ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

INTRODUCTION TO ROOFTOP NETWORKING

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Δημήτριος Ντίλης

Εξεταστική Επιτροπή
Άγγελος Μπλέτσας (επιβλέπων), Επίκουρος Καθηγητής
Γεώργιος Καρυστινός, Επίκουρος Καθηγητής
Πολυχρόνης Κουτσάκης, Επίκουρος Καθηγητής

Ιούνιος 2012

Ευχαριστίες

Θα ήθελα να εκφράσω την ευγνωμοσύνη μου προς τον καθηγητή Άγγελο Μπλέτσα για την ανάθεση αυτής της διπλωματικής εργασίας, την καθοδήγησή του και την υποστήριξη από το αρχικό μέχρι το τελικό στάδιο της υλοποίησης. Η ευρεία γνώση και εμπειρία του στον τομέα των τηλεπικοινωνιών αποτέλεσαν σημαντική βοήθεια στην επίτευξη των στόχων μου.

Τέλος θα ήθελα να ευχαριστήσω τα αγαπημένα μου πρόσωπα που πάντοτε ήταν εκεί για μένα.

Abstract

Τα ασύρματα mesh δίκτυα παρέχουν τη δυνατότητα για γρήγορη και εύκολη συμμετοχή σε τοπικά δίκτυα και πρόσβαση στο internet αυξάνοντας σε μεγάλο βαθμό την περιοχή κάλυψης που παρέχουν τα ενσύρματα δίκτυα. Η γενική περιγραφή τους περιλαμβάνει ένα σύνολο από κόμβους οι οποίοι δρομολογούν πακέτα μέσω διαδοχικών αλμάτων από τον ένα κόμβο στον άλλο.

Για την υλοποίηση αυτής της διπλωματικής εργασίας χρησιμοποιήθηκαν κάρτες ασύρματης επικοινωνίας 802.11 πολύ χαμηλού κόστους (9 ευρώ η κάθε μία), οι οποίες είναι συμβατές με τους οδηγούς MadWifi. Εγκαταστήσαμε έξι από αυτές τις κάρτες σε τρεις υπολογιστές (δύο κάρτες ανά υπολογιστή) έτσι ώστε να έχουμε μεγαλύτερη ελευθερία στην επιλογή των καναλιών λειτουργίας τους. Δημιουργήθηκε λογισμικό που ελέγχει τη λειτουργικότητα αυτών των καρτών και το οποίο χρησιμοποιεί γέφυρες στο επίπεδο 2 με τη βοήθεια του STP (Spanning Tree Protocol).

Το αποτέλεσμα της εργασίας ήταν η δημιουργία ενός φθηνού δικτύου αναμετάδοσης από προσωπικούς υπολογιστές οι οποίοι δρομολογούν τα πακέτα μέσω ασύρματων αλμάτων και την ίδια στιγμή εξυπηρετούν ασύρματους χρήστες που επιθυμούν πρόσβαση στο internet ή τη συμμετοχή σε τοπικό δίκτυο.

Περιεχόμενα

Λίστα των Figures

Κεφάλαιο 1

Εισαγωγή

Ο όρος rooftop networking αναφέρεται στην ασύρματη διασύνδεση απομακρυσμένων περιοχών (συνήθως διαφορετικών κτιρίων) μέσω κόμβων που μπορεί να είναι δρομολογητές ή και κανονικοί υπολογιστές. Επιτρέπει τη γρήγορη αλλά και εύκολη δημιουργία δικτύων με μεγάλο αριθμό κόμβων δίνοντας τη δυνατότητα σε χρήστες να αποκτήσουν πρόσβαση στο internet. Μόνο λίγοι από τους κόμβους είναι συνδεδεμένοι με κάποιον πάροχο διαδικτύου. Τα πακέτα διαδίδονται μέσω διαδοχικών αλμάτων από τον ένα κόμβο στον άλλο, έως ότου φτάσουν σε κάποιον που έχει πρόσβαση στο διαδίκτυο.

Η απουσία καλωδίων, το πολύ χαμηλό πλέον κόστος των WiFi καρτών αλλά και η λειτουργία τους στη ζώνη των κοινόχρηστων ISM (Industrial, Scientific, Medical) συχνοτήτων, καθιστούν τα παραπάνω δίκτυα ιδανικά για ένα πλήθος εφαρμογών όπως η επέκταση της εμβέλειας λειτουργίας ενός ασύρματου παρόχου (access point) για την κάλυψη μεγαλύτερου αριθμού χρηστών.

Κατά καιρούς έχουν γίνει αρκετά projects με σκοπό τη δημιουργία τέτοιων ασύρματων δικτύων. Παρακάτω παρουσιάζουμε μερικά από τα σημαντικότερα.

1. Roofnet: Πρόκειται για ένα 802.11b/g mesh δίκτυο που αναπτύχθηκε στο MIT με σκοπό να παρέχει στους χρήστες πρόσβαση στο internet. Οι κόμβοι του δικτύου είναι υπολογιστές που ο καθένας διαθέτει μία κεραία και μία ethernet θύρα. Το ethernet χρησιμοποιείται από τους υπολογιστές που μπορούν να έχουν άμεση πρόσβαση στο internet, ενώ οι κεραίες εγκαταστάθηκαν σε οροφές κτιρίων των φοιτητών στους οποίους είχαν δοθεί κόμβοι. Οι κάρτες δικτύου χρησιμοποιούνται μόνο για την επικοινωνία μεταξύ των κόμβων και όχι σαν access points. Κάθε ένας από τους κόμβους βρίσκεται εντός εμβέλειας μερικών μόνο από τους κόμβους του δικτύου. Η επικοινωνία με τους υπόλοιπους γίνεται με multi-hop forwarding [1].

Ο υπολογιστής είναι ένα iDOT Slim PC με Mini-ITX μητρική κάρτα, 500 MHz x86 CPU και έχει το μέγεθος ενός μεγάλου laptop. Η μικρή ταχύτητα του επεξεργαστή μειώνει την ποσότητα θερμότητας που παράγεται. Με εξαίρεση τα gateways, κάθε κόμβος διαθέτει μια Hyperlink Technologies 8 dBi omni-directional κεραία. Τέλος η κάρτα δικτύου είναι η Senao/Engenius SL-2511CD PLUS EXT2 802.11b PCMCIA card. Το mode λειτουργίας που επιλέχθηκε για αυτήν είναι το Pseudo-IBSS που αποτελεί μια απλουστευμένη έκδοση του 802.11 ad hoc mode και ο driver που χρησιμοποιεί είναι ο Linux HostAP 802.11 [1].

Οι κόμβοι τρέχουν σε Red Hat 9 Linux, kernel version 2.4.20. Για route discovery και packet forwarding χρησιμοποιούν το Click software router toolkit. Επίσης σε κάθε έναν έχει υλοποιηθεί ένας Web Server, NAT και ένας DHCP server στην ethernet θύρα. Το NAT και ο DHCP server επιτρέπουν στους χρήστες να χρησιμοποιήσουν τον κόμβο ως router χωρίς να απαιτούνται άλλες ρυθμίσεις. Ο Web server διαθέτει ένα interface για απλές ρυθμίσεις/έλεγχο και δείχνει ποιες διαδρομές είναι διαθέσιμες [1].

Το πρωτόκολλο δρομολόγησης του Roofnet είναι εμπνευσμένο από το Dynamic Source Routing και λέγεται SrcRR. Σκοπός του είναι να βρει διαδρομές με υψηλό throughput. Η βασική διαφορά του από το DSR είναι ότι χρησιμοποιεί την ETX (Expected Transmission Count) μετρική για να επιλέξει καλές διαδρομές. Το ETX μετράει συνεχώς το loss rate και στις δύο κατευθύνσεις μεταξύ

κάθε κόμβου και των γειτόνων του κάνοντας περιοδικά broadcasts. Σε κάθε link δίνει μια μέτρηση που εκτιμά τον αριθμό των φορών που πρέπει να σταλεί ένα πακέτο ώστε αυτό να ληφθεί επιτυχώς [1][2].

Προτού ένας κόμβος στείλει ένα πακέτο κατασκευάζει στην επικεφαλίδα (header) του μια διαδρομή (source route) όπου θα καταγράφεται από ποιους κόμβους πέρασε το πακέτο μέχρι να φτάσει στον προορισμό του. Αν ο αποστολέας δε γνωρίζει κάποια διαδρομή προς τον προορισμό, κάνει broadcast Route Request πακέτα. Όταν κάποιο από αυτά ληφθεί από τον τελικό προορισμό, αυτός απαντάει με ένα Route Reply πακέτο προς τον αρχικό αποστολέα. Σε αυτό το πακέτο αναγράφεται και η διαδρομή προς τον τελικό προορισμό.

2. One Laptop Per Child mesh: Η εταιρία cozybit σε συνεργασία με τη Marvell δημιούργησαν το πρώτο laptop που μπορεί να επικοινωνήσει μέσω ενός mesh δικτύου (που δημιουργείται από αυτά τα laptops). Η πραγματοποίηση του project έγινε δυνατή χάρη στο λογισμικό o11s της cozybit που αποτελεί την πρώτη υλοποίηση του IEEE 802.11s draft. Το One Laptop Per Child mesh αναπτύχθηκε για χρήση σε αναπτυσσόμενες χώρες από παιδιά που ζουν σε απομακρυσμένες περιοχές [3].

Το 802.11s είναι σημαντικό γιατί εξασφαλίζει τη διαλειτουργικότητα ανάμεσα σε συσκευές που το χρησιμοποιούν. Πολλοί κατασκευαστές μέχρι στιγμής έχουν αναπτύξει τις δικές του λύσεις για wireless mesh δίκτυα αλλά οι περισσότερες από αυτές βασίζονται στο MAC του 802.11 και δεν είναι συμβατές με συσκευές άλλων εταιριών. Το 802.11s επεκτείνει το MAC του 802.11 και υποστηρίζει broadcast/multicast και unicast παράδοση πακέτων στο MAC επίπεδο χρησιμοποιώντας μετρικές στο φυσικό επίπεδο οι οποίες επιτρέπουν τη δημιουργία αυτο-οργάνωσης σε ποικίλες τοπολογίες.

3. SMesh: Το SMesh είναι ένα ασύρματο mesh δίκτυο που αναπτύχθηκε στο πανεπιστήμιο John Hopkins . Παρέχει peer-to-peer συνδεσιμότητα και πρόσβαση στο internet. Βασικός του στόχος είναι να επιτρέπει στους χρήστες ελεύθερη μετακίνηση και συνδεσιμότητα (roaming) στην περιοχή που καλύπτουν οι κόμβοι του δικτύου [4][5].

Οι κόμβοι του SMesh λειτουργούν στο 802.11 IBSS mode και πρέπει να είναι συσκευές που χρησιμοποιούν Linux και διαθέτουν μια ασύρματη 802.11 κάρτα δικτύου. Όλοι οι clients δρομολογούν τα πακέτα τους μέσω ενός κόμβου που λειτουργεί ως virtual default gateway και χρησιμοποιείται το DHCP πρωτόκολλο για να πάρουν ip διευθύνσεις [4].

Η δημιουργία της τοπολογίας ξεκινάει με τον κάθε κόμβο να κάνει broadcast την παρουσία του. Όταν ένας άλλος κόμβος λάβει ένα τέτοιο broadcast πακέτο μετράει το επίπεδο σήματος (signal strength) και αν αυτό ξεπερνάει ένα ελάχιστο όριο, συνδέεται με τον προηγούμενο κόμβο. Τα internet gateways εντάσσονται σε ένα multicast group που λέγεται Internet Gateway Multicast Group (IGMG) και πάνω στο οποίο διαφημίζουν περιοδικά την ip της ενσύρματης διεπαφής τους. Όταν δύο internet gateways λάβουν τα παραπάνω advertisements, δημιουργούν ενσύρματες συνδέσεις [5].

Διάφορες εφαρμογές όπως το voice over ip απαιτούν την αποστολή και λήψη πακέτων με σταθερό ρυθμό. Όταν οι 802.11 συσκευές λειτουργούν σε infrastructure mode, πραγματοποιούν από μόνες τους τη σάρωση (scanning) για να εντοπίσουν τα καλύτερα access points. Αυτή η διαδικασία όμως μπορεί ορισμένες φορές να διαρκέσει έως και δευτερόλεπτα που σημαίνει ότι στην περίπτωση του VoIP θα χαθούν πολλά πακέτα. Για το λόγο αυτό στο SMesh ρυθμίζονται και οι clients να

λειτουργούν σε ad hoc (IBSS) mode. Οι κόμβοι του δικτύου είναι τώρα υπεύθυνοι για τον έλεγχο της ποιότητας των links που έχουν με τους διάφορους clients και όταν χρειάζεται αναγκάζουν τους τελευταίους να αλλάξουν τον κόμβο με τον οποίο είναι συνδεδεμένοι [5].

Καθώς οι mobile clients κινούνται στο δίκτυο υπάρχει το ενδεχόμενο κάποιος από αυτούς να φτάσει πιο κοντά σε ένα διαφορετικό internet gateway. Τότε τα πακέτα του που δρομολογούνται πάντοτε στο κοντινότερο gateway, δε θα φτάσουν στο αρχικό και όλες οι ανοιχτές συνδέσεις που είχε ο κόμβος θα χαθούν. Σε αυτή την περίπτωση λοιπόν το νέο gateway είναι υπεύθυνο για τη μεταφορά των πακέτων όλων των παλαιών συνδέσεων του client στο προηγούμενο gateway. Για νέες συνδέσεις, χρησιμοποιείται το νέο gateway [5].

4. WING: Το Wireless Mesh Network for Next-Generation Internet δημιουργήθηκε από την CREATE-NET (Center for Research And Telecommunication Experimentation for NETworked communities) και το πανεπιστήμιο Technion. Το WING βασίστηκε πάνω στο Roofnet project και κατάφερε να το επεκτείνει παρέχοντας υποστήριξη για περισσότερα ραδιόφωνα, αυτόνομη επιλογή καναλιού και χρήση της WCETT μετρικής [4].

Μπορεί να εγκατασταθεί και σε ενσωματωμένες πλατφόρμες που χρησιμοποιούν OpenWRT αλλά και σε desktop Linux υπολογιστές. Όμοια με το Roofnet, το Wing βασίζεται στο Click software router toolkit και χρησιμοποιεί και το ίδιο πρωτόκολλο δρομολόγησης, το SrcRR [4].

Η ETX μετρική που χρησιμοποιήθηκε στο Roofnet project έχει σαν ελάττωμα το ότι δε λαμβάνει υπόψη του το interference και το γεγονός ότι διαφορετικά links μπορεί να έχουν διαφορετικό ρυθμό μετάδοσης. Η ETT (Expected Transmission Time) μετρική λαμβάνει υπόψη του το διαφορετικό ρυθμό μετάδοσης των links. Το ETT μιας ζεύξης ορίζεται ως η αναμενόμενη διάρκεια του MAC επιπέδου για την επιτυχή μετάδοση ενός πακέτου σε αυτή τη ζεύξη. Και αυτή η μετρική όμως αγνοεί τις πιθανές παρεμβολές (interference). Το πρόβλημα λύνεται με τη χρήση του WCETT (Weighted Cumulative Estimated Transmission Time). Το WCETT μιας διαδρομής p ορίζεται ως εξής [6]:

$$WCETT(p) = (1-\beta)*\Sigma_{\text{link } \in p} [ETT_l + \beta * \max_j (X_j)]$$

Η παράμετρος β μπορεί να πάρει τιμές ανάμεσα στο 0 και στο 1. Το X_j είναι ο αριθμός των φορών που το κανάλι j χρησιμοποιείται στο μονοπάτι p και είναι η παράμετρος που διαφοροποιεί το WCETT από το ETT αφού χάρη στο X_j το WCETT metric λαμβάνει υπόψη του το interference. Ο όρος $\max_j (X_j)$ μετράει το μέγιστο αριθμό των φορών που το ίδιο κανάλι εμφανίζεται κατά μήκος μιας διαδρομής [6].

5. BerlinRoofNet: Εμπνευσμένο και αυτό από το Roofnet project, έχει σκοπό να εξετάσει τη δυνατότητα δημιουργίας ασύρματου δικτύου παρόμοιου ή και μεγαλύτερου μεγέθους στο Βερολίνο το οποίο θα είναι πλήρως self-organizing/self-configuring. Υπεύθυνοι για αυτό το project είναι μια ομάδα εθελοντών φοιτητών από το Computer Science Department του Humboldt πανεπιστημίου του Βερολίνου [7].

Το BerlinRoofNet (BRN) λειτουργεί στο επίπεδο 2 του OSI οπότε δεν το ενδιαφέρει ποιο πρωτόκολλο ανωτέρου επιπέδου χρησιμοποιείται. Οι BRN κόμβοι ανακαλύπτουν γειτονικούς BRN κόμβους και συνδέονται αυτόματα μαζί τους σχηματίζοντας ένα μεγαλύτερο δίκτυο. Κάθε ένας από αυτούς τους κόμβους μπορεί να απενεργοποιηθεί χωρίς να αλλάξει δραστικά η ομαλή λειτουργία του δικτύου [7].

6. Hyacinth: Πρόκειται για ένα 802.11 multi-channel ασύρματο mesh δίκτυο που αναπτύχθηκε στο πανεπιστήμιο Stony Brook της Νέας Υόρκης. Το project προτείνει τη δημιουργία ενός δικτύου όπου κάθε κόμβος διαθέτει πολλαπλά 802.11 ραδιόφωνα. Για να είναι δυνατή η επικοινωνία μεταξύ δύο κόμβων πρέπει αυτοί να λειτουργούν στο ίδιο κανάλι συχνοτήτων. Όσο περισσότερα ραδιόφωνα όμως λειτουργούν στο ίδιο κανάλι, τόσο θα μειώνεται το συνολικό bandwidth για κάθε ένα από αυτά. Για το λόγο αυτό δημιουργήθηκε ένας channel assignment αλγόριθμος ο οποίος προσπαθεί να αυξήσει το συνολικό bandwidth διατηρώντας ταυτόχρονα τη συνδεσιμότητα του δικτύου. Οι κόμβοι του Hyacinth είναι RouterBoard RB-230 και ο καθένας διαθέτει τρεις 802.11a κάρτες δικτύου [8].

Μερικοί από τους κόμβους του Hyacinth είναι συνδεδεμένοι στο internet μέσω καλωδίων και οι υπόλοιποι αποκτούν πρόσβαση σε αυτό σχηματίζοντας ένα multi-hop wireless mesh δίκτυο [9].

Η δρομολόγηση βασίζεται στην κατασκευή ενός δένδρου παρόμοιο με το spanning tree που χρησιμοποιείται στο bridging. Η βασική διαφορά είναι το metric που χρησιμοποιεί ο Hyacinth είναι δυναμικό ώστε να επιτυγχάνεται καλύτερη κατανομή φορτίου (load balancing). Το project ερευνά τη λειτουργικότητα τριών διαφορετικών μετρικών. Το πρώτο είναι ο αριθμός αλμάτων ανάμεσα σε ένα κόμβο και ένα gateway. Το δεύτερο είναι το gateway link capacity και το τρίτο το path capacity που αντιπροσωπεύει το διαθέσιμο εύρος ζώνης (bandwidth) ενός μονοπατιού που συνδέει τον κόμβο με ένα gateway [9].

Υποθέτουμε πως ένας κόμβος έχει ανακαλύψει μια διαδρομή προς το ενσύρματο δίκτυο. Περιοδικά κάνει broadcast αυτή την πληροφορία στους one-hop γείτονές του με ένα ADVERTISE πακέτο που περιέχει το κόστος της διαδρομής. Αν κάποιος από τους γείτονες θέλει να συνδεθεί με τον κόμβο (είτε επειδή δε συμμετέχει ακόμα στο δίκτυο είτε επειδή νέα διαδρομή έχει μικρότερο κόστος από αυτή που χρησιμοποιεί) στέλνει ένα JOIN μήνυμα [9].

Το βασικό πρόβλημα στην ανάπτυξη ενός κατανεμημένου αλγορίθμου κατανομής καναλιών, είναι η συχνοτική εξάρτηση (channel dependency) μεταξύ των κόμβων του δικτύου. Η αλλαγή δηλαδή στη συχνότητα λειτουργίας της ζεύξης (link) μεταξύ δύο κόμβων (ίσως επειδή έχουμε μεγάλες παρεμβολές) μπορεί να οδηγήσει στην ανάγκη αλλαγής καναλιού και σε πολλά άλλα links του δικτύου. Για το λόγο αυτό ο Hyacinth επιβάλει κάποιους περιορισμούς στους κόμβους του. Το σύνολο των καρτών δικτύου που ένας κόμβος χρησιμοποιεί για την επικοινωνία με τον parent κόμβο, λέγονται UP-NICs και είναι διαφορετικό από το σύνολο των καρτών δικτύου με τις οποίες επικοινωνεί με τους κόμβους-παιδιά και λέγονται DOWN-NICs. Κάθε κόμβος είναι υπεύθυνος για την ανάθεση καναλιού στα DOWN-NICs του και κάθε UP-NIC του είναι συνδεδεμένο με ένα μοναδικό DOWN-NIC του κόμβου-πατέρα [9].

Για την επιλογή του καναλιού λειτουργίας των DOWN-NICs, κάθε κόμβος πρέπει να υπολογίσει τη χρήση όλων των καναλιών στην interference γειτονιά του. Στη συνέχεια κάθε κόμβος στέλνει αυτή την πληροφορία στους k-hop γείτονές του (όπου k ο λόγος του interference range με το communication range και συνήθως ισούται με 2 ή 3). Τελικά κάθε ένας ανακαλύπτει ένα σύνολο από τα κανάλια που χρησιμοποιούνται λιγότερο και επιλέγει κάποια από αυτά για τα DOWN-NICs του [9].

Κεφάλαιο 2

Το πρωτόκολλο επικοινωνίας και ασύρματης προσπέλασης 802.11

Σε αυτό το κεφάλαιο δίνουμε μια γενική περιγραφή του 802.11 καθώς και βασικές έννοιες που αφορούν τον έλεγχο προσπέλασης (Medium Access Control-MAC).

2.1 Γενική Περιγραφή

Το IEEE 802.11 είναι ένα πρότυπο για την υλοποίηση επικοινωνίας υπολογιστών μέσω ασύρματων τοπικών δικτύων στις μπάντες συχνοτήτων των 2.4 και 5 Ghz. Υπάρχουν διάφορες προδιαγραφές στην οικογένεια του 802.11 όπως 802.11a, 802.11b, 802.11g, 802.11e, 802.11n. Σε αυτή την εργασία μας απασχόλησαν τα 802.11b και 802.11g, τα οποία αναφέρονται σε ασύρματα δίκτυα και χρησιμοποιούνται για εκπομπές σε μικρές αποστάσεις (μέχρι 100 μέτρα περίπου σε εξωτερικούς χώρους) με ταχύτητες που φτάνουν τα 54 Mbps (802.11g).

Υπάρχουν αρκετές διαφορές μεταξύ των ασύρματων και των παραδοσιακών ενσύρματων δικτύων. Το χαμηλό κόστος αλλά και η ευκολία δημιουργίας ασύρματων δικτύων μας κάνει να τα προτιμάμε σε διάφορες περιπτώσεις όπως δικτύωση κτιρίων ή ακόμα και μέσα στο σπίτι όταν θέλουμε να συνδέσουμε υπολογιστές χωρίς τη χρήση πολλών καλωδίων. Ένα άλλο σημαντικό πλεονέκτημα των ασύρματων δικτύων είναι η ικανότητα διαχείρισης χρηστών οι οποίοι βρίσκονται σε κίνηση χωρίς απώλεια της σύνδεσής τους. Από την άλλη πλευρά, επειδή το μέσο μετάδοσης που χρησιμοποιούν είναι ο ίδιος ο αέρας, είναι λιγότερο αξιόπιστα από τα ενσύρματα δίκτυα. Ένα από κυριότερα προβλήματα που αντιμετωπίζουν είναι οι παρεμβολές από άλλες 802.11 συσκευές που αυξάνουν την πιθανότητα ύπαρξης σφάλματος κατά τη μετάδοση. Άλλο πρόβλημα είναι η εξασθένιση του σήματος μέχρι να φτάσει στον παραλήπτη όπως επίσης και οι διαλήψεις (fading) που αφορούν τις αποκλίσεις στην εξασθένιση της ισχύος του σήματος κατά τη μετάδοσή του.

Βασικό δομικό στοιχείο της αρχιτεκτονικής του 802.11 είναι το Basic Service Set (BSS). Χονδρικά ένα BSS απεικονίζει την περιοχή μέσα στην οποία τα μέλη αυτού του BSS μπορούν να παραμείνουν ώστε να έχουν μεταξύ τους επικοινωνία.

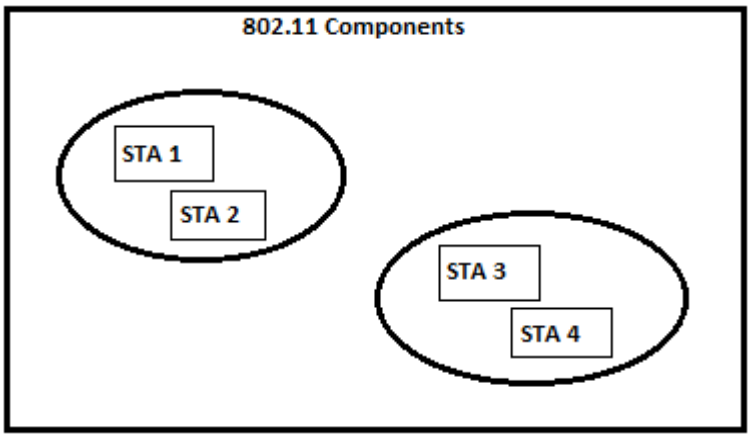


Figure 1-Basic Service Sets με 2 stations το κάθε ένα [10].

Για την αύξηση της περιοχής κάλυψης ενός δικτύου χρειάζεται η διασύνδεση μεταξύ των BSS, όπως για παράδειγμα η σύνδεση διαφορετικών access points μεταξύ τους. Αυτό επιτυγχάνεται με τη χρήση του Distribution System (DS) του 802.11 το οποίο παρέχει τις υπηρεσίες που είναι απαραίτητες για την επίτευξη των παραπάνω. Η αρχιτεκτονική του 802.11 δεν αναφέρει συγκεκριμένες λεπτομέρειες για την υλοποίηση του Distribution System. Για παράδειγμα μπορεί να υλοποιηθεί είτε σε επίπεδο MAC (layer 2) είτε σε επίπεδο δικτύου (layer 3). Από την άλλη ξεκαθαρίζει ποιες υπηρεσίες πρέπει να προσφέρει ένα DS. Αυτές είναι οι εξής [10]:

1. Association: χρησιμοποιείτε για τη σύνδεση ενός client με ένα access point.

2. Disassociation: χρησιμοποιείται για την αποσύνδεση ενός client από ένα access point.

3. Re-association: χρησιμοποιείται για την επανασύνδεση ενός client με ένα access point.

4. Distribution: χρησιμοποιείται όποτε χρειάζεται η μετάδοση ενός πακέτου (frame) από ένα client μέσω του DS.

5. Integration: αυτή η υπηρεσία συνδέει ένα 802.11 WLAN με άλλα τοπικά δίκτυα, ενσύρματα ή και ασύρματα.

Υπάρχουν και οι υπηρεσίες που προσφέρονται από οποιοδήποτε 802.11 station:

1. Authentication: παρέχει έλεγχο πρόσβασης στο ασύρματο δίκτυο.

2. De-authentication: διαγραφή client ο οποίος είχε προηγουμένως αποκτήσει δικαίωμα πρόσβασης στο δίκτυο.

3. Privacy: λόγω του ότι το 802.11 χρησιμοποιεί ως μέσο μετάδοσης τον αέρα, χρειαζόμαστε μια υπηρεσία η οποία εξασφαλίζει ότι η μετάδοση των πακέτων είναι ασφαλής, όσο και σε ένα ενσύρματο δίκτυο. Αυτό επιτυγχάνεται με τη χρήση του Wired Equivalent Privacy (WEP) αλγορίθμου ο οποίος εφαρμόζεται σε όλα τα πακέτα που μεταφέρουν δεδομένα καθώς και σε ορισμένα πακέτα διαχείρισης (management frames). Πλέον είναι ξεπερασμένος και χρησιμοποιείται κυρίως ο Wi-fi Protected Access (WPA).

4. Data Delivery: προσφέρει μετάδοση πακέτων από το MAC ενός station, στο MAC ενός άλλου ή περισσότερων.

Με τη χρήση των εννοιών του BSS και DS, είναι δυνατή η δημιουργία ενός αρκετά μεγαλύτερου δικτύου στο οποίο αναφερόμαστε με τον όρο Extended Service Set (ESS). Ένα ESS ασύρματο δίκτυο, αποτελείται από πολλαπλά BSS με αποτέλεσμα να προσφέρει στους χρήστες μεγαλύτερη περιοχή κάλυψης.

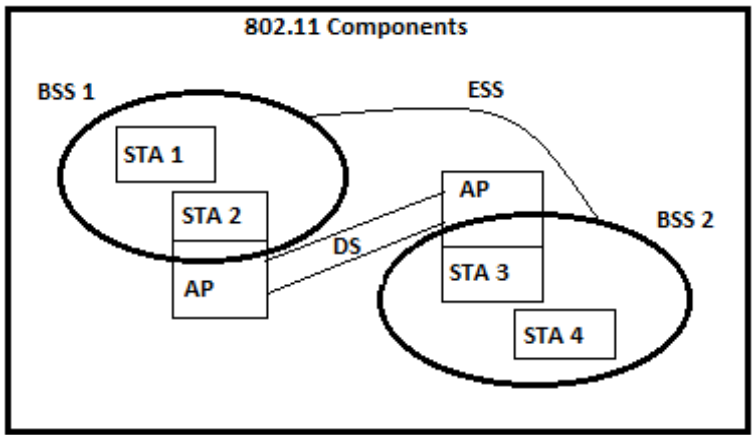


Figure 2-Extended Service Set [10].

Τα παραπάνω προϋποθέτουν ότι διαθέτουμε υπολογιστές οι οποίοι είναι συνδεδεμένοι σε access points και επικοινωνούν μεταξύ τους μέσω των τελευταίων. Είναι δυνατή όμως η δημιουργία ενός 802.11 δικτύου στο οποίο οι υπολογιστές επικοινωνούν άμεσα μεταξύ τους χωρίς την ύπαρξη κάποιου access point. Αυτά είναι τα λεγόμενα ad hoc δίκτυα.

2.2 Ορισμός Medium Access Control και Frame Format

Το MAC του 802.11 παρέχει λειτουργικότητα για την αξιόπιστη μεταφορά δεδομένων, μέσω του ασύρματου μέσου του φυσικού επιπέδου (physical layer), σε υψηλότερα επίπεδα. Η μεταφορά αυτή βασίζεται σε μια ασύγχρονη, connectionless και best effort μετάδοση πακέτων MAC επιπέδου. Δεν υπάρχει καμία εγγύηση ότι η μετάδοση δηλαδή των δεδομένων θα είναι επιτυχής.

Η πρόσβαση στο μέσο για το 802.11 γίνεται με το λεγόμενο Carrier-Sense Multiple Access with Collision Avoidance (CSMA/CA). Αυτό ορίζει ότι όταν υπάρχει κάποιος υπολογιστής που επιθυμεί να μεταδώσει δεδομένα, θα πρέπει πρώτα να "ακούσει" το κανάλι και να διαπιστώσει αν αυτό χρησιμοποιείται από άλλο υπολογιστή ο οποίος μεταδίδει εκείνη τη στιγμή. Αν το μέσο μετάδοσης είναι ελεύθερο, μπορεί να πραγματοποιηθεί η αποστολή δεδομένων.

Τα MAC frames αποτελούνται από ένα συγκεκριμένο αριθμό πεδίων τα οποία εμφανίζονται με συγκεκριμένη σειρά σε όλα τα πακέτα. Εξαίρεση σε αυτό είναι τα πεδία διευθύνσεων και το πεδίο frame body, τα οποία δεν εμφανίζονται σε κάποια πακέτα. Παρακάτω, εξηγούμε συνοπτικά τα πεδία:

Octets 2	2	4	6	6	2	6	0-2312	4
Frame Control	Duration/ ID	Address 1	Address 2	Address 3	Sequence Control	Address 4	Frame Body	FCS

Figure 3-MAC frame format [10].

Duration/ID: Δείχνει πόσο θα διαρκέσει η συγκεκριμένη μετάδοση.

Address fields: Στο frame format υπάρχουν τέσσερα πεδία διευθύνσεων. Χρησιμοποιούνται για να δείξουν το BSS ID, διεύθυνση εκπομπής, διεύθυνση προορισμού, διεύθυνση του τελευταίου κόμβου που μετέδωσε το συγκεκριμένο frame και διεύθυνση του κόμβου που είναι ο άμεσος παραλήπτης αυτού του frame. Ο τύπος του frame (management, control, data) καθορίζει ποια και πόσα από αυτά τα πεδία χρησιμοποιούνται κατά τη μετάδοση.

Sequence Control: Αποτελείται από δύο άλλα πεδία, τα sequence number και fragment number. Το πρώτο περιέχει τον αριθμό ακολουθίας του πακέτου και το δεύτερο τον αριθμό του fragment.

Frame Body: Περιέχει τα δεδομένα.

FCS: Περιέχει τον 32 bit Cyclic Redundancy Code (CRC) για ανίχνευση λάθους.

Το πεδίο frame control περιέχει διάφορα άλλα πεδία, τα οποία επεξηγούνται παρακάτω:

bits	2	2	4	1	1	1	1	1	1	1	1
Protocol Version	Type	Subtype	To DS	From DS	More Frag	Retry	Pwr Mgt	More Data	WEP	Order

Figure 4-Frame Control Field [10].

Protocol Version: Έχει σταθερή τιμή 0.

Type-Subtype fields: Αυτά τα δύο πεδία καθορίζουν μαζί τη λειτουργικότητα του frame. Για παράδειγμα αν πρόκειται για acknowledgment frame ή αν περιέχει δεδομένα.

To DS: Τίθεται σε 1 για όσα frames προορίζονται για το Distribution System. Διαφορετικά έχει τιμή 0.

From DS: Τίθεται σε 1 για όσα frames εξέρχονται από το Distribution System. Διαφορετικά έχει τιμή 0.

More Fragments: Έχει τιμή 1 όταν υπάρχουν και άλλα fragments του ίδιου frame για μετάδοση.

Retry: Με τιμή 1 δηλώνει πως πρόκειται για επαναμετάδοση frame που είχε σταλεί ανεπιτυχώς προηγουμένως.

Power Management: Δείχνει αν το station βρίσκεται σε power save mode ή όχι.

More Data: Προορίζεται σε stations που βρίσκονται σε power save mode και δηλώνει πως έχουν αποθηκευτεί δεδομένα για αυτά τα stations στο access point.

WEP: Έχει τιμή 1 όταν τα περιεχόμενα του πεδίου frame body έχουν υποστεί επεξεργασία από τον αλγόριθμο WEP.

Order: Έχει τιμή 1 όταν κάποιο frame έχει μεταδοθεί με χρήση της υπηρεσίας StrictlyOrdered (για παράδειγμα fragments ενός πακέτου)

2.3 Distributed Coordination Function

Η Distributed Coordination Function (DCF) είναι η βασική μέθοδος που χρησιμοποιείται από τα 802.11 τερματικά, έτσι ώστε να αποκτήσουν πρόσβαση στο μέσο και να μεταδώσουν. Η DCF βασίζεται στο CSMA/CA πρωτόκολλο και είναι η καρδιά του 802.11 MAC. Το CSMA/CA είναι σχεδιασμένο έτσι ώστε να μειώνει την πιθανότητα ύπαρξης σύγκρουσης πακέτων, όταν πολλαπλά stations προσπαθούν να αποκτήσουν πρόσβαση στο κανάλι για αποστολή δεδομένων.

Σύμφωνα λοιπόν με τη DCF όταν κάποιο station έχει κάποιο πακέτο έτοιμο προς μετάδοση θα ελέγξει το κανάλι. Αν αυτό είναι αδρανές για χρονικό διάστημα ίσο με ένα Distributed Interframe Space (DIFS), τότε το station θα επιχειρήσει να μεταδώσει το πακέτο του. Σε διαφορετική περίπτωση θα συνεχίσει να "ακούει" το κανάλι έως ότου αυτό γίνει αδρανές για χρονικό διάστημα ίσο με ένα DIFS. Τότε επειδή ίσως υπάρχουν και άλλα stations που επιθυμούν να μεταδώσουν, κάθε ένα θα παράξει ένα random backoff interval και θα στείλει το πακέτο του μετά το τέλος αυτού του διαστήματος εφόσον το κανάλι παραμείνει αδρανές κατά τη διάρκειά του. Έτσι μειώνεται η πιθανότητα ύπαρξης σύγκρουσης μεταξύ των stations και υλοποιείται το collision avoidance μέρος του αλγορίθμου [10][11].

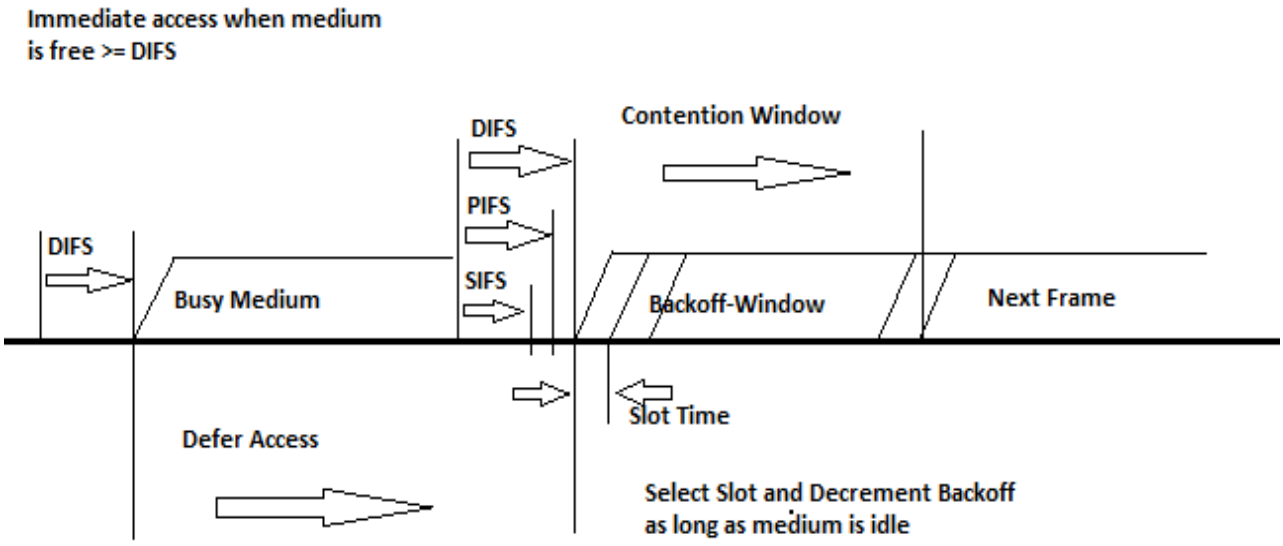


Figure 5-Basic access method [10].

Μετά τη λήψη ορισμένων frame (όπως για παράδειγμα αυτά που περιέχουν δεδομένα), απαιτείται από τον παραλήπτη να απαντήσει με ένα acknowledgment (ACK frame) για να δείξει ότι το έλαβε επιτυχώς. Αν ο αρχικός αποστολέας δε λάβει ένα τέτοιο frame θα καταλάβει ότι υπήρξε σφάλμα κατά την αποστολή του πακέτου και θα επιχειρήσει να το μεταδώσει ξανά. Το σφάλμα βέβαια μπορεί να γίνει και κατά την αποστολή του ACK frame όμως ο αποστολέας δεν είναι σε θέση να ξεχωρίσει τι από τα δύο έχει συμβεί. Η χρήση των acknowledgments είναι σημαντική καθώς οι κόμβοι δε μπορούν να ανιχνεύσουν την ύπαρξη σύγκρουσης με το να "ακούνε" τη δική τους μετάδοση όπως συμβαίνει στις ενσύρματες επικοινωνίες [10].

2.3.1 Carrier Sense Mechanism

Οι μηχανισμοί ανίχνευσης φέροντος (carrier sense) που χρησιμοποιεί το 802.11 χωρίζονται σε δύο κατηγορίες: physical carrier sense και virtual carrier sense. Η πρώτη κατηγορία στην οποία αναφερθήκαμε παραπάνω, περιλαμβάνει τους μηχανισμούς με τους οποίους ένα τερματικό "ακούει" το κανάλι για να διαπιστώσει αν εκπέμπει ήδη κάποιος άλλος κόμβος και να αναβάλει τη δική του μετάδοση ή όχι και επομένως να εκπέμψει το δικό του πακέτο.

Ο virtual carrier sense μηχανισμός παρέχεται από το MAC και αναφερόμαστε σε αυτόν ως Network Allocation Vector (NAV). Ο NAV διατηρεί μια πρόβλεψη του χρόνου διάρκειας των εκπομπών που θα γίνουν στο μέλλον από άλλα stations βασιζόμενος στις πληροφορίες που υπάρχουν στο πεδίο Duration/ID πακέτων δεδομένων που μπορεί να "ακούσει" ο κόμβος. Όπως έχει ήδη αναφερθεί το πεδίο αυτό περιέχει μια πρόβλεψη για το πόσο θα διαρκέσει η μετάδοση ολόκληρου του πακέτου, λαμβάνοντας υπόψιν και το χρόνο που χρειάζεται για την αποστολή των απαραίτητων acknowledgments. Για όσο χρόνο λοιπόν ο NAV ενός κόμβου του υποδεικνύει ότι το κανάλι χρησιμοποιείται, αυτός δε θα προσπαθήσει να μεταδώσει [10].

Άλλος τρόπος ενημέρωσης του Network Allocation Vector γίνεται με τη χρήση των Request to Send (RTS) και Clear to Send (CTS) frames. Το 802.11 δεν καθιστά υποχρεωτική τη χρήση αυτών των frames. Προτού ένας κόμβος επιχειρήσει να στείλει κάποιο πακέτο δεδομένων, μπορεί να στείλει στον ίδιο παραλήπτη ένα RTS frame για να τον ενημερώσει ότι έχει δεδομένα που θέλει να του αποστείλει. Αν ο παραλήπτης γνωρίζει ότι το μέσον είναι αδρανές (μέσω του δικού του carrier sensing) θα απαντήσει με ένα CTS frame. Όταν ο αρχικός κόμβος λάβει αυτό το frame, θα μεταδώσει το πακέτο του. Τα RTS/CTS διαθέτουν κι αυτά πεδίο που περιέχει εκτίμηση του χρόνου που θα διαρκέσει η μετάδοση του πακέτου δεδομένων [10].

Τα RTS και CTS frames βοηθούν στην εξάλειψη του προβλήματος του κρυμμένου κόμβου (hidden node).

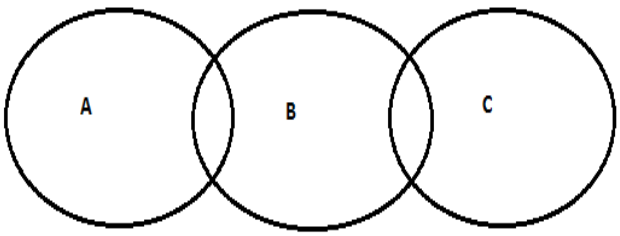


Figure 6-Hidden node problem.

Έστω ότι έχουμε τρεις υπολογιστές τους A, B, C. Ο A θέλει να στείλει δεδομένα στον B και μέσω του carrier sense νομίζει πως το κανάλι είναι αδρανές (ο C βρίσκεται μακριά κι έτσι ο A δε μπορεί να γνωρίζει για τις μεταδόσεις του). Αν όταν ο A επιχειρήσει να στείλει το πακέτο του στον B, έχουμε επίσης μετάδοση από τον C στον B, τότε θα υπάρξει σύγκρουση και τα πακέτα θα χαθούν.

Αν όμως είχαμε χρησιμοποιήσει τα RTS/CTS, τότε ο A θα έστελνε πρώτα στο B ένα RTS και αυτός με τη σειρά του θα του απαντούσε στέλνοντας ένα CTS. Αυτό το CTS θα το λάμβανε και ο C που με τη σειρά του θα ενημέρωνε κατάλληλα το NAV του και δε θα προσπαθούσε να μεταδώσει έως ότου ολοκληρωθεί η μεταφορά δεδομένων από τον A στον B.

Το virtual carrier sense χρησιμοποιείται σε συνδυασμό με το physical carrier sense. Και οι δύο αυτοί μηχανισμοί εξασφαλίζουν μικρότερη πιθανότητα ύπαρξης συγκρούσεων κατά την αποστολή πακέτων.

2.3.2 Backoff Algorithm

Στην περίπτωση που ένα station θέλει να μεταδώσει αλλά με τη χρήση του carrier sense μηχανισμού του αντιληφθεί ότι το κανάλι δεν είναι ελεύθερο, θα χρησιμοποιήσει τον αλγόριθμο backoff. Ο αλγόριθμος αυτός που χρησιμοποιείται και μετά από μια αποτυχημένη μετάδοση, περιλαμβάνει ένα επιπλέον χρονικό διάστημα που πρέπει ένα station να περιμένει (εκτός του DIFS) για να στείλει το πακέτο του. Το διάστημα αυτό ορίζεται ως εξής [10]:

```
Backoff Time = random x 1_slot_time
```

όπου:
random είναι ένας ψευδοτυχαίος ακέραιος ο οποίος επιλέγεται από μια ομοιόμορφη κατανομή στο διάστημα [0 , CW].

Για λόγους απόδοσης ο χρόνος που ακολουθεί αμέσως μετά από ένα DIFS διάστημα, χωρίζεται σε slots και μεταδόσεις από stations μπορούν να γίνουν μόνο στην αρχή των slots. Η διάρκειά τους ορίζεται έτσι ώστε σε χρόνο ενός slot, ένας κόμβος να μπορεί να αντιληφθεί τη μετάδοση κάποιου πακέτου από οποιοδήποτε άλλο κόμβο του δικτύου. Γενικά το slot time εξαρτάται από το φυσικό επίπεδο αλλά για τον υπολογισμό του πρέπει να ληφθεί υπόψιν ο χρόνος που χρειάζεται ένας κόμβος για να μεταβεί από κατάσταση λήψης σε κατάσταση εκπομπής (RX_TX_Turnaround_Time), ο χρόνος διάδοσης καθώς και ο χρόνος που απαιτείται ώστε να ενημερωθεί το MAC layer για την κατάσταση του καναλιού [11].

Το παράθυρο συνωστισμού (contention window) εξαρτάται από τον αριθμό των αποτυχημένων μεταδόσεων που έχουν συμβεί για ένα συγκεκριμένο πακέτο. Στην πρώτη προσπάθεια αποστολής του πακέτου το CW έχει τιμή CWmin. Μετά από κάθε ανεπιτυχής αποστολή του, η τιμή του contention window διπλασιάζεται μέχρι να φτάσουμε σε μια μέγιστη τιμή την CWmax [10][11].

PHY	Slot Time	CWmin	CWmax
FHSS	50μs	16	1024
DSSS	20μs	32	1024
IR	8μs	64	1024

FHSS: Frequency Hopping Spread Spectrum
DSSS: Direct Sequence Spread Spectrum
IR: Infrared

Figure 7-Slot time, min and max CW for the 3 PHY specified by 802.11 [11].

Για όσο χρόνο λοιπόν ο κόμβος ανιχνεύει πως το κανάλι είναι αδρανές, θα ελαττώνει το μετρητή του backoff αλγορίθμου. Σε περίπτωση που ανιχνευτεί κάποια μετάδοση, ο μετρητής θα σταματήσει και μόνο όταν το κανάλι είναι ελεύθερο για DIFS χρόνο, θα αρχίσει ξανά να μειώνεται. Όταν το backoff time φτάσει στο 0 θα γίνει η αποστολή του πακέτου και αν αυτή είναι επιτυχής, η τιμή του contention window θα γίνει ξανά CWmin [10][11].

Η τυχαιότητα που διέπει τον backoff αλγόριθμο οδηγεί σε λιγότερες συγκρούσεις σε περιπτώσεις συνωστισμού του καναλιού όπου πολλοί κόμβοι έχουν αναβάλει τη μετάδοσή τους επειδή το μέσον δεν είναι ελεύθερο.

Το Figure 5 μας δείχνει ένα απλοϊκό παράδειγμα πρόσβασης στο μέσο και χρησιμοποίησης του backoff αλγορίθμου:

- Ο κόμβος μας θέλει να μεταδώσει αλλά ανιχνεύει πως το κανάλι δεν είναι αδρανές και έτσι αναβάλει την αποστολή του.

- Θα πρέπει να περιμένει ώστε το κανάλι να είναι ελεύθερο για DIFS χρόνο.

- Αμέσως μετά ξεκινά τη διαδικασία του backoff αλγορίθμου. Επιλέγει ένα τυχαίο ακέραιο (από 0 έως CW) και για όσα slots το μέσον είναι αδρανές, ελαττώνει το μετρητή του.

- Όταν ο μετρητής φτάσει στο 0 ο κόμβος αποστέλλει το frame του.

2.3.3 Interframe Space

Το 802.11 standard ορίζει τεσσάρων ειδών interframe spaces. Αυτά είναι ανεξάρτητα από το ρυθμό μετάδοσης του station και έχουν σταθερή τιμή για κάθε συγκεκριμένο φυσικό μέσο (PHY):

Το Short Interframe Space (SIFS) χρησιμοποιείται για να διαχωρίζει τις μεταδόσεις πακέτων που δεν περιέχουν δεδομένα. Για παράδειγμα το SIFS χρησιμοποιείται όταν ένας κόμβος θέλει να στείλει ένα acknowledgment ως απάντηση στη λήψη ενός πακέτου δεδομένων ή όταν ένας κόμβος θέλει να μεταδώσει το επόμενο fragment ενός frame. Το SIFS είναι το μικρότερο

από τα interframe spaces και έτσι ο κόμβος που περιμένει για ένα SIFS έχει προτεραιότητα έναντι των άλλων που για να μεταδώσουν πρέπει να περιμένουν για μεγαλύτερο χρονικό διάστημα. Η τιμή του τίθεται τέτοια ώστε ένας κόμβος που μεταδίδει να προλάβει να μεταβεί σε λειτουργία δέκτη και να μπορέσει να αποκωδικοποιήσει το εισερχόμενο πακέτο. Για Frequency Hopping PHY (μέθοδος μετάδοσης ραδιοφωνικών σημάτων με συνεχή αλλαγή αλλαγή του φέροντος σε πολλά κανάλια συχνοτήτων) η τιμή του είναι 28μs [10][12].

Το Point Coordination Interframe Space (PIFS) εφαρμόζεται στην Point Coordination Function η οποία όμως δεν έχει υλοποιηθεί στους MadWIFI drivers που χρησιμοποιήθηκαν για την εκπόνηση της εργασίας. Με λίγα λόγια το PIFS χρησιμοποιείται από access points ώστε να αποκτήσουν πρόσβαση στο μέσο πριν από άλλους κόμβους [10][12].

```
PIFS=SIFS + 1_Slot_Time
```

Το Distributed Interframe Space (DIFS) όπως έχει αναφερθεί παραπάνω εφαρμόζεται όταν ένας κόμβος θέλει να ξεκινήσει τη μετάδοση ενός νέου πακέτου. Η τιμή του για Frequency Hopping PHY είναι σταθερή στα 128μs και γενικά δίνεται από τη σχέση [10][12]:

```
DIFS=SIFS + (2 x Slot_Time)
```

Τέλος το Extended Interframe Space (EIFS) χρησιμοποιείται όταν το φυσικό επίπεδο υποδείξει στο MAC ότι υπήρξε μετάδοση πακέτου το οποίο δε λήφθηκε σωστά. Το EIFS διάστημα θα ξεκινήσει όταν το φυσικό επίπεδο διαπιστώσει πως το μέσο είναι αδρανές, αγνοώντας το virtual carrier sense μηχανισμό. Το EIFS ορίζεται τόσο ώστε το άλλο τερματικό να αναγνωρίσει πως υπήρξε εσφαλμένη λήψη του πακέτου που έστειλε [12].

2.4 Frequency and Modulation

Το 802.11b standard σχεδιάστηκε για να λειτουργεί στη μπάντα των 2.4 Ghz. Το ίδιο ισχύει και για το 802.11g γιατί υπήρχε η επιθυμία οι συσκευές που χρησιμοποιούν 802.11g να είναι συμβατές με αυτές που χρησιμοποιούν 802.11b (λόγω της επιτυχίας του τελευταίου). Στον παρακάτω πίνακα φαίνονται τα κανάλια λειτουργίας και οι συχνότητες που χρησιμοποιούνται από τα 802.11b/g.

Channel	Lower Freq	Center Freq	Upper Freq
1	2.401	2.412	2.423
2	2.406	2.417	2.428
3	2.411	2.422	2.433
4	2.416	2.427	2.438
5	2.421	2.432	2.443
6	2.426	2.437	2.448
7	2.431	2.442	2.453
8	2.436	2.447	2.458
9	2.441	2.452	2.463
10	2.446	2.457	2.468
11	2.451	2.462	2.473

Figure 8-802.11b/g channels and frequencies [13].

Παρατηρώντας το Figure 8 φαίνεται ότι από τα 11 κανάλια λειτουργίας μόνο τρία μπορούν να χρησιμοποιούνται ταυτόχρονα χωρίς να υπάρχει παρεμβολή στις συχνότητες (τα 1, 6 και 11). Αυτό είναι και ένα από τα μεγαλύτερα μειονεκτήματα του 802.11b/g καθώς είναι πολύ δύσκολο να μην υπάρχουν παρεμβολές δεδομένου των πολλών συσκευών που χρησιμοποιούν αυτό το standard σήμερα.

Το 802.11a από την άλλη λειτουργεί στα 5 Ghz και δεν αντιμετωπίζει τα παραπάνω προβλήματα. Παρακάτω βλέπουμε τις συχνότητες λειτουργίας του.

Band	Channel Numbers	Frequency (MHz)
Lower Band	36	5180
	40	5200
	44	5220
	48	5240
Middle Band	52	5260
	56	5280
	60	5300
	64	5320
Upper Band	149	5745
	153	5765
	157	5785
	161	5805

Figure 9-802.11a channels and frequencies [14].

Οι ρυθμοί μετάδοσης (data rates) και η διαμόρφωση (modulation) είναι πανομοιότυπα στα 802.11a/g. Αυτά τα standards χρησιμοποιούν orthogonal frequency division multiplexing (OFDM). Στο OFDM η μπάντα χωρίζεται σε έναν αριθμό από sub-channels και σε κάθε ένα από αυτά στέλνονται ορισμένα από τα bits των δεδομένων. Ο πομπός κωδικοποιεί τις ροές των bits στους subcarriers κάνοντας χρήση των Binary Phase Shift Keying (BPSK), Quadrature Phase Shift Keying (QPSK) και Quadrature Amplitude Modulation (QAM) [15].

Data Rates (Mbps)	Modulation
6	BPSK
9	BPSK
12	QPSK
18	QPSK
24	16-QAM
36	16-QAM
48	64-QAM
54	64-QAM

Figure 10-802.11a/g modulation [16].

Στο 802.11b χρησιμοποιείται τεχνική ευρέως φάσματος με χρήση κατάλληλων ακολουθιών στο πεδίο του χρόνου (Direct Sequence Spread Spectrum)

Κεφάλαιο 3

Οδηγοί, Υλικό, Λογισμικό

Σε αυτό το κεφάλαιο θα περιγράψουμε τους οδηγούς (drivers), το λογισμικό και το υλικό που χρησιμοποιήθηκε για τη δημιουργία των rooftop κόμβων.

3.1 Οδηγός MadWifi

Το MadWifi project υλοποιήθηκε από μια ομάδα εθελοντών οι οποίοι εργάζονται πάνω στην υλοποίηση Linux drivers για ασύρματες κάρτες δικτύου που διαθέτουν chipsets της εταιρίας Atheros. Δημιουργός του MadWifi είναι ο Sam Leffler, του οποίου ο σκοπός ήταν να δημιουργήσει με αυτούς τους drivers μητροπολιτικά mesh ασύρματα δίκτυα. Επειδή όμως η δεν υπήρχε το απαιτούμενο hardware για την υλοποίηση του σχεδίου, ήρθε σε επαφή με την Atheros κι αυτή ανέλαβε την κατασκευή καρτών που θα είναι συμβατές με τους MadWifi drivers [17].

Τελικά ο Leffler εγκατέλειψε το project στις αρχές του 2005 και τότε διάφοροι άλλοι εθελοντές μπήκαν στην ομάδα για να συνεχιστεί η ανάπτυξή του. Σήμερα πλέον έχει σταματήσει η ανάπτυξη του MadWifi και οι εργασίες έχουν μετατοπιστεί στη δημιουργία των ath5k και ath9k drivers (οι οποίοι βασίζονται στο MadWifi) που μακροπρόθεσμα στοχεύουν να το αντικαταστήσουν παρέχοντας περισσότερες λειτουργικότητες. Ακόμα και σήμερα όμως το MadWifi αποτελεί έναν από τους πιο ανεπτυγμένους WLAN drivers που είναι διαθέσιμοι για τα Linux [17].

Η έκδοση των drivers που χρησιμοποιήσαμε για την εκπόνηση της εργασίας είναι η v0.9.4 (madwifi-ng). Για να τους κατεβάσουμε αρκεί να γράψουμε σε ένα τερματικό των Linux την παρακάτω εντολή [18]:

```
svn checkout http://svn.madwifi-project.org/madwifi/trunk/ madwifi-ng
```

Για να μπορέσουμε να χρησιμοποιήσουμε την παραπάνω εντολή θα πρέπει πρώτα να εγκαταστήσουμε σε κάθε κόμβο το πακέτο λογισμικού subversion (version control system)

```
apt-get install subversion
```

Τέλος ανάλογα με την έκδοση των Linux που διαθέτει ο κάθε κόμβος ίσως χρειαστεί να προσθέσουμε ακόμα διάφορα πακέτα ή και να αναβαθμίσουμε τον πυρήνα (Kernel) του λειτουργικού για να είμαστε βέβαιοι ότι δε θα προκύψει κάποιο πρόβλημα κατά την εγκατάσταση των υπόλοιπων οδηγών [18].

```
apt-get update && apt-get upgrade
```

```
apt-get install build-essential libssl-dev
```

```
apt-get install linux-headers-`uname -r`
```

Είναι συχνό φαινόμενο οι κάρτες δικτύου που είναι συμβατές με τους MadWifi drivers, να είναι συμβατές και με τους ath5k drivers. Επειδή οι νεότεροι Kernels έχουν ήδη εγκατεστημένους τους

ath5k, θέλουμε να τους αναγκάσουμε να μην τους χρησιμοποιήσουν έτσι ώστε να δουλέψουμε με τους δικούς μας drivers. Αυτό γίνεται με τις παρακάτω εντολές [18]:

```
echo "" >> /etc/modprobe.d/blacklist

echo "blacklist ath9k" >> /etc/modprobe.d/blacklist

echo "blacklist ath5k" >> /etc/modprobe.d/blacklist
```

Αφού κάνουμε όλα τα παραπάνω, η εγκατάσταση του MadWifi είναι πολύ απλή [17][18]:

```
make && make install
```

3.1.1 MadWifi Functionality

Σε αυτό το εδάφιο θα περιγράψουμε τις λειτουργικότητες που μας προσφέρουν αυτοί οι drivers για να γίνει κατανοητό γιατί επιλέξαμε τους συγκεκριμένους για την υλοποίηση του project.

Το MadWifi δίνει τη δυνατότητα στην κάρτα δικτύου μας να λειτουργήσει σε διάφορα modes. Μπορούμε λοιπόν να επιλέξουμε ένα από τα εξής:

station (sta): Η κάρτα μας λειτουργεί σαν ένας client και μπορεί να συνδεθεί σε κάποιο access point.

access point (ap): Παρέχει τη λειτουργικότητα ενός access point.

ad hoc: Το ad hoc χρησιμοποιείται για τη δημιουργία ενός Independent Basic Service Set, όπου οι κόμβοι επικοινωνούν μεταξύ τους χωρίς την παρουσία κάποιου access point.

monitor: Αυτό το mode δίνει στην κάρτα μας τη δυνατότητα να παρακολουθεί την κίνηση ενός ασύρματου δικτύου.

wireless distribution system: Παρέχει τη λειτουργικότητα ενός Distribution System, στο οποίο είχαμε αναφερθεί στο Κεφάλαιο 2. Το χρησιμοποιούμε όταν θέλουμε διαφορετικά access points να επικοινωνούν μεταξύ τους. Όταν επιλέγουμε αυτό το mode πρέπει να δώσουμε και τη MAC διεύθυνση της άλλης κάρτας δικτύου που βρίσκεται σε wds mode και με την οποία θέλουμε να επικοινωνήσει ο κόμβος μας.

Για να θέσουμε την κάρτα μας να λειτουργεί σε ένα από τα παραπάνω modes, αρκεί η εντολή [17]:

```
wlanconfig ath create wlandev wifi0 wlanmode sta/ap/adhoc/monitor/wds
```

όπου wifi0 το όνομα του physical interface (της ίδιας της κάρτας μας) και ath το όνομα που εμείς θέλουμε να δώσουμε στο interface που δημιουργούμε.

Το εργαλείο wlanconfig με το οποίο μπορούμε να προσθέσουμε, να διαγράψουμε και γενικά να χειριστούμε τις διεπαφές μας, εγκαθίσταται μαζί με τους drivers.

Αν τώρα θέλουμε να αλλάξουμε το mode λειτουργίας της κάρτας, πρέπει πρώτα να διαγράψουμε το υπάρχον interface και στη συνέχεια με την παραπάνω εντολή δημιουργούμε νέα διεπαφή στο mode

που θέλουμε. Η διαγραφή διεπαφής γίνεται με την εντολή [17]:

```
wlanconfig ath destroy
```

Ένα από τα μεγαλύτερα πλεονεκτήματα των MadWifi drivers είναι η δυνατότητα που παρέχουν για την ύπαρξη πολλαπλών διεπαφών (interfaces). Μπορούμε δηλαδή με ένα physical interface (με μία κάρτα δικτύου), να δημιουργήσουμε πολλά virtual interfaces και η κάρτα μας να λειτουργεί για παράδειγμα και σαν access point και σαν wireless distribution system και σαν station. Δεν υπάρχει κάποιος περιορισμός στον αριθμό των virtual interfaces που μπορούμε να δημιουργήσουμε κι έτσι για παράδειγμα έχουμε τη δυνατότητα να θέσουμε μια κάρτα να λειτουργεί σαν τρία διαφορετικά virtual access points που κάθε ένα από αυτά θα εξυπηρετεί διαφορετικές ομάδες πελατών.

Όσον αφορά την ασφάλεια, οι drivers μας επιτρέπουν τη χρήση του WEP αλγορίθμου με τη χρήση του iwconfig tool. Για μεγαλύτερη ασφάλεια καθώς ο WEP θεωρείται πλέον ξεπερασμένος, μπορούμε να χρησιμοποιήσουμε και WPA για κωδικοποίηση των δεδομένων. Αυτό όμως προϋποθέτει την ύπαρξη του hostap daemon στους κόμβους μας.

Το iwconfig tool μας επιτρέπει να ρυθμίσουμε και πολλές άλλες παραμέτρους της κάρτας δικτύου όπως ο ρυθμός μετάδοσης, το κανάλι λειτουργίας, το service set identifier που θα έχει ο κόμβος μας (αν είναι access point) ή του κόμβου με τον οποίο θέλουμε να συνδεθούμε (αν ο κόμβος μας είναι client), το RTS threshold (καθορίζει αν θα χρησιμοποιηθεί το RTS και ποιο είναι το ελάχιστο μέγεθος πακέτου για τα οποία θα το εφαρμόζουμε) και το fragmentation threshold (ανάλογο με το RTS threshold μόνο που εδώ ορίζουμε το μέγιστο μέγεθος ενός fragment). Αυτά πραγματοποιούνται με τις παρακάτω εντολές:

```
iwconfig ath rate X

iwconfig ath channel X

iwconfig ath rts X

iwconfig ath frag X
```

X είναι η παράμετρος που δίνει ο χρήστης (για παράδειγμα iwconfig ath rate 36M).

Το MadWifi εκτός των παραπάνω μας επιτρέπει τη ρύθμιση και πιο εξειδικευμένων παραμέτρων όπως είναι το antenna diversity, beacon interval, background scanning, DTIM period, bit rate selection algorithms.

Diversity: Αυτός ο όρος χρησιμοποιείται όταν ο δέκτης και ο πομπός έχουν περισσότερες από μία κεραίες για να βελτιώσουν τη ποιότητα τη λήψης/αποστολής.

Λειτουργία του receiver diversity: το ραδιόφωνο ακούει την αρχή της εισερχόμενης μετάδοσης (preamble) και συγκρίνει το signal strength στις κεραίες που διαθέτει. Στη συνέχεια επιλέγεται η κεραία με το καλύτερο signal strength και το σώμα του πακέτου λαμβάνεται από αυτήν. Η διαδικασία αυτή πραγματοποιείται στο hardware. Ο driver μπορεί μόνο να ενεργοποιήσει/απενεργοποιήσει το diversity [17].

Λειτουργία του transmitter diversity: Σε αυτή την περίπτωση ο driver πρέπει να πει στο ραδιόφωνο ποια κεραία να χρησιμοποιήσει. Αρχικά το ραδιόφωνο θα μεταδώσει σε ένα άλλο κόμβο μέσω της

default κεραίας του. Ταυτόχρονα θα γίνεται έλεγχος για το ποια κεραία χρησιμοποιείται για τη λήψη δεδομένων που αποστέλλονται από τον παραπάνω κόμβο. Όταν πραγματοποιηθούν τρεις συνεχόμενες λήψεις πακέτων από αυτόν τον κόμβο μέσω της ίδιας κεραίας, ο driver θα επιλέξει την κεραία λήψης σαν κεραία μετάδοσης [17].

Η εντολή με την οποία ελέγχουμε το diversity είναι [17]:

```
sysctl -w dev.wifi0.diversity=0/1
```

Με τιμή 1 ενεργοποιείται το diversity και με 0 απενεργοποιείται.

```
sysctl -w dev.wifi0.txantenna=1
sysctl -w dev.wifi0.rxantenna=1
```

Οι δύο παραπάνω εντολές χρησιμοποιούνται όταν έχουμε απενεργοποιήσει το diversity. Με αυτές επιλέγουμε σε ποια κεραία θα γίνεται λήψη δεδομένων και ποια θα χρησιμοποιείται για αποστολή (εδώ και για τις δύο εργασίες επιλέχθηκε η κεραία 1) [17].

Beacon Interval: Είναι το χρονικό διάστημα που μεσολαβεί μεταξύ των μεταδόσεων beacons από access points. Τα beacons έχουν σκοπό το συγχρονισμό των διάφορων client με το access point. Μεγάλο beacon interval ελαττώνει το overhead στο δίκτυο αλλά επίσης μειώνει την ικανότητα των χρηστών για roaming [17].

```
iwpriv ath bintval X
```

X είναι το χρονικό διάστημα σε ms.

Background Scanning: Εφαρμόζεται από clients οι οποίοι ψάχνουν για access points ίδιου service set identifier με αυτό που είναι ήδη συνδεδεμένοι. Σε περίπτωση που το νέο ap έχει καλύτερη ποιότητα σήματος, θα συνδεθούν με αυτό. Όταν το background scanning είναι ενεργό ο client προσωρινά θα σταματάει να λαμβάνει και να στέλνει δεδομένα για να ψάξει για άλλα διαθέσιμα access points. Η απενεργοποίηση του αυξάνει το συνολικό throughput αλλά μας στερεί τη δυνατότητα για roaming [17].

```
iwpriv ath bgscan 0/1
```

DTIM (Delivery Traffic Indication Message) Period: Το DTIM είναι ουσιαστικά ένας μετρητής που χρησιμοποιείται από τα access points για να ειδοποιήσουν τους clients που βρίσκονται σε power save mode ότι πρέπει να ενεργοποιηθούν και να αναμένουν broadcast και multicast πακέτα. Το Traffic Indication Message (TIM) περιέχεται σε κάθε beacon που στέλνεται από ένα ap. Σε κάθε DTIM period, το beacon θα περιέχει ένα DTIM αντί για TIM και τότε το ap θα στέλνει τα broadcast πακέτα που έχει αποθηκευμένα [10][17].

```
iwpriv ath dtim_period X
```

όπου X ο χρόνος σε ms.

Bit Rate Selection Algorithms: Επειδή τα 802.11a/b/g υποστηρίζουν πολλούς ρυθμούς μετάδοσης, πρέπει το MadWifi να διαθέτει κάποιο τρόπο ώστε να επιλέγει σε κάθε περίσταση τον πιο κατάλληλο. Υψηλότεροι ρυθμοί μετάδοσης προσφέρουν μεγαλύτερο throughput αλλά έχουν μεγαλύτερη πιθανότητα ύπαρξης σφάλματος. Το MadWifi διαθέτει τρεις διαφορετικούς αλγορίθμους για την επιλογή του bit rate.

1. ONOE: Αυτός ο αλγόριθμος ελαττώνει το ρυθμό μετάδοσης όταν τα πακέτα κατά μέσο όρο χρειάζονται να σταλούν ξανά τουλάχιστον μία φορά (δηλαδή αν το loss rate είναι μεγαλύτερο από 50%). Αυξάνει το ρυθμό μετάδοσης όταν λιγότερο από το 10% των πακέτων απαιτεί επαναμετάδοση. Ένας οποιοσδήποτε ρυθμός μετάδοσης του 802.11b θα έχει πάντα μεγαλύτερο throughput από το αμέσως μικρότερο bit rate έως ότου το loss rate γίνει μεγαλύτερο από 50%. Αυτό συμβαίνει διότι στο 802.11b ένα bit rate είναι σχεδόν διπλάσιο από το αμέσως μικρότερο και σχεδόν το μισό από το αμέσως μεγαλύτερο (1, 2, 5.5, 11 Mbps). Αυτός είναι και ο λόγος που ο ONOE λειτουργεί πολύ καλά σε 802.11b δίκτυα [17][19].

2. AMRR: Ο αλγόριθμος αυτός έχει τέσσερα ζεύγη bit-rate/transmission count r0/c0, r1/c1, r2/c2, r3/c3 όπου r είναι οι ρυθμοί μετάδοσης και c ο μέγιστος αριθμός συνολικών μεταδόσεων. Ο αρχικός ρυθμός μετάδοσης είναι r0. Αν η αποστολή ενός πακέτου αποτύχει θα γίνει προσπάθεια επαναμετάδοσής του για ακόμα c0-1 φορές. Αν δε μπορέσουμε να έχουμε επιτυχή μετάδοση, τότε ο ρυθμός μετάδοσης μειώνεται και γίνεται r1. Αυτή η διαδικασία συνεχίζεται έως ότου γίνουν c3 προσπάθειες αποστολής του πακέτου με bit rate r3. Αν και αυτές αποτύχουν η αποστολή ματαιώνεται. Σαν default τιμές το MadWifi χρησιμοποιεί: c0=4, c1=2, c2=2, c3=2. Το r3 τίθεται πάντοτε στο χαμηλότερο bit rate ενώ η τιμή του r0 είναι 11 Mbps για 802.11b και 24 Mbps για 802.11a/g. Ο ρυθμός μετάδοσης αυξάνεται μετά από συγκεκριμένο αριθμό επιτυχημένων μεταδόσεων [20].

3. SAMPLE: Ο SAMPLE αλγόριθμος ξεκινά στέλνοντας στο υψηλότερο δυνατό bit rate και το ελαττώνει όταν παρουσιαστούν τέσσερις συνεχόμενες αποτυχίες κατά την αποστολή ενός πακέτου. Αυτό συνεχίζεται έως ότου βρεθεί ένας ρυθμός μετάδοσης που είναι κατάλληλος για αποστολή πακέτων. Τότε σε κάθε δέκατο πακέτο δεδομένων, ο SAMPLE επιλέγει τυχαία ένα bit rate από αυτά που είναι ικανά να προσφέρουν μεγαλύτερο throughput από το υπάρχον και στέλνει το πακέτο με το νέο επιλεγμένο ρυθμό. Ένα bit rate δεν μπορεί να επιλεχθεί αν προηγουμένως έχουν υπάρξει τέσσερις αποτυχημένες προσπάθειες μετάδοσης με αυτό το ρυθμό, ή αν ο χρόνος μετάδοσης αυτού του bit rate θεωρώντας πάντοτε επιτυχημένες αποστολές, είναι μεγαλύτερος από το μέσο χρόνο του τρέχοντος bit rate. Για τον υπολογισμό αυτού του μέσου χρόνου μετάδοσης ο SAMPLE, που χρησιμοποιεί πληροφορίες από πακέτα που είχαν σταλεί στο παρελθόν, λαμβάνει υπ' όψιν του το μέγεθος του πακέτου, το ρυθμό μετάδοσης αλλά και τον αριθμό των επαναλήψεων που χρειάστηκαν για τη μετάδοσή του [19].

Στο MadWifi σαν προεπιλογή χρησιμοποιείται ο αλγόριθμος SAMPLE. Για να αλλάξουμε τον bit rate selection algorithm χρησιμοποιούμε την εντολή [17]:

```
make ATH_RATE=ath_rate/sample
```

Στη θέση του sample μπορούμε φυσικά να βάλουμε είτε onoe είτε amrr.

3.2 Λογισμικό

3.2.1 Bridge Utilities

Για την υλοποίηση της εργασίας εγκαταστήσαμε σε όλους του κόμβους τα bridge utilities με την εντολή:

```
apt-get install bridge-utils
```

Αυτά μας επιτρέπουν τη δημιουργία ενός bridge interface, που το χρησιμοποιούμε έτσι ώστε διαφορετικά network interfaces του ίδιου κόμβου, να μπορούν να επικοινωνήσουν μεταξύ τους. Το bridging λειτουργεί στο επίπεδο 2 και χρησιμοποιεί MAC διευθύνσεις για να μεταφέρει γρήγορα τα δεδομένα μεταξύ των διάφορων ports που διαθέτει ένα οποιοδήποτε bridge. Ένα bridge λοιπόν θα μάθει τις MAC διευθύνσεις όλων των συσκευών που είναι συνδεδεμένες στα ports του και αφού κάθε MAC frame περιέχει τις MAC του αποστολέα και παραλήπτη, θα γνωρίζει σε ποιο port θα προωθήσει το εκάστοτε frame. Το bridges τα χρησιμοποιήσαμε ώστε να επικοινωνούν τα access point με τις wireless distribution system διεπαφές, αλλά και οι τελευταίες μεταξύ τους.

3.2.2 Remote Programmability

Ένα άλλο σημαντικό κομμάτι του project μας ήταν να μπορέσουμε να έχουμε απομακρυσμένη πρόσβαση στους κόμβους του δικτύου για να μπορούμε να τους προγραμματίσουμε ξανά ή απλώς να αλλάξουμε τις παραμέτρους της κάρτας δικτύου. Υπάρχουν διάφορα προγράμματα που μας δίνουν αυτή τη δυνατότητα.

1. Vinagre: Το πρόγραμμα αυτό υπάρχει προ εγκατεστημένο στις εκδόσεις Ubuntu που χρησιμοποιήσαμε στους κόμβους μας (Ubuntu 10.04 Long-Term Support και Ubuntu 9.10). Το Vinagre είναι ένας VCN (Virtual Network Computing) client για Gnome desktops. Μας επιτρέπει την πρόσβαση σε άλλους υπολογιστές που ανήκουν όμως στο ίδιο τοπικό δίκτυο με μας. Η σύνδεση γίνεται με την εντολή:

```
vinagre X
```

Χ είναι είτε η ip διεύθυνση του κόμβου που θέλουμε να αποκτήσουμε πρόσβαση, είτε το όνομά του.

Στη συνέχεια απαιτείται να δώσουμε ένα password που έχει ορίσει ο απομακρυσμένος κόμβος. Φυσικά το Vinagre δε μας περιορίζει μόνο στο να δούμε το desktop του άλλου υπολογιστή, αλλά και να αλληλεπιδράσουμε μαζί του. Αυτό το πρόγραμμα περιορίζεται σε τοπικά δίκτυα κι έτσι αν θέλουμε να αποκτήσουμε απομακρυσμένη πρόσβαση σε άλλο κόμβο μέσω του internet, πρέπει να καταφύγουμε σε άλλες λύσεις.

2. Secure Shell (SSH): Είναι ένα πρωτόκολλο δικτύου για ασφαλή επικοινωνία υπολογιστών. Στους κόμβους μας εγκαταστήσαμε το Open SSH που είναι μια δωρεάν έκδοση του SSH. Στους Rooftop κόμβους εγκαταστήσαμε τον Open SSH server και στους clients τον Open SSH client.

```
apt-get install openssh-server

apt-get install openssh-client
```

Σε αντίθεση με το Vinagre δε μπορούμε να δούμε το desktop του απομακρυσμένου υπολογιστή, αλλά με το SSH αποκτούμε πρόσβαση στο terminal αυτού του κόμβου άρα και σε όλες τις λειτουργίες του.

Ρυθμίσεις των παραμέτρων του SSH μπορούν να γίνουν πειράζοντας το sshd_config αρχείο. Μπορούμε λοιπόν να καθορίσουμε ποιοι λογαριασμοί θα χρησιμοποιούν το SSH, αν θα έχουμε password authentication ή public key authentication και διάφορες άλλες λειτουργίες [21].

Μετά από οποιαδήποτε αλλαγή στο sshd_config αρχείο, πρέπει να ξεκινήσουμε ξανά το SSH:

```
sudo /etc/init.d/ssh restart
```

Η σύνδεση σε άλλους υπολογιστές γίνεται με την εντολή:
`ssh X`

X είναι η IP του απομακρυσμένου κόμβου ή το όνομά του.

Όσον αφορά τη μέθοδο του public key authentication, την οποία και εφαρμόσαμε στο project, κάθε client διαθέτει ένα δημόσιο και ένα ιδιωτικό κλειδί. Το ιδιωτικό κλειδί διατηρείται στον υπολογιστή από τον οποίο θα έχουμε πρόσβαση στους κόμβους του δικτύου, ενώ καθένας από αυτούς τους κόμβους έχει αποθηκευμένο το δημόσιο κλειδί στο αρχείο .ssh/authorized_keys. Όταν προσπαθούμε να αποκτήσουμε πρόσβαση σε ένα SSH server, αυτός χρησιμοποιεί το δημόσιο κλειδί ώστε να κρυπτογραφήσει τα μηνύματα με τέτοιο τρόπο που μόνο η χρήση του ιδιωτικού κλειδιού να μπορεί να εφαρμοστεί για την αποκρυπτογράφησή τους. Η λεπτομερής διαδικασία για την εξαγωγή του δημοσίου και ιδιωτικού κλειδιού και χρήση τους, περιγράφεται στο [21].

Για να αποκτήσουμε με SSH πρόσβαση σε κόμβο εκτός του τοπικού μας δικτύου (μέσω internet) πρέπει να αλλάξουμε τις ρυθμίσεις του router που χρησιμοποιούν οι κόμβοι του rooftop δικτύου.

3. TeamViewer: Τελικά για να καλύψουμε τις ανάγκες του remote programmability χρησιμοποιήσαμε το TeamViewer τόσο για λόγους απλότητας στη χρήση του, όσο και γιατί είναι διαθέσιμο και για Windows αλλά και για Linux. Είναι παρόμοιο με το Vinagre αφού και με αυτό το πρόγραμμα βλέπουμε το desktop του απομακρυσμένου κόμβου αλλά το TeamViewer επιτρέπει σύνδεση υπολογιστών μέσω internet. Κάθε υπολογιστής διαθέτει ένα ζεύγος από user name και password. Όταν θέλουμε να χρησιμοποιήσουμε το TeamViewer απλά δίνουμε το user name του άλλου υπολογιστή και το password. Το TeamViewer είναι δωρεάν για μη εμπορική χρήση.

Στο παρακάτω screen shot έχουμε συνδεθεί με χρήση του TeamViewer σε έναν από τους backbone κόμβους του δικτύου μας.

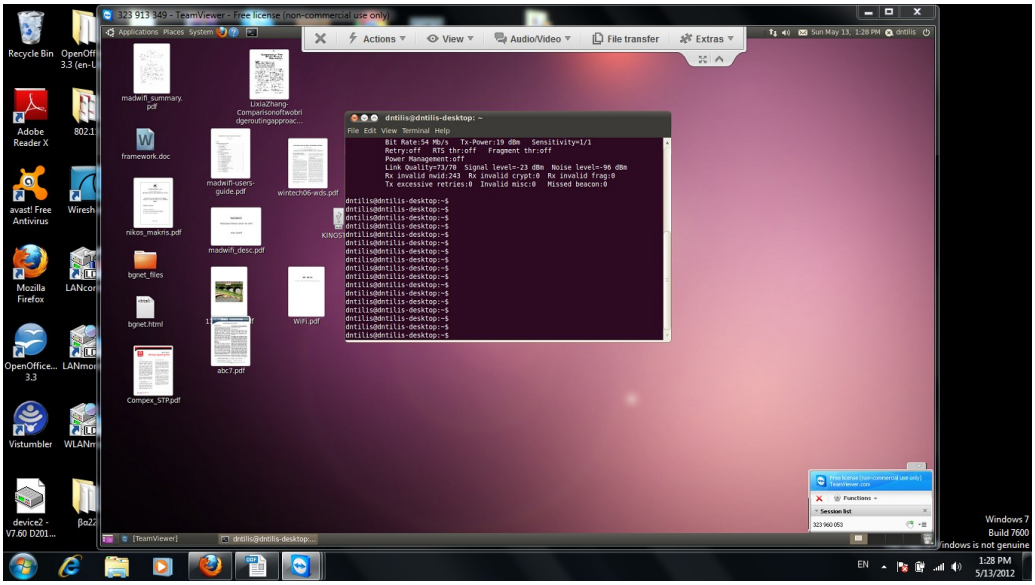


Figure 11-Remote Control ενός Rooftop node με TeamViewer.

3.3 Οι οδηγοί Ath5k και Ath9k

Ath5k: Οι ath5k drivers δημιουργήθηκαν από την ομάδα που ανέπτυξε και το MadWifi με σκοπό να αντικαταστήσουν το τελευταίο. Στους ath5k έχει αντικατασταθεί το closed source κομμάτι του MadWifi με νέο open source κώδικα. Αυτό το κομμάτι του MadWifi ονομάζεται Hardware Abstraction Layer (HAL) και αποτελεί το λογισμικό από το οποίο δρομολογείται όλη η άμεση πρόσβαση στο hardware της Atheros [6].

Οι ath5k σε αρκετές λειτουργίες τους είναι παρόμοιοι με τους drivers που χρησιμοποιήσαμε όπως η δημιουργία διάφορων αλλά και πολλαπλών virtual interfaces, επειδή όμως αναπτύχθηκαν πιο πρόσφατα, είναι συμβατοί με περισσότερες και φυσικά με τις νέες κάρτες δικτύου που δημιουργούνται με Atheros chipset.

Ένα σημαντικό πλεονέκτημα των ath5k είναι η υποστήριξη του 802.11s δηλαδή του mesh mode, πράγμα που δεν είχε υλοποιηθεί στους MadWifi drivers. Ορισμένα από τα χαρακτηριστικά του 802.11s που το καθιστούν επιθυμητό είναι τα εξής:

- προσφέρει κάλυψη σε περιοχές που είναι δύσκολο να δικτυωθούν με καλώδια

- εύκολη και με χαμηλό κόστος η δημιουργία ενός mesh δικτύου

- η αποκατάσταση διαδρομών είναι δυνατή (self-healing)

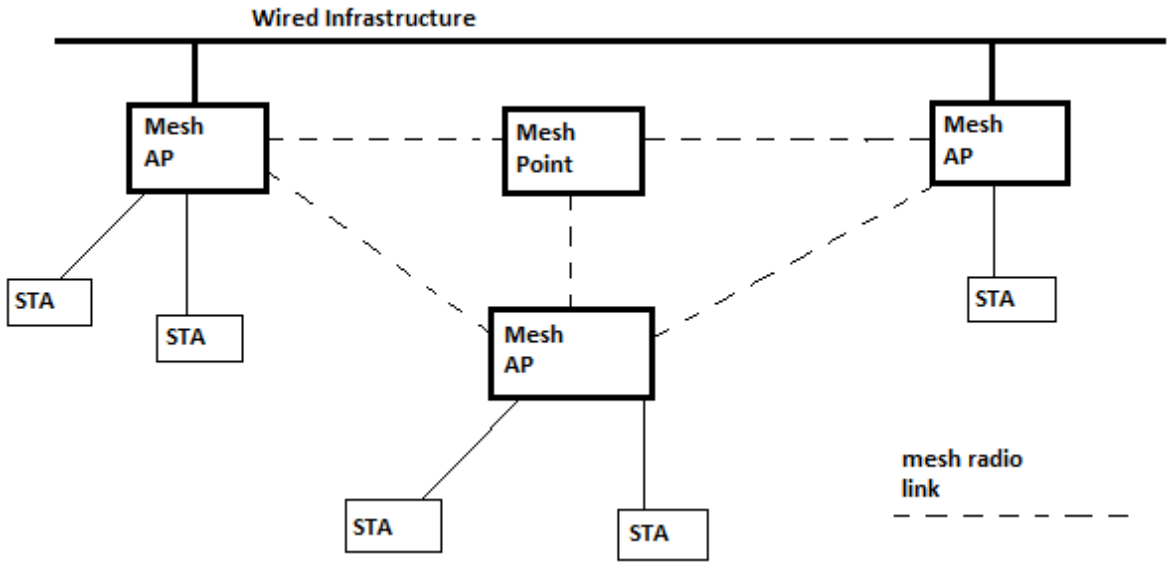


Figure 12-802.11s network [22].

Ath9k: Οι ath9k είναι drivers για Atheros PCI και PCI Express ασύρματες κάρτες δικτύου που χρησιμοποιούν το 802.11n standard. Το 802.11n, που αποτελεί και το λόγο ανάπτυξης αυτών των driver, έχει στόχο να βελτιώσει το throughput σε σχέση με τα 802.11a/g standards προσφέροντας μέγιστο ρυθμό μεταφοράς δεδομένων που φτάνει τα 600 Mbps (πολύ μεγαλύτερος από τα 54 Mbps που παρέχουν τα 802.11a/g) και επιτυγχάνεται με χρήση πολλαπλών κεραιών.

3.4 Υλικό του Rooftop Κόμβου

Η αρχική σκέψη ήταν να χρησιμοποιήσουμε laptops σαν κόμβους του συστήματος αναμετάδοσης (relay system) που αργότερα θα δημιουργούσαμε. Δυστυχώς τα HP Presario CQ56 που είχε στη διάθεσή του το εργαστήριο τηλεπικοινωνιών δε διαθέτουν θύρα PCMCIA για να εισάγουμε κάρτα δικτύου συμβατή με τους MadWifi drivers (η εσωτερική κάρτα τους δεν είναι συμβατή). Έτσι στραφήκαμε στη λύση των desktop υπολογιστών αφού οι drivers μας δε λειτουργούν με καμία usb κάρτα δικτύου.

Το πρώτο που έπρεπε να κάνουμε είναι να βρούμε μια PCI κάρτα δικτύου που να διαθέτει Atheros chipset. Τελικά καταλήξαμε στην TP-LINK TL-WN350GD η οποία χρησιμοποιεί τα 802.11b/g standard ενώ το chipset που διαθέτει είναι το AR5007G που κατασκευάζεται από την Atheros. Αυτή η κάρτα είναι συμβατή με MadWifi αλλά και με τους ath5k drivers. Επίσης το κόστος της φτάνει περίπου μόλις τα 8 ευρώ οπότε ήταν μια πολύ καλή λύση για τους σκοπούς μας.

Στην παρακάτω εικόνα φαίνεται αυτή η κάρτα.

Figure 13-TP-LINK TL-WN350GD Wireless Card.

Λόγω του ότι το MadWifi δεν επιτρέπει σε virtual interfaces του ίδιου physical interface να λειτουργούν σε διαφορετικό κανάλι συχνοτήτων, αγοράσαμε δύο από τις παραπάνω κάρτες για κάθε ένα από τους κόμβους μας. Αυτό που θέλαμε να πετύχουμε είναι τα wireless distribution system interfaces όλων τον κόμβων να λειτουργούν σε ένα κοινό κανάλι, ενώ το access point interface κάθε κόμβου να λειτουργεί σε κανάλι συχνοτήτων διαφορετικό από το άλλο. Έτσι μειώνονται οι παρεμβολές ανάμεσα στις 802.11 συσκευές. Βέβαια σε ένα μεγάλο δίκτυο αναγκαστικά θα υπάρχουν access points που θα χρησιμοποιούν το ίδιο κανάλι μιας και όπως έχει ήδη αναφερθεί στη μπάντα των 2.4 GHz υπάρχουν μόνο τρία μη επικαλυπτόμενα κανάλια.

Για τη δημιουργία του δικτύου μας χρησιμοποιήσαμε συνολικά τέσσερις υπολογιστές κι ένα smartphone (iPhone). Οι δύο ε'ιχαν εγκατεστημένη την έκδοση 10.04 LTS των Ubuntu ενώ στους άλλους δύο χρησιμοποιήθηκε η 9.10 έκδοση.

Με εξαίρεση τις κάρτες δικτύου δεν υπάρχει κάποιος άλλος περιορισμός ως προς το hardware. Θα μπορούσαμε να χρησιμοποιήσουμε οποιονδήποτε υπολογιστή που απλά έχει εγκατεστημένη κάποια έκδοση των Linux.

Μπορεί οι κόμβοι του δικτύου να απαιτούν Linux για τη λειτουργία του relay system, δε συμβαίνει όμως το ίδιο και με τους clients. Δηλαδή ένας υπολογιστής δεν είναι αναγκαίο να λειτουργεί σε Linux για να συνδεθεί σε ένα MadWifi access point.

Κεφάλαιο 4

Συνδεσιμότητα με Γέφυρες στο Επίπεδο 2 (Bridging)

Σε αυτό το κεφάλαιο περιγράφουμε το Address Resolution Protocol, την έννοια της γέφυρας στο επίπεδο 2, τα πιθανά προβλήματα βρόγχων που μπορούν να δημιουργηθούν και τέλος, το Spanning Tree Protocol όπως το χρησιμοποιήσαμε.

4.1 Address Resolution Protocol

Το Address Resolution Protocol (ARP) είναι ένα πρωτόκολλο το οποίο προσδιορίζει τη διεύθυνση μιας συσκευής στο επίπεδο ζεύξης (link layer), με βάση τη διεύθυνση της συγκεκριμένης συσκευής στο επίπεδο δικτύου (network layer). Το ARP είναι σχεδιασμένο για να χρησιμοποιείται από οποιοδήποτε link layer ή network layer πρωτόκολλο. Στην πράξη όμως χρησιμοποιείται μόνο για το Ethernet (συμπεριλαμβανομένου και του 802.11) και το IPv4.

Όπως είδαμε στα frame formats του Κεφαλαίου 2, για να πραγματοποιηθεί η μετάδοση ενός πακέτου μέσω Wi-Fi, πρέπει να είναι γνωστή στον αποστολέα η διεύθυνση MAC του παραλήπτη. Σε περίπτωση που αυτή δεν είναι γνωστή, ο αποστολέας θα κάνει broadcast ένα ARP Request που θα περιέχει την IP διεύθυνση του υπολογιστή του οποίου θέλει να μάθει τη MAC. Όταν αυτό το request ληφθεί από τον κατάλληλο υπολογιστή, αυτός θα απαντήσει με ένα unicast ARP Reply στο οποίο θα αναφέρει τη MAC του. Τα ARP Requests περιέχουν και τη MAC του αποστολέα και έτσι ο παραλήπτης γνωρίζει σε ποιο κόμβο πρέπει να στείλει το Reply.

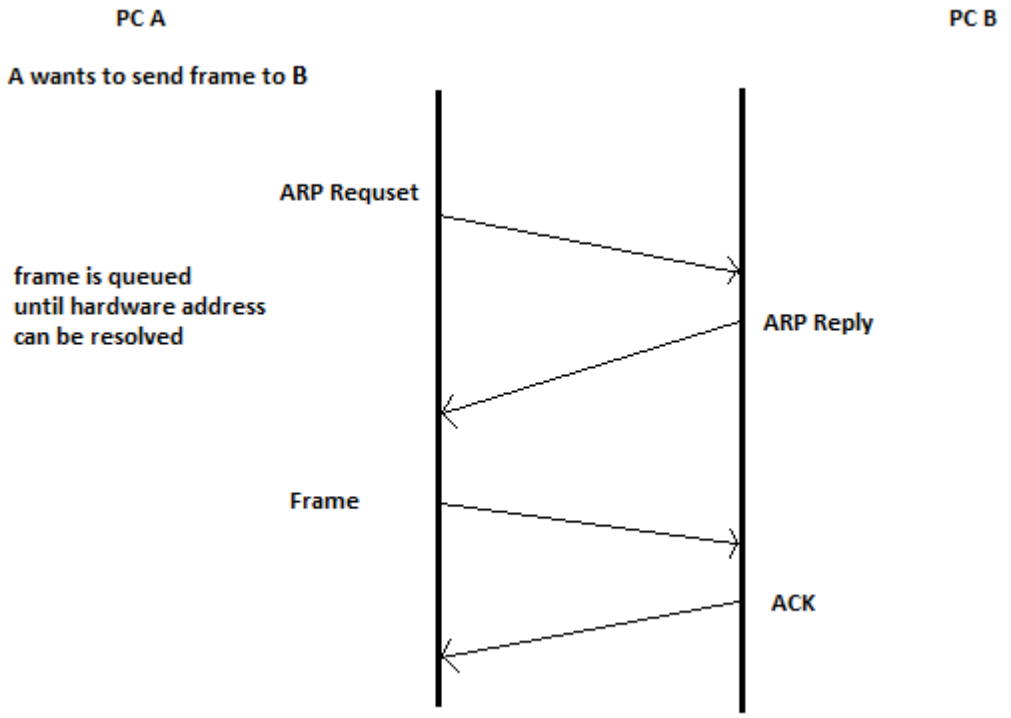


Figure 14-ARP Example [23].

Όλοι οι κόμβοι διαθέτουν μια προσωρινή μνήμη ARP (ARP cache) η οποία περιέχει τα ζεύγη IP-MAC διευθύνσεων που γνωρίζει ο εκάστοτε υπολογιστής. Αυτό συντελεί στη μείωση των ARP Requests που πρέπει να σταλούν. Η cache δεν έχει απεριόριστο μέγεθος και για αυτό πρέπει περιοδικά να διαγράφονται από αυτήν εγγραφές που δε χρησιμοποιούνται πλέον.

4.2 Bridging

Μια γέφυρα (bridge) χρησιμοποιείται για τη διασύνδεση διάφορων network segments στο επίπεδο ζεύξης (επίπεδο 2) του OSI μοντέλου. Το είδος του bridging που εφαρμόσαμε στα πλαίσια της εργασίας μας είναι το λεγόμενο transparent bridging που κυριαρχεί στα Ethernet και Wi-Fi δίκτυα. Σε αυτό κάθε πακέτο προωθείται ένα άλμα τη φορά προς τον προορισμό του.

Σε αντίθεση με τη δρομολόγηση (routing) επιπέδου δικτύου (επίπεδο 3 του OSI) που χρησιμοποιεί IP διευθύνσεις για τη δρομολόγηση, τα bridges βασίζονται στις MAC διευθύνσεις των συσκευών για να αποφασίσουν προς ποια κατεύθυνση πρέπει να προωθήσουν το εκάστοτε πακέτο.

Κάθε bridge “ακούει” όλα τα frames που στέλνονται ή λαμβάνονται από τις συσκευές δικτύου που είναι συνδεδεμένες και αποφασίζει σε ποια θύρα (port) θα προωθήσει το κάθε ένα. Για να γίνει αυτό κάθε bridge διαθέτει ένα πίνακα προώθησης ο οποίος αντιστοιχίζει τις MAC διευθύνσεις των συσκευών του δικτύου με τα κατάλληλα ports στα οποία πρέπει να προωθηθεί ένα frame προκειμένου να φτάσει στο προορισμό του (ο οποίος φαίνεται από τα address fields του MAC frame). Αρχικά ο πίνακας αυτός είναι κενός αλλά καθώς το bridge λαμβάνει frames, εισάγονται και τα κατάλληλα δεδομένα και τελικά ο πίνακας παίρνει μια μορφή όπως αυτή που φαίνεται στο Figure 15.

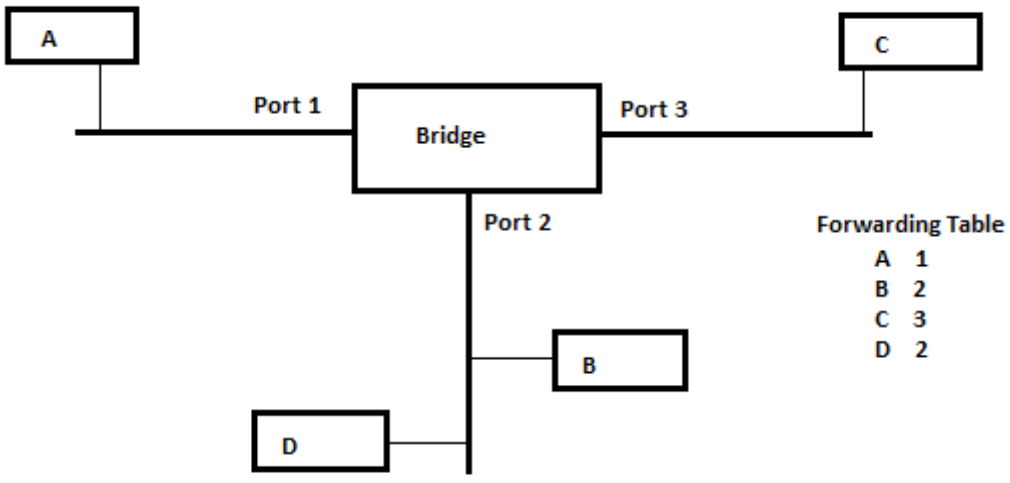


Figure 15-Bridging uses Forwarding Tables [24].

Τα A, B, C, D στο Figure 15 αναφέρονται στις MAC διευθύνσεις των συσκευών A, B, C, D.

Μένοντας στο παραπάνω Figure, βλέπουμε ότι αν το bridge λάβει ένα frame από τη συσκευή A θα εξετάσει το SA field του μαθαίνοντας τη MAC διεύθυνση του A, και έτσι θα ανανεώσει το forwarding table αφού το A βρίσκεται στο port 1. Οι καταχωρήσεις στον πίνακα προώθησης διαγράφονται αν δεν ανανεωθούν για κάποιο χρονικό διάστημα. Στην περίπτωση που το bridge λάβει ένα frame του οποίου η destination address δε βρίσκεται στο forwarding table, θα το κάνει broadcast σε όλα τα υπόλοιπα ports [24].

Η μέθοδος που περιγράψαμε παραπάνω για το πώς ένα bridge μαθαίνει τις MAC διευθύνσεις των διαφόρων συσκευών, μπορεί να επεκταθεί και σε ένα δίκτυο που διαθέτει περισσότερα του ενός bridges.

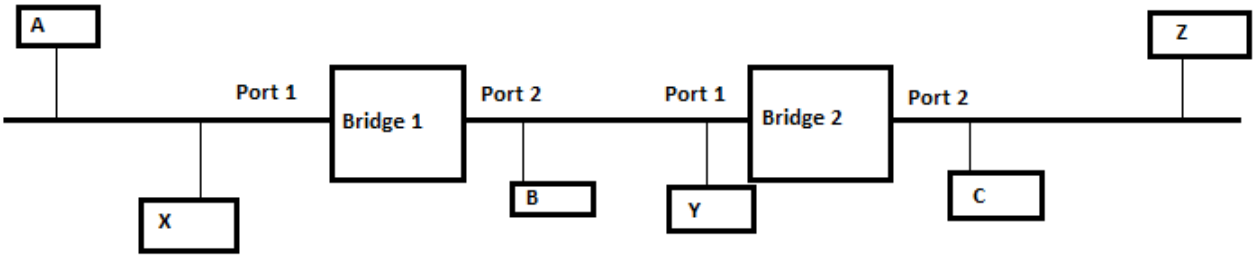


Figure 16-Δίκτυο από γέφυρες [24].

Για παράδειγμα στο Figure 16 το forwarding table για το bridge 2 είναι το παρακάτω:

A 1
X 1
B 1
Y 1
C 2
Z 2

Η εικόνα λοιπόν που το bridge 2 έχει για το δίκτυο, είναι η εξής:

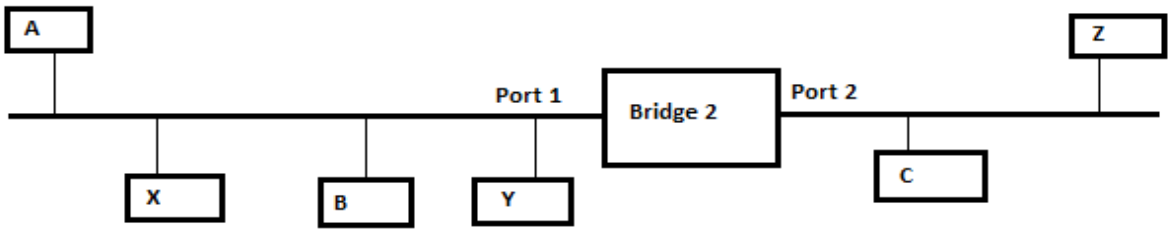


Figure 17-Πώς βλέπει μια γέφυρα το παραπάνω δίκτυο [24].

Το bridge 2 δηλαδή αγνοεί την ύπαρξη του bridge 1. Έτσι για παράδειγμα αν το Α θέλει να στείλει ένα frame στο Z, αυτό που ενδιαφέρει το bridge 2 είναι ότι το Α βρίσκεται στο port 1 και ότι για να φτάσει το frame στο Z πρέπει να προωθηθεί μέσω του port 2.

4.2.1 Loops in the Topology

Η παραπάνω διαδικασία εκμάθησης των MAC διευθύνσεων λειτουργεί προβληματικά όταν υπάρχουν loops στην τοπολογία του δικτύου μας. Έστω πως έχουμε το παρακάτω δίκτυο:

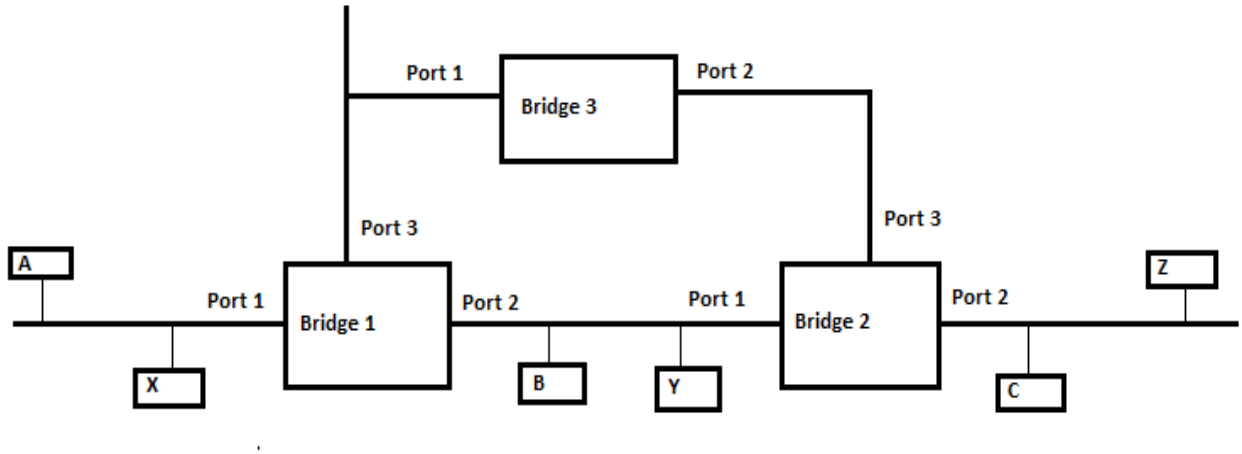


Figure 18-Loop in the network topology [24].

Μελετάμε την περίπτωση που το Α θέλει να στείλει frame στο Β και θεωρούμε ότι αρχικά τα forwarding tables είναι κενά.

- Το bridge 1 θα μάθει ότι το Α βρίσκεται στο port 1 και αφού το forwarding table του είναι κενό, θα στείλει το frame στα ports 2 και 3.

- Το bridge 3 θα το προωθήσει στο port 2 και το bridge 2 στα ports 1 και 2.

- Το bridge 1 θα μάθει τώρα ότι το Α βρίσκεται στο port 2 και θα στείλει το frame στα ports 1 και 3.

- Συνεχίζεται αυτή η διαδικασία και δημιουργούνται πολλά αντίγραφα του ίδιου frame αρκετά από τα οποία θα ληφθούν από το Β.

Γίνεται προφανές λοιπόν πως η ύπαρξη βρόγχων αν και προσφέρει μεγαλύτερη αξιοπιστία καθώς υπάρχουν περισσότερα μονοπάτια μεταξύ δύο κόμβων, καθιστά αδύνατη την ομαλή λειτουργία του δικτύου.

Για την αποφυγή του προβλήματος των βρόγχων τα bridges διατηρούν μια ενεργή τοπολογία που δεν έχει loops, με το να απενεργοποιούν κάποια από τα ports τους.

4.2.2 Spanning Tree Protocol

Το Spanning Tree Protocol (STP) εφαρμόζεται από τα bridges ώστε να αποφεύγονται τα loops στην τοπολογία. Αυτό αποφασίζει ποια ports θα απενεργοποιηθούν και ποια θα παραμείνουν σε forwarding state. Επίσης πρέπει να είναι ικανό να προσαρμόζεται σε αλλαγές στην τοπολογία του δικτύου όπως για παράδειγμα αν συμβεί βλάβη σε έναν από τους κόμβους.

Η εφαρμογή του STP σε ένα δίκτυο από bridges έχει ως αποτέλεσμα μια τοπολογία στην οποία υπάρχει μόνο ένα δυνατό μονοπάτι από έναν κόμβο προς οποιονδήποτε άλλο (δηλαδή ένα δένδρο). Αυτή μπορεί να αναπαρασταθεί ως ένας γράφος όπου κάθε vertex αντιπροσωπεύει ένα bridge και κάθε edge μια σύνδεση μεταξύ των bridges.

Το STP καθορίζει ότι τα ports ενός bridge θα είναι blocked, root ή designated [24]:

Blocked Ports: Τα ports που έχουν απενεργοποιηθεί από το STP για την αποφυγή βρόγχων.

Root Port: Υπάρχει ένα root port σε κάθε bridge. Αυτό το port οδηγεί προς το root του spanning tree (εξηγούμε παρακάτω πως γίνεται η επιλογή του root).

Designated Ports: Το port ενός link που βρίσκεται πιο κοντά προς τη ρίζα του δένδρου ορίζεται ως designated port. Κάθε link μεταξύ δύο bridges διαθέτει και ένα designated port. Γενικά κάθε port που δεν είναι root ή blocked, είναι designated.

Ας υποθέσουμε ότι στο Figure 18 το bridge 2 είναι το root του spanning tree και πως το port 1 του bridge 3 είναι blocked για να μη δημιουργηθεί βρόγχος στο δίκτυο:

-Όλα τα ports του bridge 2 είναι designated αφού βρίσκονται στο συντομότερο μονοπάτι προς τη ρίζα.

-Το port 2 του bridge 3 είναι root port αφού οδηγεί προς το root του stp. Δεν είναι designated port αφού στο link μεταξύ bridge2 και bridge 3 το port 3 του bridge 2 βρίσκεται πιο κοντά στο root.

-Το port 2 του bridge 1 είναι root port και όλα τα υπόλοιπα είναι designated ports.

Σε κάθε bridge δίνεται κατά τη δημιουργία του ένα label που εξαρτάται από τη MAC διεύθυνση του ίδιου του bridge συν ένα offset το οποίο μπορούμε αν θέλουμε να παραμετροποιήσουμε. Το bridge με το μικρότερο label επιλέγεται ως root του spanning tree. Επίσης κάθε link μεταξύ δύο bridges έχει ένα κόστος που είναι αντιστρόφως ανάλογο με το ρυθμό μετάδοσης του συγκεκριμένου link.

Το spanning tree protocol υπολογίζει ένα δέντρο συντομότερων μονοπατιών προς τη ρίζα. Για να το κάνει αυτό χρησιμοποιεί μια παραλλαγή του Bellman-Ford αλγορίθμου. Αρχικά θα μελετήσουμε την περίπτωση του centralized αλγορίθμου (το STP χρησιμοποιεί distributed Bellman-Ford).

Centralized Bellman-Ford for Bridges: Όπως έχουμε ήδη αναφέρει κάθε δίκτυο αναπαρίσταται ως ένας γράφος όπου κάθε ακμή (edge) έχει κάποιο κόστος και κάθε κόμβος (vertex) ένα label. Για να αποφασίσουμε αν ένα μονοπάτι είναι καλύτερο από κάποιο άλλο πρέπει να γνωρίζουμε τα link attributes. Το link attribute μεταξύ δύο κόμβων i και j ορίζεται ως [24]:

A(i,j) = [l(j) , c(i,j)]

Το l(j) είναι το label ενός κόμβου (MAC διεύθυνση + offset) που βρίσκεται στο edge (i,j) και το c(i,j) το κόστος ενός link. Θεωρούμε ότι ισχύουν τα παρακάτω [24]:

c(i,j) > 0
c(i,j) = ∞, αν i και j δε συνδέονται.

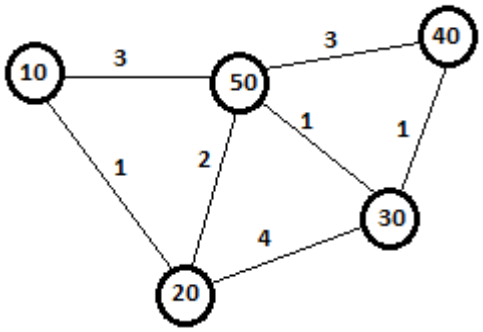
Το attribute ενός μονοπατιού, είναι η συνένωση (concatenation - ⊕) των attributes των links από τα οποία αποτελείται. Έτσι αν είχαμε δύο links με attributes [l,c] και [l',c'], το attribute του μονοπατιού που περιλαμβάνει τα δύο links θα ήταν [24]:

[l ,c] ⊕ [l' , c'] = [min(l , l') , c + c']

Σύγκριση των attributes [24]:

[l , c] ≤ [l' , c'] αν και μόνο αν [(l < l') or (l = l' and c ≤ c')]

Ένα μονοπάτι p είναι καλύτερο από ένα άλλο p' αν το attribute του p είναι μικρότερο από αυτό του p'.
Στην παρακάτω εικόνα φαίνεται ένα παράδειγμα υπολογισμού κόστους διάφορων μονοπατιών.



20 30 50 [30,5]
10 20 30 [20,5]
20 10 50 40 [10,7]
20 10 50 30 [10,5]

Figure 19-Examples of Path Attributes [24].

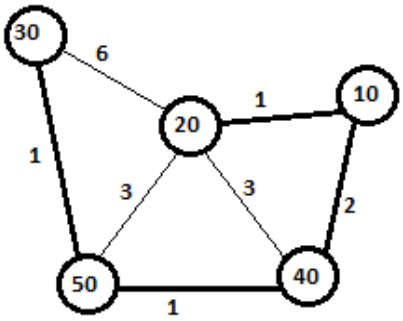
Στα παραδείγματα του Figure 19, το τελευταίο μονοπάτι είναι το καλύτερο γιατί έχει το μικρότερο label και το κόστος του είναι μικρότερο από αυτό του τρίτου path που έχει το ίδιο label.

Ο κλασικός Bellman-Ford αλγόριθμος δοσμένου ενός γράφου με link attributes όπως στην παραπάνω εικόνα, υπολογίζει ένα δέντρο συντομότερων μονοπατιών από κάθε κόμβο. Έστω ότι A(i,j) := attribute of link (i,j) = [l(j) , c(i,j)] και ότι το $p^k(i)$ είναι το attribute του καλύτερου μονοπατιού από το i προς οποιοδήποτε άλλο κόμβο που απέχει το πολύ k hops. Ο Bellman-Ford υπολογίζει το καλύτερο μονοπάτι προς τη ρίζα με τον παρακάτω τρόπο [24]:

$p^0(i) = [l(i) , 0]$ ∀i

```
for k = 1, 2, … do
      for all i
            p^k(i) = min{[l(i) , 0] , min_{j≠i}[A(i,j) ⊕ p^{k-1}(j)]}
until p^k = p^{k-1}
```

Παρακάτω έχουμε ένα παράδειγμα εκτέλεσης του αλγορίθμου. Ως pred(i) ορίζεται ο επόμενος κόμβος από τον i πάνω στο συντομότερο μονοπάτι προς τη ρίζα.



format: (label , cost)

k\i	10	20	30	40	50
0	10,0	20,0	30,0	40,0	50,0
1	10,0	10,1	20,6	10,2	20,3
2	10,0	10,1	10,7	10,2	10,3
3	10,0	10,1	10,4	10,2	10,3

i	10	20	30	40	50
pred(i)	10	10	50	10	40

Figure 20-Centralized Bellman-Ford for Bridges example [24].

Ο centralized Bellman-Ford αλγόριθμος για bridges είναι ευαίσθητος σε αρχικές συνθήκες. Χωρίς να μπούμε σε λεπτομέρειες, αν στο Figure 20 ο κόμβος 30 ήταν αρχικά συνδεδεμένος με έναν άλλο κόμβο με label 9 (μικρότερο από όλους τους άλλους κόμβους του γράφου) και στη συνέχεια το link μεταξύ του 30 και του 9 χαλούσε, ο αλγόριθμος δε θα συγκλίνει. Πιο γενικά αν οι αρχικές συνθήκες του Bellman-Ford ικανοποιούν την παρακάτω σχέση [24]:

$\forall i: \quad p^0(i) = (m_i, c_i) \ \mu\varepsilon \ m_i \geq \min_j l(j)$

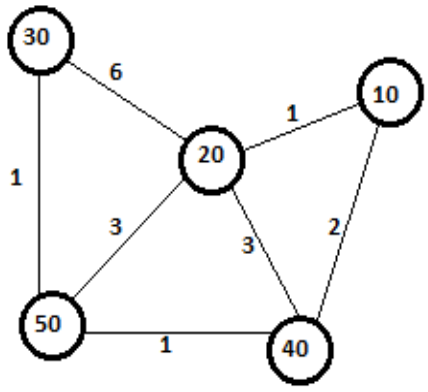
τότε ο αλγόριθμος θα συγκλίνει στη σωστή τιμή. Διαφορετικά αποκλίνει.

Distributed Bellman-Ford for Bridges: Πρόκειται για τον αλγόριθμο που χρησιμοποιεί το spanning tree protocol. Κάθε κόμβος i διατηρεί μια εκτίμηση q(i) του p(i) (του attribute του καλύτερου μονοπατιού από το i) και τον επόμενο κόμβο σε αυτό το μονοπάτι (pred(i)).

Αρχικά q(i) = [l(i),0] και pred(i) = i. Ανά τακτά διαστήματα ο κόμβος i στέλνει το q(i) στους γείτονές του. Όταν ο i λάβει ένα q(j) από ένα γειτονικό κόμβο j, θέτει το q(j) στην τιμή που έλαβε και υπολογίζει το q(i) ως εξής [24]:

```
if j == pred(i)
      then q(i):=min{A(i,j) ⊕ q(j),[l(i),0]}
      else q(i):=min{A(i,j) ⊕ q(j),q(i)}
```

Αν η παραπάνω σχέση προκαλέσει τη μεταβολή του q(i), τότε pred(i) = j. Επίσης αν υπάρξει αλλαγή στο A(i,pred(i)), έχουμε: q(i) = [l(i),0] και pred(i) = i.



i	10	20	30	40	50
	10,0	20,0	30,0	40,0	50,0
10 -> 20		10,1			
50 -> 20		10,1			
20 -> 50					10,4
20 -> 30			10,7		
50 -> 40				10,5	
20 -> 40				10,4	
10 -> 40				10,2	
40 -> 50					10,3
50 -> 20					
50 -> 30			10,4		

Figure 21-Distributed Bellman-Ford for Bridges example [24].

Εδώ όπως και στην περίπτωση του centralized Bellman-Ford για bridges, αλλαγές στην τοπολογία δεν επηρεάζουν την ομαλή εκτέλεσή του αρκεί ο κόμβος με το μικρότερο label (root) να βρίσκεται ακόμα στο δίκτυο και να είναι προσβάσιμος από όλους τους άλλους κόμβους. Αν δεν ισχύει αυτή η συνθήκη, ο αλγόριθμος δε συγκλίνει σε σωστή τιμή.

Για την αντιμετώπιση του προβλήματος το STP χρησιμοποιεί root monitoring. Ανά περίπου δύο δευτερόλεπτα το root στέλνει ένα refresh message το οποίο διαδίδεται σε ολόκληρο το spanning tree. Αν κάποιο bridge δε λάβει το refresh message για MaxAge χρονικό διάστημα, ξεκινάει από την αρχή τη διαδικασία του STP.

Η επικοινωνία μεταξύ των bridges γίνεται μέσω της αποστολής και λήψης BPDUs (Bridge Protocol Data Units). Το spanning tree μπορεί να μας εξασφαλίζει την απουσία loops στο δίκτυο, αλλά δε μας εγγυάται ότι τα τελικά μονοπάτια είναι τα βέλτιστα από άποψη κόστους.

Configuring STP: Παρακάτω παραθέτουμε μερικές εντολές (από το bridge-utils tool) για την παραμετροποίηση του STP.

`brctl stp <bridge> <state>`
Καθορίζει τη συμμετοχή του bridge στο STP. Αν το <state> είναι “on” ή “yes” το STP ενεργοποιείται.

`brctl setbridgeprio <bridge> <priority>`
Οι μικρές τιμές για το <priority> είναι καλύτερες. Το bridge με το μικρότερο <priority> επιλέγεται ως root.

`brctl setmaxage <bridge> <time>`
Καθορίζει την τιμή του MaxAge (χρόνος για τον οποίο το bridge αποθηκεύει τις πληροφορίες από τα BPDUs που έλαβε).

`brctl setpathcost <bridge> <port> <cost>`
Θέτει το κόστος του <port> στο <cost>. Αυτή η εντολή είναι ιδιαίτερα χρήσιμη όταν θέλουμε να χρησιμοποιήσουμε διαφορετικά μονοπάτια από αυτά που έχει επιλέξει το spanning tree.

```
brctl setportprio <bridge> <port> <priority>
```
Το priority ενός port χρησιμοποιείται για τον καθορισμό του ως root ή designated.

4.2.3 MadWifi and STP

Κατά την αρχική εφαρμογή του spanning tree protocol, αν και τα bridges λάμβαναν τα BPDUs, οι καταστάσεις των ports δεν άλλαζαν. Όλα τα ports βρίσκονταν πάντοτε σε forwarding state με αποτέλεσμα τη δημιουργία βρόγχων στο δίκτυο.

Αυτό είναι ένα πρόβλημα που έχουν οι MadWifi drivers με Kernels μεταγενέστερους του 2.6.16. Έπρεπε λοιπόν να απεγκαταστήσουμε το MadWifi από όλους τους κόμβους του δικτύου και να εισάγουμε ένα patch στους drivers προτού τους εγκαταστήσουμε ξανά [25].

Στο αρχείο madwifi/net80211/ieee80211_input.c έπρεπε να διαγράψουμε την εντολή:

```
return eth->h_proto;
```

και να την αντικαταστήσουμε με τις:

```
if (ntohs(eth->h_proto) >= 1536)
      return eth->h_proto;
return htons(ETH_P_802_2);
```

Κεφάλαιο 5

Υλοποίηση Συστήματος Αναμετάδοσης και Επίδειξη

Η εργασία αυτή πρόκειται ουσιαστικά για ένα relay system το οποίο θα προσφέρει ασύρματη πρόσβαση στο internet ή σε ένα τοπικό δίκτυο, σε συσκευές που βρίσκονται σε απομακρυσμένες περιοχές. Το δίκτυο που δημιουργήσαμε έχει την παρακάτω μορφή:

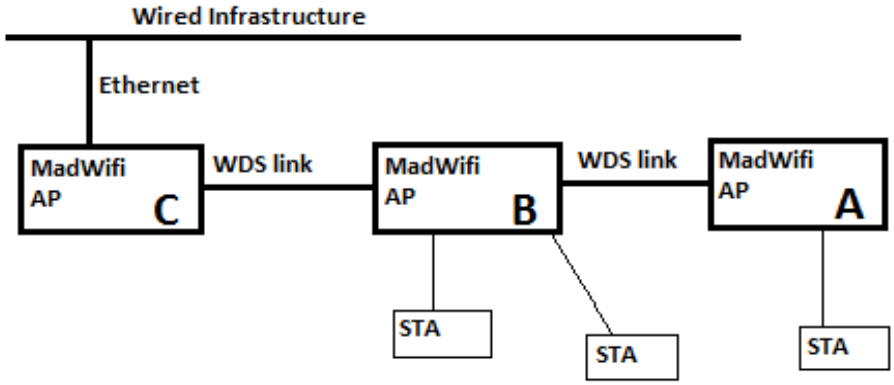


Figure 22-Ένα MadWifi Relay System, όπως υλοποιήθηκε στην παρούσα εργασία.

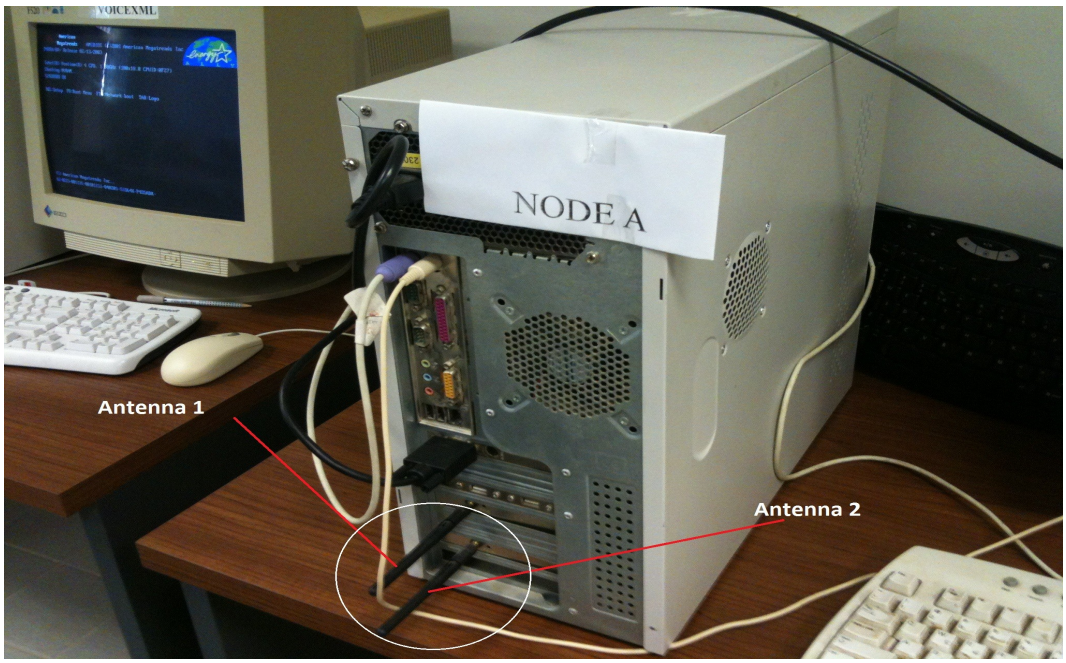


Figure 23-Node A

Επεκτείναμε δηλαδή την περιοχή που μπορεί να εξυπηρετήσει ένα access point, προσθέτοντας και άλλους υπολογιστές που διαθέτουν μία κάρτα δικτύου που λειτουργεί ως ap και άλλη μία που είναι υπεύθυνη για τη δημιουργία των ενδιάμεσων συνδετικών ζεύξεων (wds links) έτσι ώστε τα διάφορα access points να μπορούν να επικοινωνούν μεταξύ τους.

Ολόκληρη η υλοποίηση έγινε με ήδη έτοιμα εργαλεία, όπως τα bridge utilities και το spanning tree protocol, η σύνθεση των οποίων οδήγησε στη δημιουργία του relay συστήματος. Όλα τα εργαλεία που χρησιμοποιήσαμε επενεργούν στο επίπεδο 2 του Open System Interconnect (OSI) model.

5.1 Relay System

Σε αυτό το εδάφιο περιγράφουμε όλα τα βήματα που ακολουθήσαμε για τη δημιουργία του relay συστήματος.

Στην παρακάτω εικόνα παρουσιάζουμε σχηματικά τη λειτουργία του κώδικα που αναπτύξαμε και που τρέχει σε κάθε έναν από τους backbone κόμβους μας.

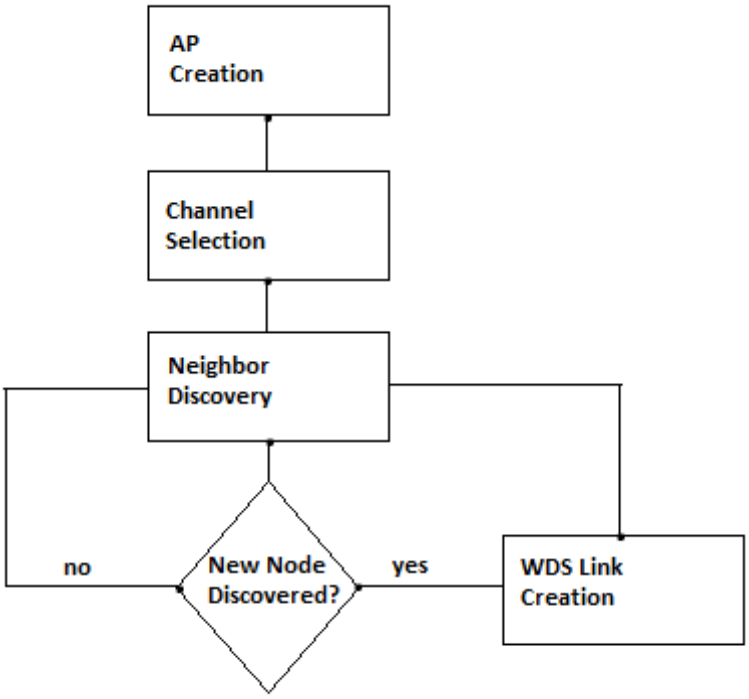


Figure 24-Relay System Finite State Machine

Αρχικά σε κάθε έναν από τους κόμβους του δικτύου μας δημιουργούμε ένα access point και ορίζουμε κάποιες από τις παραμέτρους του, όπως το rate, το beacon interval, το diversity και το κανάλι λειτουργίας. Η επιλογή του τελευταίου δεν είναι οριστική αλλά δίνουμε όλες τις παραμέτρους που είναι απαραίτητες για τη λειτουργία του, λόγω του ότι μέσω του ap κάνουμε scanning για να βρούμε ποια κανάλια συχνοτήτων είναι ελεύθερα.

Το επόμενο βήμα μετά τη δημιουργία του access point, είναι η επιλογή του καναλιού στο οποίο θα λειτουργεί. Έχουμε επιλέξει το κανάλι 11 να χρησιμοποιείται για τα wds links, οπότε για τα access points τα πιθανά κανάλια είναι το 1 και το 6 (στο 802.11b/g τα κανάλια 1, 6, 11 δεν επικαλύπτονται). Κάθε ap λοιπόν θα σαρώσει την περιοχή και θα μάθει αν και ποιες συχνότητες

χρησιμοποιούν γειτονικές 802.11 συσκευές καθώς και την ποιότητα του σήματός τους. Αν βρεθεί ότι κάποιο από τα κανάλια 1 και 6 είναι ελεύθερο ή ότι χρησιμοποιείται από συσκευές που βρίσκονται μακριά από το ap (άρα μικρό SNR), θα επιλεχθεί ως το κανάλι λειτουργίας.

Το ap είναι επίσης υπεύθυνο για την εύρεση γειτονικών κόμβων που θα εισαχθούν στο δίκτυό μας. Στην πραγματικότητα ανακαλύπτει άλλα access points που εκπέμπουν το ίδιο ssid με αυτό του δικτύου μας. Σε κάθε κόμβο όμως διατηρούμε μια λίστα με τα ζεύγη των mac διευθύνσεων των καρτών δικτύου (mac της κάρτας που λειτουργεί ως access point και mac αυτής που είναι υπεύθυνη για τα wds links μεταξύ των ap) που υπάρχουν σε όλους τους υπολογιστές μας. Αν δηλαδή ένας κόμβος ανακαλύψει έναν άλλο με διεύθυνση mac για το ap x, γνωρίζει ότι η mac της δεύτερης κάρτας που διαθέτει είναι y. Έτσι είναι δυνατή η δημιουργία των wds συνδέσεων όπου πρέπει να ορίσουμε τη mac της κάρτας του απομακρυσμένου κόμβου.

Προτού βέβαια επιχειρήσουμε να συνδεθούμε με κάποιο απομακρυσμένο κόμβο, πρέπει να προσέξουμε κάποιες λεπτομέρειες. Για παράδειγμα δε θέλουμε να συνδεθούμε με κόμβο που έχουμε ήδη συνδεθεί στο παρελθόν. Για το λόγο αυτό σε κάθε υπολογιστή διατηρούμε μια λίστα με τους κόμβους με τους οποίους έχουμε δημιουργήσει wds links. Έτσι όποτε ανακαλύπτουμε έναν νέο πιθανό γείτονα συμβουλευόμαστε αυτή τη λίστα. Πέρα από αυτόν τον έλεγχο εξετάζουμε μόνο κόμβους που εκπέμπουν ένα συγκεκριμένο service set identifier (πχ myap) που έχουμε ορίσει οι ίδιοι. Τέλος αν το θελήσουμε μπορούμε να επιλέξουμε να συνδεόμαστε μόνο με κόμβους που έχουν υψηλό SNR, που σημαίνει ότι βρίσκονται κοντά και άρα θα έχουμε μεγαλύτερο throughput κατά τη μεταφορά δεδομένων.

Από τη στιγμή που θα ανακαλύψουμε και έναν άλλο κόμβο για το δίκτυο, πέρα από τη ρύθμιση του link, δημιουργούμε και ένα bridge interface στο οποίο συνδέουμε το access point, το repeater αλλά και το ethernet interface αν αυτό υφίσταται. Με αυτόν τον τρόπο είναι δυνατή η επικοινωνία μεταξύ των διάφορων interfaces ενός κόμβου. Στη συνέχεια δίνουμε μια ip στο bridge, είτε στατική είτε μέσω ενός dhcp client (για πρόσβαση στο internet) και πλέον ο κόμβος μας είναι έτοιμος να εξυπηρετήσει άλλους υπολογιστές.

Για κάθε νέο κόμβο που ανακαλύπτουμε, θα δημιουργείται και ένα καινούριο wds interface που θα συνδέεται κι αυτό στο bridge. Μετά ξεκινάει πάλι από την αρχή η διαδικασία του neighbor discovery.

Η υλοποίηση όλων των παραπάνω έγινε στη γλώσσα C.

5.2 Demo

Στα πλαίσια του demo θέλαμε να παρουσιάσουμε μετάδοση ήχου ή και video μέσω του relay συστήματος που δημιουργήσαμε.

Στις δοκιμές μας διαθέταμε 2 ή 3 backbone κόμβους και 1 ή 2 άλλους υπολογιστές που συνδέονταν ως clients στα access points του δικτύου μας. Σε περίπτωση που κάποιος από τους backbone διέθετε πρόσβαση στο internet, οι clients συνδέονταν στους άλλους κόμβους ώστε να τεθεί σε εφαρμογή και το relay.

Στην περίπτωση λοιπόν που διαθέταμε internet χρησιμοποιήσαμε το skype για τη μετάδοση ήχου και εικόνας. Η ποιότητα της εικόνας ήταν αρκετά ικανοποιητική με εξαίρεση την περίπτωση που οι κεραίες των repeaters δεν ήταν κατάλληλα aligned. Αυτό όμως είναι κάτι που εύκολα διορθώσαμε.

Επειδή το skype απαιτεί authentication του χρήστη από κάποιο απομακρυσμένο κόμβο, δεν είναι δυνατή η χρήση του σε τοπικά δίκτυα χωρίς πρόσβαση στο internet. Για αυτό το λόγο χρησιμοποιήσαμε και δύο άλλα προγράμματα.

Το πρώτο ήταν το P2P VoIp Beta 1.1. Το πρόγραμμα αυτό είναι συμβατό μόνο με windows και χρησιμοποιείται για επικοινωνία (μόνο ήχος) σε τοπικά δίκτυα και δεν απαιτεί server authentication. Η σύνδεση με άλλους υπολογιστές επιτυγχάνεται με την παροχή της IP διεύθυνσής τους.

Για να έχουμε μετάδοση και video στο τοπικό μας δίκτυο, χρησιμοποιήσαμε το ekiga. Αυτό το πρόγραμμα είναι συμβατό και με windows και με linux και αν το θέλουμε για χρήση μόνο σε δίκτυα χωρίς internet, δεν απαιτεί κάποιο authentication. Και εδώ η σύνδεση μεταξύ υπολογιστών γίνεται παρέχοντας την IP.

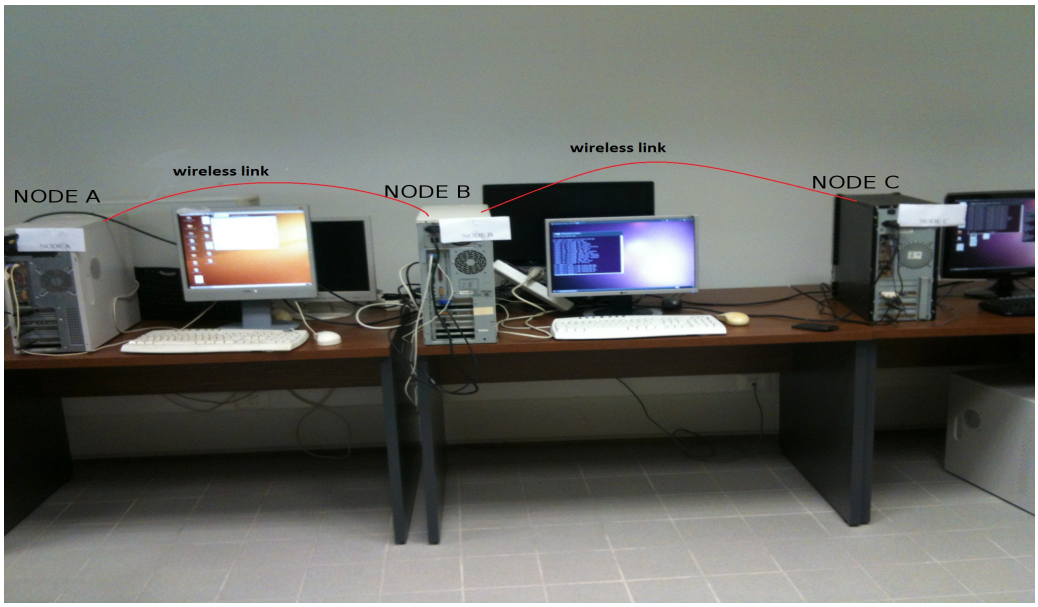


Figure 25-Nodes of our Relay System

Στην παραπάνω εικόνα ένα laptop (client) είναι συνδεδεμένο στο access point που βρίσκεται στον κόμβο A και χρησιμοποιούμε το ekiga για να έχουμε μετάδοση video από το laptop στον κόμβο C. Τα ενεργά links μεταξύ των backbone κόμβων είναι το A-B και B-C. Το link A-C έχει απενεργοποιηθεί από το spanning tree protocol για να μη δημιουργηθεί loop. Αν στον κόμβο B απενεργοποιήσουμε κάποιο από τα wds interfaces που διαθέτει και τα οποία είναι απαραίτητα για να επικοινωνεί ο B με τον A και τον C, τότε θα σταματήσει η αποστολή εικόνας μεταξύ laptop και C (μέχρι το STP να ενεργοποιήσει το A-C link). Αυτό αποτελεί απόδειξη πως η μεταφορά δεδομένων γίνεται μέσω του relay και όχι απευθείας από το laptop στον C.

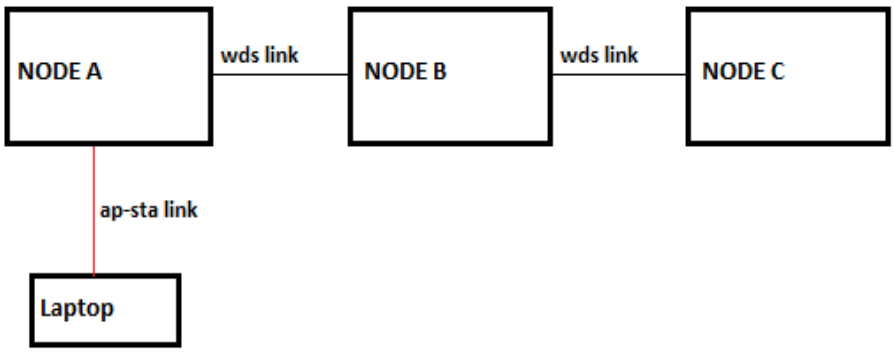


Figure 26-Demo Network Topology

Η λειτουργικότητα του relay συστήματος με πολλούς κόμβους εκτός εργαστηρίου (scaling) θα είναι δυνατή αν οι κόμβοι τοποθετηθούν σε αραιές αποστάσεις μεταξύ τους. Για το demo χρησιμοποιήσαμε μέχρι τρεις κόμβους και δεν είχαμε μεγάλα προβλήματα με παρεμβολές. Αν όμως είχαμε 20-30 κόμβους κοντά ο ένας με τον άλλο, λόγω του ότι χρησιμοποιείται το ίδιο κανάλι συχνοτήτων για το relay μεταξύ των backbone κόμβων, θα είχαμε μεγάλο interference και προβλήματα στην ομαλή λειτουργία του δικτύου. Εναλλακτικά θα μπορούσαμε να χρησιμοποιήσουμε 802.11a όπου υπάρχουν περισσότερα μη επικαλύπτόμενα κανάλια συχνοτήτων.

Στο Figure 27 βλέπουμε μια πιο κοντινή εικόνα του κόμβου C. Μπορούμε να διακρίνουμε τις κεραίες των δύο ασύρματων καρτών δικτύου που διαθέτει. Επίσης είναι προφανές πως δεν είναι ενσύρματα συνδεδεμένος με κάποιον από τους άλλους κόμβους.

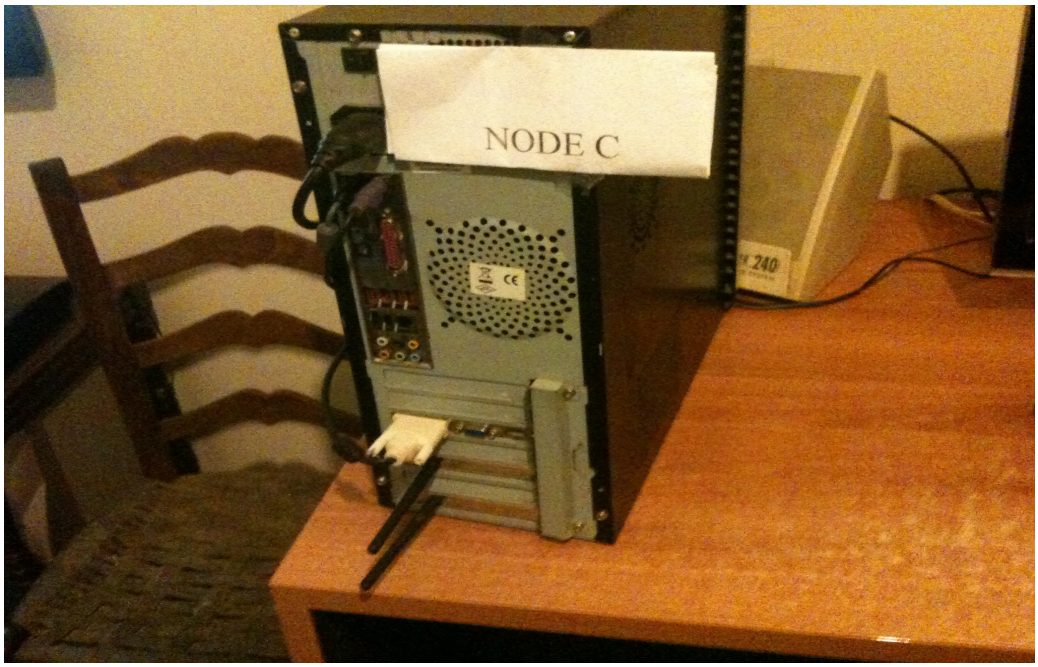


Figure 27-Node C

Figure 28-Nodes A, B and C

Στην παρακάτω εικόνα βλέπουμε στιγμιότυπο του video που στέλνεται από το laptop που είναι συνδεδεμένο στον κόμβο A (κάτω μισό της εικόνας), στον κόμβο C (πάνω μισό της εικόνας).

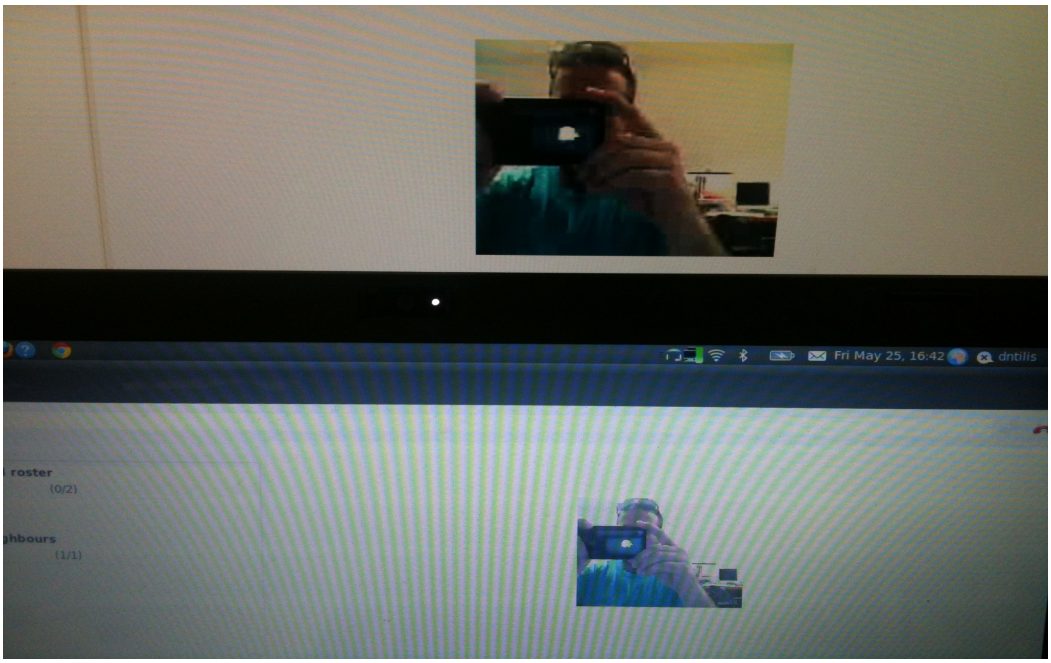


Figure 29-Transferring Video

Τέλος δοκιμάσαμε και remote control στον κόμβο C, μέσω remote control στον κόμβο A. Χρησιμοποιώντας το Vinagre αποκτήσαμε πρόσβαση από το laptop (που είναι συνδεδεμένο στο access point του A) στον κόμβο A. Στη συνέχεια από το laptop και πάλι δώσαμε εντολή στον A να αποκτήσει πρόσβαση στον C. Τελικά με μία ανοιχτή σύνδεση από το laptop, έχουμε πρόσβαση και στον A και στον C. Αυτό φαίνεται στο Figure 31. Σε αυτή την εικόνα φαίνεται ένα στιγμιότυπο που πήραμε από το desktop του laptop. Σε αυτό φαίνεται ολόκληρο το desktop του κόμβου A και ένα τμήμα του desktop του C. Το εικονίδιο στο 1 (πάνω δεξιά) μας πληροφορεί πως ο A ελέγχεται απομακρυσμένα από κάποιον άλλο υπολογιστή. Στους A και C χρησιμοποιήσαμε την iwconfig εντολή για να φανεί το ssid που κάνουν broadcast τα access points τους και να δείξουμε ότι πρόκειται για τα desktop διαφορετικών κόμβων. Έτσι βλέπουμε ότι στο 2 έχουμε “myap1” που είναι το ssid του κόμβου A και στο 3 έχουμε “myap” που είναι το ssid του C.

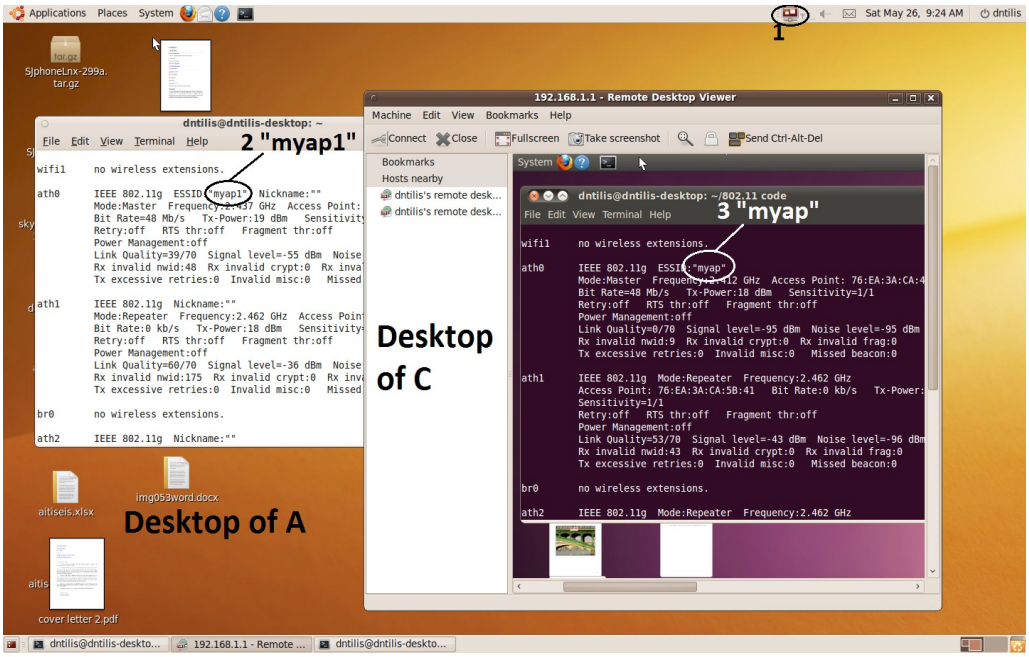


Figure 30-Laptop controls C through A

Κεφάλαιο 6

Συμπεράσματα

Συνοψίζοντας μπορούμε να πούμε ότι η λειτουργία του WiFI relay συστήματος που δημιουργήσαμε, είναι αρκετά ικανοποιητική. Εξαιτίας της φύσης των ασύρματων συνδέσεων δε μπορούμε να εγγυόμαστε πάντοτε υψηλό throughput και για αυτό το λόγο θα θέλαμε στο μέλλον να αλλάξουμε τις κεραίες που διαθέτουν οι κάρτες που χρησιμοποιήσαμε, με άλλες που θα έχουν μεγαλύτερο gain.

Σε μια μεταγενέστερη εκδοχή αυτής της εργασίας θα ήταν ίσως προτιμότερο να εργαστούμε με τους νεότερους ath5k drivers και να δημιουργήσουμε ένα mesh δίκτυο αντί για τα wds links που έχουμε τώρα μεταξύ των κόμβων. Αυτό θα συντελούσε στην καλύτερη απόδοση του συστήματος.

Επίσης ιδιαίτερα σημαντική θα είναι η χρήση laptop και γενικά μικρότερων υπολογιστών ως κόμβοι του δικτύου. Οι επιτραπέζιοι υπολογιστές που διαθέταμε για την εργασία δεν είναι κατάλληλοι λόγω μεγέθους για την εφαρμογή τους σε rooftop δίκτυα όπου οι διάφοροι κόμβοι θα εγκαθίστανται σε οροφές κτιρίων. Επιπλέον για τη συγκεκριμένη χρήση οι κόμβοι μας πέρα από το μέγεθος, πρέπει να είναι ανθεκτικοί στις αντίξοες καιρικές συνθήκες (βροχή).

Ένα άλλο σημαντικό θέμα είναι η επεξεργασία του source code του spanning tree. Θα θέλαμε αν είναι δυνατόν με κατάλληλο προγραμματισμό, να αλλάξουμε τον τρόπο λειτουργίας του STP και εκτός από την αποφυγή των βρόγχων, να μας εξασφάλιζε και βέλτιστα μονοπάτια. Αυτό ίσως μπορεί να επιτευχθεί θέτοντας τους ρυθμούς μετάδοσης των διάφορων links, να έχουν πιο σημαντικό ρόλο κατά την εκτέλεση του αλγορίθμου.

Τέλος θα μπορούσαμε να εξετάσουμε τη λειτουργικότητα του δικτύου όταν οι clients κινούνται με σχετικά υψηλές ταχύτητες και να δούμε αν οι MadWIFI (ή οι ath5k) drivers μπορούν να μας εξασφαλίσουν καλή απόδοση και σε αυτές τις περιπτώσεις.

Όσον αφορά τα προβλήματα που προέκυψαν κατά την υλοποίηση της εργασίας, μπορούμε να πούμε πως ένα από τα μεγαλύτερα ήταν η μη σωστή λειτουργία των MadWifi drivers όταν εφαρμόζαμε το STP. Αυτό λύθηκε με το patch που αναφέραμε στο εδάφιο 4.2.3 και την επανεγκατάσταση των οδηγών. Άλλο πρόβλημα που συναντήσαμε ήταν συχνή αδυναμία σύνδεσης του laptop στα access points καθώς και η συχνή αποσύνδεσή του από αυτά που οφειλόταν στη υψηλή αρχική τιμή που είχε το beacon interval. Όταν μειώσαμε την τιμή του στα 400ms, οι παραπάνω δυσκολίες έπαψαν να υφίστανται.

Α. Παράρτημα: Κώδικας και Επεξήγηση

Παρακάτω παραθέτουμε τον κώδικα που αναπτύξαμε για την υλοποίηση αυτής της εργασίας. Στο τέλος υπάρχουν και τα σχόλια που επεξηγούν τη λειτουργία του.

Κώδικας

```
#include <sys/socket.h>
#include <sys/time.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <netinet/tcp.h>
#include <netdb.h>
#include <unistd.h>
#include <time.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <string.h>
#include <errno.h>
#include <assert.h>
#include <dirent.h>
#include <sys/wait.h>
#include <sys/stat.h>
#include <ctype.h>
#include <stdarg.h>

1: #define MAXLINE         512

2: struct mylist{
3:    char nodes[100];
4:    struct mylist *next;
5: };

6: struct used_channels
7: {
8:    int channel_used;
9:    int channel_quality;
10: };

11: int channel_select()
12: {
13:   int selected_channel;
14:   int temp=0;
15:   int status1;
16:   struct used_channels ch[10];
17:   char *tr1=" ";
18:   char *buf21;
19:   char lines1[MAXLINE];
20:   FILE *fpin1  = {0};
21:   int fd1[2] = {0};
22:   pid_t child1 = -1;
23:   int i=0;
24:   for(i=0;i<11;i++)
25:   {
26:         ch[i].channel_used=0;
27:         ch[i].channel_quality=0;
```

```
28:   }
29:   srand(time(NULL));
30:   pipe(fd1);
31:   child1=fork();
32:   if(child1==-1)
33:         {
34:               exit(0);
35:         }
36:   if(child1==0)
37:         {
38:               close(fd1[0]);
39:               dup2(fd1[1], STDOUT_FILENO);
40:               system("iwlist ath0 scanning");
41:               kill (getpid(), SIGTERM);
42:         }
43:   else
44:         {
45:               close(fd1[1]);
46:               fpin1 = fdopen(fd1[0], "r");
47:               fgets(lines1, MAXLINE, fpin1);
48:               while(fgets(lines1, MAXLINE, fpin1) != NULL)
49:               {
50:                     buf21=strtok(lines1,tr1);
51:                     if(buf21[0]=='F' && buf21[1]=='r' && buf21[2]=='e'
                       && buf21[3]=='q')
52:                     {
53:                           while(buf21[0]!='(')
54:                           {
55:                                 buf21=strtok(NULL,tr1);
56:                           }
57:                           buf21=strtok(NULL,tr1);
58:                           printf("\nchannel number=%d\n",atoi(buf21));
59:                           temp=atoi(buf21);
60:                           ch[temp-1].channel_used++;
61:                           continue;
62:                     }
63:                     else if(buf21[0]=='Q' && buf21[1]=='u' &&
                                 buf21[2]=='a')
64:                           {
65:                                 if(buf21[9]!='/' && (buf21[8]=='3' ||
                                    buf21[8]=='4'))
66:                                       {
67:                                       if(ch[temp-1].channel_quality==0)
68:                                             ch[temp-1].channel_quality=1;
69:                                       }
70:                                  else if(buf21[9]=='/' || buf21[8]=='1' ||
                                          buf21[8]=='2');
71:                                 else
72:                                       ch[temp-1].channel_quality=2;
73:                           }
74:               }

75:               waitpid(-1,&status1,0);
76:               fclose(fpin1);
77:         }

78:   if(ch[0].channel_quality!=2 && ch[5].channel_quality==2)
79:         {
80:               selected_channel=1;
81:         }
```

```
82:   else if(ch[5].channel_quality!=2 && ch[0].channel_quality==2)
83:         {
84:               selected_channel=6;
85:         }
86:   else if(ch[5].channel_quality==1 && ch[0].channel_quality==1)
87:         {
88:               if(rand()%2==0)
89:               selected_channel=6;
90:               else
91:               selected_channel=1;
92:         }
93:   else if(ch[0].channel_quality==0)
94:         {
95:               selected_channel=1;
96:         }
97:   else if(ch[5].channel_quality==0)
98:         {
99:               selected_channel=6;
100:        }
101:  else
102:        selected_channel=3;

103:  return(selected_channel);


104: }



105: void find_rep_addr(char *ap)
106: {
107:  char *ap1="76:EA:3A:CA:4E:0A";
108:  char *rep1="76:ea:3a:ca:62:87";
109:  char *ap2="76:EA:3A:CA:59:9E";
110:  char *rep2="76:ea:3a:ca:59:ec";
111:  char *ap3="76:EA:3A:CA:5B:41";
112:  char *rep3="76:EA:3A:CA:55:0E";
113:  if(strncmp(ap,ap1,17)==0)
114:        strcpy(ap,rep1);
115:  else if(strncmp(ap,ap2,17)==0)
116:        strcpy(ap,rep2);
117:  else if(strncmp(ap,ap3,17)==0)
118:        strcpy(ap,rep3);
119   return;
120: }

121: int check_list(struct mylist *h,char *str)
122: {
123:  if(h==NULL)
124:        return 1;

125:  struct mylist *temp;
126:  temp=h;
127:  do
128:  {
129:        if(strncmp(temp->nodes,str,17)==0)
130:        {
131:              puts("\nnode already in list\n");
132:              return 0;
133:        }
```

```
134:  temp=temp->next;
135:  }while(temp!=NULL);

136:  return 1;
137: }

138: void add_list(struct mylist **h,char *str)
139: {
140:  struct mylist *temp,*r;
141:  temp=*h;
142:  if(*h==NULL)
143:  {
144:        temp = (struct mylist *)malloc(sizeof(struct mylist));
145:        strcpy(temp->nodes,str);
146:        temp->next=NULL;
147:        *h=temp;
148:        return;
149:  }
150:  while(temp->next!=NULL)
151:  {
152:        temp=temp->next;
153:  }

154:  r = (struct mylist *)malloc(sizeof(struct mylist));
155:  strcpy(r->nodes,str);
156:  r->next=NULL;
157:  temp->next=r;

158: }

159: void display(struct mylist *q)
160: {
161:  if(q==NULL)
162:  {
163:    printf("\n\nEmpty Link List.Can't Display The Data");

164:    goto last;
165:  }
166: while(q!=NULL)
167:  {
168:  printf("\nprinting nodes%s",q->nodes);

169:  q=q->next;
170:  }
171:  last:;
172: }


173: void create_interface(char *com, int *ifs,int *num)
174: {
175:  puts("\ncreate interface 1\n");
176:  system("ifconfig ath0 down");
177:  char ar1[50]="iwconfig ath0 channel ";;
178:  char str[10];
179:  sprintf(str,"%d",*num);
180:  strcat(ar1,str);
181:  printf("\n%s\n",ar1);
182:  system(ar1);
183:  system("ifconfig ath0 up");
184:  system("wlanconfig ath1 create wlandev wifi1 wlanmode wds");
```

```
185:  system(com);
186:  system("iwpriv ath1 wds 1");
187:  system("iwconfig ath1 channel 11");
188:  system("ifconfig ath1 up");
189:  system("brctl addbr br0");
190:  system("brctl stp br0 on");
191:  system("brctl addif br0 ath0");
192:  system("brctl addif br0 ath1");
193:  system("ifconfig eth0 up 0.0.0.0");
194:  system("brctl addif br0 eth0");
195:  system("brctl setfd br0 0");
196:  //system("ifconfig br0 192.168.0.1 up");

197:  system("dhclient br0");
198:  system("ifconfig br0 up");
199:  (*ifs)++;

200: }

201: void create_interface2(char *com, int *ifs)
202: {
203:  char ar1[50]="wlanconfig ath";
204:  char ar2[50]="iwpriv ath";
205:  char ar3[50]="iwpriv ath";
206:  char ar4[50]="ifconfig ath";
207:  char ar5[50]="brctl addif br0 ath";
208:  (*ifs)++;
209:  char str[30];
210:  sprintf(str,"%d",*ifs);
211:  strcat(ar1,str);
212:  strcat(ar1," create wlandev wifi1 wlanmode wds");
213:  strcat(ar2,str);
214:  strcat(ar2," wds_add ");
215:  strcat(ar2,com);
216:  strcat(ar3,str);
217:  strcat(ar3," wds 1");
218:  strcat(ar4,str);

219:  strcat(ar4," up");
220:  strcat(ar5,str);
221:  //strcat(ar5,"_rename");
222:  system(ar1);
223:  system(ar2);
224:  system(ar3);
225:  system(ar4);
226:  system(ar5);

227: }


228: int main()
229: {
230:  int nu_forks=0;
231:  struct mylist *lis;
232:  lis=NULL;
233:  int num_of_ifs=0;
234:  int var=0;
235:  int channel_no;
236:  system("service network-manager stop");
237:  system("wlanconfig ath0 destroy");
```

```
238:  system("echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward");
239:  system("wlanconfig ath0 create wlandev wifi0 wlanmode ap");
240:  system("iwconfig ath0 essid \"myap\" channel 8");
241:  system("iwconfig ath0 rate 48M");
242:  system("iwpriv ath0 wds 1");
243:  system("sysctl -w dev.wifi0.diversity=0");
244:  system("sysctl -w dev.wifi0.txantenna=1");
245:  system("sysctl -w dev.wifi0.rxantenna=1");
246:  system("sysctl -w dev.wifi1.diversity=0");
247:  system("sysctl -w dev.wifi1.txantenna=1");
248:  system("sysctl -w dev.wifi1.rxantenna=1");
249:  system("iwpriv ath0 bgscan 0");
250:  system("iwpriv ath0 bintval 400");
251:  system("iwpriv ath0 protmode 0");
252:  system("iwpriv ath0 rssi11a 11");
253:  system("iwpriv ath0 rssi11b 11");
254:  system("iwpriv ath0 rssi11g 11");
255:  system("iwpriv ath0 abolt 0");
256:  system("ifconfig ath0 up");

257:  system("iwlist ath0 scanning");
258:  system("sleep 3");
259:  system("iwlist ath0 scanning");
260:  system("sleep 3");
261:  system("iwlist ath0 scanning");
262:  system("sleep 3");
263:  system("iwlist ath0 scanning");
264:  system("sleep 3");


265:  char lines[MAXLINE];
266:  int i=0;
267:  char help[100];

268:  int help2=0;
269:  int help3=0;

270:  channel_no=channel_select();


271:  start:;
272:  char com[50]="sudo iwpriv ath1 wds_add ";
273:  pid_t child = -1;
274:  int status;
275:  FILE *fpin  = {0};
276:  int fd[2] = {0};
277:  char *tr=" ";
278:  char *buf2;
279:  pipe(fd);
280:  child=fork();
281:  if(child==-1)
282:        {
283:              puts("error in fork");
284:              exit(0);
285:        }
286:  if(child==0)
287:        {
288:              close(fd[0]);
289:              dup2(fd[1], STDOUT_FILENO);
290:              system("iwlist ath0 scanning");
```

```
291:              fflush(NULL);
292:              kill (getpid(), SIGTERM);
293:        }
294:  else
295:        {     waitpid(-1,&status,0);
296:               close(fd[1]);
297:              fpin = fdopen(fd[0], "r");
298:              fgets(lines, MAXLINE, fpin);
299:              while(fgets(lines, MAXLINE, fpin) != NULL)
300:              {
301:                    buf2=strtok(lines,tr);

302:                    if(help3==1)
303:                    {
304:                          if(buf2[0]=='Q' && buf2[1]=='u')
305:                          {

306:                                if((buf2[8]!='1' && buf2[8]!='2') &&
                                        buf2[9]!='/')
307:                                {
308:                                      break;
309:                                }
310:                                else
311:                                {
312:                                      break;
313:                                      //help3=0;
314:                                      //help2=0;
315:                                }
316:                          }
317:                          else
318:                          {
319:                                continue;
320:                          }
321:                    }

322:                    if(help2==1)
323:                    {
324:                          if(buf2[7]=='m' && buf2[8]=='y' &&
                              buf2[9]=='a'&& buf2[10]=='p')
325:                          {
326:                          help3=1;
327:                          continue;
328:                          }
329:                          help2=0;
330:                    }
331:                    while(buf2!=NULL)
332:                    {
333:                          if(buf2[0]=='A' && buf2[1]=='d' &&
                        buf2[2]=='d' && buf2[3]=='r' && buf2[4]=='e')
334:                          {
335:                                buf2=strtok(NULL,tr);
336:                                strcpy(help,buf2);
337:
338:                                find_rep_addr(help);

339:                                if(!check_list(lis,help))
340:                                {
341:                                      continue;
342:                                }
```

```
343:                                help2=1;
344:                                break;
345:                          }

346:                    buf2=strtok(NULL,tr);

347:                    }
348:                    i++;
349:              }
350:              fclose(fpin);
351:              display(lis);
352:              fflush(stdout);

353:              if(help3==1 && check_list(lis,help))
354:              {
355:              help3=0;
356:              help2=0;
357:              puts("\nla1\n");
358:              add_list(&lis,help);
359:              display(lis);
360:              fflush(stdout);
361:              strcat(com,help);
362:              var=1;
363:              if(num_of_ifs==0)
364:              {
365:                    create_interface(com, &num_of_ifs,&channel_no);
366:              }
367:              else
368:              {
369:                    create_interface2(help, &num_of_ifs);
370:              }

371:                    printf("\nhelp=%d\n",help2);
                  }
372:              system("sleep 10");
373:              goto start;
374:              }
375:              else
376:              {
377:                    if(num_of_ifs>=2)
378:                    system("sleep 100");
379:                    system("sleep 10");
380:                    goto start;
381:              }


382:        }

383:  return 0;
384: }
```

Επεξήγηση του κώδικα:

2: Πρόκειται για μια λίστα στην οποία κάθε κόμβος διατηρεί τις MAC διευθύνσεις όλων των γειτονικών κόμβων που έχει ανακαλύψει και συνδεθεί μαζί τους. Τη χρησιμοποιούμε όταν θέλουμε να μάθουμε αν έχουμε ήδη συνδεθεί με κάποιον υπολογιστή που βρήκαμε κατά το neighbor discovery.

6: Δημιουργούμε ένα array από αυτό το struct όπου κάθε θέση του αντιπροσωπεύει ένα από τα 11 κανάλια λειτουργίας του 802.11b/g. Η μεταβλητή channel_used δείχνει πόσα γειτονικά access points χρησιμοποιούν το συγκεκριμένο κανάλι ενώ η channel_quality πόσο μεγάλο interference θα προκαλέσουν αυτά τα ap. Το channel_quality παίρνει 3 δυνατές τιμές: 0 αν το κανάλι δεν είναι κατειλημμένο, 1 αν το access point που λειτουργεί σε αυτό προκαλεί μικρή παρεμβολή και 2 αν προκαλεί μεγάλη παρεμβολή. Αν κάποιο κανάλι χρησιμοποιείται από περισσότερα του ενός access points, το channel_quality θα πάρει την τιμή του από το ap που προκαλεί το μεγαλύτερο interference.

11: Η συνάρτηση που καλούμε για την επιλογή του καναλιού λειτουργίας του access point ενός κόμβου.

36: Στο child process εκτελείται η εντολή `iwlist ath0 scanning` η οποία επιστρέφει μια λίστα με τα γειτονικά access points και διάφορες πληροφορίες για αυτά όπως bssid, snr, MAC διεύθυνση και το κανάλι λειτουργίας τους.

43: Στο parent process επεξεργαζόμαστε τα δεδομένα που επιστρέφονται από το child ώστε τελικά να επιλεγεί το κατάλληλο κανάλι.

51: Σε αυτό το if statement προσπαθούμε να βρούμε το κομμάτι των δεδομένων που αναφέρεται σε ποια συχνότητα λειτουργεί το access point. Όταν βρούμε το κανάλι, αυξάνουμε κατά ένα το channel_used (γραμμή 60).

63: Εδώ ψάχνουμε το SNR του παραπάνω access point για να μάθουμε πόσο μεγάλη παρεμβολή μπορεί αυτό να προκαλέσει. Αν το SNR είναι μικρότερο από 30, η τιμή του channel_quality για το συγκεκριμένο κανάλι δε θα αλλάξει (έχει αρχική τιμή 0). Αν το SNR είναι μεταξύ 30 και 49 το channel_quality θα πάρει την τιμή 1 εκτός και αν η τρέχουσα τιμή του είναι 2. Σε αυτή την περίπτωση δε θα μεταβληθεί. Τέλος αν το SNR είναι μεγαλύτερο από 49, το channel_quality ισούται με 2.

78: Αφού ολοκληρωθεί η διαδικασία επεξεργασίας των δεδομένων που προέρχονται από το child process, διαθέτουμε αρκετές πληροφορίες για να μπορέσουμε τελικά να επιλέξουμε κάποιο κανάλι λειτουργίας. Όπως έχουμε πει η επιλογή είναι μεταξύ των καναλιών 1 και 6. Μεταξύ των δύο επιλέγουμε αυτό που έχει μικρότερη τιμή στο channel_quality.

105: Η συνάρτηση αυτή καλείται όταν θέλουμε να πάρουμε τη MAC της κάρτας που χρησιμοποιείται για τα wds interfaces ενώ γνωρίζουμε τη MAC της κάρτας που εκτελεί τη λειτουργία του access point.

121: Συνάρτηση που ελέγχει όλα τα πεδία της λίστας mylist για να δούμε αν όντως έχουμε ανακαλύψει κάποιον καινούριο κόμβο, ή αν έχουμε ήδη συνδεθεί μαζί του.

138: Η add_list() προσθέτει στη λίστα mylist τους κόμβους που ανακαλύπτουμε κατά το neighbor discovery και με τους οποίους θέλουμε να δημιουργηθούν wds links. Έτσι αν στο μέλλον βρούμε τον ίδιο κόμβο, η συνάρτηση check_list() θα μας ενημερώσει και δε θα προχωρήσουμε στη δημιουργία νέων interfaces.

159: Τυπώνει τα περιεχόμενα της mylist για να μπορέσουμε να ελέγξουμε αν γίνεται σωστά η όλη διαδικασία του neighbor discovery.

173: Αυτή η συνάρτηση θα δημιουργήσει το πρώτο wds interface του κόμβου (γραμμή 184), το bridge (γραμμή 189) και θα θέσει το κανάλι λειτουργίας του access point σε αυτό που επιλέχθηκε από την channel_select() (γραμμή 182).

183: Ενεργοποίηση του access point.

185: Εντολή για τη σύνδεση του κόμβου μας με το γείτονα που ανακαλύψαμε.

186: Frames που στέλνονται μεταξύ repeaters χρησιμοποιούν και τα τέσσερα πεδία διευθύνσεων που υπάρχουν στο MAC header. Με αυτή την εντολή το interface μπορεί και αναγνωρίζει τέτοια frames.

187: Το κανάλι 11 αποφασίσαμε να είναι αυτό που θα χρησιμοποιούν οι repeaters για επικοινωνία μεταξύ τους.

188: Ενεργοποίηση του wds interface (repeater).

190: Εντολή για ενεργοποίηση του spanning tree protocol στο bridge.

191, 192, 194: Δέσμευση του access point, του repeater και του ethernet interface στο bridge.

196: Ορισμός ip διεύθυνσης για το bridge και ενεργοποίησή του. Αυτή η εντολή χρησιμοποιείται μόνο όταν οι κόμβοι μας δε έχουν πρόσβαση στο internet. Σε αντίθετη περίπτωση με τις εντολές στις γρμμές 197 και 198 γίνεται η ανάθεση ip από κάποιον dhcp server και η ενεργοποίηση του bidge.

199: Η μεταβλητή ifs κρατά των αριθμό των repeaters που έχουν δημιουργηθεί σε ένα κόμβο.

201: Εκτελεί την ίδια εργασία με την create_interface() εκτός της δημιουργίας του bridge. Η create_interface2() καλείται για τη δημιουργία όλων των υπολοίπων (εκτός του πρώτου δηλαδή) repeaters. Οι εντολές εδώ είναι παραμετροποιημένες γιατί κάθε repeater πρέπει να έχει και διαφορετικό όνομα.

228: Έναρξη της συνάρτησης main(). Εδώ δημιουργούμε το access point (αργότερα αλλάζουμε το κανάλι λειτουργίας του) και εκτελείται το neighbor discovery και selection.

243-248: Απενεργοποιούμε το diversity και στις δύο κάρτες δικτύου και ορίζουμε να χρησιμοποιείται κεραία 1 (και στις δύο κάρτες) για αποστολή και λήψη.

250: Θέτουμε το beacon interval στα 400 ms.

257-264: Από τη στιγμή που θα εκτελέσουμε την εντολή για την ενεργοποίηση του access point μέχρι τη στιγμή που το interface πραγματικά θα λειτουργεί, πρέπει να περάσει ένα μικρό χρονικό διάστημα. Έτσι με την εντολή `sleep` εξασφαλίζουμε ότι αργότερα που θα χρειαζόμαστε το ap να ψάξει για νέους γείτονες, αυτό θα λειτουργεί.

286: Όπως και στη συνάρτηση channel_select(), το child process εκτελεί την `iwlist ath0 scanning`, τα αποτελέσματα της οποίας θα επεξεργαστούμε στο parent process.

294: Έναρξη του parent process.

299: Σε αυτό το while loop επεξεργαζόμαστε τα δεδομένα που παίρνουμε από το child και γίνεται το neighbor discovery.

302: Αν το help3 ισούται με 1, σημαίνει ότι ο κόμβος που βρήκαμε είναι καινούριος και επίσης ανήκει στο δίκτυό μας. Ελέγχουμε αν το RSSI από αυτό τον κόμβο είναι αρκετά υψηλό για να επιχειρήσουμε να συνδεθούμε μαζί του. Στις δοκιμές που κάναμε είχαμε απενεργοποιήσει αυτή τη λειτουργία συνδεόμασταν ανεξαρτήτως του RSSI. Για να την ενεργοποιήσουμε αρκεί να βάλουμε σε σχόλια τη γραμμή 312 και να αφαιρέσουμε τα σχόλια από τις γραμμές 313 και 314.

322: Αν το help2 ισούται με 1 τότε σημαίνει ότι ανακαλύψαμε ένα κόμβο με τον οποίο δεν έχουμε συνδεθεί. Εδώ ελέγχουμε αν ο κόμβος πράγματι ανήκει στο δίκτυό μας (δηλαδή αν το access point έχει ssid myap).

331: Σε αυτό το while loop προσπαθούμε να βρούμε τις MAC των access points που επέστρεψε η εντολή `iwlist ath0 scanning` στο child process. Αφού βρούμε μία, καλούμε τη find_rep_addr() για να μάθουμε τη MAC του αντίστοιχου repeater (γραμμή 338). Στη συνέχεια καλούμε την check_list() (γραμμή 339) και βλέπουμε αν ήδη υπάρχει wds link με αυτόν.

358: Αφού είμαστε βέβαιοι πως ο κόμβος που βρήκαμε είναι νέος και ανήκει στο δικό μας δίκτυο, καλούμε την add_list() για να τον προσθέσουμε στη mylist.

363: Αν δεν έχουν δημιουργηθεί ακόμα άλλα repeaters καλούμε την create_interface() (γραμμή 365) για να υλοποιηθεί και το bridge interface. Διαφορετικά καλούμε την create_interface2() (γραμμή 369).

372-373 και 377-380: Περιμένουμε για κάποιο χρονικό διάστημα και στη συνέχεια ξεκινάμε από την αρχή τη διαδικασία του neighbor discovery.

Βιβλιογραφία

[1] http://pdos.csail.mit.edu/roofnet/doku.php

[2] D. De Couto, D. Aguayo, J. Bicket, R. Morris, "A high-throughput path metric for multi-hop wireless routing". Wireless Networks – Special issue: Selected papers from ACM MobiCom 2003, Volume 11, Issue 4, July 2005.

[3] http://laptop.org/en/laptop/

[4] http://emergentbydesign.com/2011/02/11/16-projects-initiatives-building-ad-hoc-wireless-mesh-networks/

[5] Y. Amir, C. Danilov, R. Musaloiu-Elefteri, N. Rivera, "The SMesh Wireless Mesh Network". ACM Transactions on Computer Systems (TOCS), Volume 28, Issue 3, September 2010.

[6] Y. Yang, J. Wang, R. Kravets, "Designing Routing Metrics For Mesh Networks". In Proceedings of the IEEE Workshop on Wireless Mesh Networks (WiMesh), Santa Clara, CA, IEEE Press (2005).

[7] http://sar.informatik.hu-berlin.de/research/projects/2005-BerlinRoofNet/berlin_roof_net.htm

[8] http://www.ecsl.cs.sunysb.edu/multichannel/

[9] A. Raniwala, T. Chiueh, "Architecture and Algorithms for an IEEE 802.11-Based Multi-Channel Wireless Mesh Network". In INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communication Societies, Miami, USA (August 2005).

[10] ANSI/IEEE Std 802.11, 1999 Edition, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.

[11] G. Bianchi, "Performance Analysis of the IEEE 802.11 Distributed Coordination Function". IEEE Journal on Selected Areas in Communications, Volume 18, No 3 (March 2000).

[12] P. Brenner, "A Technical Tutorial on the IEEE 802.11 Protocol". BreezeCOM Wireless Comunication, 2007.
http://courses.csail.mit.edu/6.885/spring06/papers/Brenner.pdf

[13] http://www.moonblinkwifi.com/2point4freq.cfm

[14] http://www.vocal.com/redirect/802_11a.html

[15] https://www.okob.net/texts/mydocuments/80211physlayer/

[16] http://www.moonblinkwifi.com/80211_fallback_rates.cfm

[17] http://madwifi-project.org/

[18] http://ubuntuforums.org/showthread.php?t=1309072

[19] J. Bicket, “Bit-rate Selection in Wireless Networks”. Master's thesis, Massachussets Institute of Thechnology, February 2005

[20] Giyeong Son, “Experimental Performance Evaluation of Bit-Rate Selection Algorithms in Multi-Vehicular Networks”. Master's thesis, University of Waterloo, January 2011.

[21] https://help.ubuntu.com/community/SSH/OpenSSH/

[22] S. Conner, J. Kruys, K. Kim, J. C. Zuniga, “IEEE 802.11s Tutorial. Overview of the Amendment for Wireless Local Area Mesh Networking”. IEEE 802 Plenary, Dalas, November 2006.

[23] http://www.erg.abdn.ac.uk/~gorry/eg3561/inet-pages/arp.html

[24] Jean-Yves Le Boudec, “Bridging”. Fall 2009.
http://icawww1.epfl.ch/cn2/0910/slides/11.bridging.pdf

[25] http://madwifi-project.org/ticket/839.