με χρήση εργαλείων του προγράμματος Simplify Pro 7.3.3 [42]. Οι προσομοιώσεις έγιναν με την χρήση του προγράμματος ModelSim 5.7f. [43] Στις παραγράφους που ακολουθούν θα παρουσιαστούν οι προσομοιώσεις που έγιναν, καθώς, και όλες οι απαραίτητες λεπτομέρειες από αυτό το στάδιο υλοποίησης.

5.1 Οι προσομοιώσεις των Hardware Σχεδιάσεων

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. Η διαδικασία της προσομοίωσης διακρίνεται σε Functional, Post Synthesis και Post Place & Route και παρουσιάζεται στις υποπαραγράφους που ακολουθούν. Επίσης στην εικόνα 36 παρουσιάζεται η διαδικασία διαδοχικών προσομοιώσεων.

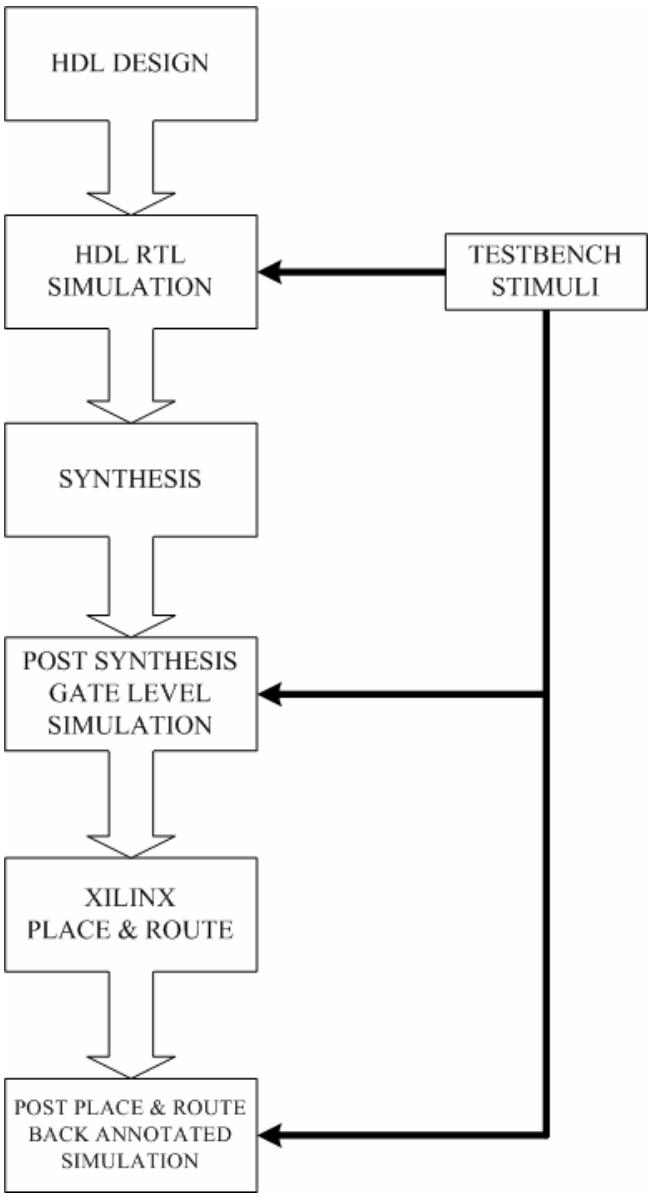
5.1.1 Προσομοίωση Function.

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.

5.1.2 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 Simplify Pro 7.3.3 της εταιρείας Simplicity. 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, όσον αφορά τα επίπεδα λογικής που χρησιμοποιούνται.



Εικόνα 37: Η διαδικασία των διαδοχικών προσομοιώσεων

5.1.3 Προσομοίωση Post Place & Route

Μετά τη διαδικασία Synthesis λαμβάνει χώρα η διαδικασία Place & Route. 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 πλησιάζει πάρα πολύ τα δεδομένα της πραγματικής κατασκευής και τα αποτελέσματά της διαβεβαιώνουν ότι η απεικονιζόμενη σχεδίαση έχει σχεδόν 100% πιθανότητα να δουλέψει σε πραγματικό hardware.

5.1.4 Παροχή τιμών στις εισόδους της σχεδίασης

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 των τιμών επιτυγχάνεται με δύο τρόπους: είτε με την δημιουργία ενός αρχείου macro (*.do, *.tcl) στο οποίο περιγράφονται οι τιμές των σημάτων συναρτήσει του χρόνου, είτε με την δημιουργία ενός test bench.

Το test bench γράφει τις τιμές στα σήματα εισόδου είτε γεννώντας αυτές on the fly με κάποιο τρόπο (τυχαίο, ή με κάποια συνάρτηση), είτε λαμβάνοντάς τις από κάποιο αρχείο με διανύσματα, που περιέχουν προεπιλεγμένα δεδομένα.

5.1.5 Παρακολούθηση των τιμών των εξόδων της σχεδίασης

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. 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.

5.1.6 Η προσομοίωση της παρούσας σχεδίασης

Η προσομοίωση του συστήματος ήταν μία από τις πολλές διαδικασίες που λάμβαναν χώρα σχεδόν παράλληλα με την σχεδίαση. Ο λόγος ο οποίος ωθεί στις επανειλημμένες προσομοιώσεις είναι η ανάγκη της δοκιμής για ορθή λειτουργία από πολύ νωρίς, πριν η σχεδίαση ολοκληρωθεί.

Επειδή, στόχος της διπλωματικής αυτής ήταν το υποσύστημα αποστολής/λήψης πλαισίων TCP, το συντριπτικό βάρος της προσομοίωσης έπεσε στο υποσύστημα αυτό, δεδομένου ότι όλα τα υπόλοιπα υποσυστήματα είχαν περάσει επιτυχώς την διαδικασία προσομοίωσης, ήδη από τις προηγούμενες εκδόσεις. Στο υποσύστημα, όμως, αποστολής λήψης πλαισίων TCP, που κατά κύριο λόγο υπέστη ολοκληρωτικό επανασχεδιασμό, η διαδικασία της προσομοίωσης έγινε με μεγάλη λεπτομέρεια. Κάθε επιμέρους υποσύστημα εξετάστηκε προσεκτικά με χρήση αρχείων macro και έξοδο που απεικονίζονταν με την βοήθεια κυματομορφών. Μετά την ολοκλήρωση όλων των επιμέρους υποσυστημάτων, το υποσύστημα αποστολής/λήψης πλαισίων εξετάστηκε τόσο με χρήση της προηγούμενης μεθόδου όσο και test bench.

Το test bench που χρησιμοποιήθηκε για την δοκιμή του υποσυστήματος TCP, δημιουργήθηκε σε VHDL. Οι τιμές των εισόδων δημιουργούνταν τόσο on the fly από το ίδιο το test bench, όσο και από αρχεία τα οποία προορίζονταν για την χρήση αυτή. Οι τιμές των εξόδων δοκιμάστηκαν με πολύ μεγάλη λεπτομέρεια και προσοχή με δύο τρόπους. Πρώτον, με ειδικές συναρτήσεις του test bench οι οποίες παρακολουθούσαν τα σήματα και έγραφαν διαγνωστικά μηνύματα σε αρχεία εξόδου και δεύτερον, με χρήση κυματομορφών παράλληλα με την λειτουργία των monitors οι οποίες επέτρεπαν την εξέταση κάθε περίπτωσης σε βάθος (ανάλυση) κύκλου ρολογιού για κάθε επιμέρους υποσύστημα. Τέλος οι έξοδοι προς άλλα υποσυστήματα όπως το CRC και μνήμες όπως η TxRAM και η Receive Memory καταγράφονταν χωρίς κανένα έλεγχο σε αρχεία.

Όσον αφορά στα «περιφερειακά» ως προς το TCP υποσυστήματα οι έλεγχοι που έγιναν, είχαν να κάνουν με λήψη και αποστολή αιτήσεων ARP, αποστολές ICMP echo request, αποστολή και λήψη πακέτων UDP. Όλοι ολοκληρώθηκαν με επιτυχία,

από την πρώτη κιόλας δοκιμή, γεγονός που δεν μας εξέπληξε αφού, τα παραπάνω υποσυστήματα παρέμειναν ως είχαν και στην έκδοση αυτή της σχεδίασης.

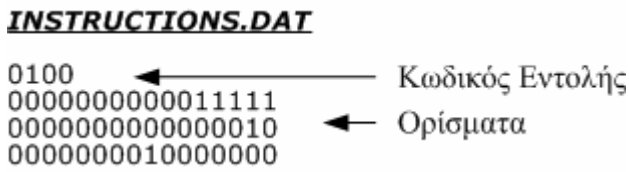
Στην επόμενη παράγραφο, θα εστιάσουμε την προσοχή μας στο test bench που δημιουργήθηκε προκειμένου να δοκιμαστεί το υποσύστημα TCP ενώ στις αμέσως επόμενες θα μελετήσουμε κάποιες από τις πολλές δοκιμασίες στις οποίες υπεβλήθη.

5.2 Το Test Bench του υποσυστήματος TCP

Όπως αναφέρθηκε παραπάνω, το μέγιστο βάρος όσον αφορά στις προσομοιώσεις για την επαλήθευση της σωστής λειτουργίας της τέταρτης έκδοσης της σχεδίασης, δόθηκε στο υποσύστημα που είχε αλλάξει, στο TCP. Στην παράγραφο αυτή θα γνωρίσουμε τις «ιδιαιτερότητες» του εν λόγω test bench, ενώ, στην αμέσως επόμενη θα μελετήσουμε τα βασικότερα αποτελέσματα των δοκιμών που προήλθαν από την χρήση του.

Τα περισσότερα εκ των σημάτων εισόδου του υποσυστήματος αποστολής/λήψης πακέτων TCP στο test bench αλλάζουν με «αυτόματο» τρόπο χρησιμοποιώντας είτε αρχεία εισόδου, είτε συναρτήσεις. Όμως, κάποια συνέχισαν να αλλάζουν με «χειροκίνητο» τρόπο και αυτό διότι το χαρακτηριστικό αυτό διευκόλυνε τον έλεγχο του συστήματος. Τα σήματα που δεν αλλάζουν με αυτόματο τρόπο είναι το Reset, τα acknowledgements που έρχονται από το control ως απαντήσεις σε αιτήσεις εγγραφής (write request) κ.α.

Το ρολόι δημιουργείται από μία ειδική διεργασία με περίοδο 100 nanoseconds. Οι εντολές δίνονται κωδικοποιημένες, έτσι ώστε να δοκιμαστεί και το Instruction Decode. Το σύνολο των εντολών αυτών φυλάγονται σε ένα αρχείο (το “instructions.dat”) με την μορφή που φαίνεται στην εικόνα 37. Σημειώνεται ότι στο ίδιο αρχείο μπορούν να συνυπάρχουν και περισσότερες της μίας εντολής ταυτόχρονα και αυτό, διότι, το test bench είναι με τέτοιο τρόπο δομημένο ώστε να μπορεί να διακρίνει ποια είναι η εντολή, πόσα ορίσματα ακολουθούν κλπ.



Εικόνα 38: Οι διάταξη των εντολών και των αντίστοιχων ορισμάτων στο αρχείο "instructions.dat".

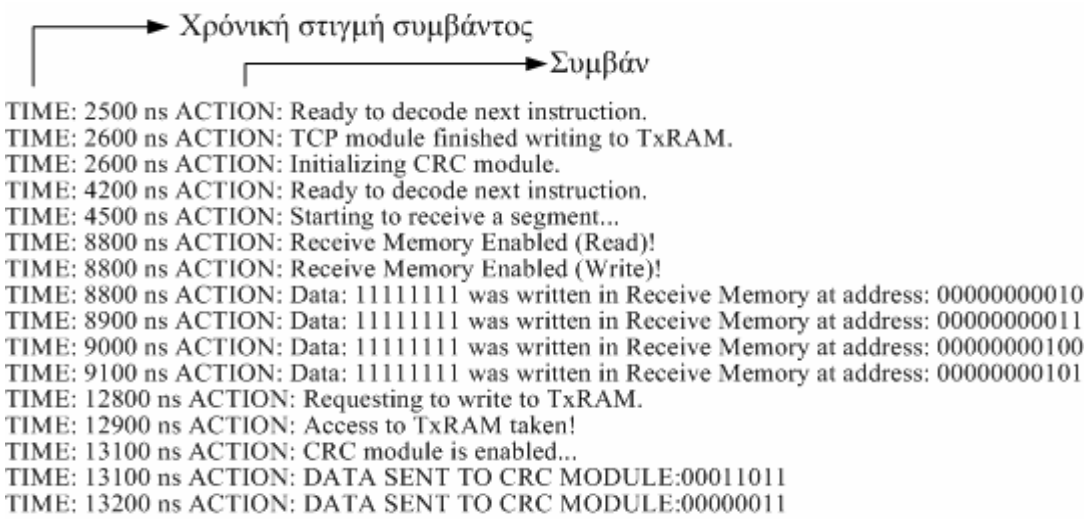
Τα δεδομένα της μνήμης αποστολής είναι αποθηκευμένα και διαβάζονται από το αρχείο “smem_data.dat”. Όμοια, τα δεδομένα που υποτίθεται ότι λαμβάνει το επιμέρους υποσύστημα λήψης, διαβάζονται από το αρχείο “in_data.dat”. Ουσιαστικά τα δεδομένα που περιέχει το αρχείο αυτό είναι τα εισερχόμενα πακέτα συμπληρωμένα με οκτώ γραμμές επιπλέον, τέσσερις εκ των οποίων φέρουν το IP του αποστολέα (με αυτήν την μορφή παρέχονται από το υποσύστημα IP).

Όσον αφορά στα monitors, το test bench που δημιουργήθηκε διαθέτει πέντε αρχεία στα οποία γράφονται αναλυτικά οι τιμές τνη πιο σημαντικών εξόδων του υποσυστήματος TCP. Τα αρχεία αυτά είναι τα “tcp_monitor.dat”, “crc.dat”, “txram.dat”, “smem.dat” και “rmem.dat”.

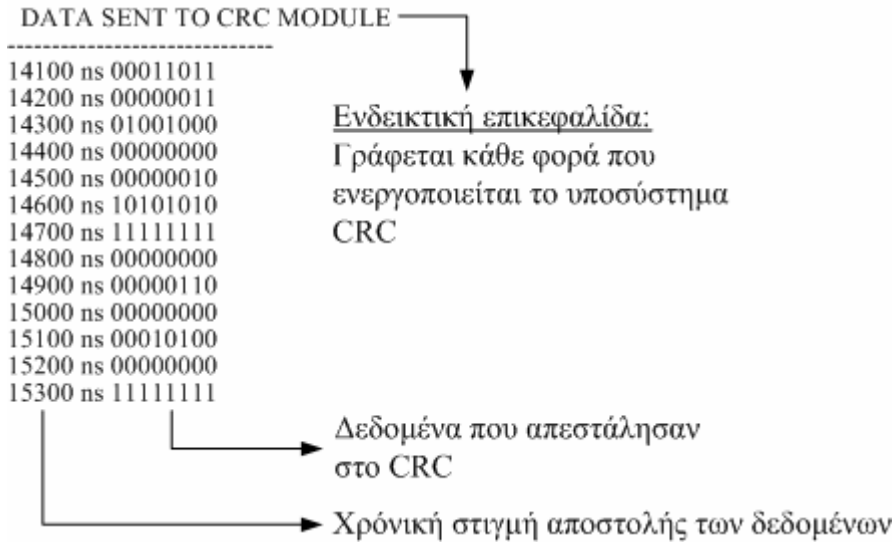
Στο αρχείο “tcp_monitor.dat” γράφονται οι πιο σημαντικές αναφορές της προσομοίωσης. Το test bench παρακολουθεί τις αλλαγές των εξόδων και γράφει στο συγκεκριμένο αρχείο σύντομα μηνύματα. Παράλληλα και πριν από κάθε μήνυμα, σημειώνεται ο χρόνος εναλλαγής του εκάστοτε σήματος. Στην εικόνα 38 παρουσιάζεται ένα μικρό δείγμα κάποιων εκ των εγγραφών του αρχείου “tcp_monitor.dat”.

Στο αρχείο “crc.dat” γράφονται όλα τα δεδομένα που αποστέλλονται από το υποσύστημα TCP στο υποσύστημα CRC. Και στο αρχείο αυτό, πριν από κάθε εγγραφή αναφέρεται η συγκεκριμένη χρονική στιγμή. Στιγμιότυπο από τις εγγραφές του “crc.dat” φαίνεται στην εικόνα 39.

Στο αρχείο “txram.dat” γράφονται τα δεδομένα που αποστέλλονται από το υποσύστημα TCP στην μνήμη TxRAM. Σε κάθε εγγραφή υπάρχει ο αντίστοιχος χρόνος, καθώς και η διεύθυνση στην οποία προορίζεται να αποθηκευτούν τα δεδομένα. Στιγμιότυπο των εγγραφών στο αρχείο “txram.dat” φαίνεται στην εικόνα 40.



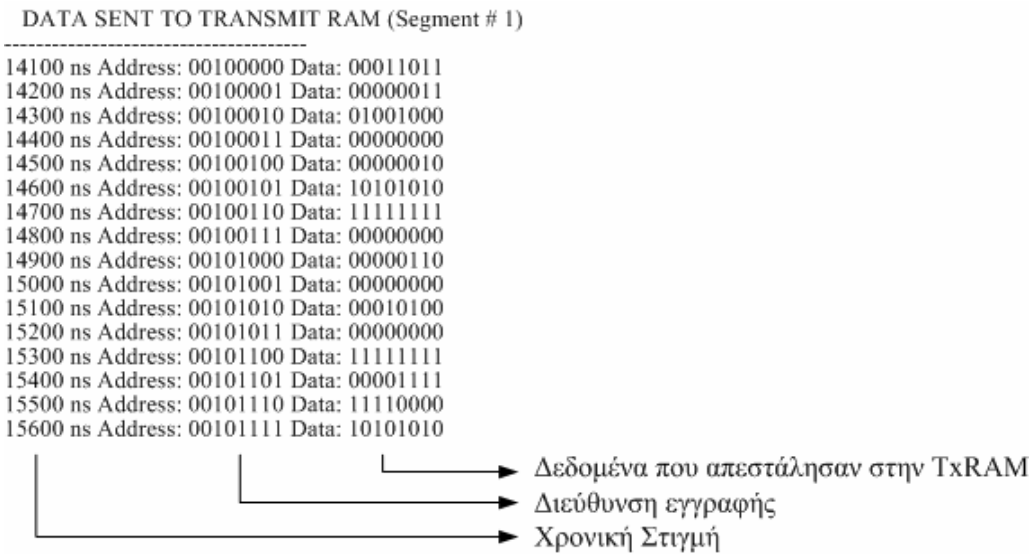
Εικόνα 39: Στιγμιότυπο εγγραφών στο αρχείο "tcp_monitor.dat".



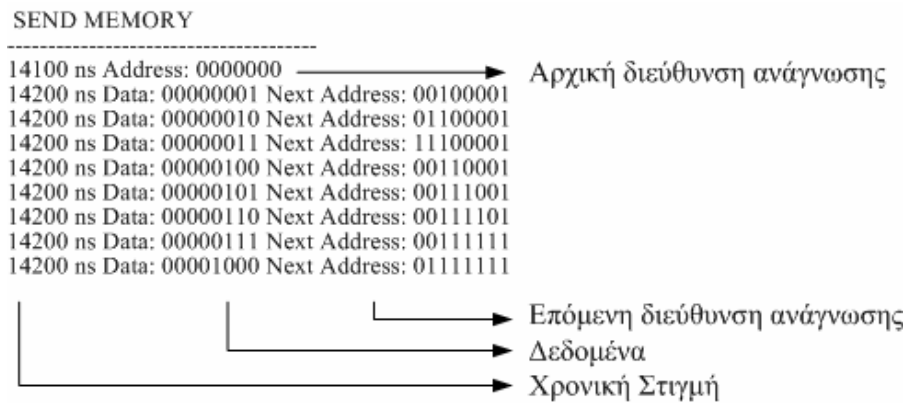
Εικόνα 40: Στιγμιότυπο εγγραφών στο αρχείο "crc.dat".

Στο αρχείο “smem.dat” αποθηκεύονται οι προσβάσεις του υποσυστήματος στην μνήμη αποστολής. Όπως φαίνεται και στην εικόνα 41, κάθε εγγραφή αναφέρει χρόνο πρόσβασης, διεύθυνση και δεδομένα.

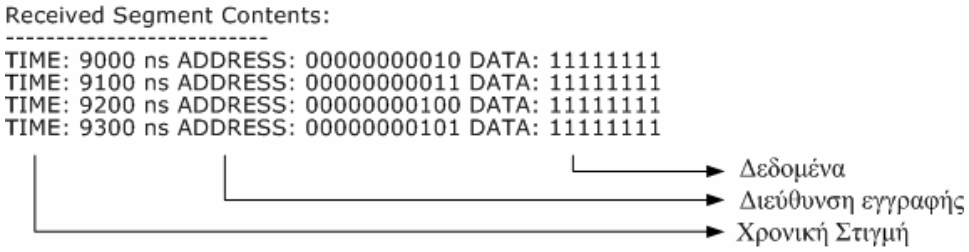
Τέλος στο αρχείο “rmem.dat” αποθηκεύονται τα δεδομένα από τα εισερχόμενα πακέτα. Κάθε εγγραφή αναφέρει την διεύθυνση αποθήκευσης, τα δεδομένα και τον χρόνο πρόσβασης (εικόνα 42).



Εικόνα 41: Στιγμιότυπο εγγραφών στο αρχείο "txram.dat".



Εικόνα 42: Στιγμιότυπο εγγραφών στο αρχείο "smem.dat".



Εικόνα 43: Στιγμιότυπο εγγραφών του αρχείου "rmem.dat".

5.3 Τα «σενάρια» δοκιμών του υποσυστήματος TCP.

Προκειμένου να διαπιστωθεί η ομαλή λειτουργία του υποσυστήματος TCP, επινοήθηκαν αρκετά σενάρια, τα οποία «έτρεξαν» με την βοήθεια του test bench, που αναλύθηκε στην προηγούμενη παράγραφο. Ο όρος σενάριο χρησιμοποιείται επειδή

ορισμένες διαδικασίες, που ακολουθήθηκαν για τον έλεγχο, είναι πιο πολύπλοκες από το τρέξιμο μίας απλής εντολής. Τα αποτελέσματα των δοκιμών αποθηκεύτηκαν στα monitors για ανάλυση. Παράλληλα όμως, χρησιμοποιήθηκαν και οι κυματομορφές προκειμένου να ελεγχθούν σήματα και ενδότερα του υποσυστήματος TCP σε επίπεδο κύκλου ρολογιού. Παρακάτω παρουσιάζονται κάποια από τα πιο σημαντικά σενάρια που κλίθηκε να «ακολουθήσει» το TCP.

- Αποδοχή αίτησης για σύνδεση μέσω της εντολής Bind: Αποδεχόμαστε κάποια σύνδεση μέσω της εντολής bind, ανοίγουμε την κατάλληλη εγγραφή στο socket table και αποστέλλουμε SYN Segment. Η έξοδος στο "tcp_monitor.dat" είναι η ακόλουθη:

```
TIME: 0 ns ACTION: Reset module...
TIME: 2600 ns ACTION: Initializing CRC module.
TIME: 4900 ns ACTION: Ready to decode next instruction.
TIME: 7000 ns ACTION: Requesting to write to TxRAM.
TIME: 7100 ns ACTION: Access to TxRAM taken!
=== Preparing to send 1st segment of flightsize to TxRAM or plain ack ===
TIME: 9200 ns ACTION: CRC module is enabled...
TIME: 9200 ns ACTION: DATA SENT TO CRC MODULE:10010011
...
TIME: 10300 ns ACTION: DATA SENT TO CRC MODULE:00010100
TIME: 10400 ns ACTION: Tx_RAM enabled  for write...
TIME: 10400 ns ACTION: DATA SENT TO TxRAM: 00000000, ADDRESS: 00101011
....
TIME: 12000 ns ACTION: DATA SENT TO CRC MODULE:00000000
TIME: 12100 ns ACTION: CRC OUTPUT: 0000111100001111 WAS WRITTEN IN TxRAM
(Adresses 3B and 3C).
...
TIME: 13900 ns ACTION: TCP module finished writing to TxRAM.
```

- Εντολή Open_Active_Socket: Η σύνδεση είναι CLOSED και το επίπεδο εφαρμογών στέλνει την εντολή Open_Active_Socket. Η σχεδίαση στέλνει SYN και μεταβαίνει στην SYN-SEND. Δεχόμαστε επιβεβαίωση του SYN-ACK και έτσι η σύνδεση γίνεται ESTABLISHED και αποστέλλεται το σχετικό μήνυμα επιτυχίας στην εφαρμογή μαζί με τα σχετικά ορίσματα. Η έξοδος στο "tcp_monitor.dat" είναι η ακόλουθη:

```
TIME: 0 ns ACTION: Reset module...
TIME: 2600 ns ACTION: Initializing CRC module.
TIME: 4100 ns ACTION: Requesting arp for address: 00000000111111110000000000110001
TIME: 4200 ns ACTION: Arp Request acknowledge for IP adress
00000000111111110000000000110001 recieved!
TIME: 5600 ns ACTION: Ready to decode next instruction.
TIME: 7700 ns ACTION: Requesting to write to TxRAM.
TIME: 7800 ns ACTION: Access to TxRAM taken!
=== Preparing to send 1st segment of flightsize to TxRAM ===
TIME: 9800 ns ACTION: CRC module is enabled...
TIME: 9800 ns ACTION: DATA SENT TO CRC MODULE:10010011
TIME: 9900 ns ACTION: DATA SENT TO CRC MODULE:00011011
…
TIME: 10900 ns ACTION: DATA SENT TO CRC MODULE:00010100
```

```
TIME: 11000 ns ACTION: Tx_RAM enabled  for write...
TIME: 11000 ns ACTION: DATA SENT TO TxRAM: 00000000, ADDRESS: 00101011
TIME: 11000 ns ACTION: DATA SENT TO CRC MODULE:00000000
…
TIME: 12700 ns ACTION: CRC OUTPUT: 0000111100001111 WAS WRITTEN IN TxRAM
(Adresses 3B and 3C).
TIME: 12700 ns ACTION: DATA SENT TO TxRAM: 00000000, ADDRESS: 00111100
TIME: 12700 ns ACTION: DATA SENT TO CRC MODULE:00000000
…
TIME: 13100 ns ACTION: Tx_RAM enabled  for write...
TIME: 13100 ns ACTION: DATA SENT TO TxRAM: 00001111, ADDRESS: 00111011
TIME: 13200 ns ACTION: CRC OUTPUT: 0000111100001111 WAS WRITTEN IN TxRAM
(Adresses 3B and 3C).
TIME: 13200 ns ACTION: Initializing CRC module.
TIME: 13200 ns ACTION: DATA SENT TO TxRAM: 00001111, ADDRESS: 00111100
TIME: 14500 ns ACTION: TCP module finished writing to TxRAM.
TIME: 15500 ns ACTION: Starting to receive a segment...
TIME: 20200 ns ACTION: Client socket opened successfully! Socket ID:11111
TIME: 22600 ns ACTION: Requesting to write to TxRAM.
TIME: 22700 ns ACTION: Access to TxRAM taken!
=== Preparing to send 1st segment of flightsize to TxRAM ===
TIME: 22900 ns ACTION: CRC module is enabled...
TIME: 22900 ns ACTION: DATA SENT TO CRC MODULE:00011011
TIME: 23000 ns ACTION: DATA SENT TO CRC MODULE:00000011
…
TIME: 27500 ns ACTION: TCP module finished writing to TxRAM.
```

- Εντολή Open_Passive_Socket: Το επίπεδο εφαρμογών στέλνει την εντολή Open_Passive_Socket. Το εν λόγω socket που είναι κλειστό, ανοίγει (στην κατάσταση LISTEN) και επιστρέφεται μήνυμα επιτυχίας. Η έξοδος στο "tcp_monitor.dat" είναι η ακόλουθη:

```
TIME: 0 ns ACTION: Reset module...
TIME: 2600 ns ACTION: Initializing CRC module.
TIME: 3900 ns ACTION: 0001: Server socket opened successfully! Socket ID: 01111
TIME: 4000 ns ACTION: Ready to decode next instruction.
```

- Αποστολή Segments (σενάριο 1): Η σύνδεση είναι σε κατάσταση ESTABLISHED και διαθέτει παράθυρο συμφόρησης μεγέθους ίσο με τρία πακέτα. Αποστέλλονται τρία πακέτα και αναμένεται επιβεβαίωση. Η έξοδος στο "tcp_monitor.dat" είναι η ακόλουθη:

```
TIME: 0 ns ACTION: Reset module...
TIME: 2600 ns ACTION: TCP module finished writing to TxRAM.
TIME: 2600 ns ACTION: Initializing CRC module.
TIME: 4200 ns ACTION: Ready to decode next instruction.
TIME: 5800 ns ACTION: Ready to decode next instruction.
TIME: 8000 ns ACTION: Requesting to write to TxRAM.
TIME: 8100 ns ACTION: Access to TxRAM taken!
===Preparing to send 1st segment of flightsize to TxRAM ===
TIME: 10200 ns ACTION: CRC module is enabled...
TIME: 10200 ns ACTION: DATA SENT TO CRC MODULE:10010011
TIME: 10300 ns ACTION: DATA SENT TO CRC MODULE:00011011
...
TIME: 11400 ns ACTION: Tx_RAM enabled  for write...
TIME: 11400 ns ACTION: DATA SENT TO TxRAM: 00000000, ADDRESS: 00101011
```

```
...
TIME: 13100 ns ACTION: CRC OUTPUT: 0000111100001111 WAS WRITTEN IN TxRAM
(Adresses 3B and 3C).
...
TIME: 13400 ns ACTION: Send Memory Enabled.
TIME: 13400 ns ACTION: READING FROM SEND MEMORY ADDRESS: 00000110001
...
TIME: 13600 ns ACTION: READING FROM SEND MEMORY ADDRESS: 00000110011
TIME: 13600 ns ACTION: CRC module is enabled...
TIME: 13600 ns ACTION: Tx_RAM enabled  for write...
TIME: 13600 ns ACTION: DATA SENT TO TxRAM: 11111111, ADDRESS: 00111111
TIME: 13600 ns ACTION: DATA SENT TO CRC MODULE:11111111
...
TIME: 32300 ns ACTION: DATA SENT TO TxRAM: 00001111, ADDRESS: 00111011
TIME: 32400 ns ACTION: CRC OUTPUT: 0000111100001111 WAS WRITTEN IN TxRAM
(Adresses 3B and 3C).
TIME: 32400 ns ACTION: Initializing CRC module.
TIME: 32400 ns ACTION: DATA SENT TO TxRAM: 00001111, ADDRESS: 00111100
TIME: 34400 ns ACTION: Requesting to rewrite to TxRAM.
TIME: 34500 ns ACTION: Access to TxRAM taken (rewrite)!
=== Preparing to send next segment of the same flightsize to TxRAM (Segment # 2) ===
TIME: 36700 ns ACTION: CRC module is enabled...
TIME: 36700 ns ACTION: DATA SENT TO CRC MODULE:00000011
...
TIME: 37700 ns ACTION: Tx_RAM enabled  for write...
TIME: 37700 ns ACTION: DATA SENT TO TxRAM: 00000000, ADDRESS: 00101011
TIME: 37700 ns ACTION: DATA SENT TO CRC MODULE:00000000
...
TIME: 39400 ns ACTION: CRC OUTPUT: 0000111100001111 WAS WRITTEN IN TxRAM
(Adresses 3B and 3C).
...
TIME: 39700 ns ACTION: Send Memory Enabled.
TIME: 39700 ns ACTION: READING FROM SEND MEMORY ADDRESS: 00011101100
....
TIME: 60700 ns ACTION: Requesting to rewrite to TxRAM.
TIME: 60800 ns ACTION: Access to TxRAM taken (rewrite)!
=== Preparing to send next segment of the same flightsize to TxRAM (Segment # 3) ===
TIME: 63000 ns ACTION: CRC module is enabled
...
TIME: 87300 ns ACTION: TCP module finished writing to TxRAM.
```

- Αποστολή Segments (σενάριο 2): Η σύνδεση είναι σε κατάσταση ESTABLISHED. Διαθέτουμε τέσσερα segments προς αποστολή και το παράθυρο συμφόρησης είναι ίσο με τρία. Στέλνονται τα τρία segments, λαμβάνεται επιβεβαίωση και έπειτα στέλνεται και το τέταρτο segment. Για το τελευταίο segment δεν λαμβάνεται ποτέ επιβεβαίωση και έτσι δημιουργείται time out, που οδηγεί σε διπλασιασμό του RTT και επαναμετάδοση. Τα περιεχόμενα του αρχείου "tcp_monitor.dat" είναι όμοια με εκείνα που έχουν παρουσιαστεί μέχρι στιγμής.
- Αποστολή Segments (σενάριο 3): Η σύνδεση είναι σε κατάσταση ESTABLISHED και το παράθυρο συμφόρησης έχει μέγεθος ίσο με τρία πακέτα. Διαθέτουμε ένα segment προς αποστολή. Μετά την αποστολή λαμβάνεται επιβεβαίωση και επιστρέφεται στο επίπεδο των εφαρμογών

μήνυμα επιτυχίας. Τα περιεχόμενα του αρχείου "tcp_monitor.dat" έχουν ως εξής:

```
TIME: 0 ns ACTION: Reset module...
TIME: 2600 ns ACTION: Initializing CRC module.
TIME: 4200 ns ACTION: Ready to decode next instruction.
TIME: 5900 ns ACTION: Ready to decode next instruction.
TIME: 8100 ns ACTION: Requesting to write to TxRAM.
TIME: 8100 ns ACTION: Access to TxRAM taken!
=== Preparing to send 1st segment of flightsize to TxRAM ===
TIME: 10300 ns ACTION: CRC module is enabled...
TIME: 10300 ns ACTION: DATA SENT TO CRC MODULE:10010011
...
TIME: 11500 ns ACTION: Tx_RAM enabled  for write...
TIME: 11500 ns ACTION: DATA SENT TO TxRAM: 00000000, ADDRESS: 00101011
TIME: 11500 ns ACTION: DATA SENT TO CRC MODULE:00000000
...
TIME: 13200 ns ACTION: CRC OUTPUT: 0000111100001111 WAS WRITTEN IN TxRAM
(Adresses 3B and 3C).
TIME: 13200 ns ACTION: DATA SENT TO TxRAM: 00000000, ADDRESS: 00111100
TIME: 13200 ns ACTION: DATA SENT TO CRC MODULE:00000000
...
TIME: 13500 ns ACTION: Send Memory Enabled.
TIME: 13500 ns ACTION: READING FROM SEND MEMORY ADDRESS: 00000110001
...
TIME: 13700 ns ACTION: CRC module is enabled...
TIME: 13700 ns ACTION: Tx_RAM enabled  for write...
TIME: 13700 ns ACTION: DATA SENT TO TxRAM: 11111111, ADDRESS: 00111111
TIME: 13700 ns ACTION: DATA SENT TO CRC MODULE:11111111
...
TIME: 32500 ns ACTION: CRC OUTPUT: 0000111100001111 WAS WRITTEN IN TxRAM
(Adresses 3B and 3C).
TIME: 32500 ns ACTION: Initializing CRC module.
TIME: 32500 ns ACTION: DATA SENT TO TxRAM: 00001111, ADDRESS: 00111100
TIME: 34800 ns ACTION: TCP module finished writing to TxRAM.
TIME: 90200 ns ACTION: Starting to receive a segment...
TIME: 94800 ns ACTION: 0011: Success Send! Socket ID: 11111
```

- Λήψη Segments (σενάριο 1): Το υποσύστημα TCP λαμβάνει ένα πακέτο το οποίο δεν φέρει ack και αποτελείται από 4 bytes δεδομένων. Μετά την λήψη του πακέτου, το υποσύστημα αποστολής ξεκινά την αποστολή ack. Παράλληλα επιστρέφεται μήνυμα επιτυχίας στο επίπεδο των εφαρμογών. Τα περιεχόμενα του αρχείου "tcp_monitor.dat" έχουν ως εξής:

```
TIME: 0 ns ACTION: Reset module...
TIME: 2600 ns ACTION: Initializing CRC module.
TIME: 4200 ns ACTION: Ready to decode next instruction.
TIME: 4500 ns ACTION: Starting to receive a segment...
TIME: 8800 ns ACTION: Receive Memory Enabled (Read)!
TIME: 8800 ns ACTION: Receive Memory Enabled (Write)!
TIME: 8800 ns ACTION: Data: 11111111 was written in Receive Memory at address:
00000000010
TIME: 8900 ns ACTION: Data: 11111111 was written in Receive Memory at address:
00000000011
TIME: 9000 ns ACTION: Data: 11111111 was written in Receive Memory at address:
00000000100
TIME: 9100 ns ACTION: Data: 11111111 was written in Receive Memory at address:
00000000101
TIME: 9700 ns ACTION: 0100: Success Receive! Socket ID: 11111
                      Buffer Offset: 0000000000000010
```

```
Byte Count: 0000000000000100
TIME: 13300 ns ACTION: Requesting to write to TxRAM.
TIME: 13900 ns ACTION: Access to TxRAM taken!
=== Preparing to send 1st segment of flightsize to TxRAM ===
TIME: 14100 ns ACTION: CRC module is enabled...
TIME: 14100 ns ACTION: DATA SENT TO CRC MODULE:00011011
...
TIME: 15200 ns ACTION: Tx_RAM enabled  for write...
TIME: 15200 ns ACTION: DATA SENT TO TxRAM: 00000000, ADDRESS: 00101011
TIME: 15200 ns ACTION: DATA SENT TO CRC MODULE:00000000
...
TIME: 16900 ns ACTION: CRC OUTPUT: 0000111100001111 WAS WRITTEN IN TxRAM
(Adresses 3B and 3C).
TIME: 16900 ns ACTION: DATA SENT TO TxRAM: 00000000, ADDRESS: 00111100
TIME: 16900 ns ACTION: DATA SENT TO CRC MODULE:00000000
...
TIME: 17400 ns ACTION: CRC OUTPUT: 0000111100001111 WAS WRITTEN IN TxRAM
(Adresses 3B and 3C).
TIME: 17400 ns ACTION: Initializing CRC module.
TIME: 17400 ns ACTION: DATA SENT TO TxRAM: 00001111, ADDRESS: 00111100
TIME: 18200 ns ACTION: TCP module finished writing to TxRAM.
```

- Λήψη Segments (σενάριο 2): Το σενάριο αυτό είναι σχεδόν πανομοιότυπο με το προηγούμενο, με την διαφορά ότι έχει μεγαλύτερο εύρος χρόνου. Αυτό σημαίνει ότι το υποσύστημα TCP εξετάζεται περισσότερο χρόνο με αποτέλεσμα να γίνεται αντιληπτό το idle time out λόγω αδράνειας της σύνδεσης. Το time out προκαλεί την άμεση αλλαγή του παραθύρου συμφόρησης. Στα δεδομένα του "tcp_monitor.dat", το time out δεν είναι «ορατό».
- Λήψη Segments (σενάριο 3): Λήψη δύο πακέτων, εκ των οποίων το δεύτερο είναι εκτός σειράς. Το σύστημα στέλνει επιβεβαίωση για το πρώτο και μετά την λήψη του δεύτερου ξαναστέλνει την ίδια. Τα περιεχόμενα του αρχείου "tcp_monitor.dat" έχουν ως εξής:

```
TIME: 0 ns ACTION: Reset module...
TIME: 2600 ns ACTION: Initializing CRC module.
TIME: 4200 ns ACTION: Ready to decode next instruction.
TIME: 4500 ns ACTION: Receiving and checking a new segment ...
TIME: 8900 ns ACTION: Receive Memory Enabled (Read)!
TIME: 8900 ns ACTION: Receive Memory Enabled (Write)!
TIME: 8900 ns ACTION: Data: 11111111 was written in Receive Memory at address:
00000000010
TIME: 9000 ns ACTION: Data: 11111111 was written in Receive Memory at address:
00000000011
TIME: 9100 ns ACTION: Data: 11111111 was written in Receive Memory at address:
00000000100
TIME: 9200 ns ACTION: Data: 11111111 was written in Receive Memory at address:
00000000101
TIME: 13300 ns ACTION: Requesting to write to TxRAM.
TIME: 13900 ns ACTION: Access to TxRAM taken!
=== Preparing to send 1st segment of flightsize to TxRAM or plain ack ===
TIME: 14100 ns ACTION: CRC module is enabled...
TIME: 14100 ns ACTION: DATA SENT TO CRC MODULE:00011011
...
TIME: 15200 ns ACTION: Tx_RAM enabled  for write...
```

```
TIME: 15200 ns ACTION: DATA SENT TO TxRAM: 00000000, ADDRESS: 00101011
TIME: 15200 ns ACTION: DATA SENT TO CRC MODULE:00000000
....
TIME: 17400 ns ACTION: CRC OUTPUT: 0000111100001111 WAS WRITTEN IN TxRAM
(Adresses 3B and 3C).
TIME: 17400 ns ACTION: Initializing CRC module.
TIME: 17400 ns ACTION: DATA SENT TO TxRAM: 00001111, ADDRESS: 00111100
TIME: 18200 ns ACTION: TCP module finished writing to TxRAM.
TIME: 20000 ns ACTION: Receiving and checking a new segment ...
TIME: 26900 ns ACTION: Requesting to write to TxRAM.
TIME: 28500 ns ACTION: Access to TxRAM taken!
=== Preparing to send 1st segment of flightsize to TxRAM or plain ack===
TIME: 28700 ns ACTION: CRC module is enabled...
.....
TIME: 29800 ns ACTION: Tx_RAM enabled  for write...
TIME: 29800 ns ACTION: DATA SENT TO TxRAM: 00000000, ADDRESS: 00101011
TIME: 29800 ns ACTION: DATA SENT TO CRC MODULE:00000000
...
TIME: 32000 ns ACTION: CRC OUTPUT: 0000111100001111 WAS WRITTEN IN TxRAM
(Adresses 3B and 3C).
TIME: 32000 ns ACTION: Initializing CRC module.
TIME: 32000 ns ACTION: DATA SENT TO TxRAM: 00001111, ADDRESS: 00111100
TIME: 32800 ns ACTION: TCP module finished writing to TxRAM.
```

- Λήψη Segments (σενάριο 4): Το υποσύστημα αποστολής έχει ολοκληρώσει την αποστολή μίας ροής σε κάποιον προορισμό. Ο ίδιος ο προορισμός στέλνει ένα πακέτο που περιέχει acknowledgement για τα σταλθέντα πακέτα και δεδομένα. Το υποσύστημα λήψης, «αναγκάζεται» να δώσει εντολή για plain ack επειδή το υποσύστημα αποστολής δεν στέλνει στην παρούσα στιγμή, γεγονός που απορρίπτει την μετάδοση του ack ως piggy back. Τα περιεχόμενα του αρχείου "tcp_monitor.dat" έχουν ως εξής:

```
TIME: 0 ns ACTION: Reset module...
....
TIME: 2600 ns ACTION: Initializing CRC module.
TIME: 4200 ns ACTION: Ready to decode next instruction.
TIME: 4500 ns ACTION: Receiving and checking a new segment ...
TIME: 8800 ns ACTION: Receive Memory Enabled (Read)!
TIME: 8800 ns ACTION: Receive Memory Enabled (Write)!
TIME: 8800 ns ACTION: Data: 11111111 was written in Receive Memory at address:
00000000010
TIME: 8900 ns ACTION: Data: 11111111 was written in Receive Memory at address:
00000000011
TIME: 9000 ns ACTION: Data: 11111111 was written in Receive Memory at address:
00000000100
TIME: 9100 ns ACTION: Data: 11111111 was written in Receive Memory at address:
00000000101
TIME: 9700 ns ACTION: 0100: Success Receive! Socket ID: 11111
                      Buffer Offset: 0000000000000010
                      Byte Count: 0000000000000100
TIME: 13800 ns ACTION: Requesting to write to TxRAM.
TIME: 13900 ns ACTION: Access to TxRAM taken!
==== Preparing to send 1st segment of flightsize to TxRAM or plain ack ===
TIME: 14100 ns ACTION: CRC module is enabled...
TIME: 14100 ns ACTION: DATA SENT TO CRC MODULE:00011011
....
TIME: 17000 ns ACTION: DATA SENT TO CRC MODULE:00000000
TIME: 17100 ns ACTION: DATA SENT TO TxRAM: 00000000, ADDRESS: 00111110
```

```
TIME: 17100 ns ACTION: DATA SENT TO CRC MODULE:00000000
TIME: 17300 ns ACTION: Tx_RAM enabled  for write...
TIME: 17300 ns ACTION: DATA SENT TO TxRAM: 00001111, ADDRESS: 00111011
TIME: 17400 ns ACTION: CRC OUTPUT: 0000111100001111 WAS WRITTEN IN TxRAM
(Adresses 3B and 3C).
TIME: 17400 ns ACTION: Initializing CRC module.
TIME: 17400 ns ACTION: DATA SENT TO TxRAM: 00001111, ADDRESS: 00111100
TIME: 18200 ns ACTION: TCP module finished writing to TxRAM.
```

- Λήψη Segments (σενάριο 5): Το υποσύστημα αποστολής βρίσκεται στην διαδικασία μετάδοσης μίας ροής σε κάποιον προορισμό μέσω μίας σύνδεσης. Ο προορισμός μέσω της ίδιας σύνδεσης αποστέλλει δεδομένα που περιλαμβάνουν και επιβεβαιώσεις. Το υποσύστημα λήψης διαπιστώνει ότι η σύνδεση είναι ενεργή και αποστέλλει ακόμα δεδομένα. Έτσι, δεν αναγκάζει το σύστημα αποστολής για μετάδοση plain ack. Αντίθετα, η επιβεβαίωση του εισερχόμενου πακέτου ενθυλακώνεται σε κάποιο από τα εξερχόμενα. Τα περιεχόμενα του αρχείου "tcp_monitor.dat" έχουν ως εξής:

```
TIME: 0 ns ACTION: Reset module...
TIME: 2600 ns ACTION: Initializing CRC module.
TIME: 4200 ns ACTION: Ready to decode next instruction.
TIME: 4500 ns ACTION: Receiving and checking a new segment ...
TIME: 8800 ns ACTION: Receive Memory Enabled (Read)!
TIME: 8800 ns ACTION: Receive Memory Enabled (Write)!
TIME: 8800 ns ACTION: Data: 11111111 was written in Receive Memory at address:
00000000010
TIME: 8900 ns ACTION: Data: 11111111 was written in Receive Memory at address:
00000000011
TIME: 9000 ns ACTION: Data: 11111111 was written in Receive Memory at address:
00000000100
TIME: 9100 ns ACTION: Data: 11111111 was written in Receive Memory at address:
00000000101
TIME: 9700 ns ACTION: 0100: Success Receive! Socket ID: 11111
                      Buffer Offset: 0000000000000010
                      Byte Count: 0000000000000100
```

- Λήψη Segments (σενάριο 6): Λήψη πακέτου με σωστό αριθμό επιβεβαίωσης (acknowledgement number) αλλά λάθος αριθμό σειράς (sequence number). Το σύστημα αποστέλλει ack με τον ίδιο αριθμό επιβεβαίωσης.
- Λήψη Segments (σενάριο 8): Το TCB είναι αρχικοποιημένο έτσι ώστε ο Fast Retransmit Counter να είναι 1. Λαμβάνονται 2 πακέτα στην σωστή σειρά. Το πρώτο πακέτο γίνεται δεκτό και αμέσως στέλνεται το αντίστοιχο ack. Το δευτερο πακέτο γίνεται επίσης δεκτό αλλά επειδή έχουμε πλέον τριπλότυπη ack στέλνεται άμεσα το ζητούμενο πακέτο στο οποίο έχει ενθυλακωθεί η ack που έπρεπε να στείλουμε (και η οποία εάν δεν είχαμε

"time out" του FRC θα στέλνονταν με την μορφή plain ack) Σημειώνεται ότι το tcp δεν στέλνει στην εφαρμογή μηνύματα επιτυχίας λήψης επειδή πρώτον δεν έχουμε το psh flag set και δεύτερον, επειδή υπάρχει άφθονος χώρος στην μνήμη λήψης.

- Λήψη Segments (σενάριο 8): Το σενάριο αυτό αποτελεί παραλλαγή του προηγούμενου. Τα πακέτα που λαμβάνονται είναι εκτός σειράς, αλλά φέρουν έγκυρο acknowledgement number. Το σύστημα, αφού απορρίψει τα εισερχόμενα πακέτα, αποστέλλει άμεσα τα πακέτα από το σημείο του τελευταίου επιβεβαιωμένου πακέτου και μετά.
- Λήψη Segments (σενάριο 9): Η σύνδεση είναι σε κατάσταση ESTABLISHED, λαμβάνεται ένα πακέτο fin και μεταβαίνει στην CLOSE WAIT. Παράλληλα, μεταδίδεται μήνυμα στο επίπεδο εφαρμογών και αποστέλλεται επιβεβαίωση. Τα περιεχόμενα του αρχείου "tcp_monitor.dat" έχουν ως εξής:

```
TIME: 0 ns ACTION: Reset module...
TIME: 2600 ns ACTION: Initializing CRC module.
TIME: 4200 ns ACTION: Ready to decode next instruction.
TIME: 4500 ns ACTION: Receiving and checking a new segment ...
TIME: 9600 ns ACTION: 1001: Close wait! SocketID: 11111
TIME: 12300 ns ACTION: Requesting to write to TxRAM.
TIME: 17100 ns ACTION: Access to TxRAM taken!
=== Preparing to send 1st segment of flightsize to TxRAM or plain ack ===
TIME: 17300 ns ACTION: CRC module is enabled...
TIME: 17300 ns ACTION: DATA SENT TO CRC MODULE:00011011
...
TIME: 18400 ns ACTION: Tx_RAM enabled  for write...
TIME: 18400 ns ACTION: DATA SENT TO TxRAM: 00000000, ADDRESS: 00101011
TIME: 18400 ns ACTION: DATA SENT TO CRC MODULE:00000000
...
TIME: 21400 ns ACTION: TCP module finished writing to TxRAM.
```

- Λήψη Segments (σενάριο 10): Η σύνδεση είναι σε κατάσταση LISTEN και λαμβάνεται πακέτο SYN. Το σύστημα απαντά με επιβεβαίωση για τη λήψη του SYN και εμφανίζει μήνυμα connection request. Τα περιεχόμενα του αρχείου "tcp_monitor.dat" έχουν ως εξής:

```
TIME: 0 ns ACTION: Reset module...
TIME: 2600 ns ACTION: Initializing CRC module.
TIME: 4500 ns ACTION: Receiving and checking a new segment ...
TIME: 9800 ns ACTION: 1111: Connection Request! Socket ID:01011
                      Remote IP Address: 00000000111111110000000000110001
                      Destination Port Number: 0000000000000111
TIME: 12300 ns ACTION: Requesting to write to TxRAM.
TIME: 17100 ns ACTION: Access to TxRAM taken!
=== Preparing to send 1st segment of flightsize to TxRAM or plain ack ===
TIME: 17300 ns ACTION: CRC module is enabled...
TIME: 17300 ns ACTION: DATA SENT TO CRC MODULE:00011011
...
```

```
TIME: 18400 ns ACTION: Tx_RAM enabled  for write...
TIME: 18400 ns ACTION: DATA SENT TO TxRAM: 00000000, ADDRESS: 00101011
...
TIME: 21400 ns ACTION: TCP module finished writing to TxRAM.
```

Τα σενάρια που αναφέρθηκαν είναι κάποια χαρακτηριστικά μόνο από τα πολλά που εφαρμόστηκαν προκειμένου να δοκιμαστεί το σύστημα αποστολής/λήψης TCP με λεπτομέρεια. Θα πρέπει να σημειωθεί ότι τα περιεχόμενα του κεντρικού monitor (tcp_monitor.dat) από το test bench δίνονται αποσπασματικά, προκειμένου να γίνει αντιληπτή η λειτουργία του συστήματος. Ο λόγος για τον οποίο δεν παρατίθενται αυτούσια, είναι καθαρά ο περιορισμένος χώρος της παρούσας αναφοράς.

5.4 Προγράμματα για την ανάλυση του δικτύου

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.

5.5 Η υλοποίηση της σχεδίασης σε αναδιατασσόμενη λογική

Μέσα από τις σελίδες της παρούσας αναφοράς έχει γίνει αντιληπτό ότι η ανάπτυξη του TCP/IP stack σε IP core στοχεύει στην ευελιξία που προσφέρει η αποτύπωσή του σε αναδιατασσόμενη λογική, ανεξάρτητα από τον κατασκευαστή, αλλά και σε τεχνολογία VLSI/ASIC με τη χρήση κατάλληλων μοντέλων για τα στοιχεία μνήμης που υπάρχουν στη σχεδίαση. Στην παρούσα εργασία, 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.

5.5.1 FPGA vs. Microcontroller

Ένα από τα εύλογα ερωτήματα που προκύπτουν μέσα από την παρούσα εργασία είναι γιατί η υλοποίηση της σχεδίασης έγινε σε FPGA και όχι σε έναν απλούστερο μικροελεγκτή. Η απάντηση στο ερώτημα αυτό δίνεται από την τάση των σχεδιαστών στις μέρες μας. Η πλειοψηφία των σχεδιαστών χρησιμοποιούν FPGAs. Η προτίμηση των FPGAs οφείλεται σε πολλούς λόγους. Ένας πολύ σημαντικός λόγος, είναι το γεγονός ότι η τεχνολογία της αναδιατασσόμενης λογικής έχει πλέον εξελιχθεί σε τέτοιο βαθμό, που είναι σε θέση να προσφέρει ακριβώς την ίδια λειτουργικότητα με οποιοδήποτε μικροελεγκτή αλλά σε μεγαλύτερες ταχύτητες.

Βέβαια, το γεγονός ότι η λειτουργικότητα ενός μικροελεγκτή μπορεί να υλοποιηθεί και με την χρήση FPGA, δεν είναι ένας παράγοντας ικανός, έτσι ώστε ο σχεδιαστής να προτιμήσει την FPGA, δεδομένου ότι το κόστος της δεύτερης μπορεί να είναι αρκετές φορές μεγαλύτερο από εκείνο του μικροελεγκτή.

Οι FPGAs έχουν να δείξουν πολλά επιπλέον χαρακτηριστικά. Κάθε μικροελεγκτής παρέχει συνήθως ένα με δύο υποσυστήματα που χρησιμοποιούνται για την επικοινωνία με άλλες συσκευές. Μία FPGA, όμως, μπορεί να παρέχει σχεδόν οποιοδήποτε αριθμό υποσυστημάτων για την επικοινωνία αυτή και κατά κανόνα περιέχει μεγαλύτερο μέγεθος μνήμης ενώ μπορεί να υποστηρίξει και παράλληλες διεργασίες. Για την υποστήριξη περισσότερων υποσυστημάτων επικοινωνίας στην περίπτωση του μικροελεγκτή απαιτείται η αντικατάστασή του με έναν άλλον που

πληροί τις προϋποθέσεις, ενώ στην περίπτωση μίας FPGA, συνήθως, απαιτείται ένας απλός επαναπρογραμματισμός. Ένα άλλο χαρακτηριστικό των FPGAs έναντι των μικροελεγκτών, είναι η τάση τροφοδοσίας (1.8-3.3 VDC) σε συνδυασμό με την χαμηλότερη κατανάλωση ενέργειας, θέμα που απασχολεί ιδιαίτερα τους σχεδιαστές σήμερα. Εν γένει, η ταχύτητα ενός μικροελεγκτή, σε περίπτωση εκτέλεσης πολύπλοκων λειτουργιών, είναι σημαντικά χαμηλότερη από την αντίστοιχη μίας FPGA. Επίσης, λόγω της μεγάλης διείσδυσης της τεχνολογίας της αναδιατασσόμενης λογικής, οι εταιρίες δημιουργούν software για τον προγραμματισμό των FPGAs με συνεχώς αυξανόμενες δυνατότητες και ευκολίες. Μάλιστα, η χρήση των FPGAs είναι σχεδόν επιβλητική σε περιπτώσεις σχεδιάσεων που βρίσκονται ακόμη στο στάδιο της ανάπτυξης ή της αποσφαλμάτωσης. Τέλος, από πλευράς κόστους, η τιμή των μικροελεγκτών διαμορφώνεται με βάση τα υποσυστήματα που περικλείονται στο ολοκληρωμένο κύκλωμα, τον τύπο του hardware (αν είναι 8 bit, 16 bit, 32 bit κλπ) και τον αριθμό των εισόδων/εξόδων. Αντίθετα, επειδή όπως είναι λογικό, μέχρι να προγραμματιστεί μία FPGA δεν παρέχει κανένα υποσύστημα, η τιμή της διαμορφώνεται με βάση τον αριθμό των πυλών ή των λογικών κελιών που διαθέτει.

5.5.2 Υλοποίηση για το ολοκληρωμένο κύκλωμα XC2V8000-4FF1517 της οικογένειας Virtex II

Στο σημείο αυτό, εμφανίζεται μία ακόμη διαφορά της έκδοσης αυτής της σχεδίασης σε σχέση με την προηγούμενη. Στην τρίτη έκδοση της σχεδίασης χρησιμοποιήθηκε το ολοκληρωμένο κύκλωμα XCV1000BG560 της οικογένειας Virtex. Ο λόγος που είχε ωθήσει στην χρήση του εν λόγω ολοκληρωμένου, ήταν η ύπαρξη της πλατφόρμας PLATO στο εργαστήριο ΕΜΥ, γεγονός που επέτρεπε την καλύτερη δοκιμή της σχεδίασης σε πραγματικό δίκτυο. Η τέταρτη έκδοση της σχεδίασης, όμως, χρησιμοποιεί το ολοκληρωμένο κύκλωμα XCV8000-4FF1517 της οικογένειας Virtex II. Η αλλαγή αυτή έγινε, διότι, κρίθηκε αναγκαία η υποστήριξη και πιο νέων ολοκληρωμένων κυκλωμάτων. Βέβαια, θα πρέπει να σημειωθεί ότι το εργαστήριο ΕΜΥ δεν διαθέτει πλατφόρμα ανάλογη με την PLATO που να υποστηρίζει ολοκληρωμένα κυκλώματα της οικογένειας Virtex II. Έτσι δεν στάθηκε δυνατή η δοκιμή σε πραγματικές συνθήκες δικτύου.

Στον πίνακα 12 παρατίθενται στοιχεία για την χρήση των πόρων από την σχεδίαση. Στον πίνακα αυτό παρουσιάζονται ταυτόχρονα και οι πόροι που απορροφήθηκαν από την προηγούμενη σχεδίαση (v.3) στην XCV100BG560.

Σύμφωνα με τον πίνακα, η σχεδίαση καταλαμβάνει το 5.9% των Block Rams, που είναι διαθέσιμα στο ολοκληρωμένο κύκλωμα. Η μέγιστη εκτιμώμενη ταχύτητα λειτουργίας, σύμφωνα με το Simplify Pro 7.3.3 είναι 37.5 MHz. Βέβαια, θα πρέπει να αναφερθεί ότι η ταχύτητα αυτή, αν και ενδεικτική, αντιπροσωπεύει το worst case, γεγονός που σημαίνει ότι είναι δυνατόν η σχεδίαση να δουλεύει και σε ακόμα μεγαλύτερη ταχύτητα.

ΥΠΟΣΥΣΤΗΜΑ	**# SLICES**		**% ΤΗΣ ΣΧΕΔΙΑΣΗΣ**		**% ΤΟΥ FPGA**		**ΤΑΧΥΤΗΤΑ (MHz)**	
	v.4	***v.3***	***v.4***	***v.3***	***v.4***	***v.3***	***v.4***	***v.3***
MII	2	2	0.01	0.10	0.004	0.05	338.3	338
CRC-32	44	53	0.44	1.10	0.09	0.55	150	77
RxETHER	126	138	1.26	2.30	0.27	1.2	137	44
TxETHER	47	63	0.47	1.10	0.10	0.55	159	89
OUTMUX	29	34	0.29	0.60	0.06	0.3	218	160
ARP_RECV_REPLY	286	219	2.86	3.70	0.61	1.7	96	62
ARP_REQUEST	70	85	0.70	1.40	0.15	0.8	142	83
ETHER_SEND	107	126	1.07	2.10	0.22	1.8	157	74
CHECKSUM	23	40	0.23	0.80	0.04	0.8	111	92
IP_RECV	61	57	0.61	0.90	0.13	0.45	105	60
IP_SEND	88	99	0.88	1.70	0.18	0.8	166	83
CONTROL	94	129	0.94	2.30	0.2	1.2	248	89
ICMP_RECV_REPLY	112	111	1.12	1.90	0.24	0.9	117	78
UDP	470	506	4.71	8.60	1	4.2	77	48
TCP (total)	8318	4203	84.41	70.4	17.85	33.9	37	27
TCP: Receive	*1682*	-	*16.80*	-	*3.61*	-	*42*	-
TCP: Send	*1582*	-	*15.80*	-	*3.39*	-	*37*	-
TCP: Instruction Decode	*475*	-	*4.75*	-	*1,01*	-	*49*	-
ΣΥΝΟΛΟ	**10007**	**5931**	**100.0**		**21.4**	**48**	**37.5**	**26.6**
Memory (Block RAM)			**10 / 168 Block RAMs (5.9%)**					

Πίνακας 12: Χρήση πόρων της XCV8000-4FF1517 (v.4).

Το μεγαλύτερο και ταυτόχρονα το πιο πολύπλοκο υποσύστημα είναι το υποσύστημα λήψης και αποστολής πλαισίων TCP. Αυτό φαίνεται και από τα slices που απορροφά αλλά και από την μέγιστη εκτιμώμενη ταχύτητα. Μένοντας λίγο περισσότερο στην λεπτομέρεια αυτή, βλέπουμε ότι η ταχύτητα του υποσυστήματος TCP πλησιάζει κατά πολύ την ταχύτητα ολόκληρου του συστήματος, γεγονός που οφείλεται στο ότι το υποσύστημα αυτό έχει απόλυτα καθορισμένη διεπαφή με την

υπόλοιπη σχεδίαση και η ανταλλαγή της πληροφορίας με το πρωτόκολλο του χαμηλότερου επιπέδου γίνεται με σχετική άνεση χρόνου, καθώς οι υπολογισμοί μπορούν να γίνουν αρκετούς κύκλους πριν τα δεδομένα ζητηθούν.

Η πληροφορία που μας δίνει η ταχύτητα με την οποία «τρέχει» η σχεδίαση αποτελεί ακρογωνιαίο λίθο για τον προσδιορισμό της απόδοσης της σχεδίασης. Δεν αποτελεί, όμως, την μόνη απαιτούμενη πληροφορία. Λόγω της φύσης, αλλά και της χρήσης της σχεδίασης, κρίνεται απαραίτητη η ύπαρξη πληροφορίας σχετικά με το throughput που μπορεί να υποστηριχθεί.

Ο παράγοντας throughput είναι κρίσιμος και για έναν ακόμα λόγο: για την ανάδειξη της αναγκαιότητας χρήσης ενός hardware συστήματος που υλοποιεί τα πρωτόκολλα TCP/IP, στην θέση ενός οποιουδήποτε άλλου software συστήματος.

Αντικείμενο της παρούσας διπλωματικής εργασίας ήταν μόνο το υποσύστημα αποστολής/λήψης πλαισίων TCP. Συνεπώς, η δομή και η λειτουργία του εν λόγω υποσυστήματος είναι πλήρως καθορισμένη και γνωστή. Αντίθετα, δεν θα ήταν δυνατό να υποστηριχθεί κάτι παρόμοιο για ολόκληρο το σύστημα. Αυτό συμβαίνει, διότι, όπως έχει ήδη γίνει αντιληπτό από τα προηγούμενα κεφάλαια, το υποσύστημα TCP επανασχεδιάστηκε κρατώντας τις ίδιες διεπαφές με τα υπόλοιπα υποσυστήματα, με τέτοιο τρόπο, ώστε με μια απλή αντικατάσταση, η σχεδίαση να δουλέψει και πάλι άμεσα χωρίς να απαιτούνται αλλαγές στα υπόλοιπα υποσυστήματα.

Μέσα από τα παραπάνω, ανάγεται το συμπέρασμα ότι όλα τα υποσυστήματα πλην του TCP αποτέλεσαν για την διπλωματική εργασία αυτή «μαύρα κουτιά», καθώς δεν χρειάστηκε καμία απολύτως παρέμβαση σε αυτά. Ως εκ τούτου δεν υπάρχει καμία πληροφορία σχετικά με την εσωτερική δομή τους. Για τον σωστό υπολογισμό του throughput, όμως, απαιτούνται πολλές πληροφορίες που έχουν να κάνουν με την ύπαρξη pipelining σταδίων ή όχι από όλα τα υποσυστήματα και συνολικά από όλη την σχεδίαση.

Η έλλειψη πληροφορίας σχετικά με την λειτουργία των περισσότερων υποσυστημάτων, εισάγει μεγάλη πιθανότητα σφάλματος στον υπολογισμό του throughput με βάση την ταχύτητα. Παρόλα αυτά, μπορούμε να έχουμε μία σχετικά καλή εικόνα αν θεωρήσουμε, ότι η σχεδίαση σε κάθε κύκλο ρολογιού στέλνει και λαμβάνει τετράδες από bits. Τότε, με δεδομένο από τον πίνακα 12 ότι η συνολική ταχύτητα είναι 37,5 MHz, μπορούμε να υποστηρίξουμε ότι το throughput είναι 150 Mbps.

Επεκτείνοντας λίγο ακόμη τους προσεγγιστικούς υπολογισμούς μας, μπορούμε να θεωρήσουμε ότι το παραπάνω throughput αφορά, κυρίως, στην περίπτωση αποστολής/λήψης πακέτων TCP. Η παραπάνω υπόθεση οφείλεται στο γεγονός, ότι το υποσύστημα TCP είναι το πιο πολύπλοκο και ταυτόχρονα το πιο αργό από τα υπόλοιπα υποσυστήματα Αυτό έχει ως αποτέλεσμα, όπως φαίνεται και στον πίνακα 12, η συνολική ελάχιστη εκτιμώμενη ταχύτητα να μειώνεται και να βρίσκεται σε επίπεδα κοντά στην ταχύτητα του υποσυστήματος TCP. Άρα, σε περιπτώσεις αποστολής/λήψης πακέτων UDP ή ICMP και με δεδομένο ότι τα υπόλοιπα εμπλεκόμενα υποσυστήματα (Ethernet, CRC, MII, CheckSum, ICMP κλπ) έχουν περιθώριο να τρέξουν σε μεγαλύτερες ταχύτητες, το throughput θα μπορούσε να αυξηθεί σημαντικά.

Κεφάλαιο 6: Τα "Open Cores"

Μέσα από τα κεφάλαια που προηγήθηκαν, δόθηκε η δυνατότητα να αναλύσουμε την νέα σχεδίαση με λεπτομέρεια. Όπως έχει ήδη αναφερθεί, σκοπός της παρούσας διπλωματικής δεν ήταν μόνο η βελτιστοποίηση του συστήματος και το πέρασμα από την τρίτη στην τέταρτη έκδοση. Εξίσου σημαντικός στόχος ήταν και η δημοσίευση του "Open TCP/IP Core" στην ιστοσελίδα. OpenCores.org. Για τον λόγο αυτό, κρίνεται σκόπιμο να δούμε κάποιες λεπτομέρειες σχετικά με την ταυτότητα του συγκεκριμένου web site.

6.1 Το "www.OpenCores.org"

Θα μπορούσε κάποιος να υποθέσει, ότι το OpenCores.org είναι κάποιος αφιλοκερδής οργανισμός που ενδιαφέρεται για την δημιουργία ενός γενικού μοντέλου ανάπτυξης IP Cores. Η παραπάνω υπόθεση είναι εν μέρει σωστή. Το OpenCores είναι μία μεγάλη ομάδα ανθρώπων που υποστηρίζουν το μοντέλο του ανοιχτού λογισμικού και έχουν κάποια ενδιαφέροντα και στόχους που εντοπίζονται στον τομέα ανάπτυξης του hardware.

Η ομάδα αυτή ξεκίνησε να παίρνει μεγάλες διαστάσεις από την στιγμή που άρχισε να δίνεται παγκοσμίως έμφαση στα ψηφιακά «υποσυστήματα» που καλούνται με μία λέξη cores. Σε αυτό, φυσικά συνέβαλαν και οι FPGAs των οποίων το σχετικά χαμηλό κόστος επέτρεψε την ανάπτυξη των Cores από τους ενδιαφερόμενους χωρίς σημαντική επιβάρυνση. Τονίζεται ότι, όλη η δραστηριότητα της ομάδας επικεντρώνεται γύρω από την ιστοσελίδα www.OpenCores.org.

Το συναρπαστικό στοιχείο που συναντάται στα OpenCores, όπως και στις υπόλοιπες ομάδες που ενστερνίζονται την ιδέα του ανοιχτού λογισμικού, είναι ότι μεταξύ των μελών δεν υπάρχει κανένας απολύτως δεσμός, καμία απολύτως σχέση. Είναι, απλά, μία ομάδα ανθρώπων που έτυχε να ακολουθούν την ίδια κατεύθυνση σε μία κοινή «διαδρομή». Η ομάδα των χρηστών που αναπτύσσουν τα cores είναι κάτι το «δυναμικό»: κάθε μέρα κάποια μέλη απομακρύνονται, ενώ, τις θέσεις τους καταλαμβάνουν άλλα νέα. Στο σύνολό της, η ομάδα των OpenCores αναπτύσσεται και προοδεύει από τις συνεισφορές κάθε μέλους ξεχωριστά.

Μία ερώτηση που προκύπτει αβίαστα από την παραπάνω εισαγωγική αναφορά, αφορά στην αναζήτηση των λόγων για τους οποίους οι ενδιαφερόμενοι συνεισφέρουν στα OpenCores. Η απάντηση δεν θα μπορούσε να είναι μονολεκτική καθώς κάθε μέλος του Open Cores συμμετέχει για τους δικούς του προσωπικούς λόγους. Κάποιοι από τους λόγους είναι η μάθηση, η εκπαίδευση, η βελτίωση των γνώσεων και η προσπάθεια να κάνουμε καλύτερο τον κόσμο του hardware.

Όπως αναφέρθηκε, τα cores που συμπεριλαμβάνονται στο OpenCores συνεισφέρονται από μία μεγάλη ομάδα ανθρώπων, όπως μαθητές, φοιτητές, εκπαιδευτικά ιδρύματα, καθηγητές, εταιρίες και γενικά, από όποιον ενδιαφέρεται. Πολλά είναι και εκείνα τα μέλη που ασχολούνται με τα cores μόνο στον ελεύθερο χρόνο τους, με μόνη βοήθεια κάποια βιβλία και έχουν φθάσει σε σημείο να δημιουργήσουν σημαντικότατα projects!

Επικεντρώνοντας το ενδιαφέρον μας στον δικτυακό τόπο των OpenCores, θα πρέπει να αναφέρουμε ότι αυτός συντηρείται εδώ και χρόνια με έξοδα του Dmajan Lampret. Αυτό, βέβαια, σε καμία περίπτωση δεν σημαίνει ότι ο Dajman είναι ιδιοκτήτης των cores που φιλοξενούνται στο www.OpenCores.org! Αντιθέτως, κάθε χρήστης που δημοσιεύει κάτι στον δικτυακό τόπο παραμένει ιδιοκτήτης της δουλειάς του. Σύμφωνα, μάλιστα, με τις προτάσεις, που διατίθενται στην ιστοσελίδα, οι χρήστες που συνεχίζουν να διατηρούν αντιρρήσεις σχετικά με την ιδιοκτησία του domain name άλλά και του server από τον Dajman Lampret, καλούνται να δημιουργήσουν τον δικό τους mirror server προκειμένου να έχουν πλήρη έλεγχο της δουλειά τους. Ο κεντρικός server βρίσκεται μέχρι και σήμερα στην Σλοβενία ενώ ένας δευτερεύων στην Ινδονησία.

6.2 Η «αποστολή» του "OpenCores"

Οι σημερινές σχεδιάσεις της τάξεως των μερικών μικρών (DSM – Deep SubMicron designs), συμπεριλαμβάνουν πολλά εκατομμύρια πυλών σε ένα ολοκληρωμένο ASIC. Ωστόσο, ο αριθμός των πυλών συνεχίζει να αυξάνει με ραγδαίους ρυθμούς, γεγονός που έχει ως συνέπεια, την επιμήκυνση του χρόνου σχεδιασμού, παράγοντας ο οποίος με την σειρά του μεγαλώνει υπερβολικά τον χρόνο που μεσολαβεί μέχρι το προϊόν να βγει στην αγορά (time-to-market) άρα και το κόστος!

Η λύση στο παραπάνω πρόβλημα, από τεχνικής άποψης, είναι η δημιουργία επαναχρησιμοποιήσιμων cores. Η τεχνική αυτή, επιταχύνει τρομακτικά την διαδικασία της σχεδίασης, επιτρέποντας, παράλληλα, στους μηχανικούς να συνεργάζονται.

Μέχρι σήμερα, τα διαθέσιμα cores για την ολοκλήρωση σχεδιάσεων αποτελούσαν πνευματική ιδιοκτησία και ο μόνος τρόπος χρήσης τους ήταν η αγορά τους, συχνά σε εξωφρενικά υψηλές τιμές, από συγκεκριμένους πωλητές. Το υψηλό κόστος της άδειας χρήσης ενός core αποτελεί τροχοπέδη για τις περισσότερες ομάδες σχεδιαστών με πενιχρά έσοδα. Επίσης, τα cores που ανήκουν στο καθεστώς που περιγράψαμε είναι, συνήθως, δύσκολο να ενσωματωθούν στο design κάθε σχεδιαστή λόγω πολυπλοκότητας, ασυμβατότητας ή έλλειψης προγραμμάτων ελέγχου. Τέλος, πρέπει να αναφερθεί, ότι τα cores αυτά, ενώ, συνοδεύονται με αναλυτική τεκμηρίωση, τις περισσότερες φορές ο πωλητής δεν έχει πρόσβαση στο κώδικα, γεγονός που καθιστά την διαδικασία της ενσωμάτωσης ακόμα πιο δύσκολη έως και ακατόρθωτη.

6.2.1 Οι στόχοι

Κύριος στόχος των OpenCores είναι η σχεδίαση και η δημοσίευση cores. Η διάθεση και η χρήση των cores είναι εντελώς δωρεάν ακολουθώντας το γνωστό μοντέλο διάθεσης του ανοιχτού κώδικα. Άλλοι στόχοι είναι η δημιουργία standards για τους κώδικες των cores, η δημιουργία εργαλείων και μεθόδων ανάπτυξης και τέλος, η παροχή επαρκούς υποστήριξης για κάθε core. Οι μέθοδοι και τα εργαλεία που αναπτύσσονται από την ομάδα OpenCores επιτρέπουν μεγάλες ομάδες,

διάσπαρτες ανά τον κόσμο, να συνεργάζονται και να αναπτύσσουν hardware με «ανοιχτό» τρόπο.

6.2.2 Τα οφέλη

Το hardware ανοιχτού κώδικα αποτελεί την λύση όλων των προβλημάτων που παρουσιάζονται σήμερα από την χρήση κώδικα επί πληρωμή. Η χρήση του προσφέρει πολύ σημαντικά οφέλη όπως:

- Κάθε core παρέχεται από μία μεγάλη ομάδα χρηστών-σχεδιαστών οι οποίοι είναι σε θέση να παρέχουν άψογη υποστήριξη, καλύτερη τεκμηρίωση και καλύτερα παραδείγματα του τρόπου λειτουργίας.
- Ο κώδικας είναι «ανοιχτός», γεγονός που επιτρέπει στον ενδιαφερόμενο να τον ελέγξει και να διαπιστώσει αν καλύπτει στην πραγματικότητα τις ανάγκες του.
- Οι εκατοντάδες λύσεις με cores που παρέχονται από τα OpenCores, παρέχονται χωρίς κανένα, απολύτως, κόστος.
- Τέλος, όσο πιο πολύ αναπτύσσονται τα «ανοιχτά» cores τόσο πιο πολύ θα δομούνται και θα καθιερώνονται τα στάνταρ, φθάνοντας σε σημείο να παρουσιάζουν μεγαλύτερη συμβατότητα απ'ότι εκείνα του «κλειστού» κώδικα.

6.2.3 Οι προβληματισμοί

Η ύπαρξη και η ανάπτυξη των «ανοιχτών» cores είναι κάτι, που αναμφισβήτητα, προκαλεί πρόοδο στον τομέα του hardware, καθώς, αντιμετωπίζει πιο αντικειμενικά τις δυσκολίες της εκάστοτε σχεδίασης και ταυτόχρονα, προάγει το πνεύμα της σωστής συνεργασίας. Παρόλ'αυτά, όπως κάθε σχετικά νέος τομέας παρουσιάζει κάποια προβλήματα, έτσι και στην περίπτωσή των OpenCores, πρόβλημα αποτελεί η εγγύηση της αξιοπιστίας και η παροχή μεθόδων αδειοδότησης για την χρήση των κωδίκων. Τα δύο σημεία αυτά αποτελούν για την κοινότητα των OpenCores όχι μόνο προβλήματα, αλλά και σημεία πρόκλησης τα οποία καλούνται να αντιμετωπίσουν και να επιλύσουν, όπως ακριβώς κατάφεραν να αντιμετωπίσουν δεκάδες προηγούμενα, προκειμένου να εξελιχθεί ο τομέας αυτός.

6.3 Προτάσεις σχεδίασης

Κάθε core για να μπορεί να συμπεριληφθεί στα OpenCores, πρέπει να ακολουθεί ορισμένους κανόνες ή – καλύτερα – προτάσεις. Οι προτάσεις αυτές είναι, κυρίως, γενικού χαρακτήρα και πρέπει να υιοθετούνται από τους σχεδιαστές, όχι μόνο επειδή έτσι προτείνεται από τα OpenCores, αλλά επειδή, έτσι δομείται ένας κώδικας σωστά και μελλοντικά έχει την δυνατότητα επέκτασης και χρήσης σε άλλη σχεδίαση. Οι λόγοι, δηλαδή, που ωθούν το OpenCores να δώσει προτεινόμενους τρόπους δόμησης του κώδικα, είναι, καθαρά και μόνο, για την υλοποίηση σωστών, συμβατών και επεκτάσιμων σχεδιάσεων.

Δεν υπάρχει απολύτως καμία προτίμηση για την γλώσσα γραφής κάθε core. Προτείνεται, όμως, στους σχεδιαστές να χρησιμοποιούν ευρέως γνωστές και ευέλικτες γλώσσες, προκειμένου η σχεδίασή τους να είναι όσο το δυνατόν περισσότερο χρήσιμη. Σημειώνεται ότι, οι περισσότεροι χρήστες προτιμούν είτε VHDL, είτε Verilog.

6.3.1 Γενικές προτάσεις δόμησης του κώδικα

Το OpenCores προσφέρει ένα σύντομο έγγραφο με γενικές οδηγίες δόμησης του κώδικα. Το έγγραφο αυτό, που αναθεωρείται αρκετά συχνά, παρέχει κάποια γενικότερα σχόλια τριών κατηγοριών ("Good Practice", "Recommendation" "Strong Recommendation"), γενικότερα για την δημιουργία του κώδικα και ειδικότερα σε περίπτωση που ως γλώσσα επιλεχθεί η VHDL ή η Verilog. Παρακάτω, συνοψίζονται κάποιες από τις βασικές προτάσεις που αναφέρονται στην γλώσσα VHDL.

Στον κώδικα κάθε core θα πρέπει να υπάρχουν αναλυτικά σχόλια για λόγους κατανόησης. Επίσης στην περίπτωση που η σχεδίαση αποτελείται από πολλά και σύνθετα υποσυστήματα, προτείνεται αυτά να ενώνονται σε ένα top level αρχείο το οποίο δεν θα περικλείει καθόλου λογική. Όσον αφορά στις active low & active high λογικές, προτείνεται η υιοθέτηση μίας εκ των δύο, αλλά όχι και των δύο.

Όσον αφορά στο reset αυτό θα πρέπει να είναι ασύγχρονο, active high και σε περίπτωση που ενεργοποιείται, όλα τα bidirectional ports που πιθανόν να υπάρχουν στην σχεδίαση, να βρίσκονται στην κατάσταση εισόδου. Εν γένει η χρήση των tri-state buffers δεν ενδείκνυται.

Οι μνήμες που θα χρησιμοποιούνται θα πρέπει να είναι σύγχρονες μονόπορτες ή διπορτες γενικού τύπου (όπως generic_spram και generic_dpram). Η σχεδίαση θα

πρέπει να βασίζεται σε σύγχρονη λογική, να μην περιέχει στοιχεία (υποσυστήματα) καθυστέρησης, μανδαλωτές (latches) ή flip-flops που κάνουν χρήση του αρνητικού παλμού του ρολογιού και να μην γίνεται χρήση της τεχνικής clock gating.

Εν γένει προτείνεται ο τύπος std_logic, ειδικά για τα εξωτερικά ports, αντενδείκνυται η χρήση του τύπου αρχικοποίησης κατά την δήλωση μίας μεταβλητής ή σήματος ή η ταυτόχρονη χρήση και των δύο στάνταρ της VHDL ('87 & '93). Επίσης προτιμάται η εντολή "case" σε αντίθεση με την "if…then…else". Τέλος, όσον αφορά στην επικεφαλίδα κάθε αρχείου προτείνεται, ανεπιφύλακτα η χρήση μίας στάνταρ επικεφαλίδας που φαίνεται στην εικόνα 43. Το Open TCP/IP ακολουθεί πλήρως τις παραπάνω συστάσεις.

```
------------------------------------------------------------------
----                                                          ----
---- WISHBONE XXX IP Core                                     ----
--                                                            ----
---- This file is part of the XXX project                     ----
---- http://www.opencores.org/cores/xxx/                      ----
----                                                          ----
---- Description                                              ----
---- Implementation of XXX IP core according to               ----
---- XXX IP core specification document.                      ----
----                                                          ----
---- To Do:                                                   ----
----                                                          ----
----                                                          ----
---- Author(s):                                               ----
---- - First & Last Name, email@opencores.org                 ----
----                                                          ----
------------------------------------------------------------------
----                                                          ----
---- Copyright (C) 2001 Authors and OPENCORES.ORG             ----
----                                                          ----
---- This source file may be used and distributed without     ----
---- restriction provided that this copyright statement is not ----
---- removed from the file and that any derivative work       ----
---- contains the original copyright notice and the associated ----
----disclaimer.                                               ----
----                                                          ----
---- This source file is free software; you can redistribute it ----
---- and/or modify it under the terms of the GNU Lesser       ----
----General Public License as published by the Free           ----
----Software Foundation;                                     ----
---- either version 2.1 of the License, or (at your option) any ----
---- later version.                                           ----
----                                                          ----
---- This source is distributed in the hope that it will be   ----
---- useful, but WITHOUT ANY WARRANTY; without even           ----
----the implied  warranty of MERCHANTABILITY or               ----
----FITNESS FOR A PARTICULAR PURPOSE. See the                ----
----GNU Lesser General Public License for more               ----
---- details.                                                 ----
----                                                          ----
---- You should have received a opy of the GNU Lesser         ----
----General Public License along with this source; if not,    ----
----download it                                              ----
---- from http://www.opencores.org/lgpl.shtml                 ----
----                                                          ----
------------------------------------------------------------------
-- $Log$
--
```

Εικόνα 44: Η επικεφαλίδα των αρχείων που δημοσιεύονται στο OpenCores.org

6.3.2 Προτεινόμενο System-On-A-Chip (SoC) bus

Το System-On-A-Chip (SoC), ουσιαστικά, αποτελεί μία σχεδιαστική μεθοδολογία με στόχο την ανάπτυξη συστημάτων, των οποίων τόσο το interface όσο και η γενικότερη δομή ακολουθεί κάποιους συγκεκριμένους κανόνες. Οι κανόνες αυτοί υπάρχουν με σκοπό να εξασφαλισθεί, κατά κάποιο τρόπο, συμβατότητα μεταξύ των σχεδιάσεων, χωρίς την παρουσία επιπλέον λογικής, για την ομαλή διασύνδεση. Η χρήση μίας σωστής σχεδιαστικής μεθοδολογίας, εξάλλου, επιτρέπει την τοποθέτηση περισσότερων συστημάτων πάνω στο ίδιο chip.

Τα OpenCores δεν θέτουν ως προϋπόθεση την αναγκαστική συμμόρφωση με κάποιο Soc Bus. Προτείνουν, όμως και υποστηρίζουν το Wishbone [44] Interconnection. Το Wishbone Interconnection αποτελεί μία καλή λύση ειδικά για τα IP Cores. Σκοπός του, όπως όλα τα υπόλοιπα SoC είναι η υποστήριξη των επαναχρησιμοποιήσιμων σχεδιάσεων παρακάμπτοντας τα γνωστά προβλήματα ασυμβατότητας μεταξύ τους.

Το interface της σχεδίασής μας μπορεί να διασπαστεί σε δύο τμήματα. Το πρώτο είναι εκείνο που εξασφαλίζει την επικοινωνία του με την κάρτα δικτύου (που υλοποιεί το φυσικό επίπεδο) και το δεύτερο εκείνο που εξασφαλίζει την επικοινωνία με το επίπεδο εφαρμογών. Το πρώτο τμήμα, όπως είναι λογικό, δεν παρουσιάζει ασυμβατότητες καθώς ακολουθεί τις προδιαγραφές των καρτών δικτύου. Το δεύτερο ακολουθεί ένα συγκεκριμένο μοντέλο, το οποίο υπήρχε ήδη από την δεύτερη έκδοση και συνεχίζει να διατηρείται και σε αυτήν. Η υποστήριξη του Wishbone δεν θεωρήθηκε αναγκαία.

Κεφάλαιο 7: Συμπεράσματα & Μελλοντικές Επεκτάσεις

Η ενασχόληση με την εργασία αυτήν οδήγησε στην εξαγωγή κάποιων συμπερασμάτων και συνάμα, δημιούργησε νέες ιδέες για μελλοντικές επεκτάσεις της παρούσας σχεδίασης ή τη χρήση της σε κάποιες δημοφιλείς εφαρμογές. Στο κεφάλαιο αυτό, που αποτελεί τον επίλογο της παρούσας αναφοράς, παρατίθενται οι παραπάνω σκέψεις, σκέψεις οι οποίες θα μπορούσαν να κεντρίσουν το ενδιαφέρον για την δημιουργία της επομένης γενιάς του συστήματος...

7.1 Συμπεράσματα

Κατά την διάρκεια των επτά και πλέον μηνών που διήρκεσε η ενασχόληση με την παρούσα διπλωματική εργασία, τα συμπεράσματα, οι παρατηρήσεις και - γιατί όχι - τα επιφωνήματα ήταν ένα καθημερινό φαινόμενο. Στο εργαστήριο Μικροεπεξεργαστών και Υλικού δόθηκε η δυνατότητα να μάθουμε, όχι μόνο να βρίσκουμε το πρόβλημα, αλλά να είμαστε σε θέση να προσφέρουμε την λύση, όχι επειδή έτσι πρέπει, αλλά, επειδή κάθε πρόβλημα που εμφανίζονταν αποτελούσε, ως ένα βαθμό πρόκληση!

Ο κώδικας της τρίτης έκδοσης της σχεδίασης και ειδικότερα το υποσύστημα αποστολής λήψης πλαισίων TCP, στο οποίο δόθηκε το βάρος της εργασίας αυτής, αποτέλεσε έκπληξη, καθώς, η δομή του ήταν εντελώς άναρχη. Μέσα σε 7500 γραμμές κώδικα γινόταν η υλοποίηση του πιο πολύπλοκου υποστηριζόμενου πρωτοκόλλου. Η πλήρης έλλειψη των σωστών διεπαφών, σε συνδυασμό με την πραγματικά ακαταλαβίστικη δομή οδήγησαν ασυζητητί στην διάσπαση του

υποσυστήματος και την αντιμετώπιση των αστοχιών, που λόγω όγκου και δομής, παρουσίαζε.

Η διαδικασία της διάσπασης ήταν χρονοβόρα και δύσκολη, καθώς απαιτούσε γνώση, όχι μόνο του προς διάσπαση κώδικα, αλλά και των στοιχείων του πρωτοκόλλου TCP. Παρόλ'αυτά, με διαδοχικά βήματα και με την απλόχερη βοήθεια του συνάδελφου κ. Κοϊδή που αντιμετώπισε το ίδιο πρόβλημα κατά την διάρκεια της προσθήκης του Congestion Control στην διπλωματική του εργασία, ο στόχος επιτεύχθηκε.

Στην παρούσα αναφορά, περιγράφεται η πορεία της διπλωματικής, ξεκινώντας από την σύντομη σχετική έρευνα και προχωρώντας προς την γενική δομή του συστήματος, μέχρι την λεπτομερή ανάλυση όλων των υποσυστημάτων. Φυσικά, δεν θα πρέπει να παραληφθεί και η διαδικασία της δοκιμής που παρουσιάζεται με αναλυτικό τρόπο στο πέμπτο κεφάλαιο. Η εν λόγω διαδικασία αποτελεί ιδιαίτερα σημαντικό μέρος , καθώς η καταγραφή της επιτρέπει την αποφυγή προβλημάτων που εμφανίστηκαν και επιλύθηκαν κατά την φάση της σχεδίασης, σε μελλοντικές επεκτάσεις ή επανασχεδιάσεις του συστήματος. Τέλος στο προηγούμενο κεφάλαιο έγινε μία μικρή αναφορά στα OpenCores.

Κλείνοντας την συζήτηση αυτή, σχετικά με τα συμπεράσματα, αυτό που πρέπει να τονιστεί και πάλι, είναι ότι, πέρα από τη γνώση και τεχνογνωσία που αποκτήθηκε σε θέματα τηλεπικοινωνιών, σχεδίασης hardware και χρήσης εργαλείων CAD, η διπλωματική αυτή βοήθησε στο να βελτιωθεί ο τρόπος σκέψης, ο τρόπος οργάνωσης και ο τρόπος που ανακαλύπτουμε ένα πρόβλημα και ψάχνουμε για την λύση του.

7.2 Μελλοντικές Επεκτάσεις

Αναμφισβήτητα, η παρούσα έκδοση της σχεδίασης βρίσκεται αρκετά βήματα μπροστά από άποψη σωστής λειτουργίας και δομής σε σχέση με τις προηγούμενες. Το γεγονός αυτό, όμως, σε καμία περίπτωση, δεν σημαίνει ότι η εξέλιξη της σχεδίασης θα μπορούσε να σταματήσει στο σημείο αυτό.

Σε όλες τις εκδόσεις της σχεδίασης, μέχρι στιγμής, γινόντουσαν «προσθετικές» παρεμβάσεις προκειμένου να εμπλουτισθεί με νέα χαρακτηριστικά. Στην εργασία αυτή ακολουθήθηκε η ίδια γραμμή. Παράλληλα, όμως, έγινε προσπάθεια βελτιστοποίησης της δομής. Μελλοντικά, θα μπορούσε να συνεχιστεί η συγκεκριμένη προσπάθεια, βελτιώνοντας ακόμη περισσότερο τόσο το υποσύστημα TCP όσο και τα υπόλοιπα υποσυστήματα ή προσθέτοντας σε αυτά επιπλέον δυνατότητες όπως η υποστήριξη περισσότερων options στα πακέτα TCP, η υποστήριξη περισσότερων υπηρεσιών του στο πρωτόκολλο ICMP κλπ.

Επειδή, όμως, κάθε παρέμβαση βελτιστοποίησης έχει άμεση συνέπεια στην συνολική απόδοση της σχεδίασης, θεωρείται άκρως αναγκαίο μελλοντικά να δοθεί περισσότερο βάρος στην απόδοση του συστήματος. Πιο συγκεκριμένα, στις μελλοντικές εργασίες θα μπορούσε να υπάρξει λεπτομερέστερη ανάλυση των παραμέτρων που συνιστούν την απόδοση του συστήματος.

Άλλες αλλαγές που θα μπορούσαν να γίνουν, είναι η αφαίρεση των υποσυστημάτων που υποστηρίζουν το πρωτόκολλο του Ethernet και η τοποθέτηση άλλων υποσυστημάτων. Με τον τρόπο αυτό μπορεί να υποστηριχθεί η διακίνηση πακέτων ATM, δημιουργώντας ένα σύστημα IP over ATM. Επιπρόσθετα, θα μπορούσε να υποστηριχθεί το πρωτόκολλο Bluetooth ή το ασύρματο 802.11 κάτω από τα IP, UDP και TCP.

Με δεδομένη την επικράτηση του IPv6 στο μέλλον, μία καλή μελλοντική επέκταση της σχεδίασης, θα ήταν η υποστήριξη και του IPv6 ώστε το σύστημα να μπορεί να χρησιμοποιηθεί και μετά από πολλά χρόνια όταν, πλέον, η χρήση της επόμενης γενιάς του πρωτοκόλλου IP οριστικοποιηθεί.

Τέλος, θα μπορούσε να αναπτυχθεί μία εφαρμογή για την υποστήριξη, είτε του VoIP, που γνωρίζει ιδιαίτερη διάδοση στις μέρες μας, είτε για την συλλογή πληροφοριών και την διαχείριση απομακρυσμένων συστημάτων, όπως για παράδειγμα τα συστήματα ανανεώσιμων πηγών ενέργειας. 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.

Παράρτημα

Στο σημείο αυτό, παρουσιάζονται περιληπτικά κάποια από τα πολύ βασικά στοιχεία από την θεωρία των δικτύων για λόγους καλύτερης κατανόησης της παρούσας εργασίας. Για περισσότερες λεπτομέρειες, ωστόσο, συνίσταται η αναφορά στο βιβλίο Computer Networking των Kurose & Ross [1] και στις προηγούμενες διπλωματικές εργασίες.

Π.1 Τα Επίπεδα OSI

Η αρχιτεκτονική του δικτύου ακολουθεί μία διάταξη επιπέδων, τα Επίπεδα OSI (Open Systems Interconnection reference model). Τα επίπεδα παρουσιάζονται στην εικόνα 44.

Applications *User Application*
Presentation *Upper Layer Protocols*
Session *Sockets*
Transport *UDP, TCP*
Network *ARP, IP, ICMP*
Data Link *LLC, MAC*
Physical *MII, PMI, MMD, PMD*

Εικόνα 45: Τα επίπεδα OSI.

Κάθε ένα από τα επίπεδα έχει αυστηρά καθορισμένο ρόλο και λειτουργία. Έτσι, το κατώτερο επίπεδο, το Physical, είναι υπεύθυνο για την μετάδοση των bits της πληροφορίας, το Data Link, για την μετάδοση πακέτων μέσα από μία σύνδεση, το Network για την μετάδοση πακέτων από άκρο σε άκρο, το Transport για την

παράδοση μηνυμάτων από άκρο σε άκρο, το Session για την εγκατάσταση και διαχείριση συνομιλίας από άκρο σε άκρο, το Presentation για την διαμόρφωση, κρυπτογράφηση και συμπίεση των δεδομένων και τέλος, το Application για την παροχή υπηρεσιών του δικτύου (ηλεκτρονικό ταχυδρομείο, μεταφορά αρχείων κλπ).

Π.2 Το πακέτο Ethernet

Η διάταξη ενός πακέτου Ethernet φαίνεται στην εικόνα 44. Η μέγιστη πληροφορία που μπορεί να μεταφερθεί ανά πακέτο Ethernet είναι 1500 bytes. Αυτό σημαίνει ότι κάθε πακέτο IP, που είναι μεγαλύτερο από 1500 bytes, θα πρέπει υποχρεωτικά να υποστεί κατακερματισμό.

Preamble	Destination Address	Source Address	Type	Data	CRC

Εικόνα 46: Η διάταξη ενός πακέτου Ethenet.

- Preamble (8 bytes): Κάθε πακέτο Ethernet ξεκινά με αυτό το πεδίο. Σκοπός της ύπαρξης του είναι ο συγχρονισμός των ρολογιών μεταξύ των δύο άκρων της σύνδεσης.
- Destination Address (6 bytes): Η διεύθυνση LAN του προορισμού.
- Source Address (6 bytes): Η διεύθυνση LAN της αποστολέα.
- Type (2 bytes): Χρησιμοποιείται για την αποπολύπλεξη των πακέτων. Κάθε πρωτόκολλο του επιπέδου Data Link (IP, ARP, Novell IPX, AppleTalk klp) έχει τον δικό του αριθμό type.
- Data (~1500 bytes): Τα δεδομένα του πακέτου.
- CRC (4 bytes): Ο αριθμός αυτός χρησιμοποιείται για τον έλεγχο σφάλματος στα δεδομένα κατά την αποστολή.

Π.3 Το πρωτόκολλο ARP

Όπως έχει αναφερθεί αρκετές φορές μέσα στην παρούσα αναφορά, το πρωτόκολλο ARP είναι υπεύθυνο για την αντιστοίχηση των διευθύνσεων IP με

εκείνες του Ethernet. Χωρίς να μπαίνουμε σε λεπτομέρειες ας δούμε την δομή ενός πακέτου του πρωτοκόλλου αυτού.

00	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
Hardware type																Protocol type															
Hardware address length								Protocol address length								Opcode															
Source hardware address																															
Source protocol address																															
Destination hardware address																															
Destination protocol address																															

Εικόνα 47: Το πακέτο ARP.

- Hardware Type (16 bits): Χρησιμοποιείται για την αναφορά του hardware που χρησιμοποιείται για την μετάδοση (πχ Ethernet, ATM, IP, Serial κλπ).
- Protocol Type (16 bits) : Φέρει την τιμή 0x800 για το πρωτόκολλο IP.
- Hardware/ Protocol address length (8 bits): Το μήκος των πεδίων hardware και protocol address σε bytes.
- Opcode (16 bits): Χρησιμοποιείται για την διάκριση των υπηρεσιών ARP (Arp Request, Arp Reply).
- Source hardware address (32 bits): η διεύθυνση hardware της πηγής.
- Source protocol address (32 bits): η διεύθυνση πρωτοκόλλου της πηγής.
- Destination hardware address (32 bits): η διεύθυνση hardware του προορισμού.
- Destination hardware address (32 bits): η διεύθυνση πρωτόκολλου του προορισμού.

Π.4 Το πρωτόκολλο ICMP

Το πρωτόκολλο ICMP σχεδιάστηκε, με σκοπό να επιτρέπει την ανταλλαγή πληροφοριών μεταξύ των πυλών και των σταθμών του δικτύου. Οι πληροφορίες έχουν να κάνουν με την αναζήτηση ενός σταθμού μέσα στο δίκτυο (εφαρμογή Ping), την αναφορά σφαλμάτων κλπ. Η δομή ενός πακέτου φαίνεται στην εικόνα 47.

00 01 02 03 04 05 06 07	08 09 10 11 12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Type	Code	ICMP header checksum
Identifier		Sequence Number
Data (Optional – Variable Length)		

Εικόνα 48: Η δομή του πακέτου ICMP.

- Type: Περιγράφει την υπηρεσία που μεταφέρει το πακέτο ICMP
- Code: Χρησιμοποιείται για να δηλωθούν οι κωδικοί της υηρεσίας.
- ICMP Checksum: Χρησιμοποιείται για τον έλεγχο των σφαλμάτων κατά την μετάδοση του πακέτου.
- Identifier: Το process ID της διεργασίας που στέλνει το μήνυμα.
- Sequence Number: Ένας αριθμός που επιλέγεται από το σύστημα που εκτελεί την αίτηση. Ο αριθμός αυτός πρέπει να επιστραφεί από την συσκευή που απαντά.

Π.5 Το πρωτόκολλο IP

Το πρωτόκολλο IP ανήκει στο επίπεδο Transport και όπως είδαμε, δεν παρέχει καμία απολύτως εγγύηση για την αξιόπιστη μετάδοση. Η δομή του πακέτου IP φαίνεται στην εικόνα 48.

00 01 02 03	04 05 06 07	08 09 10 11 12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Version	IHL	TOS	Total length
Identification			Flags / Fragment offset
TTL		Protocol	Header checksum
Source IP address			
Destination IP address			
Options			

Εικόνα 49: Η δομή του πακέτου IP.

- Version (4 bits): Η έκδοση του πρωτοκόλλου IP (v.4 – v.6).

- Internet Header Length (4 bits): Το μήκος της επικεφαλίδας του πακέτου σε 32 bit λέξεις (32 bit words).
- Type Of Service (8 bits): Χρησιμοποιείται για να δηλώσει την προέλευση και τις απαιτήσεις κάθε πακέτου. Για παράδειγμα άλλη καθυστέρηση μπορεί να ανεχθεί ένα πακέτο δεδομένων και άλλο ένα πακέτο που προέρχεται από τηλεφωνία VoIP.
- Total Length (16 bits): Το συνολικό μήκος του πακέτου.
- Identification (16 bits): Χρησιμοποιείται για την επανασύνδεση των κατακερματισμένων πακέτων.
- Flags (3 bits): Φέρουν πληροφορία σχετικά με το αν ένα πακέτο δύναται να κατακερματιστεί ή αν είναι όντως κατακερματισμένο και αποτελεί το τελευταίο τμήμα της ροής του κατακερματισμένου πακέτου ή όχι.
- Fragment Offset (13 bits): Και αυτό το πεδίο έχει να κάνει με τον κατακερματισμό των πακέτων. Χρησιμοποιείται για την αναγνώριση της σειράς του κατακερματισμένου πακέτου στο αρχικό ενιαίο πακέτο.
- Time To Live (8 bits): Περιγράφει τον χρόνο παραμονής του πακέτου στο δίκτυο. Το πεδίο αυτό μειώνεται όταν το πακέτο περνά από router σε router. Όταν μηδενισθεί τότε το πακέτο απορρίπτεται.
- Protocol (8 bits): Ο κωδικός του πρωτοκόλλου, στο οποίο αναφέρεται στο ενθυλακωμένο πλαίσιο στο IP πακέτο.
- Header Checksum (16 bits): Χρησιμοποιείται για τον έλεγχο σφάλματος κατά την μετάδοση.
- Source IP Address/Destination IP Address (32 bits): Η διεύθυνση πηγής και προορισμού.
- Options: Το πεδίο αυτό έχει μεταβλητό μήκος και χρησιμοποιείται για την μετάδοση κάποιων δευτερεύουσας σημασίας πληροφοριών.

Π.6 Το πρωτόκολλο UDP

Το πρωτόκολλο UDP, αν και μοιάζει σε κάποια σημεία με το πρωτόκολλο TCP, εντούτοις παρουσιάζει και σημαντικές διαφορές. Οι διαφορές συνίστανται σε πολλά χαρακτηριστικά που παρουσιάζει το TCP και δεν διαθέτει το UDP. Έτσι, το UDP είναι connectionless (δεν εγκαθίσταται σύνδεση μεταξύ των δύο άκρων) και το βασικότερο, δεν εξασφαλίζει καμία απολύτως εγγύηση αξιόπιστης μεταφοράς. Η δομή του πακέτου UDP φαίνεται στην εικόνα 49, ενώ τα πεδία της επικεφαλίδας εξηγούνται περιληπτικά παρακάτω.

00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Source Port	Destination Port
Length	Checksum
Data	

Εικόνα 50: Η δομή του πακέτου UDP.

- Source Port/ Destination Port (16 bits): Το port πηγής/προορισμού.
- Length (16 bits): Το μήκος του πακέτου σε 32 bit λέξεις.
- Checksum (16 bits): Αριθμός που χρησιμοποιείται για τον έλεγχο σφαλμάτων κατά την μετάδοση.

Π.7 Το πρωτόκολλο TCP

Το πρωτόκολλο TCP, όπως ήδη αναφέρθηκε από την προηγούμενη παράγραφο,είναι αρκετά πιο πολύπλοκο, καθώς, είναι σε θέση να εγγυηθεί την αξιόπιστη μεταφορά δεδομένων. Δημιουργεί πάντα connection oriented συνδέσεις. Στην παρούσα παράγραφο, θα δούμε την δομή ενός TCP πακέτου, θα αναφέρουμε την διαδικασία εγκαθίδρυσης σύνδεσης και τερματισμού αυτής και τέλος, θα μιλήσουμε για τους αλγόριθμους του congestion control.

Π.7.1 Η δομή του TCP πακέτου

00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Source Port	Destination Port
Sequence Number	
Acknowledgment Number	
Data Offset / Unsused / Flags	Receive Window
Checksum	Urgent Pointer
Options and padding	
Data	

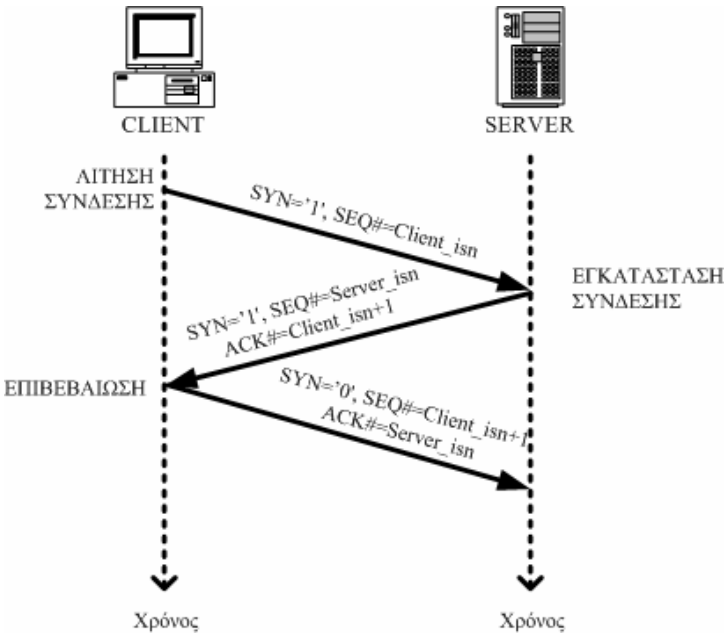
Εικόνα 51: Η δομή του πακέτου TCP.

- Source Port/Destination Port (16 bits): Το port number της πηγής και του προορισμού.
- Sequence Number (32 bits): Ο αριθμός σειράς του πακέτου. Χρησιμοποιείται για την ανίχνευση της σειράς των πακέτων.
- Acknowledge Number (32 bits): Ο αριθμός επιβεβαίωσης των ληφθέντων πακέτων. Ισούται με τον αριθμό του τελευταίου byte που λήφθηκε σωστά συν ένα.
- Data Offset (4 bits): Το μήκος της επικεφαλίδας σε 32 bit λέξεις.
- Flags (6 bits): Σημαντικό πεδίο, καθώς διαδραματίζει πρωταγωνιστικό ρόλο στην καθιέρωση και τον τερματισμό σύνδεσης, καθώς και σε άλλες περιπτώσεις. Τα flags είναι τα εξής:
    - URG: Το πεδίο Urgent Pointer περιέχει έγκυρες πληροφορίες.
    - ACK: Το πεδίο Acknowledge Number περιέχει έγκυρη πληροφορία.
    - PSH: Τα δεδομένα μόλις ληφθούν να οδηγηθούν άμεσα στην εφαρμογή.
    - RST: Επανεκκίνηση σύνδεσης.
    - SYN: Αίτηση για σύναψη σύνδεσης.
    - FIN: Αίτηση για το τερματισμό της σύνδεσης.
- Receive Window (16 bits): Η μέγιστη ποσότητα της πληροφορίας που είναι σε θέση να αποδεχθεί ο παραλήπτης.
- Checksum (16 bits): Το άθροισμα ελέγχου: Χρησιμοποιείται στον έλεγχο των σφαλμάτων κατά την μετάδοση.

- Urgent Pointer (16 bits): Pointer που δείχνει στο τελευταίο byte των δεδομένων που έχουν χαρακτηρισθεί ως «επείγοντα».
- Options: Πεδίο που φιλοξενεί κάποιες επιπλέον πληροφορίες και έχει μεταβλητό μέγεθος. Το πρώτο byte του πεδίου ενημερώνει τον παραλήπτη για τις πληροφορίες που πρόκειται να ακολουθήσουν.

Π.7.2 Η διαδικασία σύναψης και ο τερματισμός μίας σύνδεσης

Η διαδικασία καθιέρωσης σύνδεσης TCP ονομάζεται και three way handshaking επειδή εξελίσσεται σε τρία στάδια.



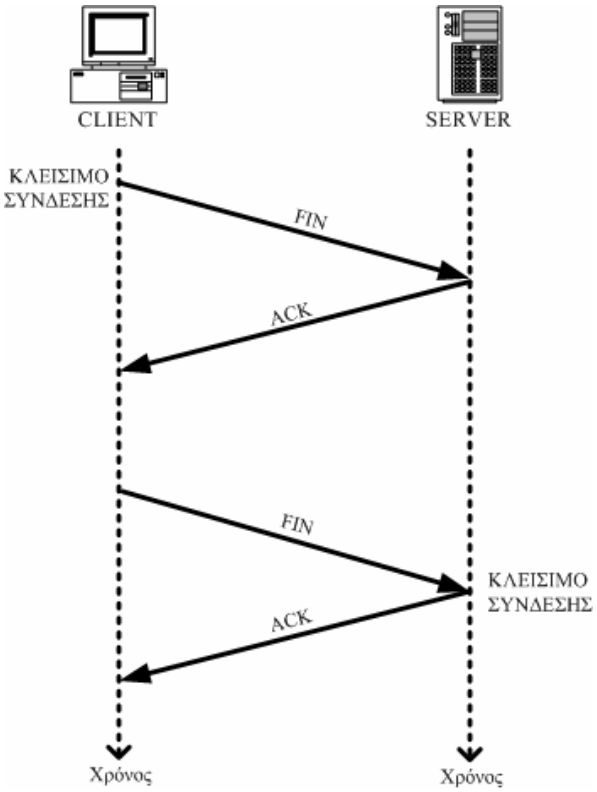
Εικόνα 52: Η διαδικασία εγκαθίδρυσης σύνδεσης.

Όπως φαίνεται και στην εικόνα 51, πρώτο βήμα για την εγκατάσταση μίας σύνδεσης είναι η αποστολή ενός ειδικού πακέτου από την πλευρά του client (πακέτο με το SYN flag set). Ο αρχικός sequence number είναι ο client_isn.

Όταν το πακέτο αυτό φθάσει στον server τότε εκείνος απαντά με ένα παρόμοιο πακέτο που φέρει επιβεβαίωση, γνωστοποιώντας, έτσι, δύο πράγματα: τον δικό του sequence number και ότι η σύνδεση έγινε δεκτή.

Τέλος ο client αποστέλλει ένα τελευταίο πακέτο, το οποίο μπορεί να φέρει και δεδομένα, γνωστοποιώντας στον server ότι έλαβε το πακέτο του. Για λόγους ευκολίας, το πρώτο πακέτο συχνά ονομάζεται «πακέτο syn» και το δεύτερο «πακέτο synack».

Στην εικόνα 52 περιγράφεται η ακριβώς αντίθετη διαδικασία: το κλείσιμο μίας σύνδεσης. Ο client αποστέλλει ειδικό πακέτο στον server (πακέτο με το FIN flag set ή αλλιώς, «πακέτο FIN») ο οποίος το λαμβάνει και αποστέλλει επιβεβαίωση. Έτσι, κλείνει η σύνδεση από την πλευρά του client. Για να κλείσει και από την πλευρά του server, γίνεται η ακριβώς αντίστοιχη διαδικασία: Ο server αποστέλλει πακέτο FIN στον client, ο οποίος απαντά με επιβεβαίωση. Με τον τρόπο αυτό η σύνδεση τερματίζεται και από τις δύο πλευρές.



Εικόνα 53: Η διαδικασία κλεισίματος μίας σύνδεσης.

Π.7.3 Οι αλγόριθμοι του Congestion Control: Slow Start & 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 την υλοποίηση τους 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, που μπορεί να στείλει στο δίκτυο ο αποστολέας πριν

πάρει κάποια επιβεβαίωση και το παράθυρο του δέκτη το οποίο είναι ένα όριο από την μεριά του δέκτη για το μέγεθος των δεδομένων, που μπορεί να δεχτεί και να αποθηκεύσει στους buffers του. Είναι λογικό, η ελάχιστη τιμή του παραθύρου συμφόρησης και του παραθύρου του δέκτη είναι αυτή, που καθορίζει το μέγεθος των ανεπιβεβαίωτων δεδομένων στο δίκτυο. Είναι σαφές, ότι αν στείλουμε κι άλλα δεδομένα στο δίκτυο πέρα από το παράθυρο συμφόρησης, τότε το επιβαρύνουμε, και ότι 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 την κατακόρυφη πτώση της απόδοσης του δικτύου. Από την άλλη, αν ένας αποστολέας στείλει περισσότερα δεδομένα απ'όσα υποδεικνύει το παράθυρο του δέκτη, τότε είναι λογικό, 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 την αποθήκευσή τους.

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 μπορεί να επιφέρει μειωμένη απόδοση και προβλήματα. Έτσι, καλείται ο αποστολέας να διερευνήσει σιγά-σιγά το δίκτυο για να βρει την διαθέσιμη χωρητικότητα, έτσι ώστε να αποφύγει την συμφόρηση. Ο slow start αλγόριθμος χρησιμοποιείται στην αρχή ή μετά από απώλεια κάποιου πακέτου για αυτόν τον σκοπό.

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 εξίσωση :

$$\boxed{\text{InitWin} = \min\{4\text{xSMSS}, (\max\{2\text{xSMSS}, 4380 \text{ bytes}\})\}}$$

όπου

- InitWin (Initial Window) η αρχική τιμή του παραθύρου συμφόρησης του αποστολέα μετά την διαδικασία σύναψης της σύνδεσης.

- SMSS (Sender Maximum Segment Size): Το μέγιστο μέγεθος του πακέτου που δύναται να στείλει ο αποστολέας. Τονίζεται ότι το μέγεθος αυτό δεν συμπεριλαμβάνει την επικεφαλίδα.

Στις περιπτώσεις αυτές, 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 αναφέρεται για να τονιστούν κάποιες ερευνητικές προσπάθειες βελτίωσης της απόδοσης του πρωτοκόλλου.

Η αρχική τιμή του slow start threshold μπορεί να είναι αυθαίρετα υψηλή, αλλά 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 παραπάνω 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 νέα επιβεβαίωση που λαμβάνει. Ο slow start σταματά, 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 κάθε φορά που στέλνεται ένα πακέτο και λαμβάνεται η αντίστοιχη επιβεβαίωση. 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 φαίνεται παρακάτω :

$$\text{CongWin} \mathrel{+}= \text{SMSS}^2/\text{CongWin}$$

όπου

- CongWin (Congestion Window): Το παράθυρο συμφόρησης.

- SMSS (Sender Maximum Segment Size): Το μέγιστο μέγεθος του πακέτου που δύναται να στείλει ο αποστολέας. Τονίζεται ότι το μέγεθος αυτό δεν συμπεριλαμβάνει την επικεφαλίδα.

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, παρέχει μια αποδεκτή προσέγγιση στον πρακτικό κανόνα που διατυπώθηκε προηγουμένως.

Διαισθητικά καταλαβαίνουμε, όμως, ότι και ο congestion avoidance δεν μπορεί να αυξάνει το παράθυρο συμφόρησης επ'άπειρον. Όπως είναι λογικό, 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, την οποία ο αποστολέας την αντιλαμβάνεται μετά από την απώλεια κάποιου πακέτου.

Έστω, ότι βρισκόμαστε κατά την διάρκεια κάποιας μετάδοσης, και αντιλαμβανόμαστε ότι έχει χαθεί κάποιο από τα πακέτα, που έχουμε στείλει. Σε μια τέτοια περίπτωση, είτε η μετάδοση ελέγχεται από τον slow start, είτε από τον congestion avoidance αλγόριθμο η αντίδραση του TCP είναι η ίδια: 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, δίνει στο παράθυρο συμφόρησης τιμή ίση με ένα πακέτο και η μετάδοση συνεχίζεται από το πακέτο το οποίο χάθηκε με τον αλγόριθμο slow start να ελέγχει πάλι την μετάδοση.

Π.7.4 Οι αλγόριθμοι του Congestion Control: Fast Retransmit & 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 τον ενημερώσει 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.

Καταρχήν μπορεί να προκληθούν από πακέτα τα οποία έχουν απορριφθεί. Διπλότυπες επιβεβαιώσεις μπορούν, επίσης, να προκληθούν από επαναδιάταξη των πακέτων στο 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. 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. Με αυτόν τον τρόπο ο αποστολέας ο οποίος προσπαθεί να ανακάμψει μετά από απώλεια θα πάρει πιο γρήγορα πληροφορίες.

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. Μετά την λήψη τριών διπλότυπων επιβεβαιώσεων, ο αποστολέας προχωρά σε επαναμετάδοση του πακέτου το οποίο φαίνεται να έχει χαθεί άμεσα.

Από την στιγμή, που ο αλγόριθμος fast retransmit στέλνει το πακέτο, που φαίνεται να έχει χαθεί, τον έλεγχο της μετάδοσης αναλαμβάνει ο αλγόριθμος fast recovery, έως ότου σταματήσουν να έρχονται πλέον διπλότυπες επιβεβαιώσεις. Ο λόγος για τον οποίο δεν προτείνεται η χρησιμοποίηση του αλγορίθμου slow start, είναι 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. Άρα το TCP μπορεί να χρησιμοποιήσει προς όφελός του τον ελεύθερο χώρο, που 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 $\boxed{\text{Ssthres} = \max\{\text{FlightSize}/2,\ 2\text{xSMSS}\}}$

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 και SMSS το μέγιστο μέγεθος του πακέτου που μπορεί να στείλει ο αποστολέας.

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, που λαμβάνουμε αυξάνουμε την τιμή του παραθύρου συμφόρησης κατά ένα πακέτο μέγιστου μεγέθους, χρησιμοποιώντας την ίδια λογική την οποία περιγράψαμε παραπάνω.
4. Μεταδίδουμε ένα ακόμα πακέτο, αν 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" το παράθυρο συμφόρησης.

Βιβλιογραφία

[1] James F. Kurose, Keith W. Rosh *"Computer Networking – A top down approach featuring internet"* Addison Wesley, Second International Edition.

[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] Christophoros Kachris *"Design and Implementation of a TCP/IP Core"*, Thesis, Technical University of Crete 2001

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

[11] 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

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

[13] Γ. Ζήσης, *"Ανάπτυξη πρωτοκόλλου TCP/IP πολλών θυρών από Application ως Session Layer, βασισμένου σε αναδιατασσόμενη λογική"*, Διπλωματική Εργασία, Πολυτεχνείο Κρήτης, 2002

[14] Ι. Κοϊδής *"Έλεγχος συμφόρησης σε επαναχρησιμοποιήσιμο σχεδιασμό του πρωτόκολλου TCP/IP για αναδιατασσόμενη λογική"*, Διπλωματική Εργασία, Πολυτεχνείο Κρήτης, 2003

[15] **WEB:** http://www.rfc-editor.org, Σύνδεσμος στα RFC του οργανισμού IETF.

- RFC 768 "User Datagram Protocol"
- RFC 791 "Internet Protocol"
- RFC 792 "Internet Control Message Protocol"
- RFC 793 "Transmission Control Protocol"
- RFC 826 "Ethernet Address Resolution Protocol"
- RFC 894 "Standard for the Transmission of IP Datagrams over Ethernet networks"
- RFC 2581 "TCP Congestion Control"
- RFC 2988 "Computing TCP'S Retransmission Timer"
- RFC 2861 "TCP Congestion Window Validation"

[17] **WEB:** http://www.networksorcery.com

[18] **WEB:** http://www.beyondlogic.org

[19] **WEB:** http://www.cmx.com

[20] **WEB:** http://www.kadak.com/html/kdkp1030.htm

[21] **WEB:** http://www.microchip.com

[22] **WEB:** http://www.microdigital.co.uk/

[23] **WEB:** http://www.teamf1.com/NetF1.htm

[24] **WEB:** http://www.lantronix.com/products/eds/micro100

[25] **WEB:** http://www/interpeak.com

[26] **WEB:** http://www.edevice.com

[27] **WEB:** http://www.hynix.com/eng/

[28] **WEB:** http://www.altera.com/products/devkits/altera/kit-dev_nios_ethernet.html

[29] **WEB:** http://www.xilinx.com/publications/xcellonline/partners/xc_insight41.htm

[30] **WEB:** http://www.digi.com

[31] **WEB:** http://www.connectone.com

[32] **WEB:** http://www.wiznet.co.kr

[33] **WEB:** http://savannah.nongnu.org/projects/lwip/

[34] **WEB:** http://www.elektor.gr (τεύχος 263-264, Ιούλιος – Αύγουστος 2004)

[35] **WEB:** http://www.hynix.com/eng/products/system_ic/sp/down/HMS91C7432.pdf

[36] **WEB:** http://www.iready.com

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

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

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

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

[42] **WEB:**http://www.synplicity.com

[43] **WEB:** http://www.model.com

[44] **WEB:** http://www.opencores.org