Παρακολούθηση & Διαχείριση Κινούμενων Σταθμών Μέσω Ασυρμάτων Δικτύων & SNMP

SNMP & Wi-Fi Positioning

Βιολέττας Ε. Γεώργιος

Θεοδώρου Λ. Τρύφων

Πανεπιστήμιο Μακεδονίας

Μεταπτυχιακό Εφαρμοσμένης Πληροφορικής

Επιβλέπων: Γεωργιάδης Κ. Χρήστος , Ph.D

Θεσσαλονίκη 2008

SNMP & Wi-Fi Positioning

Έστιν ...οφθαλμός, ος τα πανθ' ορά

Γεώργιος Ε. Βιολέττας
Τρύφων Λ. Θεοδώρου

Ευχαριστίες:

Καταρχήν θα θέλαμε να ευχαριστήσουμε τους καθηγητές μας, στο Πα.Μακ., γιατί μας έδωσαν την ευκαιρία να είμαστε και πάλι μαζί, δίνοντας μας την δυνατότητα να συγγράψουμε αυτήν την εργασία «παρέα».

Ελπίζουμε ότι το αποτέλεσμα δεν είναι κατώτερο των προσδοκιών τους.

Θα θέλαμε επίσης να ευχαριστήσουμε ειδικά τον Επιβλέποντα καθηγητή μας Δρ. Χρήστο Γεωργιάδη, τόσο για τις πολύτιμες συμβουλές και την καθοδήγηση του, αλλά κυρίως γιατί στάθηκε δίπλα μας σαν φίλος και συνεργάτης

Γιώργος – Τρύφων

Η εργασία αυτή είναι αφιερωμένη στα παιδιά μου Βαγγέλη &Μαρία- Δανάη, γιατί τους «έκλεψα» αρκετές ώρες για να την ολοκληρώσω,

Είναι αφιερωμένη στην γυναίκα μου Σοφία, που δεν βαρέθηκε να ακούει «…έχω διάβασμα»,
και στους γονείς μου που ρωτούν συνέχεια «..ακόμα αυτό το πανεπιστήμιο;..»,

Είναι επίσης αφιερωμένη σε όλους αυτούς, φίλους και συναδέλφους, που έχουν μοιραστεί μαζί μου λίγη από την γνώση τους, της οποίας κάποια ψήγματα καταστάλαξα εδώ μέσα. Στην επιστήμη που διαλέξαμε, χωρίς φίλους που «να ξέρουν» είσαι χαμένος. Και εγώ έχω ευτυχήσει να ξέρω κάμποσους από δαύτους,

Επίσης η εργασία αυτή είναι αφιερωμένη στο Ε.Α.Π. το πρώτο Πανεπιστήμιο μου (μας, αφού με τον Τρύφωνα εκεί γνωριστήκαμε) και σε όλους τους συμφοιτητές μας εκεί, γιατί οι παλιές αγάπες δεν ξεχνιούνται,

Και στο τέλος (last but not least) θέλω να ευχαριστήσω τον Τρύφωνα, για το κυριότερο πράγμα που μου έχει χαρίσει:
Την φιλία του…..

Γιώργος Βιολέττας , Θεσσαλονίκη, Φθινόπωρο 2008

Η εργασία αυτή είναι αφιερωμένη στην γυναίκα μου Μαρία για την υπομονή και υποστήριξη της,

Είναι αφιερωμένη στους δύο γιούς μου, τον Λάζαρο και το (2 μηνών) νέο μέλος της οικογενείας για τις ώρες που έλειψα από κοντά τους,

Είναι αφιερωμένη στους γονείς μου που «είναι πάντα εκεί όταν τους χρειάζομαι..»,

Επίσης η εργασία αυτή είναι αφιερωμένη στον φίλο μου και συνεργάτη Γιώργο για την άψογη συνεργασία μας στην εκπόνηση αυτής της εργασίας,

Τρύφων Θεοδώρου, Θεσσαλονίκη, Φθινόπωρο 2008

***Άργος ο Πανόπτης**:*

Εικόνα 1: Ο Ερμής σκοτώνει τον Άργο

Στην ελληνική μυθολογία ο Άργος ο Πανόπτης ήταν τερατόμορφος γίγαντας, απόγονος (όχι όμως ο γιος) του ομώνυμου ιδρυτή του Άργους, γιος του Ινάχου ή του Αγήνορα ή του Αρέστορα και της κόρης του Ασωπού ή του Ινάχου Ισμήνης. Θρυλικός φρουρός της ερωμένης του Δία Ιούς, την οποία ο Δίας είχε μεταμορφώσει σε αγελάδα για να αποφύγει τις σκηνές ζηλοτυπίας της Ήρας (κατ' άλλους η μεταμόρφωση έγινε από τη ζηλιάρα Ήρα), ο Άργος αυτός ανήκει στα ισχυρότερα και φοβερότερα τέρατα της ελληνικής μυθολογίας: ήταν γνωστός ως πανόπτης ή μυριωπός γιατί είχε εκατό μάτια διάσπαρτα σε όλο του το σώμα (σύμφωνα με άλλη εκδοχή όμως, μόνο τρία ή 4). Υποτίθεται, έτσι, ότι μόνο μερικά από τα μάτια «κοιμούνταν» σε κάθε στιγμή, ενώ πάντα θα υπήρχαν κάποια που θα έμεναν ανοικτά. Σε αγγειογραφίες εμφανίζεται ως διπρόσωπος και με σώμα γεμάτο από μάτια (Βικιπαίδεια, 2008).

Περιεχόμενα

Περιεχόμενα Εικόνων

Περίληψη

Η παρούσα εργασία περιγράφει τα απαραίτητα θεωρητικά τμήματα που ερευνήθηκαν για την δημιουργία μίας εφαρμογής η οποία έχει την δυνατότητα να παρακολουθεί μέσω SNMP οποιοδήποτε δίκτυο ασύρματο ή ενσύρματο, με όλες τις πιθανές παραμέτρους του. Επίσης η εφαρμογή υλοποίει πρωτότυπους αλγόριθμους εύρεσης θέσης ενός κινούμενου σταθμού, εκμεταλλευόμενη την πληροφορία της στάθμης του σήματος εκπομπής των ασύρματων δικτύων (802.11).

Η εφαρμογή είναι ανοιχτή, επιδέχεται παραμετροποίησης από την μεριά του χρήστη και εκμεταλλεύεται πλήρως τις σύγχρονες τεχνολογίες. Η εφαρμογή είναι αρθρωτή, αποτελούμενη από 3 βασικά τμήματα, παρέχοντας έτσι την δυνατότητα, το κάθε τμήμα της εφαρμογής να είναι εγκαταστημένο στο προσφορότερο σημείο του δικτύου. Π.χ. το κομμάτι γραφικού περιβάλλοντος του χρήστη μπορεί να βρίσκεται μακριά από την βάση δεδομένων και να έχει πρόσβαση στα δεδομένα μέσω τοπικού δικτύου ή μέσω διαδικτύου.

Προς την κατεύθυνση της παρουσίασης μίας ολοκληρωμένης εφαρμογής, έγινε έρευνα πεδίου και δειγματοληψία, ώστε να λυθούν πρακτικά προβλήματα σχετικά με την στάθμη σήματος (SNR) των σταθμών σχετικά με την απόσταση και τα επιμέρους υλικά που συνεισφέρουν στην στάθμη του σήματος (καλώδια, κεραίες κτλ). Παράλληλα εντοπίστηκαν ελαττώματα και ασάφειες του πρωτοκόλλου 802.11, όπως το ότι δεν ορίζει σαφώς το είδος της πληροφορίας της στάθμης του σήματος μεταξύ των σταθμών εκπομπής και λήψης, και το εύρος διακύμανσης της.

Λύθηκαν τα μεγάλα πρακτικά προβλήματα που παρουσιάστηκαν για την εύρεση θέσης (positioning) χρησιμοποιώντας την πληροφορία της στάθμης σήματος (RSSI). Μεταξύ άλλων, δημιουργήθηκε εφαρμογή που αντλεί πληροφορία από χαμηλού επιπέδου βιβλιοθήκες (WMI, NDIS) της Microsoft.

Δημιουργήθηκε εξ αρχής, μία 3-tier εφαρμογή για να μπορεί να διαχειριστεί τον απαιτούμενο όγκο πληροφορίας. Μία βάση δεδομένων (SQL) αποθηκεύει σε πραγματικό χρόνο την διαθέσιμη πληροφορία SNMP, μαζί με την πληροφορία της στάθμης σήματος που αποστέλλουν οι υπό έρευνα σταθμοί μέσω της εφαρμογής που δημιουργήθηκε για αυτό το σκοπό. Μία server εφαρμογή επικοινωνεί με την βάση αντλεί συνεχώς δεδομένα

και υπολογίζει συνεχώς την θέση των κινούμενων σταθμών, και τέλος η εφαρμογή πελάτης, απεικονίζει σε γραφικό περιβάλλον τους σταθμούς βάσης σε x,y συντεταγμένες, και τους σταθμούς πελάτες να κινούνται στον χώρο σε πραγματικό χρόνο.

Λέξεις - κλειδιά: Δίκτυα, Ασύρματα Δίκτυα, Εύρεση θέσης, Παρακολούθηση δικτύων, Wi-Fi, SNMP, positioning, RTLS, SNR, FSPL

Εισαγωγή

"I'm trying to free your mind, Neo. But I can only show you the door. You're the one that has to walk through it." (Morpheus, MATRIX I, 1999).

Οι εξελίξεις στον τομέα των δικτύων (των ασύρματων περισσότερο αλλά και των ενσύρματων) ακολουθούν την δυναμική της επιστήμης της πληροφορικής γενικότερα. Αναπτύσσονται και εξελίσσονται ραγδαία, καθιστώντας την παρακολούθηση και συντήρηση τους ένα ιδιαίτερα δύσκολο και απαιτητικό στόχο. Ιδιαίτερα στον τομέα των ασύρματων δικτύων, τα τελευταία χρόνια βιώνουμε μία μεγάλη αναβάθμιση, τόσο σε επίπεδο ταχυτήτων, όσο και σε επίπεδο ποιότητας της υπηρεσίας (Quality of Service, QoS). Τα ασύρματα δίκτυα εξαπλώνονται ραγδαία εκμεταλλευόμενα στο έπακρο τις ευκολίες που προσφέρουν (χαμηλό κόστος αγοράς και εγκατάστασης, άμεση υλοποίηση, εγκατάσταση σε οποιοδήποτε περιβάλλον και κτίριο). Μαζί με τα πλεονεκτήματα αυτά όμως έρχονται και η ανάγκη παρακολούθησης, συντήρησης, ελέγχου και περιφρούρησης αυτών των υποδομών. Οι άνθρωποι που αναλαμβάνουν αυτές τις εργασίες, χρειάζονται εργαλεία που να τους δίνουν άμεση απάντηση στα πολύπλοκα ερωτήματα που εγείρονται, εργαλεία που να προσαρμόζονται στις ανάγκες τους, και να μπορούν να εξελιχθούν μαζί με τα δίκτυα που υποστηρίζουν.

Ένα από τα πολύτιμα εργαλεία για αυτήν την απαιτητική αποστολή είναι το πρωτόκολλο SNMP. Το πρωτόκολλο αυτό είναι απλό στην χρήση του, είναι αρκετά ασφαλές (όταν τηρούνται οι κανόνες ασφαλείας), και είναι ευρέως διαδομένο, αφού η συντριπτική πλειοψηφία των όποιων δικτυακών συσκευών, το υλοποιεί. Είναι ένα πρωτόκολλο με το οποίο μπορούμε να συντάξουμε πολύ απλά και εύκολα υλοποιήσιμα (σε γραμμή εντολών) ερωτήματα, να τα αποστείλουμε σε μία συσκευή, και να λάβουμε την απάντηση στο ερώτημα μας. Επίσης δίνει την δυνατότητα να παραμετροποιήσουμε μία συσκευή ή να αλλάξουμε τις όποιες παραμέτρους. Είναι ανοιχτό στην χρήση του (χωρίς δικαιώματα) και τα βασικά του ερωτήματα υλοποιούνται με τον ίδιο τρόπο σε ομοειδείς συσκευές.

Αυτή η εργασία έχει ως σκοπό να βάλει τις βάσεις για την δημιουργία μίας πρότυπης εφαρμογής –μοντέλου, μέσω της οποίας μπορούν να επιτευχθούν δύο στόχοι:

- Η παρακολούθηση ενός δικτύου συσκευών μέσω του πρωτοκόλλου SNMP με όλες τις δυνατότητες που αυτό δίνει. Οι δυνατότητες αυτές είναι απεριόριστες, και μέσω αυτών μπορεί κάποιος να παρακολουθεί σε πραγματικό χρόνο τα δεδομένα που ανταλλάσει μία συσκευή με μία άλλη και αφορούν ένα συγκεκριμένο πρωτόκολλο δικτύου (π.χ. ftp δεδομένα) ή το πότε ήταν η τελευταία φορά που μία συσκευή έκανε επανεκκίνηση, πόσα λάθη έχει σε κάθε αποστολή δεδομένων κτλ.
- Μέσω της υπάρχουσας πληροφορίας της στάθμης σήματος (SNR) μεταξύ ενός σταθμού–πελάτη (wireless client) και ενός σταθμού εκπομπής (access point ή AP) μπορούμε χρησιμοποιώντας την εξίσωση της διάδοσης του ηλεκτρομαγνητικού κύματος (ή συνεπακόλουθα σήματος) στον χώρο, να βρούμε την απόσταση μεταξύ τους και στην συνέχεια χρησιμοποιώντας τις εξισώσεις της τριγωνοποίησης να προσδιορίσουμε με σχετική ακρίβεια την θέση του σταθμού - πελάτη στον χώρο. Είναι αξιοσημείωτο ότι η πληροφορία αυτή υπάρχει (για διαφορετικό σκοπό) και δεν χρειάζεται επιπλέον εξοπλισμός για την άντληση της.

Η εργασία αυτή χωρίζεται σε τέσσερα βασικά τμήματα:

- Στο πρώτο τμήμα περιγράφονται οι δυνατότητες του πρωτοκόλλου SNMP. Είναι αρκετά εκτενές, λόγω της σπουδαιότητας (παραγνωρισμένης αρκετές φορές) του πρωτοκόλλου αυτού στην διαχείριση των δικτύων, αλλά και γιατί δεν βρέθηκε κάποιος αντίστοιχος οδηγός στα Ελληνικά.
- Στο δεύτερο τμήμα περιγράφονται οι ιδιότητες των ασύρματων δικτύων που ήταν απαραίτητες για την εργασία. Επίσης περιγράφονται οι δυσκολίες και τα προβλήματα που παρουσιάστηκαν κατά την εύρεση και καταγραφή της στάθμης του σήματος ενός ασύρματου σταθμού, στοιχείου απαραίτητου για το επόμενο τμήμα της εργασίας.
- Στο τρίτο τμήμα περιγράφονται οι συναρτήσεις και ο αλγόριθμος που κατασκευάστηκαν για την εύρεση ενός σημείου στον χώρο μέσω της γνώσης της απόστασης του από τουλάχιστον τρία γνωστά σημεία στον χώρο (τριγωνοποίησης)
- Στο τέταρτο τμήμα περιγράφεται η εφαρμογή που αναπτύχθηκε, η οποία μέσω γραφικού περιβάλλοντος παρακολουθεί σε πραγματικό χρόνο την θέση των κινούμενων στον χώρο ασύρματων σταθμών. Παρακολουθεί συνεχώς όλες τις λειτουργίες αυτών των σταθμών ενημερώνοντας μία βάση δεδομένων, ενημερώνει τον χρήστη για την όποια δυσλειτουργία του συνολικού δικτύου. Είναι τελείως

ανοιχτή και παραμετροποιήσιμη, μπορεί δηλαδή να προσαρμοστεί στις ανάγκες και δυνατότητες οποιουδήποτε δικτύου.

Η εφαρμογή υλοποιεί συνοπτικά:

- το πρωτόκολλο SNMP και επικοινωνεί μέσω αυτού με οποιεσδήποτε συσκευές το υλοποιούν αντίστοιχα,
- τους αλγόριθμους εύρεσης θέσης ενός κινούμενου σταθμού μέσω της γνώσης των αποστάσεων από γνωστά σημεία εκπομπής. Οι αλγόριθμοι εκτελούνται συνεχώς και βρίσκουν την θέση του κινούμενου σταθμού σε πραγματικό χρόνο.

1. Συναφείς Εργασίες – Προϊόντα

Γενικώς όλα τα συστήματα εύρεσης θέσης (positioning), έχει καθιερωθεί να ονομάζονται RTLS (Real Time Location System, Εύρεση θέσης σε πραγματικό χρόνο).

Η Cisco έχει υλοποιημένο ένα αντίστοιχο σύστημα, που στρέφεται όμως σε κλειστούς χώρους. Είναι συνδυασμός software και hardware (wi-fi tags)[1] και φυσικά είναι ένα κλειστό σύστημα, του οποίου είναι γνωστές μόνο οι εμπορικές παράμετροι. Συνεργάζεται με hardware tags διαφόρων κατασκευαστών για να λειτουργήσει (Cisco, 2008).

Η Aeroscout είναι από τις ελάχιστες (η μόνη μαζί με την Ekahau) της οποίας το προϊόν μπορεί να λειτουργήσει και σε εσωτερικούς και σε εξωτερικούς χώρους. Το σύστημα της είναι επίσης συνδυασμός software και hardware. Είναι η πρώτη που παρουσίασε ένα RFID tag βασισμένο σε wi-fi το 2003. Το σύστημα της βασίζεται στην εκπομπή ενός συμβατού με το 802.11 σήματος από τα RFID tags προς τους σταθμούς εκπομπής. Έτσι όποιος κινούμενος «στόχος» έχει επάνω του το tag μπορεί να καταγραφεί η θέση του (AeroScout , 2008).

Το κοντινότερο σύστημα σε αυτό που περιγράφεται στην παρούσα εργασία, είναι το σύστημα της Ekahau. Είναι φυσικά ένα κλειστό εμπορικό σύστημα, και ικανό να αποδώσει τόσο σε ανοιχτό όσο και σε κλειστό χώρο. Τα συγκριτικά του πλεονεκτήματα είναι ότι φαίνεται να κυριαρχεί στην αγορά βασικά λόγω αξιοπιστίας, αλλά και λόγω του ότι δεν κάνει χρήση κανενός είδους hardware και είναι «ανοιχτό» σύστημα, μπορώντας να συνεργαστεί τόσο με αρκετούς σταθμούς εκπομπής (Access Points) όσο και με αρκετές κάρτες δικτύου διαφορετικών κατασκευαστών (Ekahau, 2008).

Γενικώς, αυτά τα συστήματα (RTLS) απευθύνονται σε ιδιαιτέρως προσοδοφόρες (αλλά και πολύ απαιτητικές) αγορές. Έχουν άμεσες εφαρμογές σε νοσοκομεία για την παρακολούθηση και εύρεση της θέσης πολύτιμων φορητών εργαλείων, είτε για την άμεση

1 Εδώ πρέπει να σημειωθεί ότι θεωρητικά δεν υπάρχει τίποτα που να υπαγορεύει την χρήση και hardware για την υλοποίηση τέτοιων μεθόδων. Προφανώς οι εταιρίες στην προσπάθεια τους να αποτρέψουν την παράνομη χρήση των συστημάτων τους, καταφεύγουν στην εμπλοκή του hardware στις υλοποιήσεις τους μόνο και μόνο για να κάνουν απαραίτητη την αγορά του υλικού και του συνεπακόλουθου λογισμικού από τους ενδιαφερόμενους-υποψήφιους πελάτες.

χρήση τους, είτε για την ασφάλεια έναντι κλοπής και απώλειας. Επίσης μπορεί να παρακολουθεί εξειδικευμένο προσωπικό (γιατρούς επειγόντων περιστατικών, φρουρούς εγκαταστάσεων) και να καλεί-εντοπίζει τον κοντινότερο. Μπορεί να χρησιμοποιηθεί σε εφαρμογές logistics π.χ. σε λιμάνια για την παρακολούθηση των εμπορευματοκιβωτίων (container) ή/και των οχημάτων μεταφοράς αυτών (carriers- straddles).

Μία ιδιαιτέρως απαιτητική και ενδιαφέρουσα εφαρμογή τέτοιων συστημάτων περιγράφεται στην εργασία για την (Interlink Networks Inc, 2002) όπου ουσιαστικά περιγράφεται η προσπάθεια καταγραφής της φυσικής θέσης ενός μη-εξουσιοδοτημένου σταθμού πελάτη που προσπαθεί να εισβάλει σε ένα εταιρικό δίκτυο. Είναι προφανές ότι είναι αναγκαία η καταγραφή της φυσικής θέσης ενός τέτοιου εισβολέα, στην προσπάθεια των όποιων υπηρεσιών ασφαλείας (ιδιωτική ασφάλεια – security, αστυνομικές αρχές) να σταματήσουν –συλλάβουν τον επίδοξο εισβολέα.

Οι σχετικές με το SNMP εργασίες και εφαρμογές είναι ανεξάντλητες. Υπάρχουν πάρα πολλές εμπορικές εφαρμογές και πολλές από αυτές είναι δωρεάν. Το πρόβλημα μέχρι σήμερα με την πλειονότητα αυτών των εφαρμογών είναι ότι ήταν πολύ μονολιθικές. Δηλαδή δεν υπήρχε δυνατότητα προσαρμοστικότητας (adaptability) και της συνεπαγόμενης ευχρηστίας (usability). Τελευταία όμως παρουσιάστηκαν εφαρμογές που είναι αρκετά (έως πλήρως παραμετροποιήσιμες. Ενδεικτικά αναφέρουμε την εφαρμογή Dude (Mikrotik, 2008) η οποία είναι δωρεάν και αρκετά παραμετροποιήσιμη. Επίσης ένα πολύ καλό εμπορικό προϊόν, είναι το προϊόν της SolarWinds για τα ασύρματα δίκτυα (SolarWinds, 2008). Είναι πλήρως παραμετροποιήσιμο και ευέλικτο (αλλά αρκετά ακριβό ειδικά για μικρές εγκαταστάσεις).

Στην έρευνα μας βρήκαμε κάποιες εργασίες (papers) εκ των οποίων οι περισσότερες ήταν (ίσως σκοπίμως) ασαφείς ως προς την πρακτική πλευρά του θέματος.

Μία συλλογή αρκετά θεωρητικών τέτοιων προσεγγίσεων υπάρχει στο (Qiang Yang, 2007). Κάποιες από τις ιδέες που αναφέρονται χρειάστηκαν αρκετή προσαρμογή στην πρακτική πλευρά του πράγματος.

Ακόμη μία πολύ ενδιαφέρουσα εργασία, είναι η (Paramvir Bahl, 2008). Αντιμετωπίστηκαν όλα τα προβλήματα που αντιμετώπισε η παρούσα εργασία, όπως η εξαγωγή της

πληροφορίας της στάθμης του σήματος (χωρίς να αποκαλύπτονται λεπτομέρειες…), το στατιστικό λάθος, ο αριθμός των σταθμών και η απόσταση από τον κινούμενο σταθμό.

Στην εργασία (Antti Seppänen, 2008) μεταξύ άλλων περιγράφεται η δυνατότητα εξαγωγής της στάθμης του σήματος μέσω SNMP. Και αυτή η εργασία φαίνεται ότι έγινε σε θεωρητικό επίπεδο, αφού κάποια από τα συμπεράσματα όπως αυτό, θα δούμε αργότερα ότι δεν επαληθεύονται στην πράξη (Το SNMP δεν δίνει πάντα αυτήν την πληροφορία).

Η εργασία (Antti Kotanen, Positioning with IEEE 802.11b Wireless LAN, 2008) έχει κάποιες πολύ ενδιαφέρουσες ιδέες, δείχνει ότι έγινε πρακτική αντιμετώπιση του θέματος με σημαντικά αποτελέσματα, και ήταν η βασική πηγή για την ιδέα του φίλτρου (kalman filter) που προτείνεται ως βελτίωση της αποτύπωσης της πορείας του κινούμενου σταθμού.

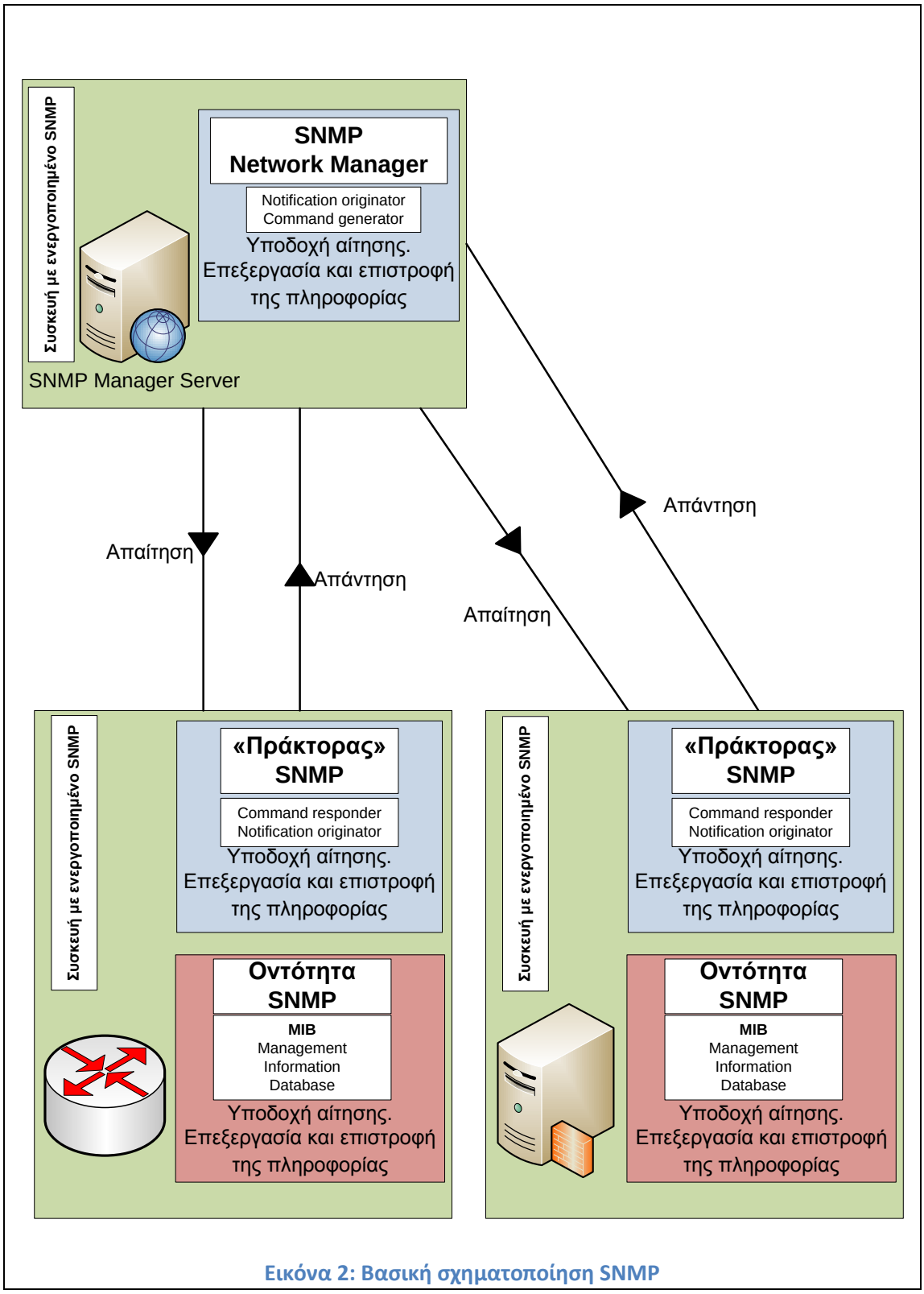
Από την μέχρι τώρα έρευνα μας δεν προκύπτει κάποιο προϊόν στην αγορά που να συγκεράζει τα δύο παραπάνω περιγραφόμενα πεδία. Πιστεύουμε ότι υπάρχει χώρος για ένα τέτοιο προϊόν στην παγκόσμια αγορά, αν φυσικά εξελιχθεί και δοκιμαστεί κατάλληλα.

2. SNMP

«Στα σημερινά πολύπλοκα δίκτυα γεμάτα από router, switches και servers, μοιάζει πολύ κουραστική η διαχείριση όλων αυτών των συσκευών, ώστε να γνωρίζουμε όχι μόνο ότι βρίσκονται σε λειτουργία, αλλά ότι επιτελούν την εργασία τους με τον καλύτερο τρόπο..»(Mauro, Mauro, & Schmidt, 2005).

2.1. Τι είναι το SNMP

Το SNMP είναι ένα σύνολο από απλές εντολές (ή διεργασίες), και από τα δεδομένα που αυτές οι εντολές-διεργασίες συλλέγουν. Διαθέτοντας αυτήν την πληροφορία και την δυνατότητα αποστολής εντολών αλλαγής κατάστασης, οι διαχειριστές (administrators) ενός δικτύου, μπορούν *«… να ελέγξουν την ταχύτητα και το φόρτο εργασίας σε συγκεκριμένη διεπαφή ενός ρούτερ, ή να ελέγχουν συνεχώς την θερμοκρασία λειτουργίας ενός switch και να παρεμβαίνουν ανάλογα με την στάθμη της…»* (Mauro, Mauro, & Schmidt, 2005).



Εικόνα 2: Βασική σχηματοποίηση SNMP

2.2. Σύντομη περιγραφή του SNMP

Ένα σύστημα διαχείρισης SNMP αποτελείται από:

- Αρκετές (κατά προτίμηση πολλές) οντότητες δικτύου, κάθε μία από τις οποίες μπορεί να ανταποκρίνεται σε εντολές SNMP με κατάλληλα μηνύματα και ενέργειες
- Τουλάχιστον μία οντότητα η οποία έχει την δυνατότητα να δεχθεί μηνύματα SNMP[2] ή να δημιουργήσει τέτοιου είδους εντολές (που συνήθως αποκαλείται "Διαχειριστής SNMP"), και
- Ένα πρωτόκολλο διαχείρισης το οποίο χρησιμοποιείται για να μεταφέρει τις πληροφορίες μεταξύ των συμμετεχόντων μερών (RFC 3411, 2002)

Οι συμβατές με SNMP οντότητες δημιουργούν σύμφωνες με το πρωτόκολλο εντολές ή πληροφορίες και διαχειρίζονται συμβατές συσκευές. Συμβατές είναι συσκευές όπως Η/Υ, δρομολογητές δικτύου (routers), switches/hubs, συσκευές απομακρυσμένης διαχείρισης (terminal server), κάμερες ή όποια άλλη συσκευή έχει δυνατότητα δικτύωσης, των οποίων η διαχείριση γίνεται εφικτή μέσω της πρόσβασης στην πληροφορία SNMP (RFC 3411, 2002).

Στόχος του SNMP είναι να διαχειριστεί αποτελεσματικά ανομοιογενή περιβάλλοντα και καταστάσεις. Μία τέτοια αρχιτεκτονική πρέπει να διαχειρίζεται τα εξής:

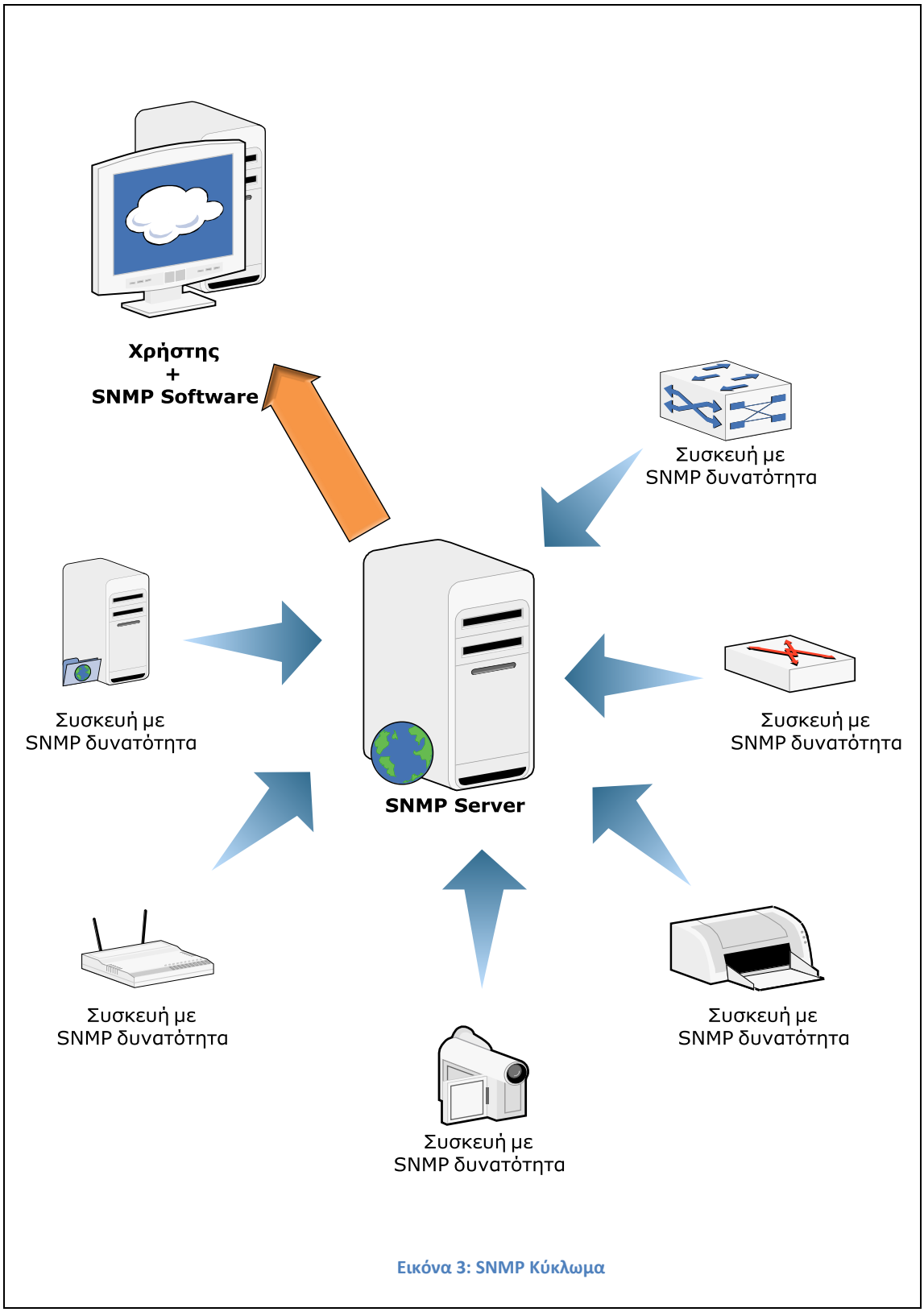
- Οντότητες που υλοποιούν το SNMP και ανταποκρίνονται σε εντολές ή «απαντούν» σε κατάλληλες ερωτήσεις εφαρμογών που έχουν τέτοιες δυνατότητες και ονομάζονται «Πράκτορες» SNMP (Agents)
- Οντότητες που μπορούν να δεχθούν και να αποστείλουν εντολές και ονομάζονται Διαχειριστές γραμμής εντολών (command line managers)
- Οντότητες που μπορούν να δεχθούν και να αποστείλουν εντολές και επίσης ανταποκρίνονται σε εντολές ή «απαντούν» σε κατάλληλες ερωτήσεις εφαρμογών και ονομάζονται οντότητες διπλού ρόλου (dual role entities)
- Οντότητες SNMP που μπορούν να δεχθούν και να αποστείλουν εντολές ενώ παράλληλα μπορούν να υλοποιήσουν πολλαπλές άλλες εφαρμογές και

[2] Ο όρος SNMP οντότητα, δηλώνει μία οποιαδήποτε συσκευή (συνήθως δικτυακή αλλά με την ευρεία έννοια) που έχει ενεργοποιημένο το συγκεκριμένο πρωτόκολλο – δυνατότητα. Απλουστευτικά μιλώντας, μία ψηφιακή συσκευή από όλο το φάσμα της τεχνολογίας όπως Η/Υ, router, hub/switch, network camera, αλλά και οποιαδήποτε συσκευή προηγμένης τεχνολογίας όπως π.χ. ένας ψηφιακός αποκωδικοποιητής, μπορεί να υλοποιεί το πρωτόκολλο SNMP και συνεπακόλουθα να δεχθεί διαχείριση ή να παράσχει πληροφορίες μέσω του πρωτοκόλλου.

προγράμματα και μπορούν έτσι να διαχειριστούν ένα μεγάλο αριθμό οντοτήτων και αποκαλούνται Σταθμοί διαχείρισης (δικτύου), (Network Management Stations)(RFC 3411, 2002).

2.3. Εφαρμογές SNMP

Μία οντότητα SNMP συνήθως υλοποιεί έναν πλήθος εφαρμογών. Οι εφαρμογές χρησιμοποιούν τις υπηρεσίες ενός πράκτορα SNMP για να υλοποιήσουν συγκεκριμένες εργασίες. Συγχρονίζουν τις διαδικασίες διαχείρισης πληροφορίας και μπορεί να χρησιμοποιούν τις δυνατότητες του SNMP για να επικοινωνήσουν (στέλνοντας κατάλληλα μηνύματα) σε άλλες SNMP οντότητες.



Εικόνα 3: SNMP Κύκλωμα

2.4. Διάρθρωση της πληροφορίας διαχείρισης

Η πληροφορία διαχείρισης είναι μία συλλογή από παραμετροποιήσιμα (διαχειρήσιμα) αντικείμενα) τα οποία βρίσκονται αποθηκευμένα σε μία ιδεατή βάση η οποία ονομάζεται MIB (Management Information Base). Συλλογές συναφών (συγγενικών) αντικειμένων καθορίζονται στις υπομονάδες ενός MIB (MIB modules).
Μοντέλο (Model) ονομάζεται η συγκεκριμένη σχεδίαση ενός υποσύστηματος (subsystem) SNMP ώστε να ικανοποιεί απαιτήσεις και ιδιαιτερότητες των συσκευών στις οποίες απευθύνεται.

2.5. Εκδόσεις του SNMP

Υπάρχουν 3 εκδόσεις (Versions) του SNMP:

- SNMP έκδοση 1 (SNMPv1), η αρχική έκδοση, η οποία περιγράφεται στα RFCs 1155, 1157, and 1212
- SNMP έκδοση 2 (SNMPv2), είναι ο άμεσος απόγονος της έκδοσης 1 και περιγράφεται στα STD 58, RFCs 2578, 2579, 2580,και STD 62, RFCs 3416, 3417, 3418
- SNMP έκδοση 3 (SNMPv3) που είναι μία επέκταση της έκδοσης 2 και υποστηρίζει επιπλέον:
    - Νέα διάρθρωση μηνύματος SNMP
    - Ασφάλεια μηνυμάτων
    - Διαχείριση πρόσβασης, και
    - Απομακρυσμένη διαχείριση παραμέτρων SNMP

2.6. Δομικά στοιχεία της αρχιτεκτονικής του SNMP

Μία οντότητα SNMP είναι μία συσκευή που υλοποιεί αυτήν την δυνατότητα (SNMP). Κάθε οντότητα SNMP αποτελείται από την μηχανή SNMP που υλοποιεί, και μία ή περισσότερες συνδεδεμένες με αυτή εφαρμογές.

2.6.1. Μηχανή SNMP

Παρέχει τις απαραίτητες λειτουργίες αποστολής και λήψης μηνυμάτων SNMP, αυθεντικοποίησης και κρυπτογράφησης τους. Επίσης ελέγχει την πρόσβαση στις οντότητες

(συσκευές) υπό διαχείριση. Υπάρχει μία αντιστοιχία ένα προς ένα, μεταξύ της μηχανής SNMP και της οντότητας που την περιέχει. Έτσι κάθε συσκευή SNMP έχει ένα μοναδικό αναγνωριστικό που την ταυτοποιεί πέραν πάσης αμφιβολίας σε ένα περιβάλλον πολλαπλών όμοιων συσκευών (π.χ. πολλαπλοί ίδιοι ρούτερ). Η μηχανή SNMP αποτελείται από:

- Τον αποστολέα (dispatcher):
- Το υποσύστημα επεξεργασίας μηνυμάτων
- Το υποσύστημα ασφαλείας
- Το υποσύστημα έλεγχου πρόσβασης

2.6.2. Εφαρμογές SNMP

Υπάρχουν διάφοροι τύποι εφαρμογών SNMP, οι οποίες περιλαμβάνουν:

- Γεννήτριες εντολών (command generator), οι οποίες παρακολουθούν και διαχειρίζονται την παραγόμενη πληροφορία
- Γεννήτριες απόκρισης (command responder), παρέχουν πρόσβαση στην πληροφορία
- Γεννήτριες ειδοποιήσεων (notification originator), δημιουργούν ασύγχρονα μηνύματα
- Εφαρμογές διαμεταγωγής (proxy), μεταφέρουν μηνύματα μεταξύ οντοτήτων (RFC 3413, 2002)

2.6.3. Διαχειριστής SNMP (SNMP Manager)

Μία οντότητα SNMP η οποία εμπεριέχει μία ή περισσότερες εφαρμογές γεννήτριας εντολών, και/ή μία γεννήτρια ειδοποιήσεων, καλείται Διαχειριστής (Manager) SNMP.

2.6.4. Πράκτορας SNMP (SNMP Agent)

Μία οντότητα SNMP που εμπεριέχει μία ή περισσότερες γεννήτριες απόκρισης και/ή εφαρμογές με γεννήτρια ειδοποιήσεων, ονομάζεται «Πράκτορας» SNMP (SNMP Agent).

2.6.5. Πλαίσιο SNMP (SNMP context)

Το πλαίσιο SNMP είναι μία συλλογή από διαχειρίσιμη πληροφορία από μία οντότητα SNMP. Ένα συγκεκριμένο αντικείμενο (ίδιο) μπορεί να εμπεριέχεται μέσα σε περισσότερα του ενός, πλαίσια. Μία οντότητα SNMP, δυνητικά έχει την δυνατότητα πρόσβασης σε

πολλά πλαίσια. Μέσα σε ένα δίκτυο, πολλές ίδιες συσκευές, παράγουν το ίδιο είδος πλαισίου (άρα και κατηγορία πληροφορίας) ταυτοποιούνται όμως μοναδικά μέσω διαφορετικών παραγόμενων πλαισίων με το μοναδικό αναγνωριστικό (ID) της συγκεκριμένης συσκευής.

2.7. MIB

Οι διαχειρίσιμες από (με) SNMP οντότητες είναι προσβάσιμες μέσω μίας ιδεατής βάσης δεδομένων που καλείται **Management Information Base (MIB).** Τα αντικείμενα-περιεχόμενα αυτής της βάσης είναι προσβάσιμα και επεξεργάσιμα μέσω του SNMP(RFC 3410, 2002). Συλλογές συγγενών αντικειμένων αυτής της βάσης ομαδοποιούνται στις αντίστοιχες υπομονάδες της βάσης (MIB) (RFC 2578, 1999).

2.7.1. Δομή Πληροφορίας (Structure of Management Information, SMI)

Η πληροφορία όπως ειπώθηκε μπορεί να ιδωθεί ως συλλογή από διαχειρίσιμα αντικείμενα της ιδεατής βάσης (MIB). Οι συλλογές ομοειδών αντικειμένων της βάσης ομαδοποιούνται σε υπομονάδες ακολουθώντας την δενδρική διάταξη και την αυστηρά καθορισμένη αναπαράσταση της πληροφορίας όπως περιγράφεται λεπτομερώς στο πρότυπο ASN.1 (ITU-T, 2002).

Η δομή της πληροφορίας μπορεί να χωριστεί σε 3 βασικές κατηγορίες: (module definitions), (object definitions), και (notification definitions)(RFC 3410, 2002).

- **Module definitions** χρησιμοποιείται για να περιγράψει τις υπομονάδες που παράγουν πληροφορία. Χρησιμοποιεί το MODULE-IDENTITY του ASN.1 για να περιγράψει την παραγόμενη πληροφορία
- **Object definitions** χρησιμοποιείται για να περιγράψει διαχειρίσιμες οντότητες-αντικείμενα. Το OBJECT-TYPE του ASN.1 χρησιμοποιείται για να περιγράψει την παραγόμενη πληροφορία
- **Notification definitions** χρησιμοποιείται για να περιγράψει την παραγωγή πληροφορίας άνευ προηγούμενης ζήτησης. Το NOTIFICATION-TYPE του ASN.1 χρησιμοποιείται για να περιγράψει την παραγόμενη πληροφορία (RFC 3410, 2002)

2.7.2. Περιγραφή της πληροφορίας βάσης SMI

Οι κατηγορίες των δεδομένων που παράγει μία MIB βάση είναι:

Πίνακας 1: Τύποι Δεδομένων MIB

Integer32	Gauge32	TimeTicks	STRING	Opaque
Enumerated integers	Counter32	INTEGER	Unsigned32	BITS
OBJECT IDENTIFIER	Counter64	OCTET	IpAddress	

Επίσης ως δομές δεδομένων περιγράφονται και οι εξής:

- **IMPORTS** επιτρέπει την περιγραφή αντικειμένων μίας βάσης MIB Module, μέσα σε μία άλλη ολόκληρη υπομονάδα MIB module.
- **MODULE-IDENTITY** Περιγράφει την ταυτότητα και τα χαρακτηριστικά μίας τέτοιας μονάδας
- **OBJECT-IDENTITY & OID** Η λεπτομερής αριθμητική δενδροειδής αναπαράσταση της συγκεκριμένης παραμέτρου της υπομονάδας
- **OBJECT-TYPE** Περιγράφει το είδος των παραγόμενων δεδομένων, την κατάσταση της συσκευής και την δυνητικά παραγόμενη πληροφορίας
- **SEQUENCE** type assignment Περιγράφει τα διαδοχικά στοιχεία ενός παραγόμενου πίνακα
- **NOTIFICATION-TYPE** Περιγράφει το είδος της παραγόμενης ειδοποίησης (RFC 3410, 2002)

2.7.3. Αρχικοποίηση αντικειμένων στο MIB

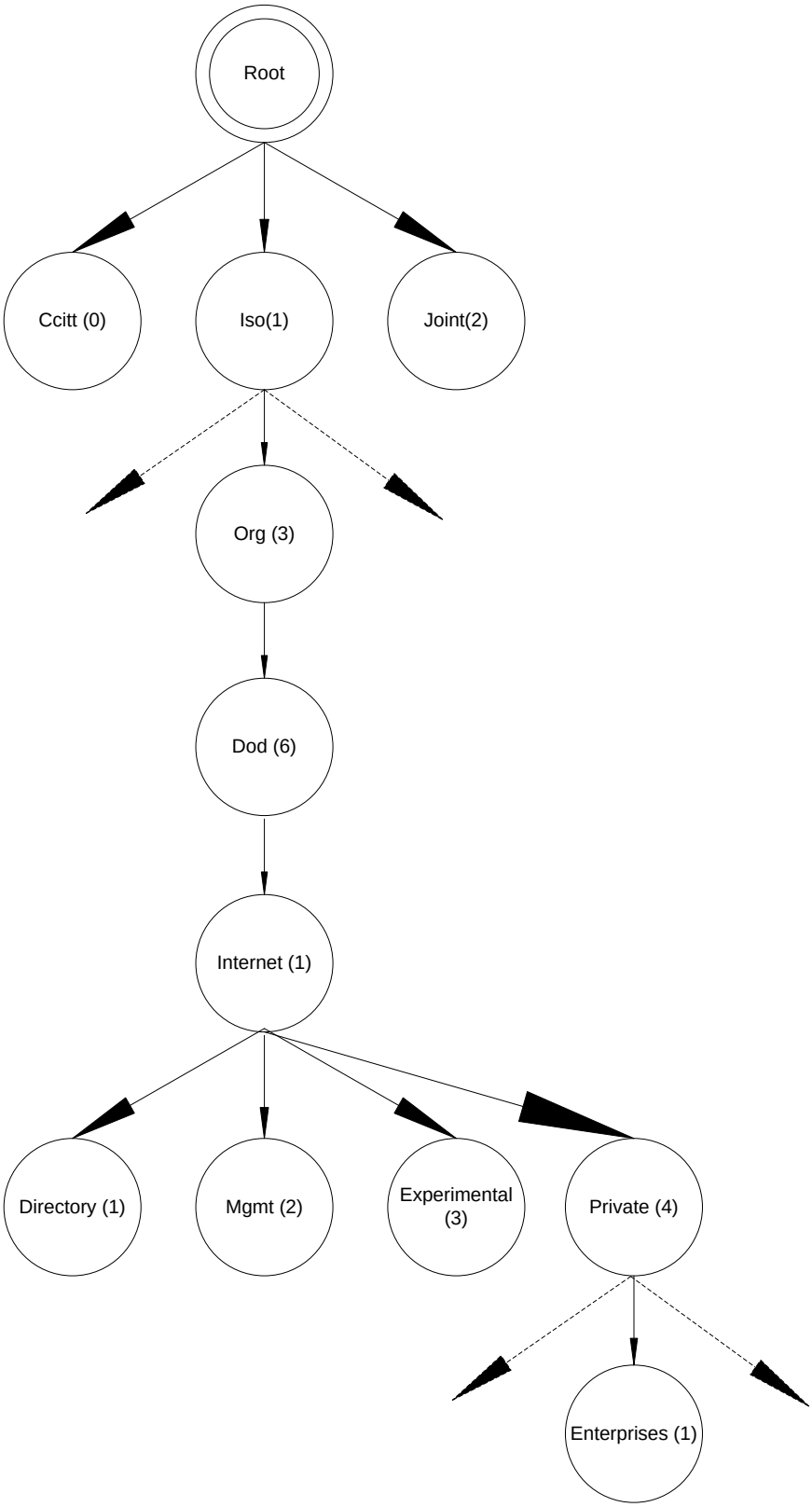
Κάθε τύπος αντικειμένου (object type) έχει ένα όνομα, μία λογική σύνταξη και μία κωδικοποίηση. Το όνομα ορίζεται μοναδικά από το OBJECT IDENTIFIER.

Η λογική σύνταξη του αντικειμένου καθορίζεται από τον τύπο του αντικειμένου. Π.χ. για κάποιο δεδομένο αντικείμενο μπορεί να είναι ακέραιος (INTEGER) ή οκταδική συμβολοσειρά (OCTET STRING). Η κωδικοποίηση είναι απλώς ο κωδικοποιημένος τρόπος αναπαράστασης του συγκεκριμένου αντικειμένου σύμφωνα με το προαναφερθέν ASN.1.

2.8. Ονόματα (OBJECT IDENTIFIER)

Το όνομα (OBJECT IDENTIFIER) χρησιμοποιείται για να αναγνωρίσει μοναδικά ένα αντικείμενο ή και ολόκληρη οικογένεια αντικειμένων. Σε κάθε περίπτωση αναγνωρίζει μοναδικά, είτε μία συσκευή δικτύου όπως π.χ. ένα ρούτερ, είτε μία κατηγορία π.χ. εγγράφων.

Το όνομα είναι μία αλληλουχία ακεραίων αριθμών οι οποίοι διασχίζουν ένα μοναδικό καθολικό δέντρο. Το δέντρο ξεκινά από την ρίζα που συνδέεται με κόμβους. Ο κάθε κόμβος με την σειρά του συνδέεται με άλλους κόμβους-παιδιά. Οι κόμβοι αυτοί ονομάζονται παιδιά του προηγούμενου. Το βάθος του δέντρου δεν είναι καθορισμένο και μπορεί να συνεχίζει για όσο χρειαστεί. Τονίζεται ότι ο κάθε κόμβος μπορεί να ονομαστεί με ένα όνομα αντιπροσωπευτικό της λειτουργίας του.



Εικόνα 4: OID, Αναπαράσταση αρχικών κόμβων

Η ρίζα του δέντρου δεν έχει όνομα, αλλά έχει τουλάχιστον 3 παιδιά:

- Το 1^{ο} κόμβο τον διαχειρίζεται ο διεθνής οργανισμός προτυποποιήσεως (International Organization for Standardization)και ονομάζεται iso(1).
- Ο 2^{ος} κόμβος διαχειρίζεται από την διεθνή επιτροπή τηλέγραφου και τηλεφώνου (International Telegraph and Telephone Consultative Committee) με το όνομα ccitt(0), και
- Ο 3^{ος} κόμβος είναι προϊόν συνδιαχείρισης των δύο προηγούμενων (ISO και CCITT) με το όνομα joint-iso-ccitt(2).

Κάτω από τον κόμβο iso(1) ο ISO έχει ορίσει ένα υποδέντρο για χρήση άλλων διεθνών οργανισμών με την ονομασία org(3). Δύο από τα παιδιά αυτού του κόμβου έχουν ανατεθεί στο Αμερικανικό Ινστιτούτο τεχνολογίας και προτύπων (U.S. National Institutes of Standards and Technology – NIST). Ένα από αυτά τα υποδέντρα έχει ανατεθεί από την NIST στο υπουργείο Άμυνας της Αμερικής[3] ως dod(6). Αυτό με την σειρά του έχει ορίσει ένα υποδέντρο για το Διαδίκτυο, ώστε να διαχειρίζεται από την αρμόδια επιτροπή (Internet Activities Board - IAB) σύμφωνα με το παρακάτω ορισμό:

internet OBJECT IDENTIFIER ::= { iso org(3) dod(6) 1 }

Για τον παραπάνω λόγο, κάθε δέντρο που ασχολείται με την ονομασία οντοτήτων – αντικειμένων ξεκινά με το αναγκαστικό πρόθεμα:

1.3.6.1.xxxxx

Ακολουθώντας την κωδικοποίηση της IAB , το υποδέντρο κάτω από το κλειδί internet, έχει με την σειρά του 4 κόμβους ως εξής:

```
directory     OBJECT IDENTIFIER ::= { internet 1 }
mgmt          OBJECT IDENTIFIER ::= { internet 2 }
experimental  OBJECT IDENTIFIER ::= { internet 3 }
private       OBJECT IDENTIFIER ::= { internet 4 }
```

2.8.1. Directory

Το υποδέντρο directory(1) είναι δεσμευμένο για μελλοντικές χρήσεις σε σχέση με το OSI.

[3] Η ανάθεση του υποδέντρου στο Υπουργείο Άμυνας της Αμερικής δεν θα πρέπει να ξενίζει. Αν αναλογιστεί κανείς ότι το Ιντερνέτ είναι ο απόγονος του ARPANET το οποίο ήταν στην αρχή ένα στρατιωτικό δίκτυο, τότε εξηγείται και η ευθεία ανάμιξη του Υπουργείου Άμυνας.

2.8.2. Mgmt

Το υποδέντρο mgmt(2) χρησιμοποιείται για αντικείμενα τα οποία ορίζονται από την IAB για την IANA (Internet Assigned Numbers Authority). Εδώ δηλαδή ορίζονται οι αριθμοί για καινούρια αντικείμενα που ορίζονται μέσω των RFC (Request for Comments). Π.χ. το RFC που ορίζει το αρχικό MIB του ίντερνετ, θα πάρει τον αύξοντα αριθμό εγγράφου 1. Έτσι θα έχει το αναγνωριστικό { mgmt 1 } ή

1.3.6.1.2.1

2.8.3. Experimental

Το υποδέντρο experimental(3) χρησιμοποιείται για πειραματικούς σκοπούς και έχει ανατεθεί επίσης στην IANA.

2.8.4. Private

Το υποδέντρο χρησιμοποιείται για τα αντικείμενα που ορίζονται μονομερώς. Ορίζονται δηλαδή από τον κατασκευαστή τους κατά το δοκούν. Και αυτό το υποδέντρο έχει ανατεθεί στην IANA, όσον αφορά στον πρώτο κόμβο του, δηλαδή τον

enterprises OBJECT IDENTIFIER ::= { private 1 }

Αυτό το υποδέντρο χρησιμοποιείται από τους κατασκευαστές και τις επιχειρήσεις για να υποδηλώσουν και να περιγράψουν τα προϊόντα τους. Μόλις μία επιχείρηση αναλάβει ένα υποδέντρο, μπορεί μέσα σε αυτό να περιγράψει όλες τις συσκευές (MIB objects) που επιθυμεί, σε όποιο βάθος του κόμβου. Π.χ. στην εταιρία "Flintstones, Inc." που παράγει προϊόντα δικτύων, της ανατίθεται ένα υποδέντρο από την IANA, έστω το 1.3.6.1.4.1.42

Η εταιρία "Flintstones, Inc." μπορεί τώρα να «δηλώσει» (register) το προϊόν της "Fred Router" κάτω από το υποδέντρο 1.3.6.1.4.1.42.1.1

2.9. Τύποι δεδομένων - Primitive Types

Μόνο οι τύποι δεδομένων που περιγράφονται στο ASN.1 (Application fields of ASN.1, 2005) ως primitive types INTEGER, OCTET STRING, OBJECT IDENTIFIER, & NULL γίνονται δεκτοί.

Κατασκευαστές - Constructor Types

Επιτρέπεται η χρήση του κατασκευαστή (constructor) όπως ορίζεται στο ASN.1 με την προϋπόθεση ότι δημιουργεί (κατασκευάζει) λίστες ή πίνακες σύμφωνα με τις συντάξεις:

Λίστες

SEQUENCE { <type1>, ..., <typeN> }

Όπου <type1>, είναι ένας από τους τύπους δεδομένων που περιγράφονται στο 2.9,

Πίνακες

SEQUENCE OF <entry>

Όπου το <entry> αντιστοιχεί σε έναν «κατασκευαστή» λίστας της προηγούμενης παραγράφου

2.10. Βασικοί τύποι - Defined Types

NetworkAddress Μία διεύθυνση από πιθανά πρωτόκολλα. Προς το παρόν υποστηρίζεται μόνο το πρωτόκολλο Internet.

IpAddress Μήκους 32-bit διεύθυνση ίντερνετ ή μήκους 4 octets.

Counter Μη αρνητικός ακέραιος ο οποίος μονότονα αυξάνει μέχρι μία προκαθορισμένη μέγιστη τιμή, και τότε μηδενίζει και ξανα-αρχίζει να αυξάνεται. Μέγιστη τιμή του η 2^32-1.

Gauge Μη αρνητικός ακέραιος ο οποίος δύναται να αυξομειώνεται διαθέτωντας όμως μία μάξιμουμ τιμή. Αν όχι άλλως ορισμένη, τότε η μέγιστη τιμή του είναι επίσης 2^32-1.

TimeTicks Μη αρνητικός ακέραιος ο οποίος μετράει τον χρόνο σε εκατοστά του δευτερολέπτου ξεκινώντας από κάποια ημερομηνία.

Opaque Χρησιμοποιείται για να μπορέσει να συνταχθεί ένας τύπος αυθαίρετων δεδομένων. Τα δεδομένα χωρίζονται σε octets και μπορούν να χρησιμοποιηθούν αυτούσια από την εφαρμογή που τα παραλαμβάνει-υποδέχεται.

2.11. Διαχειρίσιμα αντικείμενα - Managed Objects

Μία περιγραφή ενός αντικειμένου - συσκευής που μπορεί να ανταλλάξει SNMP πληροφορία, αποτελείται από 5 τμήματα:

OBJECT: Ένα κείμενο το ονομαζόμενο the OBJECT DESCRIPTOR, ακολουθούμενο από το OBJECT IDENTIFIER.

Syntax: Ο ορισμός του αντικειμένου (η σύνταξη δηλαδή της γραμμής που το υποδεικνύει) θα πρέπει να αντιστοιχεί σε ένα στιγμιότυπο του δέντρου των MIB όπως περιγράφεται παρακάτω

Definition: Μία λεκτική αναπαράσταση του αντικειμένου υπο περιγραφή. Θα πρέπει να είναι μοναδική ώστε να έχει νόημα σε οποιαδήποτε άλλη μηχανή υλοποιεί το πρωτόκολλο

Access: Μπορεί να πάρει τις παρακάτω τιμές:

read-only, read-write, write-only, or not-accessible.

Status: Μπορεί να πάρει τις παρακάτω τιμές:

mandatory, optional, or obsolete.

2.12. Object Types and Instances

Ένας τύπος αντικειμένου (object type) είναι ένα είδος αντικειμένων προς διαχείριση (ικανών να δεχτούν διαχείριση, είναι δυνητικά διαχειρίσιμα). Αντίθετα ένα στιγμιότυπο του τύπου αυτού (object instance, δηλαδή ένα συγκεκριμένο άτομο ενός τύπου αντικειμένων) είναι ένα συγκεκριμένο αντικείμενο με μοναδική ταυτότητα.

Οι συλλογές αντικειμένων στο MIB αναγνωρίζονται μοναδικά από το OBJECT IDENTIFIER, το οποίο έχει επίσης ένα όνομα το OBJECT DESCRIPTOR. Έτσι το κάθε συγκεκριμένο αντικείμενο – στιγμιότυπο του συγκεκριμένου τύπου είναι προσβάσιμο μέσω ενός μηχανισμού που είναι συγκεκριμένος για αυτό το πρωτόκολλο. Είναι «ευθύνη» του κάθε μηχανισμού διαχείρισης να ορίσει τις μεθόδους με τις οποίες θα έχει πρόσβαση στα συγκεκριμένα αντικείμενα- στιγμιότυπα.

Στον πίνακα Πίνακας 2, ακολουθούν παραδείγματα αντικειμένων δηλωμένων στο MIB:

Πίνακας 2: 3 διαφορετικά αντικείμενα SNMP

OBJECT:	atIndex { atEntry 1 }	atPhysAddress { atEntry 2 }	atNetAddress { atEntry 3 }

Syntax:	INTEGER	OCTET STRING	NetworkAddress
Definition:	The interface number for the physical address.	The media-dependent physical address.	The network address corresponding to the media-dependent physical address
Access:	read-write.	read-write.	read-write.
Status:	mandatory.	mandatory.	mandatory.

Ένας 4ος τύπος αντικειμένου μπορεί να οριστεί ως εξής:

Πίνακας 3: Αντικείμενο SNMP τύπου Λίστας

OBJECT:	atEntry { atTable 1 }
Syntax:	AtEntry ::= SEQUENCE { atIndex INTEGER, atPhysAddress OCTET STRING, atNetAddressNetworkAddress }
Definition:	An entry in the address translation table.
Access:	read-write
Status:	mandatory

Κάθε στιγμιότυπο αυτού του αντικειμένου περιέχει επιπλέον πληροφορία που αναπαράγεται στα προηγούμενα στιγμιότυπα του συγκεκριμένου τύπου αντικειμένων. Αυτό το είδος αναπαράστασης ονομάζεται λίστα. Με τον ίδιο τρόπο μπορούν να δημιουργηθούν πίνακες οι οποίοι εμπεριέχουν λίστες όπως το παρακάτω 5ο στιγμιότυπο του συγκεκριμένου τύπου αντικειμένων:

Πίνακας 4: Αντικείμενο SNMP τύπου Πίνακας

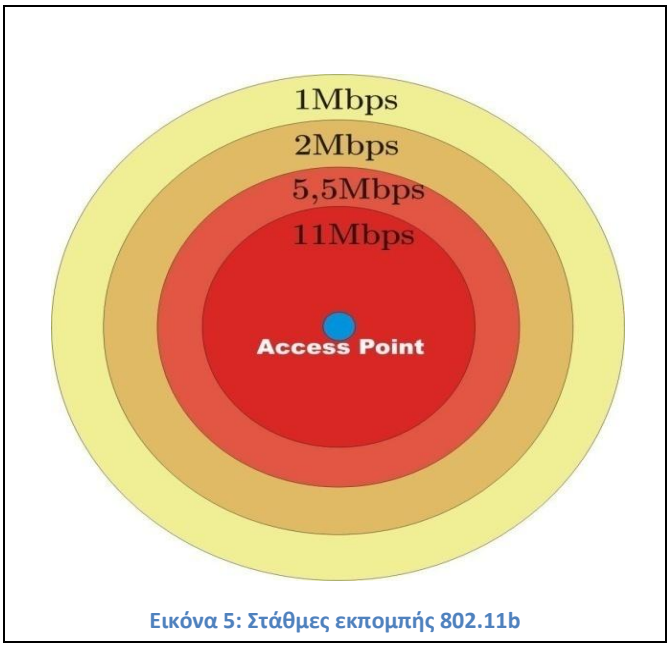
OBJECT:	atTable { at 1 }
Syntax:	atTable { at 1 }
Definition:	The address translation table
Access:	read-write
Status:	mandatory

Το κάθε πρωτόκολλο διαχείρισης προσφέρει συγκεκριμένους μηχανισμούς πρόσβασης στους τύπους που έχουν προηγουμένως οριστεί. Επίσης το πρωτόκολλο θα καθορίσει και τον τύπο της πληροφορίας που θα «επιστρέψει» το αντικείμενο όταν ερωτηθεί από το πρωτόκολλο (RFC 3410, 2002), (Mauro, Mauro, & Schmidt, 2005), (RFC 3411, 2002) .

3. Wi-Fi: Υπόβαθρο και Θεωρία

Τα δίκτυα 802.11 έχουν κατακτήσει την αγορά εδώ και αρκετά χρόνια. Χρησιμοποιούν την συχνότητα των 2,4 GHz, η οποία είναι πλέον ελεύθερη (δεν χρειάζεται άδεια λειτουργίας) στα περισσότερα κράτη του κόσμου (σε ολόκληρη την Ευρωπαϊκή Ένωση με ελάχιστες επιμέρους διαφοροποιήσεις). Αυτή την στιγμή χρησιμοποιούμε δύο «εκδόσεις». Την 802.11b η οποία μας δίνει μάξιμουμ (θεωρητική) ταχύτητα 11Mbps και 3 μη επικαλυπτόμενα κανάλια, και την έκδοση 802.11g η οποία μας δίνει μάξιμουμ (θεωρητική) ταχύτητα 54 Mbps και περισσότερα μη επικαλυπτόμενα κανάλια (το πλήθος διαφέρει ανά υλοποίηση ανά χώρα, βλέπε και (Wikipedia, 2008)). Πολύ σύντομα (καταληκτική ημερομηνία Νοέμβριος 2009) η IEEE θα έχει ολοκληρώσει την προτυποποίηση του επερχόμενου πρωτοκόλλου 802.11n με θεωρητικές ταχύτητες 300-600Mbps (IEEE-802.11, 2008).

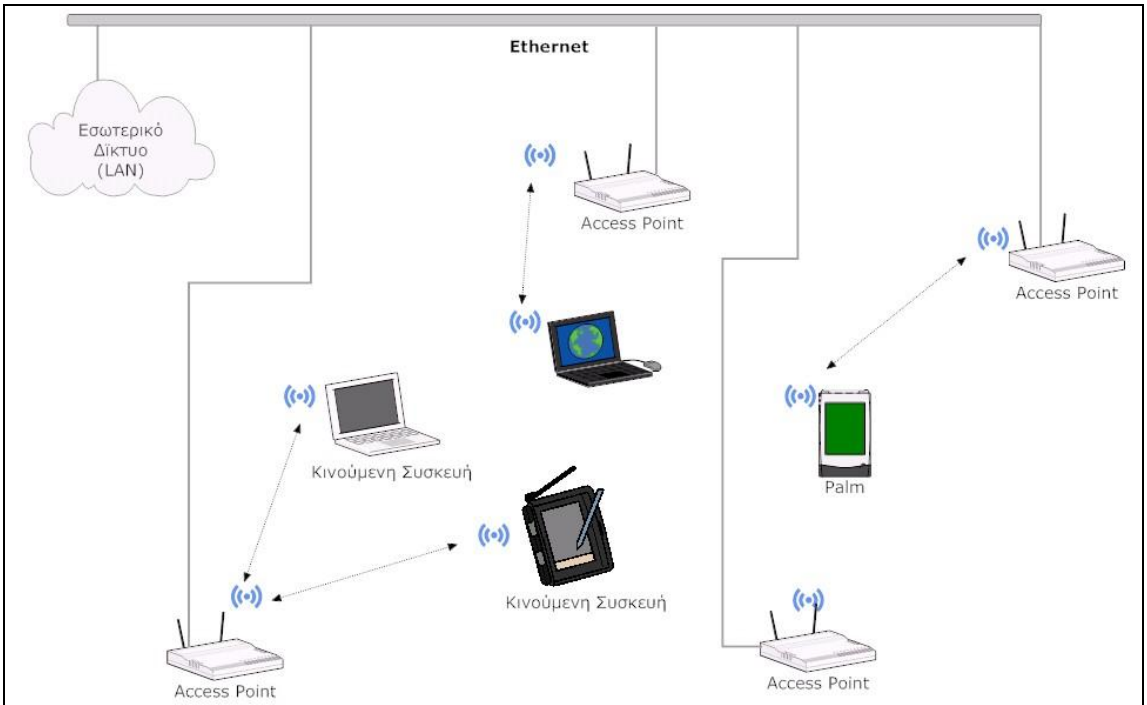
3.1. Συνοπτική περιγραφή Wi-fi



Εικόνα 5: Στάθμες εκπομπής 802.11b

Σε μία τυπική εγκατάσταση ενός τέτοιου (όπου υπάρχει ανάγκη κάλυψης μεγαλύτερου χώρου από τον χώρο κάλυψης μίας συσκευής εκπομπής – Access Point) δικτύου ένας ή περισσότεροι σταθμοί εκπομπής (Access Points) είναι συνδεδεμένοι συνήθως ενσύρματα (ή με ασύρματη σύνδεση με άλλον σταθμό) με το ευρύτερο δίκτυο. Σταθμοί – πελάτες

εφοδιασμένοι με αντίστοιχες κάρτες ασύρματης σύνδεσης Wi-Fi συνδέονται με τα Access Points και μέσω αυτών με το υπόλοιπο δίκτυο. Η ακεραιότητα των δεδομένων και η ασφάλεια και η πιστοποίηση των εγκεκριμένων χρηστών γίνεται μέσω πρωτοκόλλων ασφαλείας, συνθηματικών εισόδου κτλ.



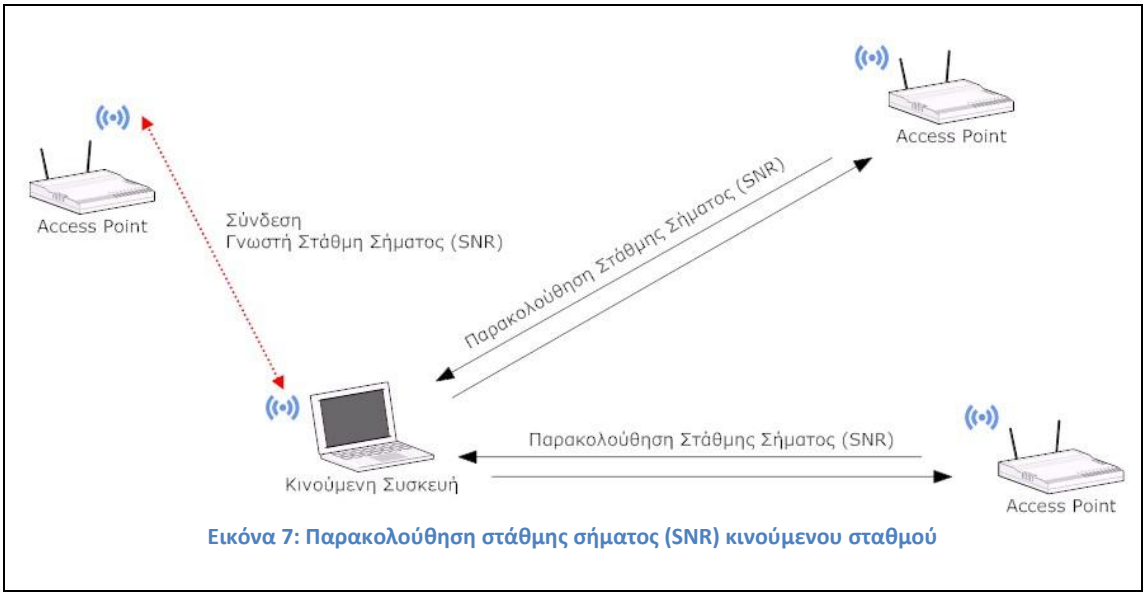
Εικόνα 6: Τυπική εγκατάσταση WLAN (Ασύρματου Δικτύου)

Όπως φαίνεται και στην Εικόνα 6, ικανός αριθμός σταθμών εκπομπής -στρατηγικά τοποθετημένοι- αναλαμβάνουν να παρέχουν ασύρματη κάλυψη σε έναν δεδομένο χώρο. Μέσα σε αυτόν το χώρο κινούνται συσκευές ασύρματης δικτύωσης (φορητοί υπολογιστές – laptops, τηλέφωνα (smart phones) ή άλλες συσκευές (PDA's) εξοπλισμένες πάντα με δυνατότητα ασύρματης δικτύωσης).

3.2. Στάθμη σήματος Wi-Fi

Τα δίκτυα όπως αυτό στην Εικόνα 6, έχουν μία ιδιαιτερότητα: Κάθε δεδομένη χρονική στιγμή, ο κάθε σταθμός-πελάτης είναι συνδεδεμένος με έναν σταθμό εκπομπής (Access Point). Η ασύρματη φύση του δικτύου όμως, προστάζει και όλους τους υπόλοιπους σταθμούς του δικτύου που βρίσκονται εντός εμβέλειας του σταθμού πελάτη, να παρακολουθούν (monitor) την στάθμη του σήματος εκπομπής/λήψης (SNR, βλέπε

κεφάλαιο 4.1) με το οποίο θα μπορούσαν να συνδεθούν με αυτόν το σταθμό. Ομοίως παρακολουθεί την στάθμη αυτού του σήματος και ο σταθμός –πελάτης. Όταν αυτή η στάθμη σήματος «πέσει» κάτω από κάποιο κατώφλι, και με την προϋπόθεση ότι υπάρχει άλλος ενεργός σταθμός εκπομπής στην ευρύτερη περιοχή, ο σταθμός-πελάτης αποσυνδέεται από τον σταθμό-εκπομπής που δεν μπορεί να τον εξυπηρετήσει πλέον, και αποφασίζει να συνδεθεί (εάν έχει ευχέρεια επιλογών) με τον σταθμό που του παρέχει το υψηλότερο σήμα σύνδεσης, την δεδομένη χρονική στιγμή. Είναι εύκολο δε να αποφασίσει για το «καλύτερο» εκείνη την στιγμή σήμα (υψηλότερο σε στάθμη), γιατί παρακολουθούν την στάθμη του σήματος με όλους τους δυνητικούς πελάτες[4] και το διαφημίζουν/παρέχουν ως πληροφορία στους «ενδιαφερόμενους» πελάτες (Kurose & Ross, 2005), (Vaughan-Nichols, 2003).



Εικόνα 7: Παρακολούθηση στάθμης σήματος (SNR) κινούμενου σταθμού

3.3. WMI, NDIS Microsoft Libraries

Ένα από τα βασικότερα θέματα προβλήματα της παρούσης εργασίας, είναι η ανεύρεση και διάθεση της πληροφορίας της στάθμης του σήματος (SNR) στην κεραία ενός σταθμού – πελάτη, από όλα τα access point που βρίσκονται γύρω του.

[4] Στα επόμενα κεφάλαια θα δούμε ότι αυτό δεν ισχύει πάντα. Υπάρχουν σταθμοί εκπομπής (access points) που δεν αποθηκεύουν αυτήν την πληροφορία, για αυτό αναγκαστήκαμε να στραφούμε στην λύση πρόσληψης της πληροφορίας αυτής, κατευθείαν από τον κινούμενο σταθμό - πελάτη.

Παρόλη την βιβλιογραφία προς αυτήν την κατεύθυνση, βρέθηκαν access point (Orinoco AP-2000) τα οποία δεν αποθηκεύουν πουθενά αυτήν την πληροφορία[5]. Έτσι αφού αυτά ήταν τα μηχανήματα που είχαμε στην διάθεση μας, αναγκαστήκαμε να στραφούμε σε άλλες λύσεις.

Η λύση ήταν να αντλήσουμε αυτήν την πληροφορία από τον ίδιο τον σταθμό – πελάτη. Δηλαδή ενώ ο σταθμός– πελάτης είναι συνδεδεμένος με ένα συγκεκριμένο σταθμό εκπομπής, ταυτόχρονα «σκανάρει» στις υπόλοιπες συχνότητες, ανιχνεύοντας την στάθμη του σήματος όλων των υπόλοιπων σταθμών εκπομπής στην περιοχή.

Για να μπορέσουμε να αντλήσουμε την παραπάνω πληροφορία πρέπει ο κώδικας της εφαρμογής να «κατέβει» σε επίπεδο κάρτας δικτύου (Στο επίπεδο 2, σύμφωνα με το πρότυπο OSI). Το συγκεκριμένο επίπεδο εξαρτάται από το λειτουργικό σύστημα στο οποίο λειτουργεί η εφαρμογή. Εφόσον οι σταθμοί πελάτες, είχαν λειτουργικό σύστημα Windows, αναζητήσαμε την πληροφορία στην Microsoft.

3.4. Βιβλιοθήκη WMI

Η Microsoft από τα Window XP Service Pack 2 και μετά, διαθέτει για τέτοιους σκοπούς την βιβλιοθήκη WMI (Microsoft, 2008).(Microsoft, 2008). Η βιβλιοθήκη αυτή μπορεί να δώσει - με την χρήση εντολών που μοιάζουν με SQL εντολές- πληροφορίες που αφορούν τις κάρτες δικτύου και φυσικά για ασύρματες κάρτες μπορεί να δώσει και όλη την σχετική πληροφορία που αφορά στο επίπεδο ισχύος του σήματος. Δυστυχώς όμως όπως ανακαλύψαμε, η μόνη πληροφορία που δεν μπορεί να δώσει η WMI είναι η ζητούμενη. Δίνει μόνο την στάθμη του σήματος του συνδεδεμένου - με τον σταθμό υπό έλεγχο – σταθμού εκπομπής, αλλά δεν δίνει την στάθμη του σήματος των υπολοίπων σταθμών. Μάλιστα σχετική έρευνα στο διαδίκτυο αποκάλυψε ότι είναι ένα από τα bug της WMI (βλέπε για παράδειγμα (Microsoft MSDN Forums, 2008), (Microsoft MSDN,

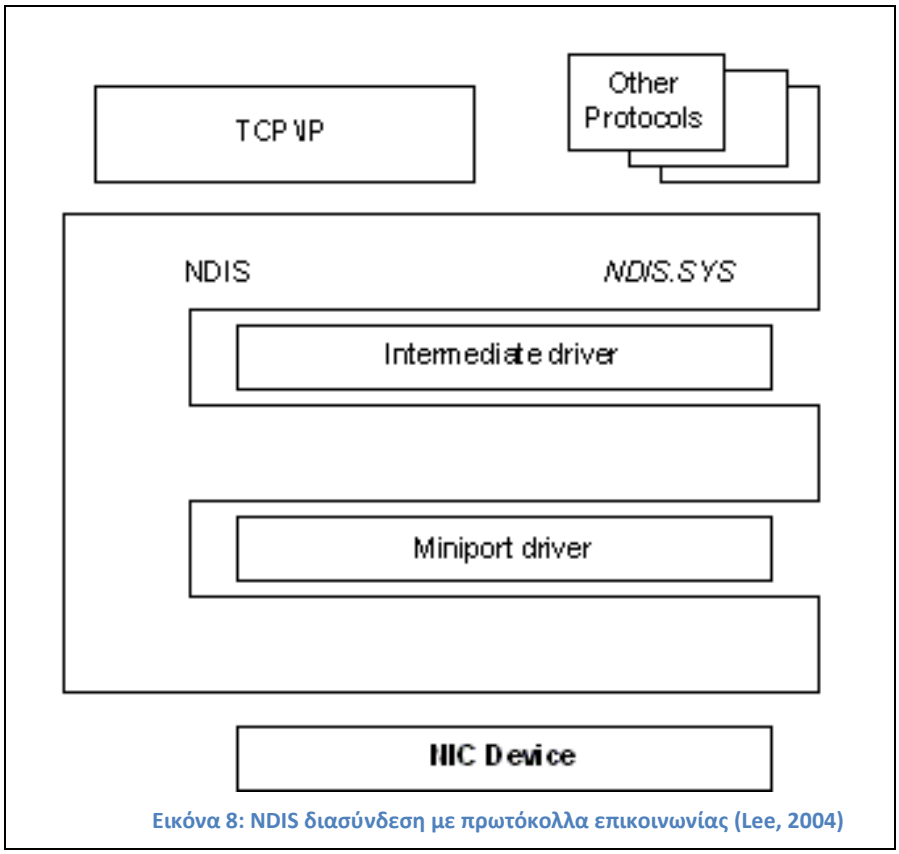
[5] Σχετικά με αυτά τα access-point (Orinoco AP-2000) ανταλλάχτηκε σχετική αλληλογραφία με την κατασκευάστρια εταιρία Proxim. Το ζητούμενο ήταν να μας καθοδηγήσει η Proxim, σε ποιο σημείο – εντολή του σχετικού MIB, μπορούσε να ανευρεθεί η πληροφορία της Στάθμης Σήματος σταθμών – πελατών Wi-Fi στην περιοχή του access point που δεν είναι συνδεδεμένοι σε αυτόν το σταθμό εκπομπής.
Η απάντηση του *Oscar Ubierna* (*Systems Engineer Southern Europe PROXIM Wireless*) ήταν επί λέξει:
*"...There is no information in our MIBs about the clients that are **not** connected to the AP..."*

2008))(Microsoft MSDN Forums, 2008), (Microsoft MSDN, 2008)) το οποίο διορθώνεται μόνο στα Windows Vista.

3.5. Βιβλιοθήκη Native Wi-Fi

Η βιβλιοθήκη Native Wi-Fi αποτελεί πρόγονο της WMI και είναι η βιβλιοθήκη που κατά κόρον χρησιμοποιείται από δικτυακές εφαρμογές για τα MS Windows XP. Δυστυχώς όμως ακόμη και στις πιο βελτιωμένες εκδόσεις της, ακόμη και αυτής των MS Windows SP3, το αποτέλεσμα ήταν ένα ζεύγος από στάθμη σήματος (RSSI[6]) & όνομα δικτύου (SSID) από όλα τα AP's της περιοχής λήψης. Αυτή η πληροφορία είναι άχρηστη γιατί ενώ έχουμε το επίπεδο του λαμβανόμενου σήματος (RSSI), το όνομα δικτύου (SSID) δεν είναι μοναδικό με αποτέλεσμα να μην μπορούμε να ξέρουμε ποιο AP έχει το αντίστοιχο RSSI. Μάλιστα στο δίκτυο που δουλεύαμε, το SSID ήταν το ίδιο σε όλους τους σταθμούς, έτσι ώστε οι σταθμοί πελάτες, να κάνουν συνεχώς διαμεταγωγή (roaming). Άρα αυτή η πληροφορία ήταν μη αξιοποιήσιμη.

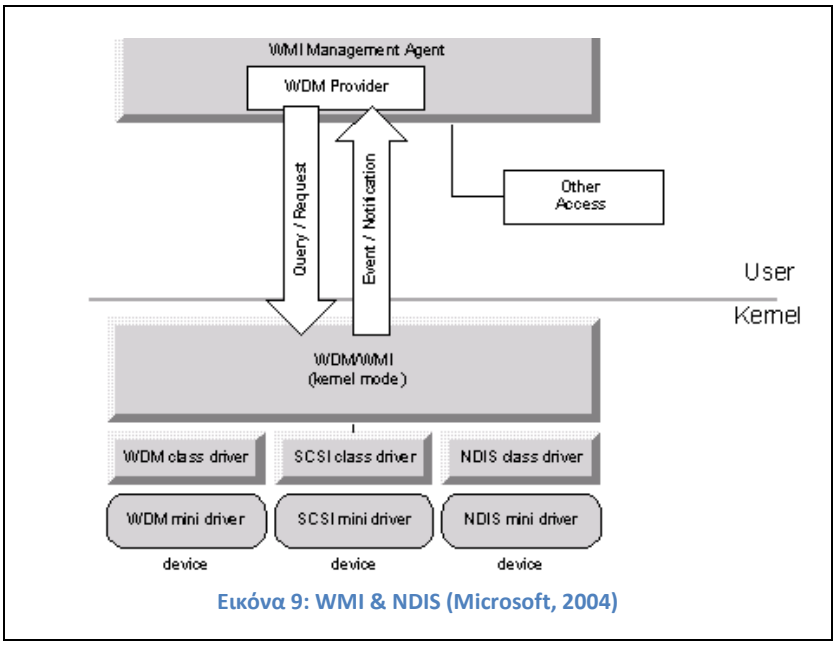
[6] Received Signal Strength Indicator



Εικόνα 8: NDIS διασύνδεση με πρωτόκολλα επικοινωνίας (Lee, 2004)

3.6. Βιβλιοθήκη NDIS

Έτσι στραφήκαμε ακόμα ποιο χαμηλότερα στο λειτουργικό σύστημα στην βιβλιοθήκη NDIS (NDIS, 2007).



Εικόνα 9: WMI & NDIS (Microsoft, 2004)

Η NDIS είναι ένα API (application programming interface) που απευθύνεται σε κάρτες δικτύου (NIC). Ανήκει στο 2^{o} επίπεδο του OSI, και λειτουργεί ως ενδιάμεσος μεταξύ των επιπέδων 2 & 3 του OSI. Είναι ουσιαστικά μία βιβλιοθήκη που ενθυλακώνει τις λειτουργίες των καρτών δικτύου, και ένα κομμάτι της το (NDISUIO) είναι αυτό που ανήκει στον οδηγό (Driver) των Windows για τις ασύρματες κάρτες δικτύου. (Wireless Zero Configuration). Με την χρήση του NDIS μπορέσαμε να ανακτήσουμε την πληροφορία της MAC Address (που είναι μοναδική) των AP's και σε συνδυασμό με την Native Wi-Fi να αντιστοιχήσουμε τις MAC Address με RSSI. Έτσι μπορέσαμε και ανακαλύψαμε την αναζητούμενη πληροφορία.

Η εμπλοκή δύο διαφορετικών βιβλιοθηκών, από δύο διαφορετικά επίπεδα, δημιούργησε την ανάγκη δημιουργίας μίας εφαρμογής, η οποία θα λειτουργεί ως ενδιάμεσος ανάμεσα στην ασύρματη κάρτα δικτύου ενός σταθμού πελάτη, και της εφαρμογής που συλλέγει την υπό έρευνα πληροφορία. Ουσιαστικά η εφαρμογή συνδυάζει την αντλούμενη από τη Native Wi-Fi - WMI πληροφορία της στάθμης του σήματος των γειτονικών σταθμών, με την MAC Address των σταθμών αυτών, πληροφορία που «έρχεται» από την NDIS. Τα επίπεδα λειτουργίας των δύο βιβλιοθηκών φαίνονται στις εικόνες Εικόνα 8, Εικόνα 9, ενώ στην τελευταία φαίνεται και η διασύνδεση της με την NDIS.

Η κατασκευή της εφαρμογής αποδείχθηκε ιδιαίτερα δύσκολη υπόθεση, αφού όπως προαναφέρθηκε , η ζητούμενη πληροφορία είναι κρυμμένη σε πολύ χαμηλά επίπεδα του οδηγού (driver) της κάθε κάρτας δικτύου. Ευτυχώς όμως η συγκεκριμένη πληροφορία

περιλαμβάνεται σε όλους τους οδηγούς καρτών δικτύου, έτσι η εφαρμογή δεν δεσμεύεται από συγκεκριμένους κατασκευαστές.

3.7. Περιγραφή ιδιοτήτων RSSI[7]

Όπως σαφώς αναφέρεται στην περιγραφή του πρωτοκόλλου 802.11 (IEEE, 2007), *«… το RSSI είναι ένα προαιρετικό μέγεθος, που κυμαίνεται από 0 έως το max, δεν καθορίζεται η ακρίβεια του, και έχει νόημα ως σχετικό νούμερο (SNR)…»*. Στην σελίδα 600 του (IEEE, 2007), καθορίζεται το μέγεθος αποθήκευσης της πληροφορίας RSSI σε στάθμη εύρους από 0-255 (δηλαδή ένα Byte). Σύμφωνα δε με την Microsoft (MSDN, 2008) το RSSI είναι ένας προσημασμένος ακέραιος, ως προαιρετικό πεδίο.

Όντως σε όλες τις περιπτώσεις σταθμών εκπομπής που μπορέσαμε να ελέγξουμε, το RSSI ήταν ένας προσημασμένος (αρνητικός στο συνηθισμένο του εύρος) ακέραιος, που αναλόγως τον κατασκευαστή, έπαιρνε τιμές από -20 έως -95 dBm, περίπου.

Ένα από τα μεγαλύτερα προβλήματα που επιφέρει και το μεγαλύτερο σφάλμα στην εύρεση θέσης, είναι ότι το εύρος τιμών του RSSI μεταβάλλεται πολύ εύκολα, ακόμη και όταν ο σταθμός πελάτης παραμένει σε ακινησία, λόγω του ότι εκφράζεται σε ακέραιο αριθμό. Αυτό επιφέρει συνεχείς μεταβολές στον προσδιορισμό της θέσης του σταθμού πελάτη, και επιβάλλει την χρήση φίλτρων, όπως το Kalman filter (Welch & Bishop, 2006). Το φίλτρο αυτό συγκρίνει την τρέχουσα ταχύτητα και θέση, με τις προηγούμενες και σε περίπτωση μεγάλης διαφοράς, επιφέρει τις απαραίτητες τροποποιήσεις ώστε να ομαλοποιήσει την κίνηση του σταθμού-στόχου.

Όντας ακέραιος αριθμός το RSSI, αποδεικνύεται επίσης ότι επιφέρει μεγάλες αποκλίσεις στο αποτέλεσμα της συνάρτησης που υπολογίζει την απόσταση.

Πίνακας 5: Σχέση SNR & Απόστασης[8]

Fspl (dBm)	Απόσταση (m)
65	17,38
66	19,5
73	43,65
74	48,98
83	138,06

[7] RSSI , Received Signal Strength Indicator: η στάθμη του σήματος μεταξύ δύο σταθμών ασύρματης εκπομπής

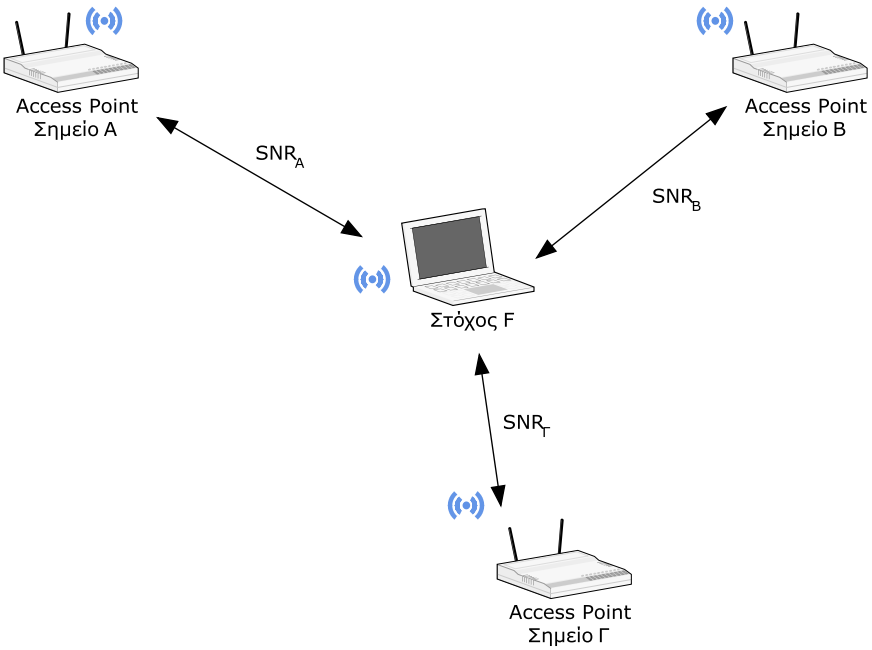
[8] Οι τιμές αυτές βγαίνουν αν αντικαταστήσουμε στην συνάρτηση $fspl = 20log_{10}d + 20log_{10}f + 32{,}44$ το *f*=2440000 Hz, και το *fspl* με τις τιμές του πίνακα.

84	154,9

Εάν στην εξίσωση (fspl=20log(d)+20log(f)-147,56), (4.2 FSPL) εφαρμόσουμε δύο διαδοχικές ακέραιες τιμές (βλέπε Πίνακας 5), βλέπουμε ότι το μικρότερο εύρος που μπορεί να αποτιμήσει η εξίσωση είναι 2 μέτρα. (Επειδή η συνάρτηση περιέχει λογάριθμους, το σφάλμα μεγαλώνει όσο μεγαλώνουν οι τιμές που λαμβάνει η fspl).

4. Εύρεση θέσης με χρήση Στάθμης σήματος

Στο κεφάλαιο αυτό αναπτύσσονται οι απαραίτητες παράμετροι που συνεισφέρουν στον υπολογισμό της θέσης ενός κινούμενου σταθμού - στόχου, σε έναν καλά ορισμένο χώρο (x,y). Για την εύρεση της θέσης του σταθμού, χρειάζεται να βρεθεί η απόσταση του σταθμού αυτού από τουλάχιστον 3 γνωστά σημεία στο χώρο. Οι αποστάσεις αυτές μπορούν να υπολογιστούν χρησιμοποιώντας την στάθμη του σήματος ασύρματης εκπομπής μεταξύ του κινούμενου σταθμού και των συντεταγμένων (x,y) των θέσεων των σταθμών εκπομπής.



Εικόνα 10: 3 Σημεία στον χώρο και σημείο- στόχος F

Η εύρεση της θέσης ενός κινούμενου σταθμού F, του οποίου γνωρίζουμε την στάθμη σήματος από 3 τουλάχιστον γνωστά σημεία στον χώρο (access points), όπως απεικονίζεται στην Εικόνα 10, οδηγεί στην εύρεση 3 εξισώσεων:

$$SNR_A = f(dA), SNR_B = f(dB), SNR_\Gamma = f(d\Gamma)$$

Αυτές οι εξισώσεις αν λυθούν ως προς την απόσταση d(A,B ή Γ), μας δίνουν 3 εξισώσεις για δύο αγνώστους:

$$dA = f(x_F, y_F),\ dB = g(x_F, y_F),\ d\Gamma = h(x_F, y_F)$$

Αυτοί οι δύο άγνωστοι είναι οι συντεταγμένες του $\mathrm{F}\,(\mathrm{x_F}, \mathrm{y_F})$, του σημείου δηλαδή που θέλουμε να προσδιορίσουμε την θέση.

4.1. Σηματοθορυβικός λόγος (Signal to Noise Ratio)

$$\boldsymbol{SNR = 10log_{10}\frac{S}{N}}$$

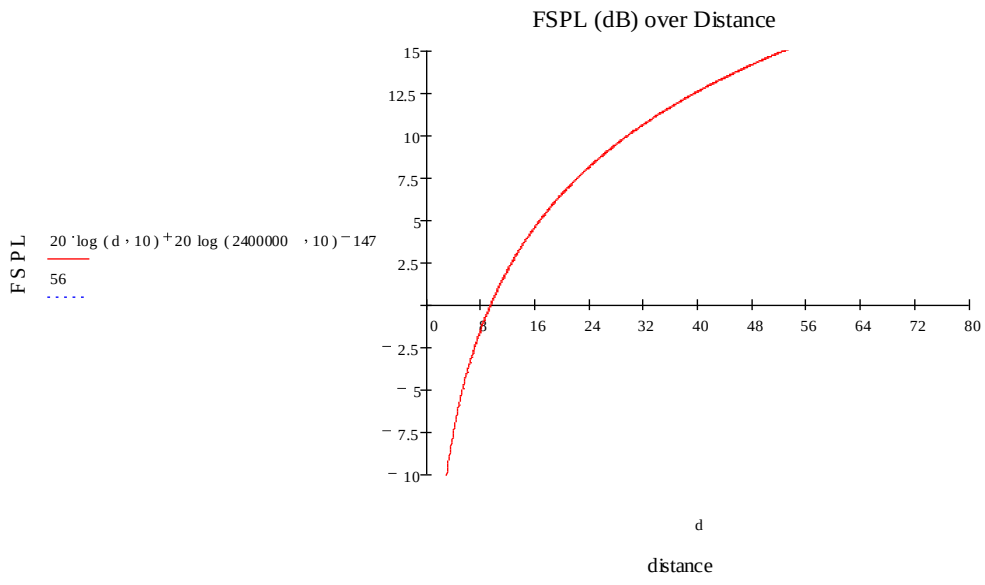
Ο σηματοθορυβικός λόγος (SNR Signal to Noise Ratio) είναι το λογικό γινόμενο της στάθμης του εκπεμπόμενου σήματος δια το ύψος του θορύβου (συνήθως λευκός θόρυβος[9]). Εκφράζεται ως ο δεκαδικός λογάριθμος του λόγου του σήματος προς θόρυβο και «.. *έχει διαφορετικές τιμές στα διάφορα σημεία του τηλεπικοινωνιακού συστήματος*..» (Φούσκας Γεώργιος - Ε.Α.Π., 2002). Εδώ θα γίνεται χρήση στην έξοδο του δέκτη ή του πομπού, εκτός αν ορίζεται διαφορετικά. Όταν ένα σήμα έχει υψηλότερη στάθμη από τον θόρυβο, τότε ο λόγος αυτός είναι θετικός. Όταν το σήμα έχει χαμηλότερη στάθμη από τον θόρυβο (πνίγεται στον θόρυβο) τότε ο λόγος αυτός είναι αρνητικός, και φυσικά δεν υπάρχει αξιοποιήσιμη πληροφορία. Το SNR μετράται πάντα σε dB.

4.2. Απώλεια σήματος σε σχέση με την απόσταση (Free Space Path Loss)

Ως απώλεια σήματος σε σχέση με την απόσταση ορίζεται ως η απώλεια ισχύος ενός ηλεκτρομαγνητικού σήματος το οποίο ταξιδεύει στην νοητή γραμμή του ορίζοντα, σε ανοιχτό χώρο χωρίς εμπόδια (χωρίς ανακλάσεις ή αντανακλάσεις). Δεν λαμβάνονται υπόψη παράγοντες όπως η απολαβή των κεραιών, είτε οι απώλειες λόγω παραγόντων όπως οι ανοχές καλωδίων κτλ. Η απώλεια αυτή είναι ανάλογη του τετραγώνου της απόστασης

9 Φυσικά στον πραγματικό κόσμο, ο θόρυβος είναι ένα από τα μεγαλύτερα προβλήματα σε οποιαδήποτε εγκατάσταση ασύρματης επικοινωνίας. Ο θόρυβος μπορεί να προέρχεται είτε από φυσικές πηγές όπως το ηλεκτρομαγνητικό πεδίο της Γης, η ηλεκτρομαγνητική ακτινοβολία του ήλιου, αλλά και από ανθρώπινες δραστηριότητες ή/και παρεμβάσεις, όπως φούρνοι μικροκυμάτων, εκπομπές αντίστοιχων κοντινών εγκαταστάσεων, κεραίες αναμετάδοσης (links) αλλά και μεταλλικών κατασκευών, κατασκευών που περιέχουν μόλυβδο κτλ.

μεταξύ πομπού και δέκτη, και ανάλογη του τετραγώνου της συχνότητας εκπομπής. Η εξίσωση είναι ακριβής μόνο στο μεσοδιάστημα μεταξύ πομπού και δέκτη (Rappaport, 1996), (Wikipedia, 2008).



Εικόνα 11: Απώλεια Σήματος σε σχέση με την απόσταση

$$\boldsymbol{fspl(dB) = 4\pi d\lambda^2 = 10log_{10}4\pi dfc^2 = 20log_{10}d + 20log_{10}f - 147,56}$$

- d = distance – απόσταση (σε μέτρα)
- λ = μήκος κύματος (σε μέτρα)
- f = frequency – συχνότητα (σε Hertz)[10]
- C = Speed of light – Ταχύτητα του φωτός = 2,99792458 * 108 m/s (μέτρα / δευτερόλεπτο)

Εάν μετρηθεί η συχνότητα f σε MHz και η απόσταση d σε χιλιόμετρα τότε η εξίσωση γίνεται ισοδύναμα:

$$\boldsymbol{fspl(dB) = 20log_{10}d + 20log_{10}f + 32,44}$$

[10] Εδώ πρέπει να σημειωθεί ότι η τιμή της *f* (Συχνότητα) ΔΕΝ είναι σταθερή. Ένα συχνό λάθος που παρατηρήθηκε στην υπάρχουσα βιβλιογραφία είναι ότι η συχνότητα θεωρείται σταθερή και ίση με 2.400GHz. Αυτό είναι ένα συχνό λάθος που συμβαίνει από την θεώρηση των δικτύων 802.11x ως δικτύων στην συχνότητα των 2,4 όπως καταχρηστικά αναφέρουμε . Στην πραγματικότητα η συχνότητα είναι μεταβλητό μέγεθος και κυμαίνεται μεταξύ 2400GHz και συνήθως μέχρι τα 2483GHz αναλόγως την χώρα υλοποίησης. Φυσικά οι διαφορές είναι μικρές αλλά δεν υπάρχει κανείς λόγος η ακριβής στάθμη της συχνότητα να ΜΗΝ υπεισέρχεται ως παράμετρος.

Εάν είναι γνωστή η στάθμη του σήματος, τότε η συνάρτηση μπορεί να λυθεί ως προς την απόσταση d:

$$d = 10^{fspl - 20log_{10}f + 147{,}56 / 20}$$

4.3. Όρια χρήσης στάθμης σήματος

Η απόσταση του κινούμενου σταθμού στόχου από έναν σταθμό εκπομπής, είναι συνάρτηση της στάθμης του σήματος σε σχέση με την απόσταση.

Σύμφωνα και με πρακτικές παρατηρήσεις η στάθμη του σήματος (SNR) μεταξύ σταθμού εκπομπής και σταθμού-πελάτη, έχει πρακτική χρησιμότητα μόνο στο μεσοδιάστημα της απόστασης. Αν η συνάρτηση χρησιμοποιηθεί περά από κάποια όρια, τότε θα επιφέρει μεγάλο σφάλμα στην εύρεση της θέσης. Τα πρακτικά αυτά όρια είναι γνωστά από τις προδιαγραφές των πρωτοκόλλων 802.11 (Wi-Fi Alliance, 2007) σε συνδυασμό με την συνάρτηση FSPL(βλέπε 4.2), και θεωρητικά οριοθετούνται από τα 10 μέτρα ως κάτω όριο και από τα 80 μέτρα περίπου, ως πάνω όριο. Επειδή όμως αυτές οι τιμές διαφοροποιούνται σημαντικά λόγω της προσθήκης των καλωδίων, των κεραιών εκπομπής κτλ (βλέπε (Proxim, 2008)),(Proxim, 2008)), είναι καλύτερα να συνδέονται αυτά τα όρια με την ισχύ εκπομπής/λήψης (dB). Πρακτικά αυτά τα όρια τέθηκαν χάριν της εργασίας σε -40 dBm, και -90dBm. Άλλωστε σε ένα τέτοιο χώρο όπου υπάρχουν αρκετοί σταθμοί εκπομπής, είναι εφικτό (και ζητούμενο) να μην χρησιμοποιηθούν οι μετρήσεις από όλους τους σταθμούς που τυχόν διατηρούν αξιοποιήσιμη πληροφορία, και για λόγους ταχύτητας των υπολογισμών, αλλά και γιατί από ένα σημείο και πέρα τα επιπλέον δείγματα δεν επιφέρουν σημαντική μεταβολή στην ακρίβεια των υπολογισμών (Krumm & Platt, 2003).

4.4. SNR & RSSI (dB vs dBm)

Το SNR περιγράφεται στο 4.1. ($SNR = 10log_{10}\frac{S}{N}$). Τα S, N είναι μονάδες ισχύος μετρήσιμες σε Watt, και το αποτέλεσμα που προκύπτει είναι ένα αριθμός (SNR) (CVEL, 2008).

Γενικώς το dB ως μονάδα μέτρησης προσλαμβανόμενης ισχύος σήματος χρησιμοποιείται από την αγορά, και η τεκμηρίωση του έγινε αργότερα (Consultative Committee for Units (CCU), 2003).(Consultative Committee for Units (CCU), 2003).

Εάν θέλουμε να δηλώσουμε την απόλυτη ισχύ εκπομπής ενός σήματος, χρησιμοποιούμε τον ίδιο τύπο, βασισμένοι όμως σε γνωστό σημείο αναφοράς. Αυτό το σημείο είναι η ισχύς του 1mW (WildPackets Inc, 2002).

Άρα ο τύπος γίνεται $SNR = 10log_{10}\frac{P_1}{P_2}$, όπου ως P_1 ορίζεται η λαμβανόμενη ισχύς , και ως P_2 ορίζεται η εκπεμπόμενη ισχύς, δηλαδή $P_2 = 1mW$. Σε αυτή την περίπτωση, ο τύπος πλέον παράγει dBm, δηλαδή $SNR(dBm) = 10log_{10}\frac{P_1}{1mW}$.

Όλες οι τιμές που εξάγουν οι ασύρματες κάρτες δικτύου που είχαμε στην διάθεση μας μετρούν την ισχύ σε απόλυτη τιμή, δηλαδή $RSSI\ (dBm) = 10log_{10}\frac{S}{P}$, όπου P είναι η βάση του 1mW (Proxim Wireless, 2008).

Η μετατροπή του RSSI σε SNR γίνεται αν θεωρήσουμε ως σταθερή την ισχύ εκπομπής ενός σταθμού (access point). Αυτή η ισχύς είναι γνωστή από το πρωτόκολλο 802.11 (IEEE-802.11, 2008), και έχει ανώτατη τιμή τα 20mW.

Άρα εάν είναι γνωστή η προσλαμβανόμενη ισχύς σε ένα σύστημα, τότε

$$SNR\ (dB) = 10log_{10}\frac{P}{20mW}$$

Όπου P, είναι η προσλαμβανόμενη ισχύς.

Σύμφωνα με την (Proxim Wireless, 2008), μπορούμε στον υπολογισμό της προσλαμβανόμενης ισχύος να συνυπολογίσουμε και τις παραμέτρους του δικτύου, δηλαδή

$$Rx = T_x - T_xΑπώλεια\ καλωδίου + T_xΚέρδος\ Κεραίας\ (Gain) - fspl + R_xΚέρδος\ κεραίας(Gain) - R_xΑπώλεια\ καλωδίου$$

Όπου

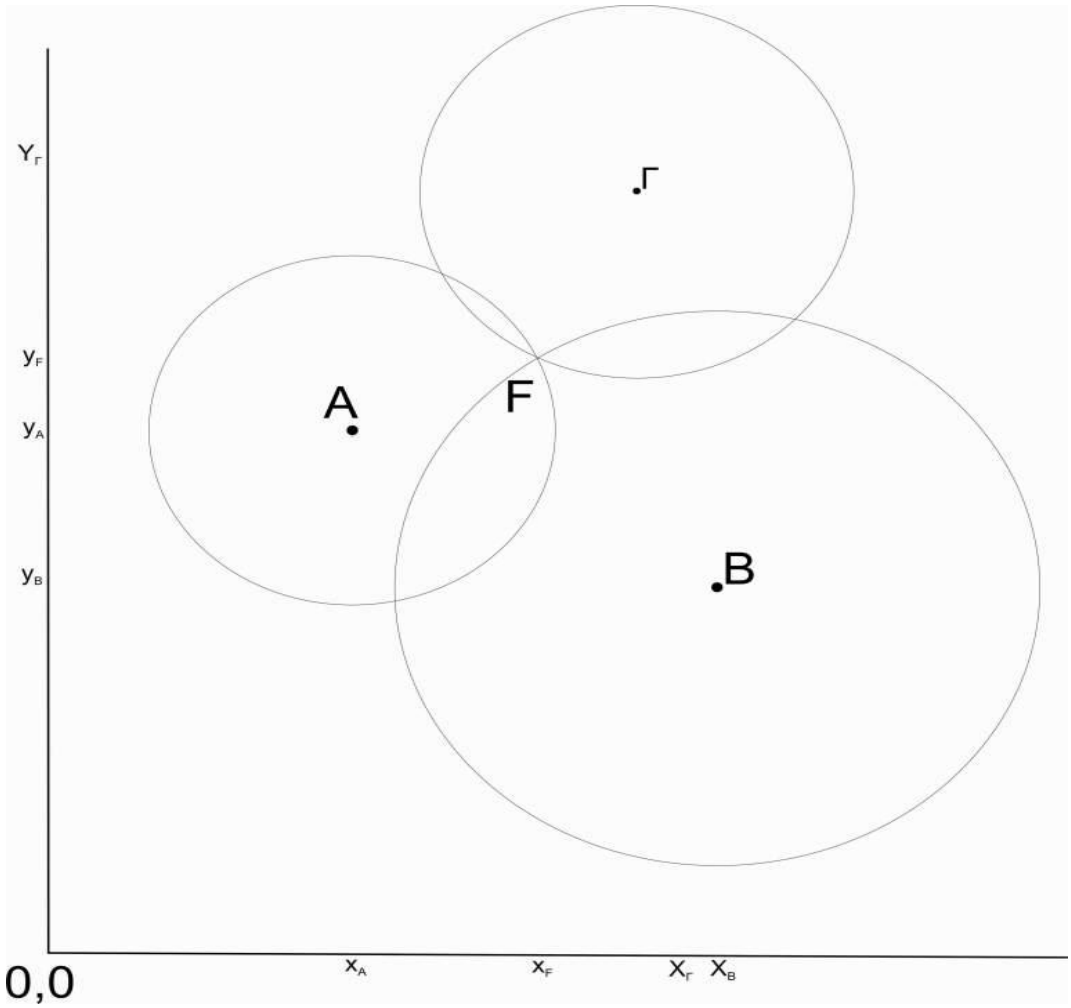
$$Rx = Προσλαμβανόμενη\ Ισχύς$$

$$T_x = εκπεμπόμενη\ ισχύς$$

Επειδή πολλές από αυτές τις παραμέτρους δεν μπορούν να μετρηθούν με ακρίβεια, αποδείχθηκε αναγκαία και η είσοδος μίας μεταβλητής S στον τύπο (βλέπε κεφάλαιο 6.3.3) η οποία μπορεί να διορθώσει κατά πολύ την αποτελεσματικότητα του.

$$Rx = T_x - T_xCable + T_xGain - fspl + R_xGain - R_xCable + S$$

4.5. Τριγωνοποίηση (Triangulation)



Εικόνα 12: Εύρεση θέσης Σημείου F

Έστω στην Εικόνα 12, ότι είναι γνωστές οι θέσεις των Α,Β,Γ (τριών σταθμών εκπομπής σήματος Wi-Fi). Σε ένα προδιαγεγραμμένο σύστημα αξόνων (x,y), είναι γνωστές (και σταθερές) οι συντεταγμένες των 3 σημείων (βλέπε Πίνακας 6). Επίσης γνωστές θεωρούνται

οι αποστάσεις του σημείου F (dA,dB,dΓ αντίστοιχα) από τα 3 προαναφερθέντα σημεία, αφού έχουν ήδη βρεθεί από την συνάρτηση FSPL (βλέπε 4.2).

Πίνακας 6: Συντεταγμένες 3 γνωστών σημείων

	Σημείο Α	**Σημείο Β**	**Σημείο Γ**
	$A(x_A, y_A)$	$B(x_B, y_B)$	$\Gamma(x_\Gamma, y_\Gamma)$
Απόσταση από το F	$dA=AF$	$dB=BF$	$d\Gamma=\Gamma F$

Έστω ότι το σημείο $F(x_F, y_F)$ αντιπροσωπεύει έναν σταθμό-πελάτη που περιφέρεται στον καλά οριοθετημένο χώρο και του οποίου σταθμού-πελάτη θέλουμε να προσδιορίσουμε την θέση. Οι συναρτήσεις που δημιουργούνται, έχουν ως εξής:

$$(\boldsymbol{x_A - x_F})^2 + (\boldsymbol{y_A - y_F})^2 = \boldsymbol{AF^2}$$
$$(\boldsymbol{x_B - x_F})^2 + (\boldsymbol{y_B - y_F})^2 = \boldsymbol{BF^2}$$
$$(\boldsymbol{x_\Gamma - x_F})^2 + (\boldsymbol{y_\Gamma - y_F})^2 = \boldsymbol{\Gamma F^2}$$

Εάν αναπτύξουμε αυτές τις εξισώσεις καταλήγουμε σε 3 δευτεροβάθμιες εξισώσεις:

$$\boldsymbol{x^2 - 2xx_A + x_A^2 + y^2 - 2yy_A + y_A^2 = AF^2}$$
$$\boldsymbol{x^2 - 2xx_B + x_B^2 + y^2 - 2yy_B + y_B^2 = BF^2}$$
$$\boldsymbol{x^2 - 2xx_\Gamma + x_\Gamma^2 + y^2 - 2yy_\Gamma + y_\Gamma^2 = \Gamma F^2}$$

Οι οποίες αν αφαιρεθούν ανά δύο ανά μέλη μας δίνουν δύο πρωτοβάθμιες εξισώσεις:

$$\boldsymbol{{x_A}^2 - {x_B}^2 - (2x_A + 2x_B)x_F + {y_A}^2 - {y_B}^2 - (2y_A - 2y_B)y_F = AF^2 - BF^2}$$
$$\boldsymbol{{x_B}^2 - {x_\Gamma}^2 - (2x_B + 2x_\Gamma)x_F + {y_B}^2 - {y_\Gamma}^2 - (2y_B - 2y_\Gamma)y_F = BF^2 - \Gamma F^2}$$

Για την απλοποίηση των υπολογισμών θεωρούμε τις παρακάτω απλοποιήσεις:

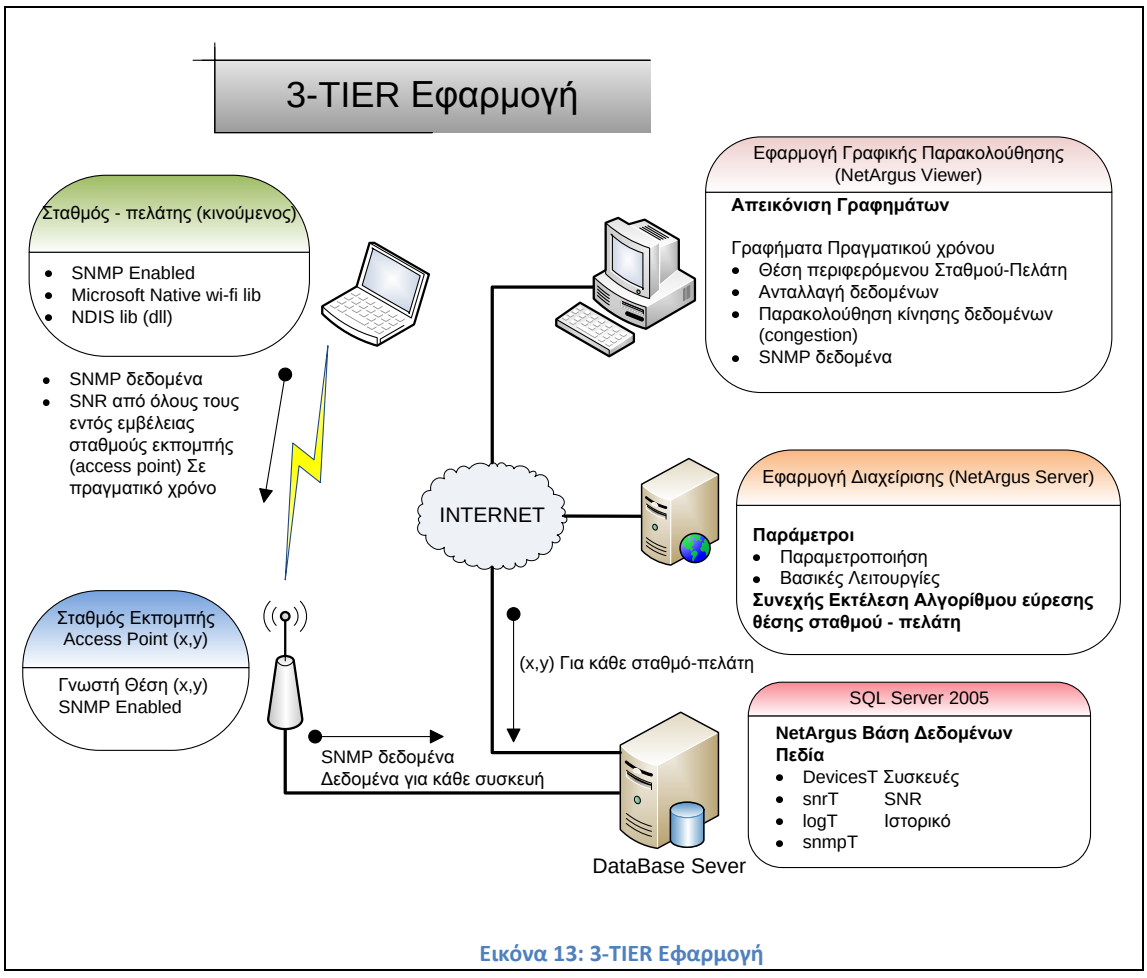
Πίνακας 7: Απλοποιήση Συναρτήσεων Τριγωνοποιήσης

$\boldsymbol{K1 = AF^2 - BF^2 - {x_A}^2 + {x_B}^2 - {y_A}^2 + {y_B}^2}$	
$\boldsymbol{M1 = 2(y_B - y_A)}$	$M2 = 2(x_\Gamma - x_A)$
$\boldsymbol{N1 = 2(x_B - x_A)}$	$N1 = 2(y_\Gamma - y_A)$

$$\boldsymbol{x_F = \frac{K1}{N1} - \frac{M1}{N1}\, y_F,} \qquad y_F = \frac{\frac{K2}{N2} - \frac{M2}{N2}\frac{K1}{N1}}{1 - \frac{M2}{N2}\frac{M1}{N1}}$$

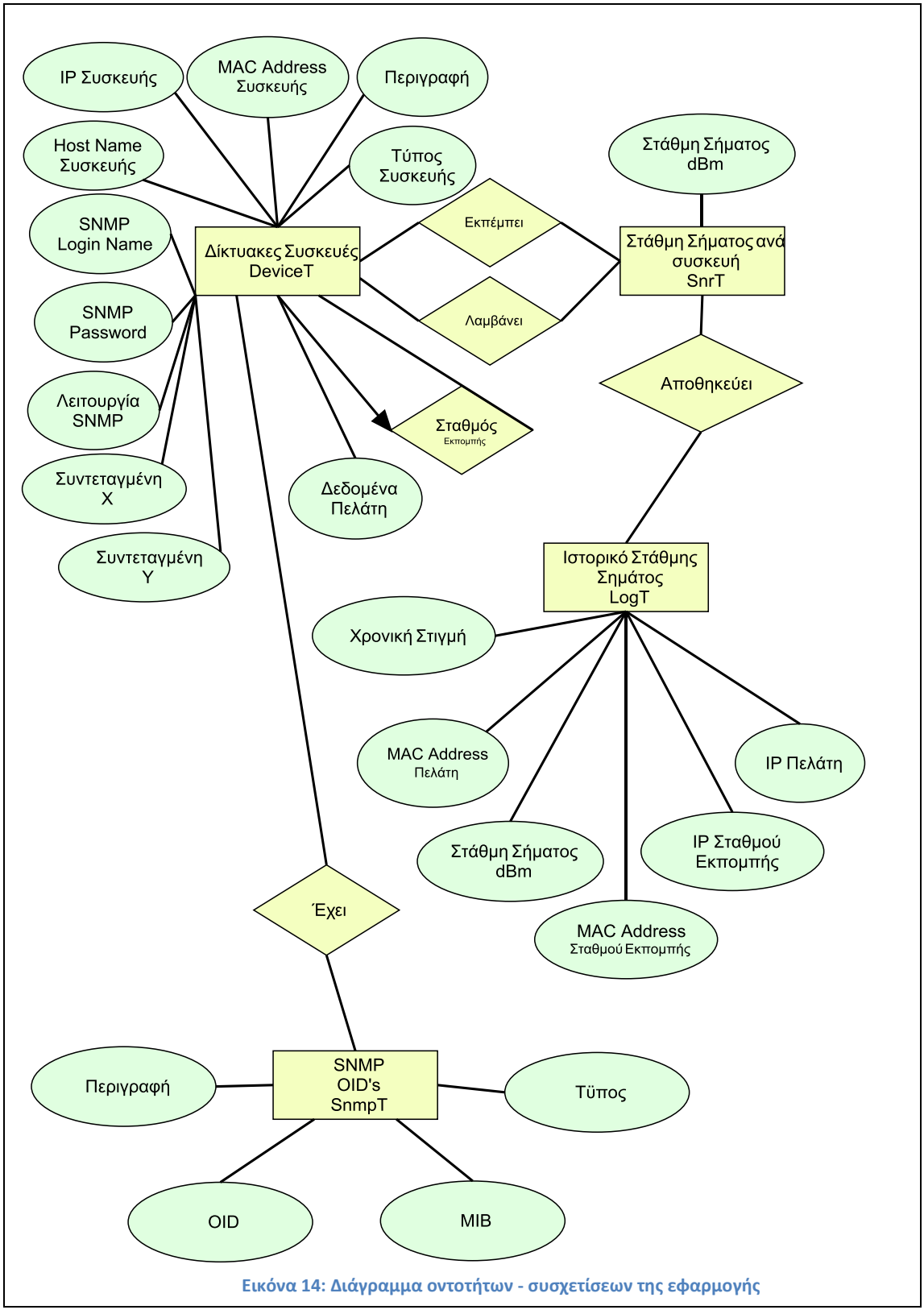
5. Αρχιτεκτονική και Σχεδίαση του Λογισμικού

Το λογισμικό που αναπτύχθηκε στα πλαίσια της εργασίας είναι ένα μία εφαρμογή τριών επιπέδων διασύνδεσης (3-tier application) για λειτουργικό σύστημα Microsoft Windows XP Service Pack 2 και ανώτερο, και δίκτυα TPC/IP. Είναι μία δικτυακή εφαρμογή η οποία αποτελείται από τρία τμήματα τα οποία επικοινωνούν μεταξύ τους με την χρήση μίας κεντρική βάσης δεομένων. Η λειτουργία τους προτείνεται να γίνεται σε ανεξάρτητους ηλεκτρονικούς υπολογιστές οι οποίοι βρίσκονται είτε στο ίδιο τοπικό δίκτυο (ενσύρματο ή ασύρματο) είτε ακόμη και μέσω διαδικτύου με υλοποίηση VPN.



Εικόνα 13: 3-TIER Εφαρμογή

5.1. Διάγραμμα Οντοτήτων και Σχέσεων



Εικόνα 14: Διάγραμμα οντοτήτων - συσχετίσεων της εφαρμογής

5.2. Βάση Δεδομένων

5.2.1. Περιγραφή Πινάκων Βάσης Δεδομένων- Data Dictionary

Η εφαρμογή για την διατήρηση των δεδομένων χρησιμοποιεί Βάση Δεδομένων με όνομα NetArgus (σε DBMS Microsoft SQL Server). Για την ανάπτυξη της κατάλληλης βάσης δεδομένων έγινε ανάλυση και σχεδιασμός της εφαρμογής. Από το παραπάνω διάγραμμα οντοτήτων συσχετίσεων προέκυψαν τέσσερεις πίνακες, δύο βασικοί πίνακες και δύο βοηθητικοί.

Ακολουθεί περιγραφή των πινάκων και των χαρακτηριστικά τους (data dictionary):

- DeviceT

  Ο πίνακας DeviceT χρησιμοποιείται για την καταχώρηση των στοιχείων της κάθε δικτυακής συσκευής που συμμετέχει στην εφαρμογής. Καταχωρεί από βασικά στοιχεία όπως η διεύθυνση IP της συσκευής έως και περιγραφικά στοιχεία που επιθυμεί ο χρήστης για την παρακολούθηση της συσκευής. Αποτελεί τον βασικό πίνακα (master-table).

Πίνακας 8: ΣΥΣΚΕΥΕΣ ΔΙΚΤΥΟΥ - DeviceT

ΟΝΟΜΑΤΑ ΠΕΔΙΩΝ	ΠΕΡΙΓΡΑΦΗ
Κωδικός Συσκευής id	Αύξον αριθμός της δικτυακής συσκευής (κλειδί μοναδικό non null)
IP Address Συσκευής ip	Διεύθυνση IP της συσκευής στο δίκτυο (μοναδικό – non null)
MAC Address Συσκευής mac	Διεύθυνση MAC address της κάρτας δικτύου της συσκευής (αλφαριθμητικό κείμενο)
Host Name Συσκευής Name	Το δικτυακό όνομα που έχει η συσκευή (κείμενο)
Τύπος Συσκευής type	Το είδος της δικτυακής συσκευής. Π.χ. access point, router, pc, laptop, antenna … (λίστα επιλογών)
Περιγραφή Description	Στο πεδίο Περιγραφή μπορεί να χρήστης να εισάγει χαρακτηριστικά της συσκευής που επιθυμεί (ελεύθερο κείμενο)
SNMP snmp	Ακέραιος αριθμός. Το 1 δηλώνει ότι η συσκευή διαθέτει ενεργοποιημένο το πρωτόκολλο SNMP. Το 0 δηλώνει ότι η συσκευή δεν διαθέτει SNMP.
IP Address Σταθμού parent	Διεύθυνση IP της συσκευής (σταθμού εκπομπής) που είναι συνδεδεμένη η συσκευή. Χρησιμοποιείται κυρίως για ασύρματα δίκτυα για να δηλώνει την IP του εξυπηρετητή σταθμού εκπομπής. Αν είναι κενό αυτό το πεδίο δηλώνει ότι η συσκευή δεν είναι κινητή και πρόκειται για access point antenna ή router.
Οριζόντια	Ακέραιος αριθμός που αποθηκεύει την οριζόντια συντεταγμένη της

Συντεταγμένη x	σχετικής θέσης της συσκευής στην γραφική απεικόνιση της σε διαστάσεις 800x600 (τιμές από 0 έως 800)
Κάθετη Συντεταγμένη y	Ακέραιος αριθμός που αποθηκεύει την κάθετη συντεταγμένη της σχετικής θέσης της συσκευής στην γραφική απεικόνιση της σε διαστάσεις 800x600 (τιμές από 0 έως 600)
Όνομα εισόδου SNMP login	Κείμενο – Το όνομα χρήστη που χρησιμοποιείται για την είσοδο στο πρωτόκολλο SNMP της συσκευής
Κωδικός εισόδου SNMP password	Κείμενο – Ο κωδικός χρήστη που χρησιμοποιείται για την είσοδο στο πρωτόκολλο SNMP της συσκευής
Στοιχεία χρήστη κινούμενου σταθμού userdata	Πεδία Κείμενου – Τα πεδία αυτά χρησιμοποιούνται ως παράδειγμα για πιθανά επιπλέων στοιχεία που θα μπορούσε να αποθηκεύει η βάση δεδομένων για την κάθε πελάτη συσκευή (Από εδώ και πέρα ανάλογα με το πεδίο στο οποίο θα λειτουργήσει το λογισμικό μπορούν να γίνουν και επεκτάσεις και να εισαχθούν πεδία που θα εξυπηρετούν την συγκεκριμένη εγκατάσταση).

- SnrT

  Ο πίνακας SnrT χρησιμοποιείται ως detail πίνακας του DeviceT και καταχωρεί την στάθμη εκπομπής από οποιοδήποτε σταθμό σε οποιοδήποτε πελάτη. Είναι στην ουσία ένα καρτεσιανό γινόμενο μεταξύ σταθμών και πελατών. Δημιουργείται δυναμικά την στιγμή που ξεκινάει η εκτέλεση του αλγορίθμου εύρεσης θέσης από την εφαρμογή NetArgus Server και ενημερώνεται από τις εφαρμογές NetArgus Client που εκτελούνται σε κάθε συσκευή πελάτη. Η χρήση του είναι πολύ σημαντική αφού από αυτόν τον πίνακα προκύπτουν οι πληροφορίες για την εκτέλεση του αλγορίθμου εύρεσης θέσης του κινητού σταθμού.

Πίνακας 9: Στάθμη Σήματος - SnrT

ΟΝΟΜΑΤΑ ΠΕΔΙΩΝ	ΠΕΡΙΓΡΑΦΗ
Διεύθυνση IP Σταθμού stationIp	Διεύθυνση IP του σταθμού εκπομπής
Διεύθυνση MAC Σταθμού stationMac	Διεύθυνση MAC της συσκευής εκπομπής
Διεύθυνση IP Πελάτη clientIp	Διεύθυνση IP του πελάτη που λαμβάνει το σήμα από τον σταθμό εκπομπής
Διεύθυνση MAC Πελάτη clientMac	Διεύθυνση MAC της συσκευής του πελάτη που λαμβάνει το σήμα από τον σταθμό εκπομπής
Στάθμη Σήματος snr	Ακέραιος Αριθμός - Η στάθμη του σήματος που λαμβάνει η συσκευή από τον σταθμός εκπομπής

- LogT

Ο πίνακας LogT είναι ένας πίνακας ημερολόγιο (log file) ο οποίος κρατάει στοιχεία για κάθε ενημέρωση (update) που έχει γίνει στον πίνακα SnrT. Η ενημέρωση του γίνεται με την χρήση της τεχνολογίας trigger του MSSQL Server. Σε πραγματικό χρόνο κάθε update στον πίνακα SnrT δημιουργεί αυτόματα ένα insert στον πίνακα LogT. Η χρήση του πίνακα LogT γίνεται για λόγους διατήρησης ιστορικού και δίνει στην εφαρμογή την δυνατότητα να παρουσιάσει σε μεταγενέστερο χρόνο την πορεία που ακολούθησαν οι κινητοί σταθμοί του δικτύου (μέσω προσομοίωσης της πορείας τους).

Πίνακας 10: Ιστορικό - LogT

ΟΝΟΜΑΤΑ ΠΕΔΙΩΝ	ΠΕΡΙΓΡΑΦΗ
Κωδικός Id	Μοναδικός Αύξων αριθμός. Χρησιμοποιείται ως πρωτεύον κλειδί
Χρονική στιγμή καταγραφής timeStamp	Πεδίο Ημερομηνίας – Ώρας. Καταχωρεί την χρονική στιγμή που εισάγετε η εγγραφή
stationIp	Διεύθυνση IP του σταθμού εκπομπής
stationMac	Διεύθυνση MAC της συσκευής εκπομπής
clientIp	Διεύθυνση IP του πελάτη που λαμβάνει το σήμα από τον σταθμό εκπομπής
clientMac	Διεύθυνση MAC της συσκευής του πελάτη που λαμβάνει το σήμα από τον σταθμό εκπομπής
snr	Ακέραιος Αριθμός - Η στάθμη του σήματος που λαμβάνει η συσκευή από τον σταθμός εκπομπής

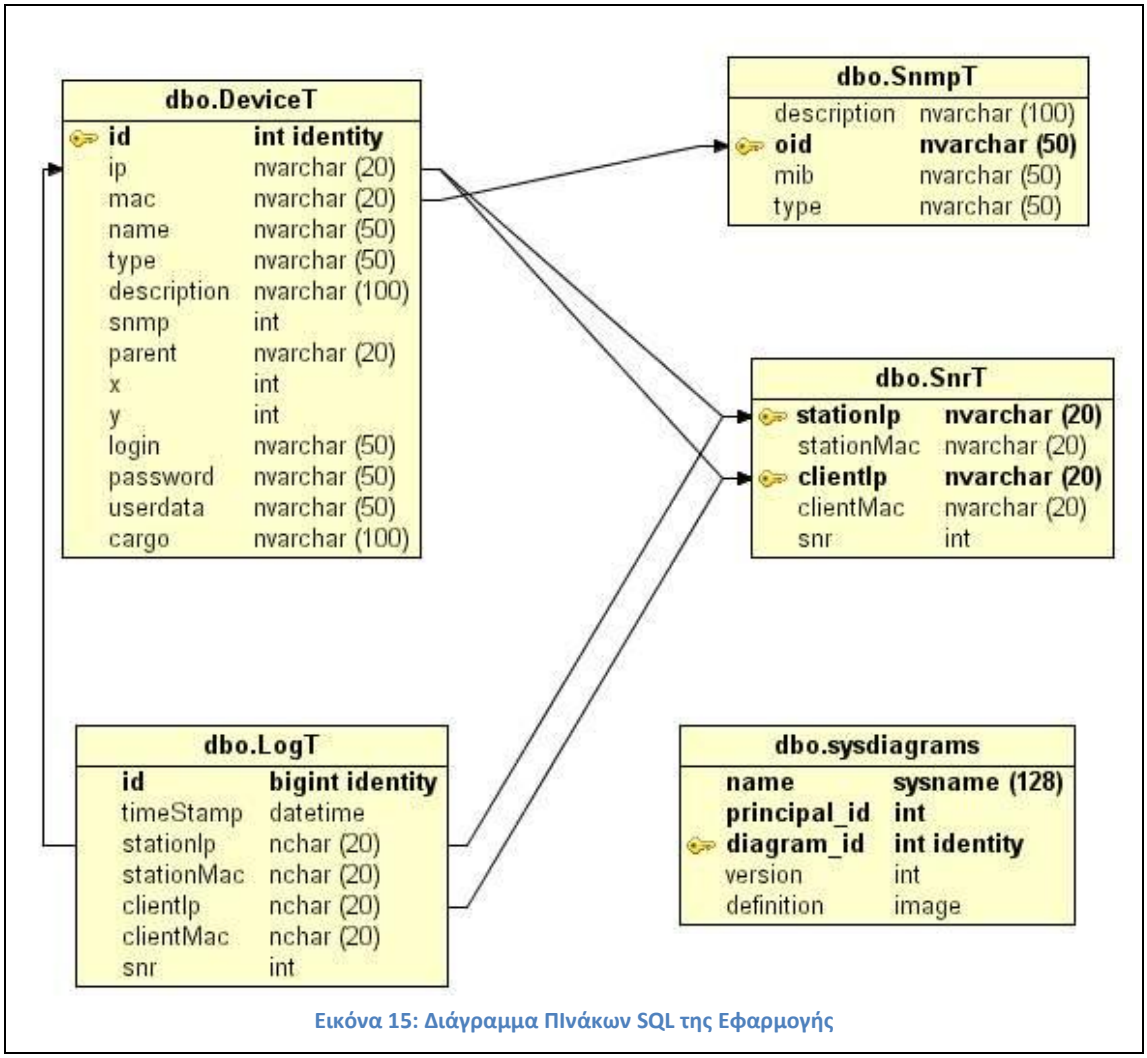
- SnmpT

  Ο πίνακας SnmpT χρησιμοποιείται για λόγους παραμετροποίησης της εφαρμογής, όσον αφορά τις εντολές SNMP που μπορεί να εκτελεί η εφαρμογή. Όπως έχει αναλυθεί σε προηγούμενη ενότητα της εργασίας (βλέπε κεφάλαιο 2.7) οι εντολές SNMP (οι κωδικοί OID's) σε πολλές περιπτώσεις είναι διαφορετικές από συσκευή σε συσκευή (device specific). Ο πίνακας SnmpT δίνει την δυνατότητα στον χρήστη της εφαρμογής να εισάγει τους κωδικούς των εντολών (OID's) της συγκεκριμένης συσκευής που χρησιμοποιεί. Η χρήση του κάνει την εφαρμογή πολύ πιο ευέλικτη και παραμετρική (device independent).

Πίνακας 11: SNMP Εντολές - SnmpT

ΟΝΟΜΑΤΑ ΠΕΔΙΩΝ	ΠΕΡΙΓΡΑΦΗ
Περιγραφή description	Κείμενο που περιγράφει την εντολή SNMP
Εντολή OID oid	Κείμενο με τον αριθμητικό κωδικό OID για την συγκεκριμένη εντολή SNMP
Εντολή MIB mib	Κείμενο με τον λεκτικό κωδικό MIB για την συγκεκριμένη εντολή SNMP
Τύπος Δεδομένων type	Τύπος δεδομένων της τιμής που επιστρέφει η SNMP εντολή

5.2.2. Διάγραμμα Σχέσεων Πινάκων



Εικόνα 15: Διάγραμμα ΠΙνάκων SQL της Εφαρμογής

5.2.3. Παραδείγματα Αποθηκευμένων Δεδομένων

Στις παρακάτω εικόνες εμφανίζονται παραδείγματα από τους πίνακες της εφαρμογής:

DeviceT: Quer...ress.NetArgus)

	id	ip	mac	name	type	description	snmp	parent	x	y	login	password	userdata	cargo
	106	10.17.1.20	00:02:2d:bf:94:2d	SEMPO.thpa.gr	Antenna		1	NULL	25	257	thepa	123	NULL	NULL
	107	10.17.1.22	00:02:2d:0b:b0:3f	linapp5.thpa.gr	Antenna		1	NULL	383	433	thepa	123	NULL	NULL
	108	10.17.1.23	00:02:2d:0b:12:b6	NP103.thpa.gr	Antenna		1	NULL	326	193	thepa	123	NULL	NULL
	109	10.17.1.24	00:02:2d:0b:10:80	SW-KLIMAKIO.t...	Antenna		1	NULL	549	323	thepa	123	NULL	NULL
▶	110	10.17.1.25	00:02:2d:06:f4:7e	SW-PARKING.th...	Antenna		1	NULL	663	432	thepa	123	NULL	NULL
	111	10.17.1.26	00:02:2d:0b:b0:47	aiantas.thpa.gr	Antenna		1	NULL	175	480	thepa	123	NULL	NULL
	112	10.17.1.46	00:02:2d:bf:94:50	sempoups.thpa.gr	Antenna		1	NULL	209	340	thepa	123	NULL	NULL
	113	10.17.1.70	00:02:2d:0b:10:87	AP-DX-20.thpa.gr	Antenna		1	NULL	212	554	thepa	123	NULL	NULL
	114	10.17.1.71	00:02:2d:1c:17:05	AP-BSU-LP42-21...	Antenna		1	NULL	748	200	thepa	123	NULL	NULL
	115	10.17.1.72	00:02:2d:0b:b0:48	AP-L48-22.thpa.gr	Antenna		1	NULL	77	125	thepa	123	NULL	NULL
	116	10.17.1.73	00:02:2d:0b:10:7c	AP-L46-24.thpa.gr	Antenna		1	NULL	558	77	thepa	123	NULL	NULL
	125	10.17.1.75	00:02:12:2b:b7:4e	ffffdd	Laptop	NULL	1	10.17.1.25	490	501	soho	123	JIM	cont1
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Εικόνα 16: Πίνακας Δικτυακών Συσκευών DeviceT

SnrT: Query(sh...ress.NetArgus)

	stationIp	stationMac	clientIp	clientMac	snr
▶	10.17.1.20	00:02:2d:bf:94:2d	10.17.1.75	nac4	0
	10.17.1.22	00:02:2d:0b:b0:3f	10.17.1.75	nac4	0
	10.17.1.23	00:02:2d:0b:12:b6	10.17.1.75	nac4	0
	10.17.1.24	00:02:2d:0b:10:80	10.17.1.75	nac4	0
	10.17.1.25	00:02:2d:06:f4:7e	10.17.1.75	nac4	0
	10.17.1.26	00:02:2d:0b:b0:47	10.17.1.75	nac4	0
	10.17.1.46	00:02:2d:bf:94:50	10.17.1.75	nac4	0
	10.17.1.70	00:02:2d:0b:10:87	10.17.1.75	nac4	0
	10.17.1.71	00:02:2d:1c:17:05	10.17.1.75	nac4	0
	10.17.1.72	00:02:2d:0b:b0:48	10.17.1.75	nac4	0
	10.17.1.73	00:02:2d:0b:10:7c	10.17.1.75	nac4	0
*	NULL	NULL	NULL	NULL	NULL

Εικόνα 17: Πίνακας Στάθμης Σήματος Πελατών SnrT

SnmpT: Query(...ress.NetArgus)

description	oid	mib	type
Station Name	1.3.6.1.2.1.1.5.0	station.name	String
Station MAC	1.3.6.1.2.1.2.2.1.6	station.mac	String
Client MAC	1.3.6.1.4.1.11898.2.1.33.1.1.2.	NULL	String
Client SNR	1.3.6.1.4.1.11898.2.1.33.1.1.20.	NULL	Int
Client IP	1.3.6.1.4.1.11898.2.1.33.1.1.3.	NULL	String
Client Name	1.3.6.1.4.1.11898.2.1.33.1.1.5.	NULL	String
Station Clients N...	1.3.6.1.4.1.11898.2.1.33.3.0	NULL	Int
NULL	NULL	NULL	NULL

Εικόνα 18: Πίνακας Παραμέτρων SNMP OID's SnmpT

LogT: Query(sh...ress.NetArgus) | SnmpT: Query(...ress.NetArgus)

id	timeStamp	stationIp	stationMac	clientIp	clientMac	snr
3126	14/10/08 1:15:13 μμ	10.17.1.23	00:02:2d:0b:12:b6	10.17.1.100		92
3127	14/10/08 1:15:13 μμ	10.17.1.72	00:02:2d:0b:b0:48	10.17.1.100		87
3128	14/10/08 1:15:16 μμ	10.17.1.20	00:02:2d:bf:94:2d	10.17.1.100		77
3129	14/10/08 1:15:16 μμ	10.17.1.23	00:02:2d:0b:12:b6	10.17.1.100		92
3130	14/10/08 1:15:16 μμ	10.17.1.72	00:02:2d:0b:b0:48	10.17.1.100		87
3131	14/10/08 1:15:18 μμ	10.17.1.20	00:02:2d:bf:94:2d	10.17.1.100		78
3132	14/10/08 1:15:18 μμ	10.17.1.23	00:02:2d:0b:12:b6	10.17.1.100		92
3133	14/10/08 1:15:18 μμ	10.17.1.72	00:02:2d:0b:b0:48	10.17.1.100		87
3134	14/10/08 1:15:21 μμ	10.17.1.20	00:02:2d:bf:94:2d	10.17.1.100		79
3135	14/10/08 1:15:21 μμ	10.17.1.23	00:02:2d:0b:12:b6	10.17.1.100		92
3136	14/10/08 1:15:21 μμ	10.17.1.72	00:02:2d:0b:b0:48	10.17.1.100		87
3137	14/10/08 1:15:23 μμ	10.17.1.20	00:02:2d:bf:94:2d	10.17.1.100		80
3138	14/10/08 1:15:23 μμ	10.17.1.23	00:02:2d:0b:12:b6	10.17.1.100		92
3139	14/10/08 1:15:23 μμ	10.17.1.72	00:02:2d:0b:b0:48	10.17.1.100		87
3140	14/10/08 1:15:26 μμ	10.17.1.20	00:02:2d:bf:94:2d	10.17.1.100		81
3141	14/10/08 1:15:26 μμ	10.17.1.23	00:02:2d:0b:12:b6	10.17.1.100		92
3142	14/10/08 1:15:26 μμ	10.17.1.72	00:02:2d:0b:b0:48	10.17.1.100		87
3143	14/10/08 1:15:28 μμ	10.17.1.20	00:02:2d:bf:94:2d	10.17.1.100		80
3144	14/10/08 1:15:28 μμ	10.17.1.23	00:02:2d:0b:12:b6	10.17.1.100		0
3145	14/10/08 1:15:28 μμ	10.17.1.72	00:02:2d:0b:b0:48	10.17.1.100		87
3146	14/10/08 1:15:31 μμ	10.17.1.20	00:02:2d:bf:94:2d	10.17.1.100		80
3147	14/10/08 1:15:31 μμ	10.17.1.23	00:02:2d:0b:12:b6	10.17.1.100		0
3148	14/10/08 1:15:31 μμ	10.17.1.72	00:02:2d:0b:b0:48	10.17.1.100		87
3149	14/10/08 1:15:33 μμ	10.17.1.20	00:02:2d:bf:94:2d	10.17.1.100		79
3150	14/10/08 1:15:33 μμ	10.17.1.23	00:02:2d:0b:12:b6	10.17.1.100		0
3151	14/10/08 1:15:33 μμ	10.17.1.72	00:02:2d:0b:b0:48	10.17.1.100		87
3152	14/10/08 1:15:36 μμ	10.17.1.20	00:02:2d:bf:94:2d	10.17.1.100		76
3153	14/10/08 1:15:36 μμ	10.17.1.23	00:02:2d:0b:12:b6	10.17.1.100		0
3154	14/10/08 1:15:36 μμ	10.17.1.72	00:02:2d:0b:b0:48	10.17.1.100		87

1 of 4848 | Cell is Read Only.

Εικόνα 19: Πίνακας Ιστορικών Στοιχείων Στάθμης Σήματος Πελατών LogT

5.3. Περιβάλλον Ανάπτυξης του Λογισμικού

Για την ανάπτυξη της εφαρμογής έχουν χρησιμοποιηθεί τα παρακάτω εργαλεία:

- Λογισμικό διαχείρισης Βάσεων Δεδομένων Microsoft SQL SERVER 2005
- Γλώσσα Δομημένων Ερωτημάτων SQL
- Γλώσσα Προγραμματισμού Visual C# 2005

5.3.1. Microsoft SQL SERVER 2005

Πρόκειται για το Σύστημα Διαχείρισης Σχεσιακών Βάσεων Δεδομένων (RDBMS) το οποίο αξιοποιείται για την αποθήκευση, ανάκτηση και αναζήτηση των δεδομένων της εφαρμογής.

Ο Microsoft SQL SERVER 2005 (Microsoft, 2008)(Microsoft, 2008) είναι ένα από τα πιο δημοφιλή συστήματα διαχείρισης βάσεων δεδομένων που κυκλοφορούν στην αγορά. Παρέχει μια ασφαλή, αξιόπιστη βάση δεδομένων που μπορεί να εκτελεί τις περισσότερες σημαντικές επιχειρηματικές εφαρμογές και εφαρμογές ανάλυσης. Με τον SQL Server 2005 μπορούμε να διαχειριζόμαστε πολύ μεγάλες ποσότητες δεδομένων με το μεγαλύτερο βαθμό διαθεσιμότητας και ασφάλειας, βελτιώνοντας τις δυνατότητες διεκπεραίωσης των ηλεκτρονικών συναλλαγών και της αποθήκευσης δεδομένων. Ο SQL Server 2005 είναι διαθέσιμος και για μικρές εφαρμογές στις εκδόσεις Standard, Workgroup και Express.

Ο Microsoft SQL SERVER 2005 επιτρέπει τη δημιουργία βάσεων που στηρίζονται στο σχεσιακό μοντέλο (relational database model). Ως ένα σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων (Relational Database Management System, RDBMS), ο Microsoft SQL SERVER 2005 είναι εφοδιασμένος με όλα εκείνα τα χαρακτηριστικά που επιτρέπουν την εύκολη και αποτελεσματική διαχείριση των δεδομένων ενός πληροφοριακού συστήματος. Αυτά τα δεδομένα, σε πλήρη εφαρμογή των αρχών που διέπουν την αρχιτεκτονική του σχεσιακού μοντέλου είναι οργανωμένα σε πίνακες, οι οποίοι συσχετίζονται μεταξύ τους. Η δομή αυτών των πινάκων καθώς και των συσχετίσεων που υφίστανται ανάμεσα στα πεδία τους, μπορεί να ορισθεί κατά τρόπο πλήρως συμβατό με το μοντέλο οντοτήτων συσχετίσεων που έχουμε δημιουργήσει κατά το στάδιο του λογικού σχεδιασμού της εφαρμογής. Αυτό σημαίνει πως θα δημιουργήσουμε πίνακες τόσο για τους τύπους οντότητας που περιλαμβάνονται στο λογικό μοντέλο του συστήματος, όσο και για εκείνους τους τύπους συσχέτισης των οποίων η πολλαπλότητα είναι M:N. Η διαχείριση των

δεδομένων της εφαρμογής, αμέσως μετά την καταχώρησή τους, μπορεί να γίνει χρησιμοποιώντας εντολές της γλώσσας SQL, η οποία υποστηρίζεται πλήρως.

5.3.2. SQL – Structured Query Language

Η SQL (Structured Query Language) είναι μια εξειδικευμένη γλώσσα προγραμματισμού για την αποστολή ερωτημάτων σε βάσεις δεδομένων. Χρησιμοποιείται για να ζητήσει πληροφορίες από μια βάση δεδομένων καθώς και να ενημερώσει και να διαχειριστεί σχεσιακές βάσεις δεδομένων (Ramakrishnan & Grherke, 2000).(Ramakrishnan & Grherke, 2000).

Η SQL είναι πρότυπο που χρησιμοποιείται από όλους τους προμηθευτές και τους προγραμματιστές βάσεων δεδομένων για να καθορίσουν, να εξαγάγουν και να έχουν πρόσβαση στις πληροφορίες που αποθηκεύονται στις βάσεις δεδομένων. Η SQL άρχισε τη ζωή ως δημιουργία της IBM αλλά τυποποιήθηκε από το αμερικανικό εθνικό ίδρυμα προτύπων (ANSI) και το διεθνή οργανισμό για την τυποποίηση (ISO) ως ANSI /*ISO SQL το 1988. Από τότε τα πρότυπα ANSI /ISO SQL συνέχισαν να εξελίσσονται.

Η SQL παρέχει δυνατότητες για:

- τον ορισμό, τη διαγραφή και τη μεταβολή πινάκων και κλειδιών,
- την σύνταξη ερωτήσεων (*queries*),
- την εισαγωγή, διαγραφή και μεταβολή στοιχείων,
- τον ορισμό όψεων (*views*) πάνω στα δεδομένα,
- τον ορισμό δικαιωμάτων πρόσβασης,
- τον έλεγχο της ακεραιότητας των στοιχείων,
- τον έλεγχο συναλλαγών (*transaction*)

5.3.3. Visual C# 2005

Η Visual C# είναι μία σχετικά καινούργια γλώσσα προγραμματισμού (Jones, 2002)(Jones, 2002) η οποία βασίζεται στο component μοντέλο του .NET Framework και στην εμπειρία του COM+. Είναι μια component-oriented γλώσσα πλήρως αντικειμενοστραφής, η οποία βασίζεται στη δύναμη της C και στην ευκολία της Visual Basic. Συντακτικά έχει πάρει τα περισσότερα στοιχεία από την μακροβιότερη γλώσσα προγραμματισμού την C. Είναι πιο σαφής από την C++ και πιο δομημένη από τη Visual Basic και επιτυγχάνει ελάχιστη καμπύλη εκμάθησης για όλους, είτε έχουν ασχοληθεί με προγραμματισμό είτε όχι.

Δίνει την δυνατότητα στους προγραμματιστές να αναπτύξουν εφαρμογές για τα λειτουργικά σύστημα MS-Windows, αλλά και εφαρμογές ιστοσελίδων για το διαδίκτυο. Συνδέεται με τις περισσότερες γνωστές βάσεις δεδομένων για την παροχή ολοκληρωμένων λύσεων. Βρίσκεται ενσωματωμένη στο ενιαίο περιβάλλον ανάπτυξης εφαρμογών Visual Studio 2005.

Η Visual C# χρησιμοποιήθηκε στην υλοποίηση του λογισμικού της εργασίας γιατί αποτελεί ένα επαγγελματικό εργαλείο ανάπτυξης μεγάλου ποσοστού εφαρμογών παγκόσμιος. Επιπλέον λόγος ήταν η πολυπλοκότητα και η πολυμορφία που χαρακτηρίζει την παρούσα εφαρμογή η οποία αποτελείται από τμήματα κώδικα τα οποία έχουν σχέση με προγραμματισμό γραφικών, client-server βάσεις δεδομένων, προγραμματισμό σε επίπεδο δικτύου (SNMP), προγραμματισμό σε επίπεδο API (NDIS) και άλλα αντίστοιχα διαφορετικά περιβάλλοντα. Έτσι η γλώσσα προγραμματισμού πρέπει να μπορεί να ανταποκριθεί τόσο σε προγραμματισμό χαμηλού επιπέδου όσο και υψηλού (4GL).

5.4. Λειτουργικές απαιτήσεις

Για την λειτουργία της εφαρμογής υπάρχουν οι εξής απαιτήσεις σε λογισμικό:

Πίνακας 12: Λειτουργικές Απαιτήσεις Εφαρμογής

Λειτουργικό Σύστημα για τους Ασύρματους Πελάτες (NetArgus Client)	Microsoft© Windows XP με Service Pack 2 με εγκατεστημένο το WindowsXP-KB918997-v6-x86-ELL.exe (native Wi-Fi library update) ή Service Pack 3 Microsoft Windows Vista
Λειτουργικό Σύστημα για τον Διαχειριστή NetArgus Server	Microsoft© Windows Server 2003 ή Microsoft Windows XP
Βάση Δεδομένων	Microsoft© SQL SERVER 2005 ή Microsoft© SQL SERVER EXPRESS 2005 (για μικρές εγκαταστάσεις)

6. Περιγραφή του Λογισμικού - Εγχειρίδιο χρήσης

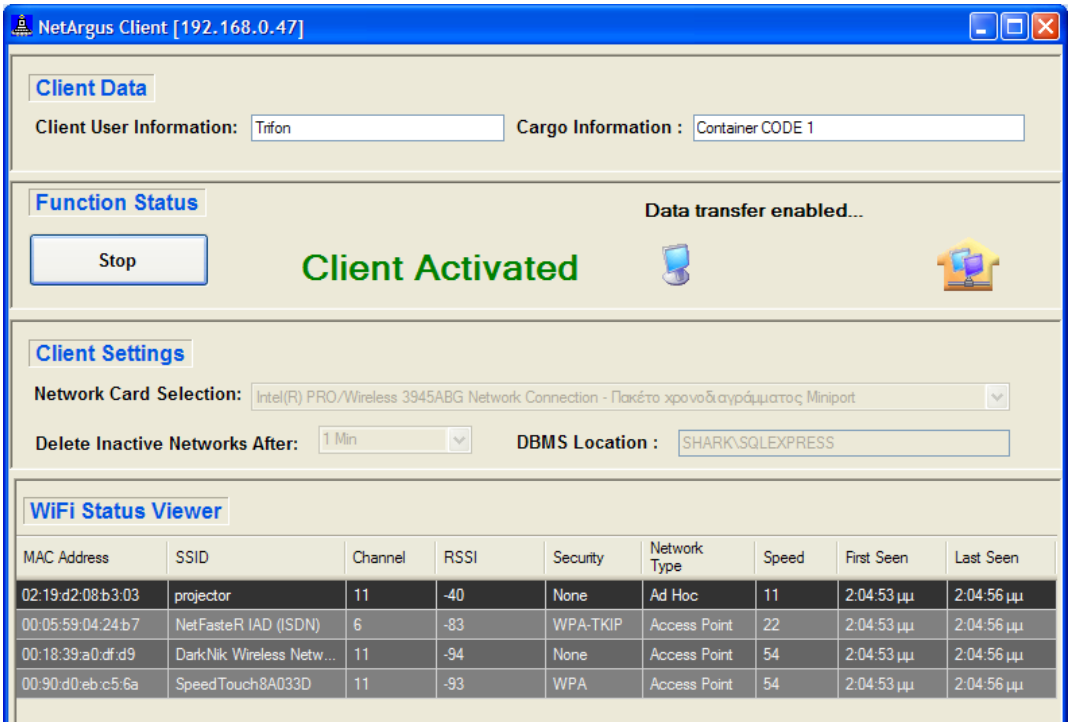
Το λογισμικό παρακολούθησης και διαχείρισης ασύρματων κινούμενων σταθμών αποτελείται από τρία ξεχωριστά τμήματα:

- **Net Argus Client**: Αποτελεί το λογισμικό που εκτελείται σε κάθε κινούμενο σταθμό και η κύρια λειτουργία του είναι η ενημέρωση της κεντρικής βάσης δεδομένων με τις τιμές της στάθμης του σήματος που λαμβάνει ο σταθμός – πελάτης, από τους εντός εμβέλειας του, σταθμούς εκπομπής.
- **Net Argus Server:** Αποτελεί την κεντρική εφαρμογή, η οποία αντλεί στοιχεία από την βάση δεδομένων και υπολογίζει την απόλυτη θέση των κινούμενων σταθμών του δικτύου, σε πραγματικό χρόνο.
- **Net Argus Viewer**: Αποτελεί περιβάλλον διαχείρισης και παρουσίασης των πληροφοριών της βάσης δεδομένων με γραφικό τρόπο.

6.1. NetArgus Client

Η εφαρμογή NetArgus Client, είναι το κομμάτι του λογισμικού που εγκαθίσταται στους Η/Υ των σταθμών πελατών, των οποίων θέλουμε να γνωρίζουμε την θέση ενώ κινούνται στον χώρο. Είναι μία εφαρμογή που εκτελείται συνεχώς, συλλέγοντας σε πραγματικό χρόνο από την ασύρματη κάρτα δικτύου (Wi-Fi NIC) την στάθμη του σήματος που λαμβάνει από όλους τους σταθμούς εκπομπής εντός της εμβέλειας της ασύρματης κάρτας. Η πληροφορία που συλλέγεται, αποστέλλεται συνεχώς στην βάση δεδομένων.

Να σημειωθεί ότι το NetArgus Client ήταν από τα δυσκολότερα και πιο απαιτητικά κομμάτια της εφαρμογής σε θέματα προγραμματισμού. Δημιουργήθηκε για να ξεπεράσουμε τα προβλήματα της συλλογής της στάθμης του σήματος των κινούμενων ασύρματων σταθμών από την μεριά των σταθμών εκπομπής (Access Points) (Βλέπε σχετικά στο 3.2). Είναι μία ιδιαίτερα πολύπλοκη εφαρμογή, η οποία εμπλέκει και τις τρείς βιβλιοθήκες που αναφέρονται στα κεφάλαια 3.3, 3.4, 3.5, 3.6 (WMI, NDIS, Native Wi-fi).



Εικόνα 20: NetArgus Client Εφαρμογή, σε λειτουργία

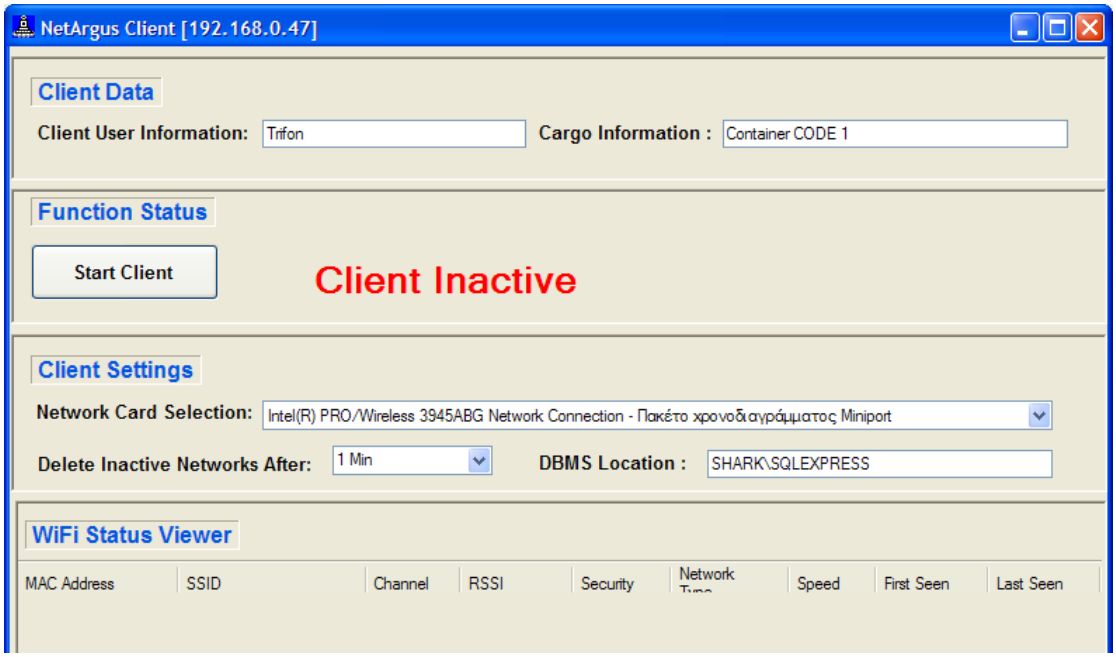
6.1.1. Περιγραφή NetArgus

Η εφαρμογή NetArgus Client έχει ένα μόνο κουμπί, έναρξης/παύσης (Start/Stop Button). Με την επιλογή Start απενεργοποιούνται όλες οι υπόλοιπες λειτουργίες αρχικής εγκατάστασης (περιγράφονται παρακάτω στο ίδιο κεφάλαιο) και ξεκινάει η συλλογή δεδομένων, και η αποστολή τους στην βάση δεδομένων. Με την επιλογή Stop, γίνονται ενεργές και προσβάσιμες οι παράμετροι αρχικής εγκατάστασης της εφαρμογής, και φυσικά σταματάει η συλλογή και αποστολή δεδομένων.

Χάριν παραδείγματος έχουν δημιουργηθεί επίσης δύο πεδία εισαγωγής:

- Στο πρώτο πεδίο (Client User Information) μπορεί ο χρήστης να δηλώσει το username που χρησιμοποιεί,
- Στο δεύτερο πεδίο ο χρήστης καταχωρεί το είδος και το νούμερο του υποθετικού φορτίου

Τα δύο αυτά πεδία δέχονται είσοδο από τον χρήστη,. Μπορούν πολύ εύκολα να συνδεθούν με οποιαδήποτε εφαρμογή εκτελείται στον ίδιο Η/Υ, και να μεταφέρουν αυτόματα αυτήν την πληροφορία στην βάση δεδομένων. Δέχεται πλήρη παραμετροποίηση σχετικά με το είδος των δεδομένων που θα μεταφέρονται από μία όποια ειδική εφαρμογή.

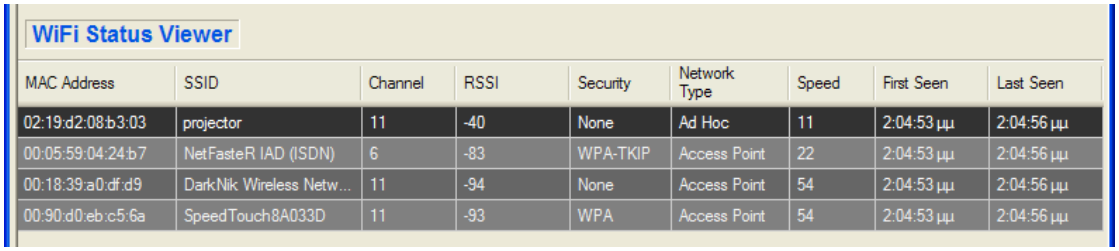


Εικόνα 21: NetArgus Client, Παραμετροποιηση αρχικής εγκατάστασης

6.1.2. Επιλογές παραμετροποίησης[11] NetArgus

- Network Card Selection: Η επιλογή της κάρτας δικτύου από την οποία πρέπει να «διαβάζει» τα δεδομένα (η εφαρμογή αναγνωρίζει –ανακαλύπτει αυτόματα όλες τις πιθανές κάρτες δικτύου NIC's που διαθέτει ο Η/Υ και τις εμφανίζει σε λίστα επιλογών, drop down list).
- Delete Inactive Networks After: Ο χρόνος που απαιτείται για να διαγράψει από τον σχετικό πίνακα ένα δίκτυο που δεν εκπέμπει πλέον. Το δίκτυο μπορεί να μην εκπέμπει είτε γιατί δεν είναι πλέον ενεργό, είτε γιατί ο Η/Υ πελάτης απομακρύνθηκε αρκετά από αυτό.
- DBMS Location: Η θέση της βάσης δεδομένων στην οποία το NetArgus αποστέλλει («γράφει») τα δεδομένα που συλλέγει. Η θέση αυτή μπορεί να δηλώνεται είτε ως όνομα Η/Υ (WINS) είτε ως IP διεύθυνση (DNS).

[11] Η εφαρμογή είναι «γραμμένη» με τέτοιο τρόπο ώστε να θυμάται όλες τις προηγούμενες παραμετροποιήσεις. Όλες οι παράμετροι αποθηκεύονται στο μητρώο (registry) του Η/Υ, ώστε ακόμη και μετά από επανεκκίνηση οι παράμετροι παραμένουν ως είχαν. Η εφαρμογή «θυμάται» και την προηγούμενη κατάσταση που βρισκόταν πριν την όποια έξοδο (κλείσιμο). Αν δηλαδή η εφαρμογή ήταν σε κατάσταση inactive (ανενεργή) όταν έκλεισε, τότε όταν ξανά-ανοίξει θα επανέλθει στην πρότερη κατάσταση (ανενεργή).



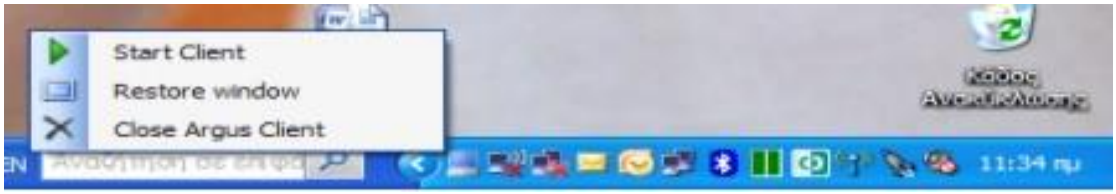
Εικόνα 22: NetArgus, Wi-Fi Status Viewer

Το κομμάτι στο κάτω μέρος της εφαρμογής (WiFi Status Viewer) απεικονίζει σε πίνακα όλη την πληροφορία που ανιχνεύει η εφαρμογή. Αυτή η πληροφορία για κάθε σταθμό εκπομπής εντός εμβέλειας της ασύρματης κάρτας του Η/Υ είναι :

- Mac Address
- SSID[12] (όνομα ασύρματου δικτύου)
- Security (Είδος πρωτοκόλλου ασφαλείας που χρησιμοποιείται)
- Network Type (Είδος δικτύου, αν ο σταθμός εκπομπής είναι access point ή όχι)
- Speed (Ταχύτητα δικτύου, η οποία αντιστοιχεί στα πρωτόκολλα 802.11b/g)
- First Seen (χρόνος που πρωτοεμφανίστηκε το συγκεκριμένο δίκτυο)
- Last Seen (χρόνος που εμφανίστηκε για τελευταία φορά το συγκεκριμένο δίκτυο)

Οι λεπτομέρειες που καταγράφονται για το κάθε δίκτυο εντός εμβέλειας, είναι πολύ περισσότερες από αυτές που χρειάζονται για τις δυνατότητες της υλοποιημένης εφαρμογής. Μπορούν όμως να χρησιμοποιηθούν άμεσα στις πιθανές επεκτάσεις και τροποποιήσεις της εφαρμογής, όπως περιγράφονται στο κεφάλαιο 8.

[12] Service Set IDentifier



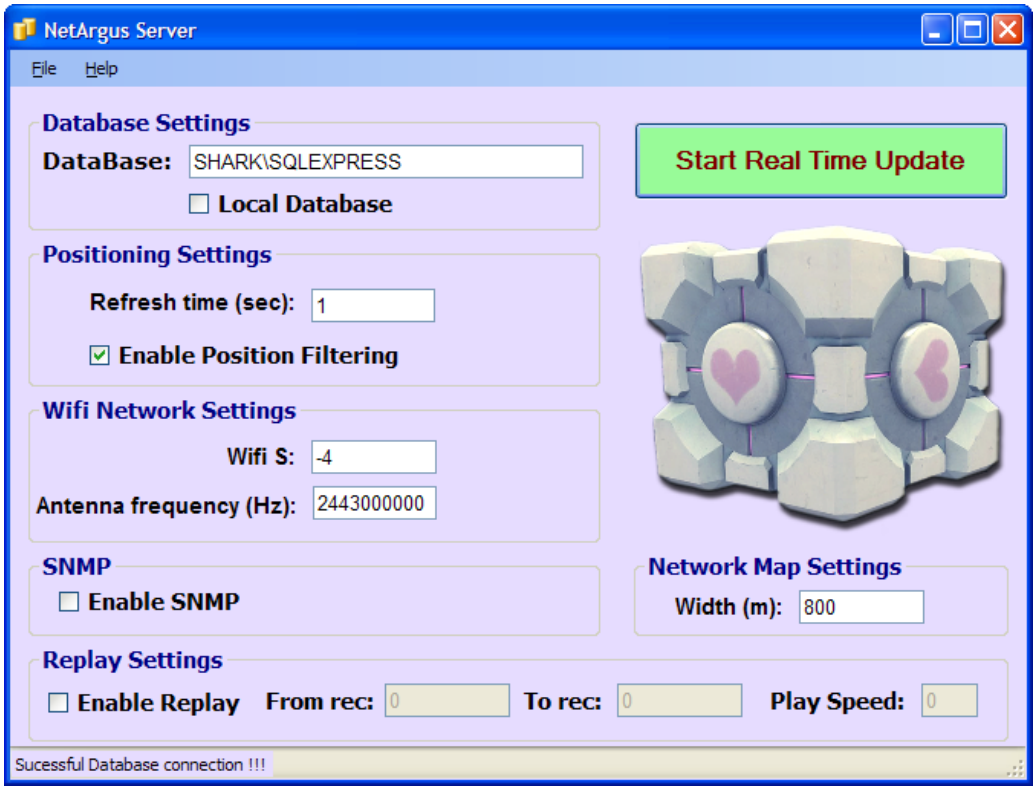
Εικόνα 23: NetArgus Client Ελαχιστοποίηση

6.1.3. Ελαχιστοποίηση της εφαρμογής NetArgus

Η εφαρμογή έχει την δυνατότητα ελαχιστοποίησης. Όταν βρίσκεται σε τέτοια κατάσταση, έχει μόνο ένα εικονίδιο στην «Γραμμή εργασιών» (Task Bar) από το οποίο με δεξί «κλικ» ο χρήστης έχει 3 επιλογές:

- Την έναρξη/ παύση εκτέλεσης (start/stop client) της εφαρμογής
- Την αποκατάσταση (restore) του παραθύρου της εφαρμογής
- Το κλείσιμο (Close Argus Client) της εφαρμογής

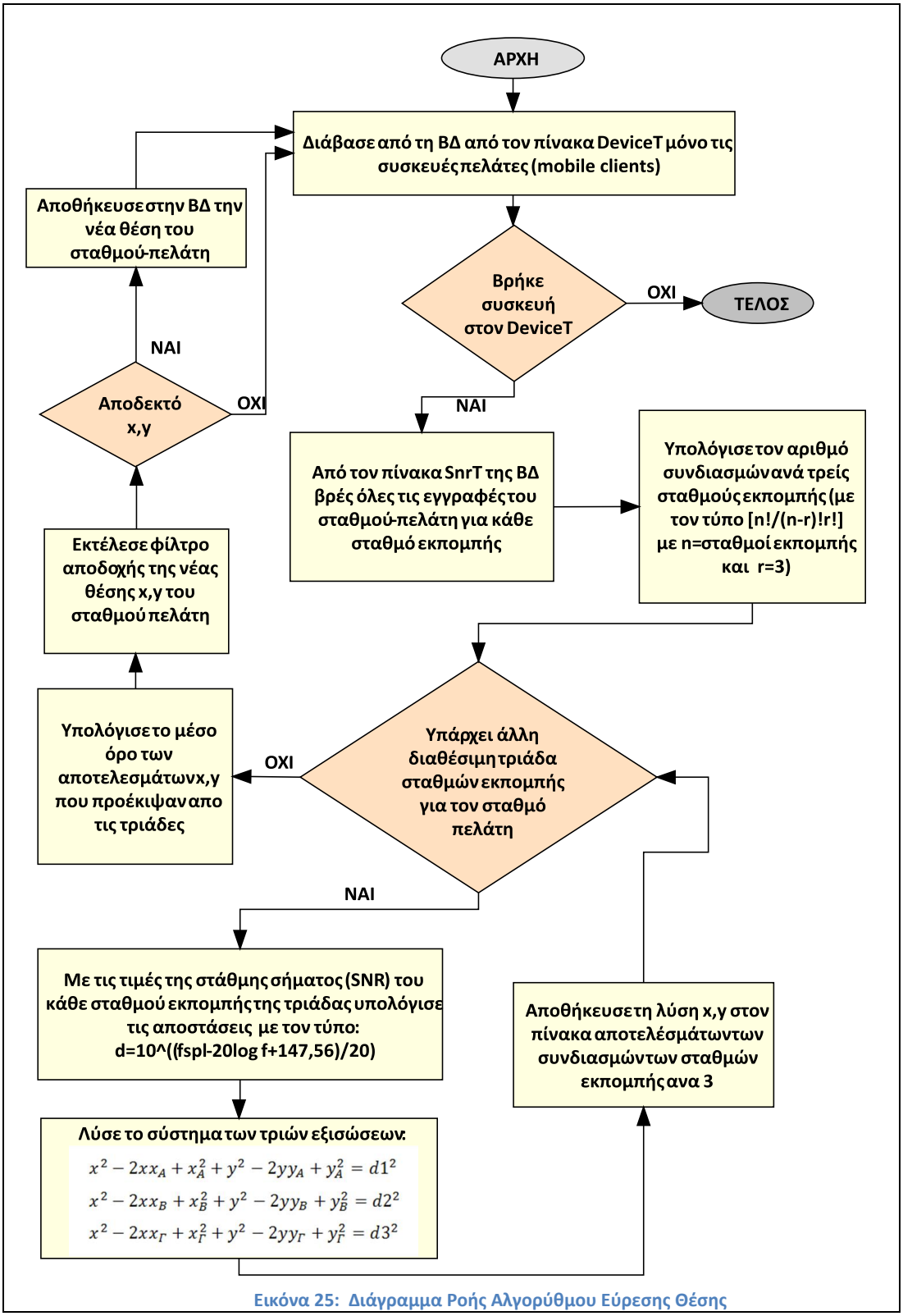
6.2. NetArgus Server



Εικόνα 24: NetArgus Server

Το δεύτερο κομμάτι της εφαρμογής είναι το κομμάτι του Server (NetArgus Server). Αυτό είναι και η καρδιά της εφαρμογής, αφού εδώ επιτελούνται οι κυριότερες λειτουργίες και εκτελούνται οι αλγόριθμοι που δημιουργήθηκαν.

6.3. Διάγραμμα ροής (flow chart) αλγορίθμου εύρεσης θέσης



Εικόνα 25: Διάγραμμα Ροής Αλγορύθμου Εύρεσης Θέσης

Παρακάτω περιγράφονται τα πεδία και οι λειτουργίες τους, της κεντρικής φόρμας της εφαρμογής:

6.3.1. Database Settings

- Database: Ορίζεται η τοποθεσία της βάσης δεδομένων. Η θέση αυτή μπορεί να δηλώνεται είτε ως όνομα Η/Υ (WINS) είτε ως IP διεύθυνση (DNS). Να σημειωθεί ότι η εφαρμογή έχει αμφίδρομη σχάση με την βάση δεδομένων. Ανακαλεί στοιχεί σχετικά με τους ασύρματους σταθμούς –πελάτες, τα επεξεργάζεται και στέλνει στην βάση μεταξύ άλλων τις συντεταγμένες του σταθμού – πελάτη, σε καθορισμένα χρονικά διαστήματα.
- Local Database: εάν η εφαρμογή είναι εγκαταστημένη στον ίδιο Η/Υ με την βάση δεδομένων, δεν χρειάζεται να οριστεί το προηγούμενο πεδίο.

6.3.2. Positioning Settings

- Refresh Time (sec): Ανά πόσα δευτερόλεπτα θα εκτελείται ο αλγόριθμος εύρεσης θέσης
- Enable Position Filtering: υπάρχει δυνατότητα βελτιστοποίησης του αλγόριθμου εύρεσης θέσης μέσω της ενεργοποίησης σχετικού φίλτρου[13]

6.3.3. WiFi Network Settings

- WiFi S: Εσωτερική παράμετρος διόρθωσης/βελτιστοποίησης του αλγόριθμου εύρεσης θέσης. Επειδή όπως αναφέρεται στο κεφάλαιο 3.2, στην λαμβανόμενη στάθμη σήματος υπάρχουν διάφοροι αστάθμητοι παράγοντες και δύσκολα υπολογίσιμες μεταβλητές (π.χ. η εξασθένιση των καλωδίων , cable attenuation), θεωρείται αναγκαία η προσθήκη μίας μεταβλητής λάθους στον αλγόριθμο (Βλέπε και (Antti Kotanen, Positioning with IEEE 802.11b Wireless LAN, 2008), όπου γίνεται η ίδια παραδοχή). Όταν η μεταβλητή τίθεται στο μηδέν, δεν υπεισέρχεται μεταβολή στον τυπικό αλγόριθμο του κεφαλαίου 4.5. Η εύρεση της τιμής της μεταβλητής

[13] Στην παρούσα υλοποίηση, υπάρχει η δυνατότητα εκτέλεσης ενός φίλτρου που αποφασίζει αν θα χρησιμοποιήσει την «καινούρια» θέση που υπολόγισε ο αλγόριθμος, εάν η τιμή είναι μικρότερη ή ίση από ένα κατώφλι (threshold) που ορίζεται εσωτερικά στον αλγόριθμο. Η περαιτέρω παραμετροποίηση και αναβάθμιση του φίλτρου αυτού, προτείνεται στις επεκτάσεις βελτιώσεις της εφαρμογής.

γίνεται μετά από έλεγχο της κάθε τοπικής εφαρμογής και συνήθως κυμαίνεται μεταξύ -0,8 έως 1,5.

- Antenna frequency (Hz): όπως αναφέρεται στην σελίδα 47, η συχνότητα δεν είναι σταθερή. Υπάρχει λοιπόν η δυνατότητα αλλαγής της.

6.3.4. SNMP

- Enable SNMP: Η εφαρμογή μπορεί να λειτουργήσει μόνο τους αλγόριθμους εύρεσης θέσης χωρίς να ανιχνεύει τις δυνατότητες SNMP των συσκευών
- Network Map Settings: Για να μπορέσει η εφαρμογή να αναπαραστήσει την κίνηση των σταθμών πελατών στο χώρο, στην περίπτωση που υπάρχει χάρτης του χώρου στην εφαρμογή του γραφικού περιβάλλοντος του χρήστη (βλέπε 6.4.1), θα πρέπει η εφαρμογή να γνωρίζει την κλίμακα του χάρτη. Σε ποιο ποσοστό σμίκρυνσης δηλαδή αναπαριστά ο χάρτης τον χώρο που κινούνται οι σταθμοί. Η εφαρμογή κάνει αυτόματα τις αναγκαίες μετατροπές από μέτρα σε pixel για να μπορέσει να αναπαραστήσει την κίνηση στην σωστή κλίμακα.

6.3.5. Replay Settings

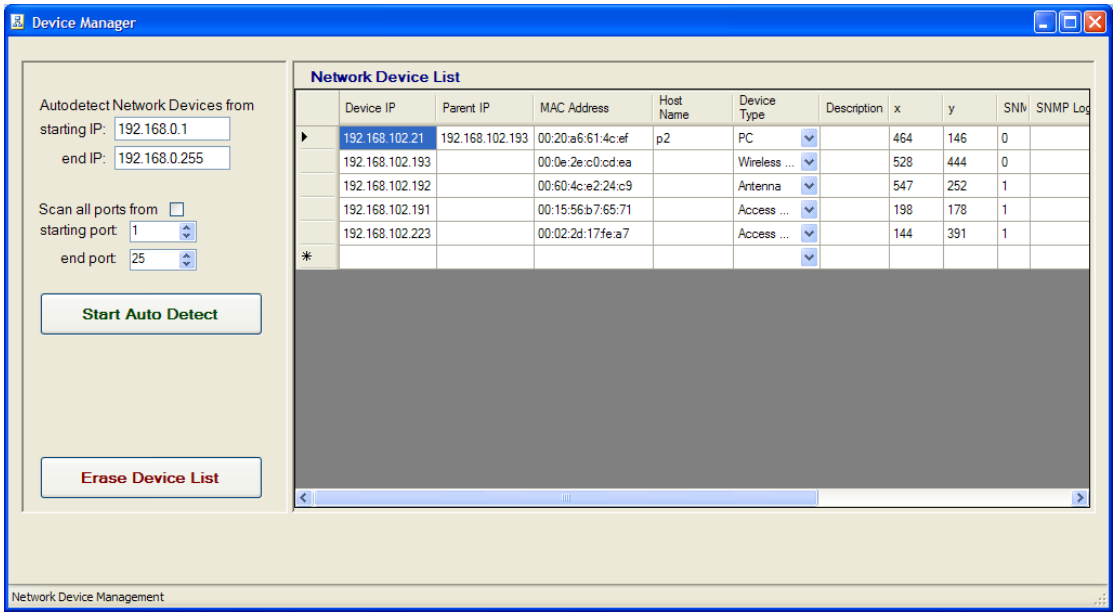
- Enable Replay: Η εφαρμογή έχει την δυνατότητα να επαναϋπολογίσει από την βάση δεδομένων, προηγούμενες κινήσεις και διαδρομές σταθμών πελατών. Αυτό μπορεί να γίνει (αφού συμβουλευτεί κάποιος τον σχετικό πίνακα LogT) ξεκινώντας από συγκεκριμένη καταχώρηση στην βάση (from rec:), μέχρι κάποια συγκεκριμένη καταχώριση (To rec:).
- Play speed: η ταχύτητα (σε δευτερόλεπτα, sec) με την οποία θα προβάλλει την κίνηση αυτή πάνω στον χάρτη.

Εδώ πρέπει να σημειωθεί ότι κάθε φορά που ζητείται από το πρόγραμμα να κάνει replay κάποιων συγκεκριμένων καταχωρήσεων στην βάση, το πρόγραμμα αντιμετωπίζει αυτές τις καταχωρήσεις με την ίδια ακριβώς αντιμετώπιση της πρώτης φοράς (πραγματικού χρόνου). Στην βάση δεδομένων η κάθε γραμμή καταχώρησης (record) αφορά στον κινούμενο σταθμό-πελάτη, τον σταθμό-εκπομπής, και την στάθμη του σήματος μεταξύ τους. Άρα όταν ζητάμε από την εφαρμογή να κάνει replay στην ουσία επαναϋπολογίζει την απόσταση μεταξύ των δύο σημείων, και με αυτή τροφοδοτεί τους αλγόριθμους εύρεσης θέσης. Άρα

κάθε φορά, η θέση επαναϋπολογίζεται. Αυτό έγινε για λόγους αποσφαλμάτωσης και τεκμηρίωσης της εφαρμογής (fine tuning).

Μέσω της χρήσης του κουμπιού “File” στο αριστερό άνω άκρο της εφαρμογής, ο χρήστης έχει πρόσβαση σε τρία μενού:

6.3.6. Device Manager



Εικόνα 26: Argus Server, Device Manager

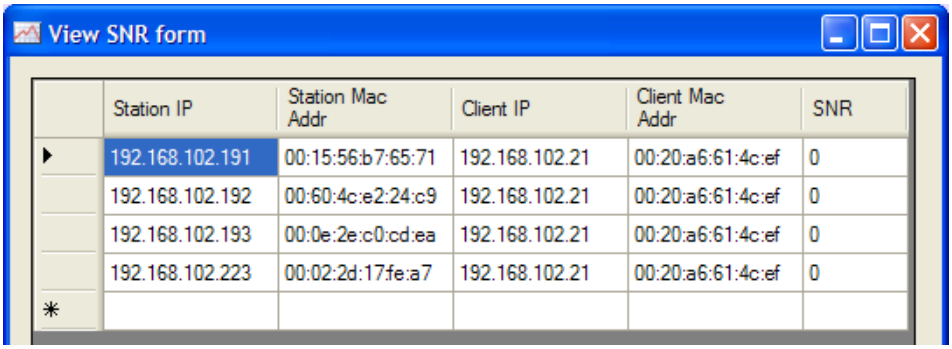
Στην αρχική εγκατάσταση της εφαρμογής, ο χρήστης μπορεί από αυτό το μενού να παραμετροποιήσει πλήρως το δίκτυο στο οποίο θα εργάζεται η εφαρμογή.

- Autodetect Network Devices from: ο χρήστης πρέπει να δηλώσει το εύρος (starting IP, end IP) των IP διευθύνσεων που η εφαρμογή θα αναζητήσει υπαρκτές συσκευές. Η εφαρμογή αναζητά ενεργές συσκευές σε αυτό το εύρος, και όσες «απαντούν» καταχωρούνται αυτόματα στην βάση δεδομένων, ενώ τα στοιχεία του εμφανίζονται στον πίνακα “Network Device List”, στο δεξί μέρος του μενού
- Scan all ports from: η εφαρμογή μπορεί να κάνει εξαντλητική αναζήτηση σε κάθε ενεργή συσκευή, ανιχνεύοντας τις «ενεργές» πόρτες (active ports) της συσκευής. Αυτό συνήθως οδηγεί τον χρήστη στο να κατανοήσει τι είδους συσκευή είναι η συγκεκριμένη (π.χ. ο web server έχει ανοιχτή την πόρτα 80)

- Start Auto Detect: η εφαρμογή θα ξεκινήσει τον έλεγχο – αναζήτηση σε όλο το εύρος του δικτύου που ορίστηκε προηγουμένως
- Erase Device List: Διαγράφονται όλες οι συσκευές που έχουν καταχωρηθεί, τόσο στον πίνακα δεξιά στο μενού, αλλά και από την βάση δεδομένων

Να σημειωθεί ότι εφόσον βρεθούν οι ενεργές συσκευές, (Device IP, Mac Address, Host Name) ο χρήστης μπορεί να καταχωρήσει τις επιπλέον πληροφορίες:

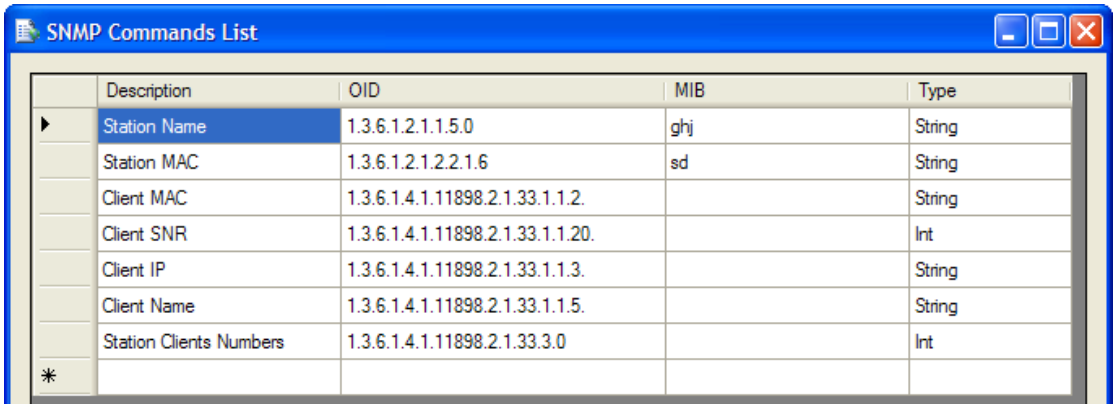
- Network Device List
    - Συντεταγμένες x,y: όταν η συσκευή είναι σταθερή (σταθμός εκπομπής, access point), ο χρήστης έχει την δυνατότητα να ορίσει τις συντεταγμένες του σημείου βάσει της κλίμακας του χάρτη που έχει οριστεί στην γραφική εφαρμογή (βλέπε κεφάλαιο..). φυσικά ο χρήστης έχει την δυνατότητα να μετακινήσει τον σταθμό αργότερα στην γραφική εφαρμογή , απλώς σύροντας τον εν λόγω σταθμό σε άλλη θέση. Αυτόματα θα ενημερωθεί και ο πίνακας στο δεξί μέρος του μενού, αλλά και η βάση δεδομένων.
    - SNMP: είναι μία Boolean μεταβλητή (0,1) η οποία (0) ορίζει ότι δεν υπάρχει (ή δεν είναι ενεργοποιημένη) η δυνατότητα SNMP σε αυτήν την συσκευή, και (1) ορίζει ότι η συσκευή δύναται να επικοινωνήσει μέσω SNMP πρωτοκόλλου
    - SNMP login: Είναι το read-only συνθηματικό της συγκεκριμένης συσκευής, για το πρωτόκολλο SNMP
    - SNMP password: Είναι το read-write συνθηματικό της συγκεκριμένης συσκευής, για το πρωτόκολλο SNMP
- View SNR from



Εικόνα 27: Argus Server - View SNR form

Εδώ φαίνονται όλες οι καταχωρήσεις της βάσης σε τριάδες που αποτελούνται από τον σταθμό εκπομπής (Station IP, Station Mac Addr), τον σταθμό πελάτη (Client IP, Client Mac Addr) και την μεταξύ τους στάθμη σήματος (SNR).

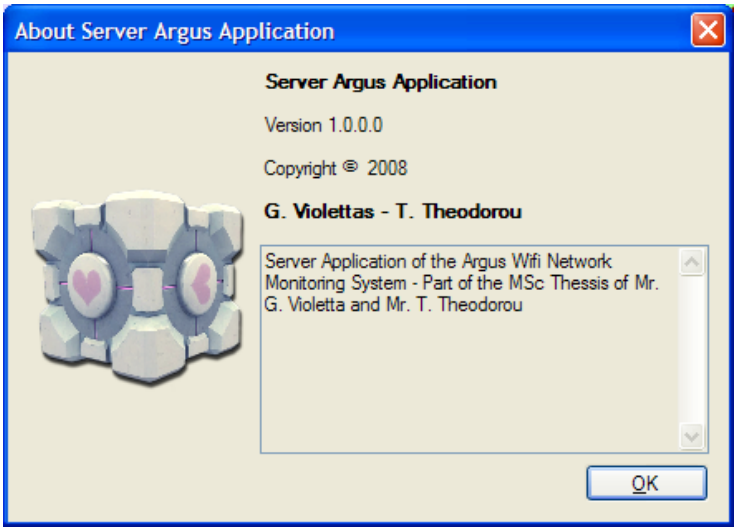
- SNMP Commands List



Εικόνα 28: Argus Server - SNMP Command List

Αυτός ο πίνακας είναι πλήρως παραμετροποιήσιμος. Ο χρήστης μπορεί εδώ να προσθέσει όσες εντολές SNMP επιθυμεί. Αυτές μπορεί να είναι είτε σε μορφή OID (βλέπε 2.8) (ακολουθία αριθμών) και να αφορούν συγκεκριμένη μάρκα και μοντέλο συσκευής, είτε να προσθέσει ένα ολόκληρο MIB (βλέπε 2.7) είτε συγκεκριμένης εταιρίας, είτε συγκεκριμένου πρωτοκόλλου. Το πρωτόκολλο MIB που εμείς δοκιμάσαμε ήταν το 802.11, όπως περιγράφεται στο (Gast, 2002).

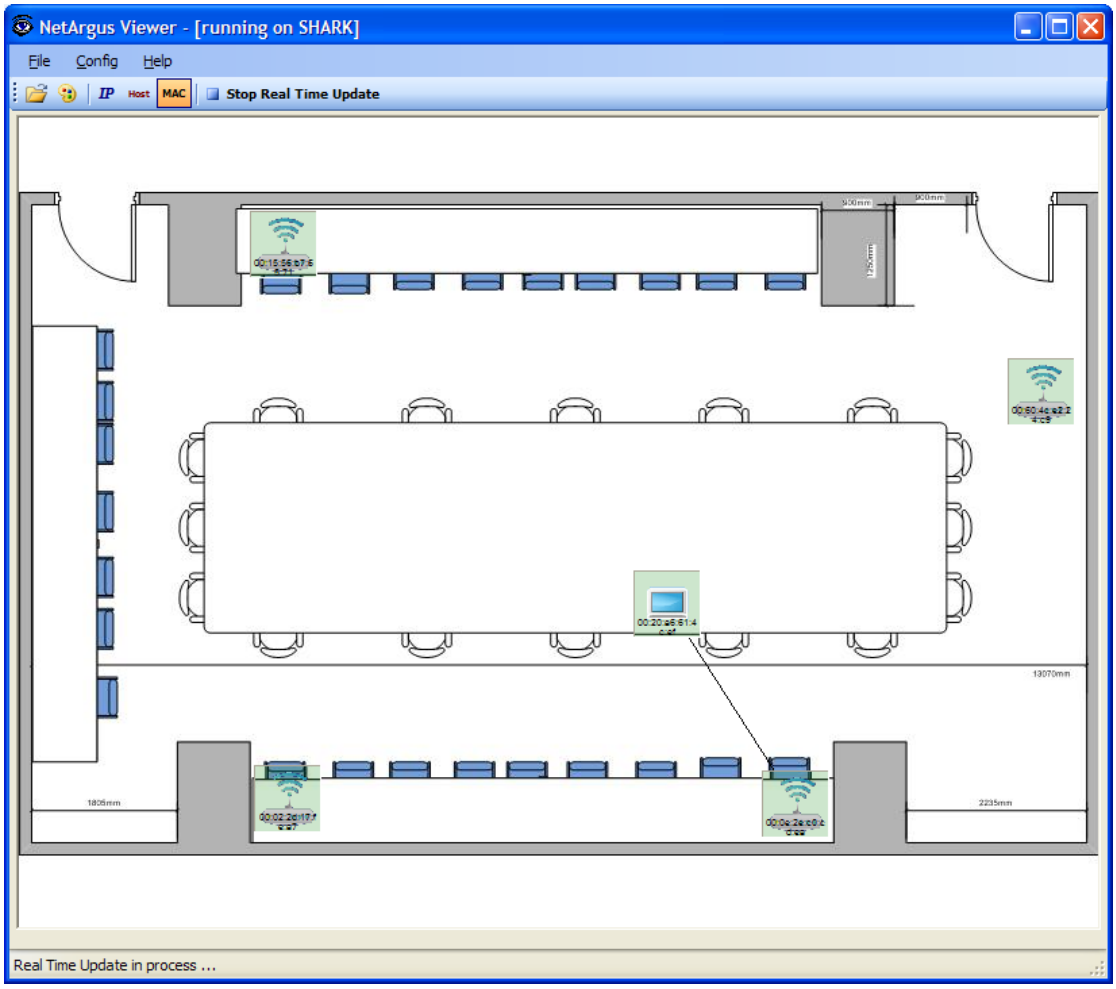
- Argus Server – About



Εικόνα 29: Argus Server - About

Κάθε εφαρμογή που σέβεται τον εαυτό της, περιγράφει τους δημιουργούς της, την έκδοση (version) και το έτος που δημιουργήθηκε.

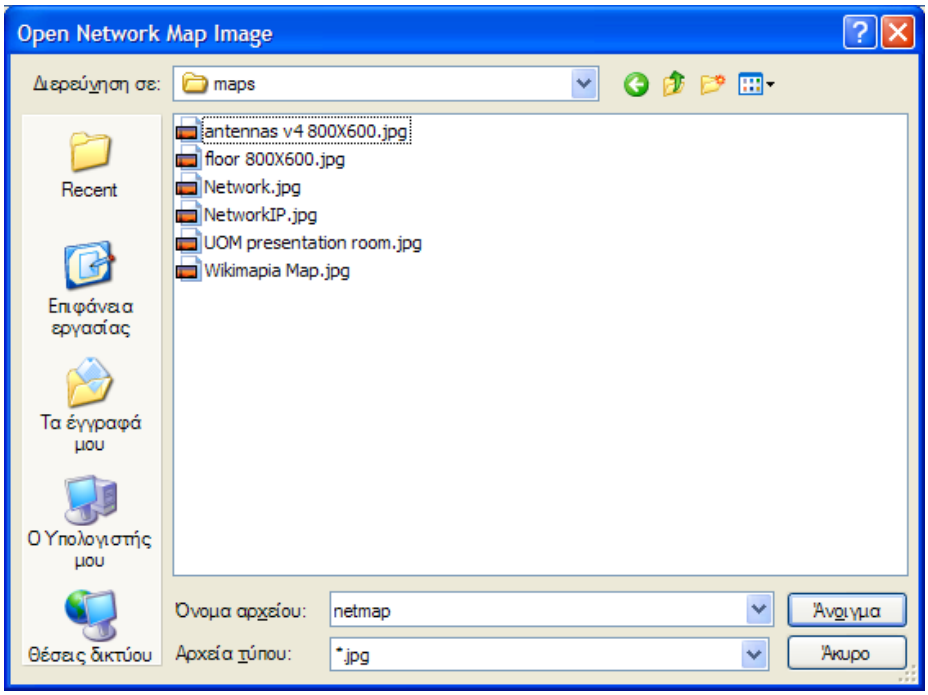
6.4. Net Argus Viewer



Εικόνα 30: NetArgus Viewer - Βασική Οθόνη

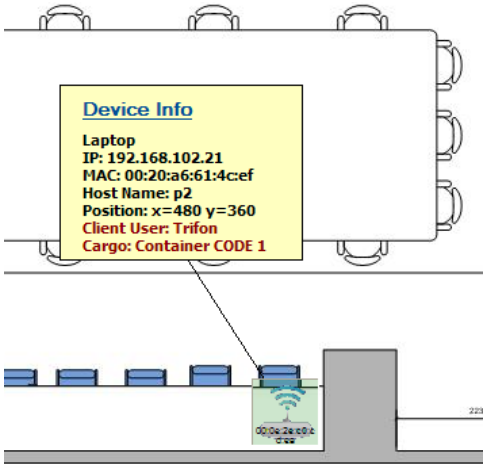
Το τρίτο κομμάτι της εφαρμογής που δημιουργήθηκε, είναι το γραφικό κομμάτι του τελικού χρήστη της εφαρμογής. Είναι τελείως ανεξάρτητο από τα άλλα δύο μέρη, και μπορεί να είναι εγκαταστημένο στο ίδιο ή διαφορετικό δίκτυο με τα άλλα δύο κομμάτια της εφαρμογής. Μπορεί εύκολα να επικοινωνεί με αυτά, ακόμη και μέσω διαδικτύου, μέσω VPN κτλ. Φυσικά μπορεί να υπάρχουν πολλαπλά στιγμιότυπα (instances) της εφαρμογής που να εκτελούνται παράλληλα για το ίδιο δίκτυο, σε διαφορετικά μέρη, και με διαφορετικά δικαιώματα το καθένα.

6.4.1. Εύρεση χάρτη



Εικόνα 31: NetArgus Client - Open Network Map Image

Η πρώτη και απαραίτητη παραμετροποίηση που πρέπει να γίνει στην εφαρμογή, είναι η εύρεση του χάρτη που θα χρησιμοποιηθεί. Εδώ να σημειωθεί ότι προς το παρόν η εφαρμογή εργάζεται σε χάρτες 800 X 600. Άρα ο χάρτης θα πρέπει να είναι σε αντίστοιχη κλίμακα. Μία από τις επεκτάσεις της εφαρμογής, θα μπορούσε να είναι η αυτόματη μετατροπή οποιαδήποτε κλίμακας χάρτη.



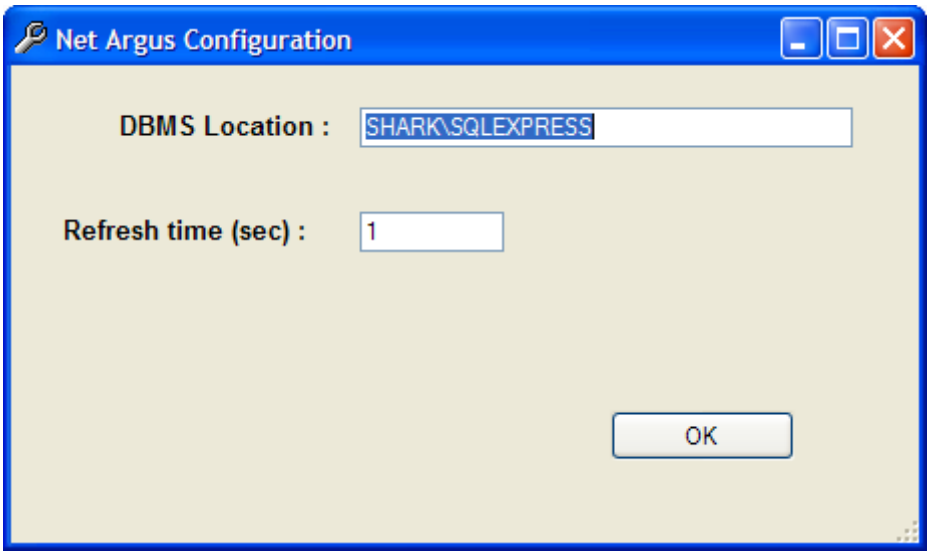
Εικόνα 32: NetArgus Client - Δεξί κλίκ, Device Info

Τα είδη των συσκευών που εμφανίζονται και οι θέσεις τους, είναι οι αντίστοιχες καταχωρήσεις στην βάση δεδομένων και ανανεώνονται σε πραγματικό χρόνο.

Με δεξί κλικ πάνω σε οποιαδήποτε συσκευή, μπορούμε να δούμε άμεσα όλη την καταγεγραμμένη στην βάση δεδομένων πληροφορία, σχετικά με αυτή την συσκευή, όπως είδος συσκευής, IP διεύθυνση, Mac Address, θέση της συσκευής (x,y), και οποιαδήποτε έξτρα πληροφορία αποστέλλει η συσκευή, π.χ. το είδος του φορτίου που μεταφέρει. Αυτή την πληροφορία, το κομμάτι της εφαρμογής που βρίσκεται εγκατεστημένο στον σταθμό πελάτη, μπορεί να την δέχεται ως αυτοματοποιημένη είσοδο (auto input) από οποιαδήποτε άλλη διεργασία- εφαρμογή εκτελείται παράλληλα στον σταθμό πελάτη.

Η εφαρμογή είναι πλήρως δυναμική. Υποστηρίζει την άμεση τροποποίηση της θέσης της οποιασδήποτε συσκευής πάνω στον χάρτη με απλό drag & drop. Άρα ο χρήστης πρέπει να είναι προσεκτικός σε τέτοιες περιπτώσεις γιατί είναι προφανές ότι ενημερώνεται αυτόματα και η βάση δεδομένων για την νέα θέση, αντικαθιστώντας τα προηγούμενα στοιχεία.

6.4.2. Net Argus Configuration



Εικόνα 33: Net Argus COnfiguration

- DBMS Location: Απαραίτητες αρχικές ρυθμίσεις της εφαρμογής, είναι η θέση της βάσης δεδομένων με την οποία θα επικοινωνεί η εφαρμογή. Να σημειωθεί εδώ ότι η εφαρμογή Net Argus Viewer δεν επικοινωνεί άμεσα με την εφαρμογή NetArgus Server. Οι δύο εφαρμογές επικοινωνούν μόνο μέσω της βάσης δεδομένων, και αυτό προσθέτει στην αξιοπιστία και στην ανοχή σε σφάλματα της όλης εφαρμογής.
- Refresh time (sec): το κλάσμα του χρόνου κατά το οποίο η εφαρμογή θα επικοινωνεί με την βάση δεδομένων για να ανανεώσει την θέση των κινούμενων σταθμών. Να σημειωθεί ότι αυτός ο χρόνος πρέπει να είναι μεγαλύτερος ή ίσος από τον χρόνο εκτέλεσης των αλγορίθμων εύρεσης θέσης, γιατί σε αντίθετη περίπτωση απλώς δεν θα υπάρχουν ανανεωμένα δεδομένα.

7. Συμπεράσματα

Η εφαρμογή που δημιουργήσαμε (NetArgus) ακολουθεί όλα τα σύγχρονα πρότυπα σχεδίασης. Είναι κατανεμημένη (distributed) ακολουθώντας το μοντέλο 3-tier.

- Με την βάση δεδομένων επικοινωνούν οι εφαρμογές – πελάτες (NetArgus Client & NetArgus Viewer) οι οποίες υποστηρίζονται από την εφαρμογή NetArgus Server.
- Η εφαρμογή NetArgus Server μπορεί να εκτελείται σε διαφορετική τοποθεσία (δίκτυο) από τις υπόλοιπες εφαρμογές. Σε αυτήν την εφαρμογή εκτελούνται σε πραγματικό χρόνο οι αλγόριθμοι εύρεσης θέσης του κινούμενους σταθμού.
- Η εφαρμογή NetArgus Client είναι η εφαρμογή που εγκαθίσταται σε κάθε κινούμενο σταθμό που θέλουμε να παρακολουθούμε την θέση του. Αποστέλλει συνεχώς στην βάση δεδομένων στοιχεία για την στάθμη του σήματος λήψης με όλους τους σταθμούς εκπομπής στην περιοχή λήψης του.
- Η εφαρμογή NetArgus Viewer είναι το γραφικό περιβάλλον παρουσίασης των δεδομένων που συλλέγονται. Επικοινωνεί με την βάση δεδομένων σε πραγματικό χρόνο, και αποτυπώνει τόσο την κίνηση των περιφερόμενων σταθμών, όσο και όλη την υπόλοιπη πληροφορία του δικτύου.

Η εφαρμογή που δημιουργήθηκε μπορεί να θεωρηθεί ότι εξυπηρετεί τους σκοπούς που έθετε η παρούσα εργασία. Βρίσκει την θέση ενός κινούμενου σταθμού μέσα σε έναν ορισμένο (x,y) χώρο με σχετική ακρίβεια, όση επιτρέπεται από τις αδυναμίες του πρωτοκόλλου 802.11 που περιγράφονται στα αντίστοιχα κεφάλαια. Υλοποιεί με ευκολία και χωρίς ιδιαίτερο φόρτο τους αλγόριθμους εύρεσης θέσης, οι οποίοι δημιουργήθηκαν εξαρχής για αυτόν το σκοπό. Δημιουργήθηκε μία εφαρμογή που εγκαθίσταται σε κάθε κινούμενο σταθμό πελάτη, για να ξεπεραστούν οι αδυναμίες των σταθμών εκπομπής που χρησιμοποιήθηκαν. Αυτή η εφαρμογή είναι έτοιμη να χρησιμοποιηθεί και σε πιθανές επεκτάσεις της όλης εφαρμογής και σε πιθανές τροποποιήσεις χρήσης της, όπως περιγράφονται στο κεφάλαιο 8. Επίσης η εφαρμογή μπορεί εύκολα να συνεργαστεί με άλλες εφαρμογές που τυχόν εκτελούνται στον ίδιο Η/Υ και να δεχθεί ως είσοδο δεδομένα προς αποστολή στην βάση δεδομένων.

Η εφαρμογή δοκιμάστηκε σε διάφορα περιβάλλοντα σε πραγματικές συνθήκες. Όντως μπορούσε να προβλέψει την θέση του κινούμενου σταθμού με σχετική ακρίβεια. Εξήχθησαν και πολύτιμα συμπεράσματα σχετικά με την ποιότητα του προσλαμβανόμενου σήματος όσον αφορά τις αποστάσεις, την ποιότητα και κατάσταση του εξοπλισμού, και τις διαφορές μεταξύ των διαφορετικών κατασκευαστών.

Ως άξιο σχολιασμού συμπέρασμα, που δεν βρήκαμε σε άλλες παρεμφερείς εργασίες, είναι το γεγονός ότι η διαδικασία εύρεσης θέσης μέσω της στάθμης σήματος, έχει θεωρητικό κάτω όριο ακρίβειας τα 2,5 μέτρα περίπου. Επειδή όπως είδαμε η στάθμη του σήματος είναι ακέραιος αριθμός, η μεταβολή από το έναν ακέραιο στον επόμενο -λόγω της λογαριθμικής συνάρτησης- επιφέρει σημαντική μεταβολή στην υπολογίσιμη απόσταση.

Σε κλειστούς χώρους όπου υπάρχουν και αρκετές παρεμβολές λόγω γειτονικού θορύβου, η εφαρμογή έχει σημαντικές αποκλίσεις στην ακρίβεια εντοπισμού. Για την βελτίωση του προβλήματος , μπορούν να γίνουν προσθήκες φίλτρων όπως για παράδειγμα το kalman filter, κάτι που θα μπορούσε να είναι το αντικείμενο μίας ξεχωριστής εργασίας.

Όσον αφορά στο SNMP, η εφαρμογή είναι έτοιμη να εργαστεί σε πολύπλοκα και ανομοιογενή δίκτυα (Adaptability). Μπορεί πολύ να δεχθεί αλλαγές και προσθήκες στα MIB που χρησιμοποιεί και μπορεί πολύ εύκολα να αναπαραστήσει την προσλαμβανόμενη πληροφορία.

8. Μελλοντικές Επεκτάσεις

Η παρούσα εργασία είναι ένα πολύ μικρό κομμάτι ενός πεδίου της πληροφορικής που μοιάζει αχανές και ατελείωτο.

Υπάρχουν πολλοί τομείς της πληροφορικής που συνυπάρχουν σε αυτήν την εργασία, με τρείς κυριότερους άξονες:

- Το SNMP και τις διάφορες πολλαπλές και αχανείς εφαρμογές του
- Τα δίκτυα Wi-Fi και τις ανεξάντλητες εφαρμογές τους
- Τις δυνατότητες εύρεσης θέσης σταθμού πελάτη μέσω αυτών των δικτύων και της υπάρχουσας πληροφορίας που προσφέρουν

Όσον αφορά στο SNMP και την παρακολούθηση συσκευών πελατών, υπάρχουν πολλαπλά στάδια επέκτασής:
Αναφέρονται χαρακτηριστικά η εξαγωγή συμπερασμάτων επιβάρυνσης κυκλοφορίας ενός δικτύου και κατάλληλη αλλαγή δρομολογήσεων ίσως και μέσω δυναμικών πρωτοκόλλων δρομολόγησης (RIP, BGP, IGMP), πιθανές υλοποιήσεις εφαρμογών παρακολούθησης δικτύου και αυτόματων αναφορών και ανάληψης πρωτοβουλιών (π.χ. σε ένα δίκτυο ύδρευσης , η αναγνώριση διαρροής ύδατος και η δρομολόγηση ή αποκοπή της ροής του ύδατος) και χιλιάδες άλλες υλοποιήσεις. Η παρούσα εργασία παρέχει το γενικό πλαίσιο στους επόμενους ενδιαφερόμενους να το χρησιμοποιήσουν κατά το δοκούν.

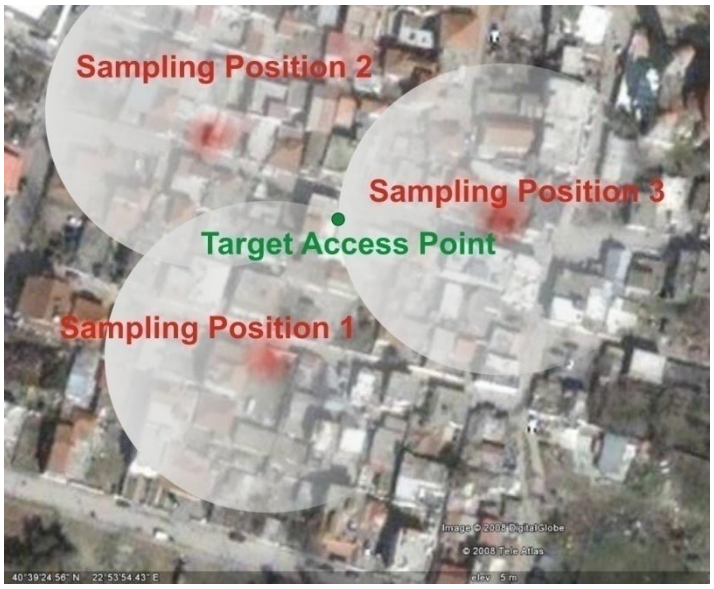
Όσον αφορά στα δίκτυα Wi-Fi υπάρχει ένα μεγάλο πεδίο έρευνας ανοιχτό:
Δεν υπάρχουν πουθενά σε όλο το 802.11x, σαφώς ορισμένοι μέθοδοι και πρωτόκολλα για το roaming (διαμεταγωγή) μεταξύ των σταθμών εκπομπής. Το 802.11x παρέχει κάποιες βασικές κατευθύνσεις και οι διάφοροι κατασκευαστές, υλοποιούν τις δικές τους μεθόδους διαμεταγωγής με αποτέλεσμα μεγάλα προβλήματα μεταξύ υλικών διαφορετικών κατασκευαστών.

Φυσικά οι αλγόριθμοι εύρεσης θέσης και το όλο κύκλωμα εν γένει, είναι σίγουρο ότι μπορεί να βελτιωθεί και να επεκταθεί, ιδίως προς την κατεύθυνση της εύρεσης θέσης άλλων συσκευών (π.χ. tags) ώστε να έχει και άλλες χρήσεις, όπως ανεύρεση συσκευών, υλικού κτλ.

Θα μπορούσε επίσης να γίνει χρήση καλύτερων αλγορίθμων και βελτιωμένων τεχνικών όπως αυτών που περιγράφονται στο (Kohout & Kolingerova, 2003).

Οι εφαρμογές που δημιουργήθηκαν μπορούν με ελάχιστες αλλαγές να χρησιμοποιηθούν ως εξής:

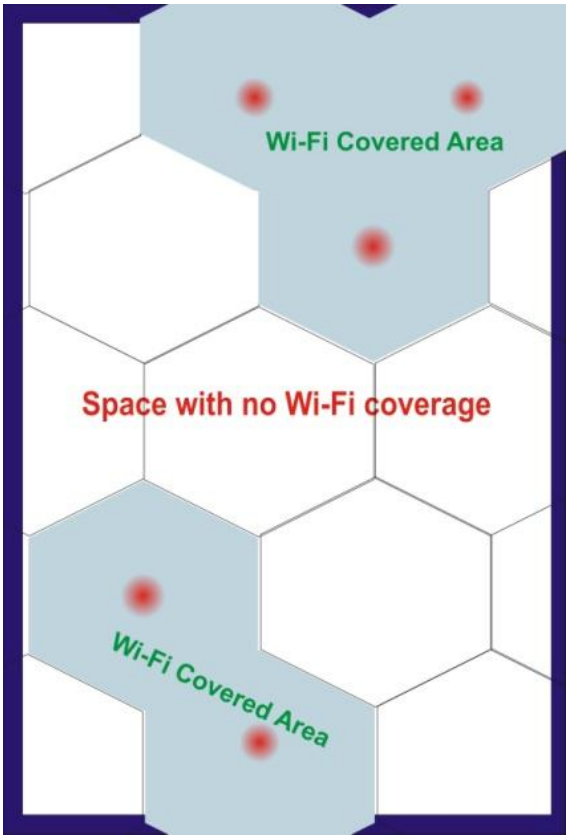
- **Τριγωνοποίηση για την εύρεση σταθμού εκπομπής στον χώρο**. Μέσω της χρήσης διανυσματικών χαρτών (π.χ. Google Maps, ή οποιοδήποτε DEM δίνει πληροφορία Longitude & latitude (Γεωγραφικές συντεταγμένες)) αν εκ των προτέρων θέσουμε 3 τουλάχιστον αυθαίρετα και γνωστά εκ των προτέρων (και φυσικά προσβάσιμα) στον οριοθετημένο χώρο, και μετρήσουμε την στάθμη του σήματος ενός υπό στόχευση σταθμού εκπομπής, τότε μπορούμε με ακρίβεια να προσδιορίσουμε την φυσική, γεωγραφική του θέση.



Εικόνα 34: Τριγωνοποιήση απο γνωστές θέσεις, θέσης άγνωστου στόχου

- **Χαρτογράφηση και αποτύπωση στάθμης σήματος σταθμών εκπομπής σε καθορισμένο χώρο**. Οι εφαρμογές μπορούν να χρησιμοποιηθούν για την μέτρηση της στάθμης σήματος σε έναν χώρο, στον οποίο υπάρχουν ήδη σταθμοί εκπομπής. Αυτό μπορεί να οδηγήσει είτε σε αναδιάταξη των υπαρχόντων σταθμών εκπομπής ώστε να πλησιάσουν την βέλτιστη διάταξη, να τοποθετηθούν δηλαδή έτσι ώστε να μην υπάρχει (σχεδόν) αλληλοεπικάλυψη (φαινόμενο κυψέλης ασύρματων δικτύων, βλέπε (Stüber, 2001),(Stüber, 2001), είτε να οδηγήσει στην προσθήκη σταθμών

εκπομπής ώστε να καλυφθούν τα νεκρά (μη ύπαρξη στάθμης εκπομπής) σημεία του χώρου.



Εικόνα 35: Κάλυψη χώρου απο σταθμούς εκπομπής Wi-Fi

Δευτερεύοντες άξονες αυτής της εργασίας και πιθανά πεδία μελλοντικής έρευνας, είναι οι εφαρμογές 3-tier οι οποίες γιγαντώνονται μαζί με την χρήση του διαδικτύου, η χρήση η ευχρηστία και η ευφυΐα των σημερινών βάσεων δεδομένων, καθώς καλούνται να ανταπεξέλθουν συνεχώς σε αυξημένες απαιτήσει και πολυσύνθετα προβλήματα.

Βιβλιογραφία

RFC 3411. (2002). Architecture for SNMP Management Frameworks. Harrington, et al.

AeroScout . (2008). *How the AeroScout System Works.* Ανάκτηση από AeroScout .

Antti Kotanen, M. H. (2008). Positioning with IEEE 802.11b Wireless LAN. Korkeakoulunkatu 1, FIN-33720 Tampere, Finland: Tampere University of Technology, Institute of Digital and Computer Systems .

Antti Kotanen, M. H. (2008). Positioning with IEEE 802.11b Wireless LAN. Korkeakoulunkatu 1, FIN-33720 Tampere, Finland : Tampere University of Technology, Institute of Digital and Computer Systems .

Antti Seppänen, J. I. (2008). *EXTRACTING AND USING POSITION INFORMATION IN WLAN NETWORKS.*

Application fields of ASN.1. (2005, Aug Wed). Ανάκτηση 08 2008, από http://asn1.elibel.tm.fr/en/uses/index.htm

Cisco. (2008). *Cisco Wireless Location Applianc.* Ανάκτηση 01 2008, από Wi-Fi Based Real-Time Location Tracking: Solutions and Technology: http://www.cisco.com/en/US/prod/collateral/wireless/ps5755/ps6301/ps6386/prod_white_paper0900aecd80477957.html

Consultative Committee for Units (CCU). (2003). *Report of the 15th meeting.* Bureau International des Poids et Mesures.

CVEL. (2008). *Electromagnetic Compatibility*. Ανάκτηση 10 21, 2008, από Working with Decibels: http://www.cvel.clemson.edu/emc/tutorials/DB_Notes/DB_Notes.html

Ekahau. (2008). *Asset Tracking.* Ανάκτηση 01 2008, από http://www.ekahau.com/?id=1012

Gast, M. (2002). *802.11® Wireless Networks: The Definitive Guide.* O'Reilly .

IEEE-802.11. (2008, 10 06). *IEEE 802.11TM WIRELESS LOCAL AREA NETWORKS*. Ανάκτηση 10 06, 2008, από The Working Group for WLAN Standards: http://www.ieee802.org/11/

Interlink Networks Inc. (2002). *A Practical Approach to Identifying and Tracking Unauthorized 802.11 Cards and Access Points.* Ann Arbor, MI 48108 USA.

ITU-T. (2002). Abstract Syntax: Notation One (ASN.1).

Jones, B. L. (2002). *Teach Yourself C# in 21 Days.* SAMS.

Kohout, J., & Kolingerova, I. (2003). *Parallel Delaunay Triangulation in E^3. Make it simple.* Plzen, Czech Republic: University of West Bohemia, Univezsitni 8, 36 14.

Lee, C.-H. (2004). *NDIS RTP Quality Monitor.* New York, NY 10027-6902: Columbia University, Computer Science.

Mauro, R., Mauro, D. R., & Schmidt, K. J. (2005). *Essential SNMP.* O'Reilly.

Microsoft MSDN Forums. (2008). *MSDN Forums*. Ανάκτηση 07 12, 2008, από WMI MSNdis_80211_ServiceSetIdentifier query returns garbage: http://social.msdn.microsoft.com/Forums/en-US/netfxbcl/thread/967654b4-56d7-4617-8de7-94b47b903a88/

Microsoft MSDN. (2008). *MSDN*. Ανάκτηση 10 21, 2008, από how can i get visual basic to scan for wireless networks?: http://forums.microsoft.com/MSDN/ShowPost.aspx?PostID=2043509&SiteID=1

Microsoft. (2008). *SQL Server 2005*. Ανάκτηση 04 12, 2008, από http://www.microsoft.com/sqlserver/2005/en/us/default.aspx

Microsoft. (2004, 12 4). *Windows Hardware Developer Central*. Ανάκτηση 6 2008, από Windows Management Instrumentation Extensions to WDM: http://www.microsoft.com/taiwan/whdc/system/pnppwr/wmi/wmi.mspx

Microsoft. (2008). *WMI overview*. Ανάκτηση 08 23, 2008, από Microsoft TechNet: http://www.microsoft.com/technet/scriptcenter/guide/sas_wmi_overview.mspx?mfr=true

Mikrotik. (2008). *Routers & Wireless*. Ανάκτηση 11 2008, από The Dude for Windows: http://www.mikrotik.com/thedude.php

MSDN. (2008). *802.11 RSSI*. Ανάκτηση 04 18, 2008, από 2.2.1.1.13 802.11 RSSI: http://msdn.microsoft.com/en-us/library/cc234011.aspx

NDIS. (2007, 01 20). *NDIS Developer's Reference*. Ανάκτηση από http://www.ndis.com/

Paramvir Bahl, V. N. (2008). *A Software System for Locating Mobile Users: Design, Evaluation, and Lessons.* RADAR.

Proxim. (2008). *Proxim Wireless*. Ανάκτηση 10 2, 2008, από Calculations: System Operating Margin (SOM): http://www.proxim.com/learn/library/calculations/som.aspx

Proxim Wireless. (2008). *Calculations: System Operating Margin (SOM)*. Ανάκτηση 03 12, 2008, από http://www.proxim.com/learn/library/calculations/som.aspx

Qiang Yang, S. J. (2007). Estimating Location Using Wi-Fi. *IEEE ICDM Contest* , Hong Kong University of Science and Technology.

Ramakrishnan, R., & Grherke, J. (2000). *Database Managment Systems, Second Editiion.* McGraw-Hill.

Rappaport, ,. T. (1996). *Wireless Communications: Principles and Practice.* New Jersey: Prentice-Hall Inc.

RFC 2578. (1999). Structure of Management Information Version 2 (SMIv2). Case, et Al,.

RFC 3410. (2002). Applicability Statements for SNMP. Case, et Al,.

RFC 3413. (2002). Simple Network Management Protocol (SNMP) Applications. D. Levi, P. Meyer, B. Stewart.

SolarWinds. (2008). *SolarWinds Wireless*. Ανάκτηση 10 2008, από Orion NPM : http://www.solarwinds.com/products/orion/wireless/features.aspx

Stüber, G. L. (2001). *Principles of mobile communication (2nd ed.).* Atlanta, USA: Kluwer Academic Publishers Norwell, MA, USA.

Welch, G., & Bishop, G. (2006). *An Introduction to the Kalman Filter.* Chapel Hill, NC 27599-3175: University of North Carolina at Chapel Hill, Department of Computer Science.

Wikipedia. (2008, 08 22). *List of WLAN channels*. Ανάκτηση 08 28, 2008, από http://en.wikipedia.org/wiki/List_of_WLAN_channels

Wikipedia. (2008, 09 15). *Wikipedia*. Ανάκτηση 10 02, 2008, από Free-space path loss: http://en.wikipedia.org/wiki/FSPL#Free-space_path_loss_in_decibels

WildPackets Inc. (2002). Converting Signal Strength Percentage to dBm Values. *Executive Summary* .

Βικιπαίδεια. (2008, 08 12). Ανάκτηση 10 12, 2008, από Άργος ο Πανόπτης: http://el.wikipedia.org/wiki/%CE%86%CF%81%CE%B3%CE%BF%CF%82_%CE%BF_%CE%A0%CE%B1%CE%BD%CF%8C%CF%80%CF%84%CE%B7%CF%82

Βιολέττας, Γ. (2008). *Wi-Fi Positioning.* Θεσσαλονίκη: Πανεπιστήμιο Μακεδονίας.

IEEE. (2007, Ξθνε 12). Wireless LAN Medium Access Control (MAC). *IEEE Std 802.11™-2007* , σ. 489.

Φούσκας Γεώργιος - Ε.Α.Π. (2002). Βασικά Ζητήματα Δικτύων Η/Υ. Στο *Τόμος Β', Ψηφιακές Επικοινωνίες.* Πάτρα: Ελληνικό ΑΝοιχτό Πανεπιστήμιο.

Παράρτημα

Το μέγεθος του πηγαίου κώδικα της εφαρμογής NetArgus είναι περίπου 3.250 γραμμές. Στις ενότητες των παραρτημάτων που ακολουθούν παραθέτουμε ορισμένα κομμάτια τα οποία θεωρούμε κρίσιμα και ουσιώδη στην λειτουργία της εφαρμογής.

Α. Πηγαίος Κώδικας Αλγορίθμου Positioning

```
        // POSITIONING ALGORITHM
        //
=====================================================================

        public static long Factorial(long number)
        {
            long factorial = 1;

            for(long i = number; i >= 1; i--)
            {
                factorial *= i;
            }
            return factorial;
        }

        // Calculate d distance
        private double d(double dbm)
        {
            double S = Convert.ToDouble(txtS.Text);
            double f = Convert.ToDouble(txtf.Text);
            double dx = Math.Pow(10,((S+dbm-
20*Math.Log10(f)+147.56)/20));
            return (dx);
        }

        // Solve xy system
        private double[] xy(double xa, double ya, double dbma, double
xb, double yb, double dbmb, double xc, double yc, double dbmc)
        {
            xa = xa / rate;
            xb = xb / rate;
            xc = xc / rate;

            ya = ya / rate;
            yb = yb / rate;
            yc = yc / rate;
```

```
            double dap = Convert.ToDouble(Math.Pow(d(dbma), 2));
            double dbp = Convert.ToDouble(Math.Pow(d(dbmb), 2));
            double dcp = Convert.ToDouble(Math.Pow(d(dbmc), 2));
            double xap = Math.Pow(xa, 2);
            double xbp = Math.Pow(xb, 2);
            double xcp = Math.Pow(xc, 2);
            double yap = Math.Pow(ya, 2);
            double ybp = Math.Pow(yb, 2);
            double ycp = Math.Pow(yc, 2);

            double K1 = dap - dbp - xap - yap + xbp + ybp;
            double K2 = dap - dcp - xap - yap + xcp + ycp;
            double M1 = 2 * (yb - ya);
            double M2 = 2 * (xc - xa);
            double N1 = 2 * (xb - xa);
            double N2 = 2 * (yc - ya);

            double y = (K2 / N2 - ((M2 / N2) * (K1 / N1))) / (1 -
((M2 / N2) * (M1 / N1)));
            double x = (K1 / N1) - (M1 / N1) * y;

            double[] results = new double[2];
            results[0] = x*rate;
            results[1] = y*rate;
            return (results);
        }

        // Calculate all posible points
        private double[,] CalcPoints(String IP)
        {
            this.snrTTableAdapter.Fill(this.netArgusDataSet.SnrT);
            DataRow[] SNRRows, a, b, c;
            String strExpr;

            // Filter clients
            strExpr = "clientIP=\'" + IP + "\'";
            strExpr = strExpr + " AND snr>=30 AND snr<=90";
            SNRRows =
this.netArgusDataSet.Tables["SnrT"].Select(strExpr);

            long Size = Factorial(SNRRows.GetUpperBound(0) + 1) /
(Factorial(SNRRows.GetUpperBound(0) + 1 - 3) * 6);
            double[,] points = new double[2, Size];
            int pi = 0;

            for (int ai = 0; ai < SNRRows.GetUpperBound(0); ai++)
            {
                for (int bi = ai + 1; bi <= SNRRows.GetUpperBound(0);
bi++)
                {
                    for (int ci = bi + 1; ci <=
SNRRows.GetUpperBound(0); ci++)
                    {
                        strExpr = "ip=\'" +
SNRRows[ai]["stationIp"].ToString() + "\'";
                        a =
this.netArgusDataSet.Tables["DeviceT"].Select(strExpr);
```

```
                        int xa =
Convert.ToInt32(a[0]["x"].ToString());
                        int ya =
Convert.ToInt32(a[0]["y"].ToString());
                        int dbma =
Convert.ToInt32(SNRRows[ai]["snr"].ToString());

                        strExpr = "ip=\'" +
SNRRows[bi]["stationIp"].ToString() + "\'";
                        b =
this.netArgusDataSet.Tables["DeviceT"].Select(strExpr);
                        int xb =
Convert.ToInt32(b[0]["x"].ToString());
                        int yb =
Convert.ToInt32(b[0]["y"].ToString());
                        int dbmb =
Convert.ToInt32(SNRRows[bi]["snr"].ToString());

                        strExpr = "ip=\'" +
SNRRows[ci]["stationIp"].ToString() + "\'";
                        c =
this.netArgusDataSet.Tables["DeviceT"].Select(strExpr);
                        int xc =
Convert.ToInt32(c[0]["x"].ToString());
                        int yc =
Convert.ToInt32(c[0]["y"].ToString());
                        int dbmc =
Convert.ToInt32(SNRRows[ci]["snr"].ToString());

                        double[] point = xy(xa, ya, dbma, xb, yb,
dbmb, xc, yc, dbmc);
                        points[0, pi] = point[0];
                        points[1, pi] = point[1];
                        pi++;
                    }
                }
            }
            return (points);
        }

        // Find Mean point algorithm
        private int[] MeanPoint(double[,] points)
        {
            int[] xy = new int[2];
            double x = 0, y = 0;

            for (int i = 0; i < points.Length/2; i++)
            {
                x += points[0, i];
                y += points[1, i];
            }
            xy[0] = Convert.ToInt32(x / (points.Length / 2));
            xy[1] = Convert.ToInt32(y / (points.Length / 2));

            return xy;
        }

        // For all mobile devices find position
```

```
        private void Positioning()
        {

this.deviceTTableAdapter.Fill(this.netArgusDataSet.DeviceT);
            Application.DoEvents();

            for (int i = 0; i <
this.netArgusDataSet.Tables["DeviceT"].Rows.Count; i++) // All
clients devices
            {
                if
(this.netArgusDataSet.Tables["DeviceT"].Rows[i]["parent"].ToString().
Trim() != "") //Client not station
                {
                    double[,] points =
CalcPoints(this.netArgusDataSet.Tables["DeviceT"].Rows[i]["ip"].ToStr
ing());
                    if (points.Length > 0)
                    {
                        int[] xy = MeanPoint(points);

                        // Filtering
                        if (chkFilter.Checked == true)
                        {
                            // Limit Filter
                            if (xy[0] > 800) xy[0] = 800;
                            if (xy[0] < 0) xy[0] = 0;
                            if (xy[1] > 600) xy[1] = 600;
                            if (xy[1] < 0) xy[1] = 0;
                            // History Filter
                            int xdif =
Math.Abs(Convert.ToInt32(this.netArgusDataSet.Tables["DeviceT"].Rows[
i]["x"]) - xy[0]);
                            int ydif =
Math.Abs(Convert.ToInt32(this.netArgusDataSet.Tables["DeviceT"].Rows[
i]["y"]) - xy[1]);
                            if (xdif > 200)
                            {
                                xy[0] =
Convert.ToInt32(this.netArgusDataSet.Tables["DeviceT"].Rows[i]["x"]);
                            }
                            if (ydif > 200)
                            {
                                xy[1] =
Convert.ToInt32(this.netArgusDataSet.Tables["DeviceT"].Rows[i]["y"]);
                            }
                        }
                        Global.sql("UPDATE DeviceT SET x = \'" +
xy[0].ToString() + "\', y = \'" + xy[1].ToString() + "\' where ip=\'"
+ this.netArgusDataSet.Tables["DeviceT"].Rows[i]["ip"].ToString() +
"\'");
                    }
                }
            }
        }
// END POSITIONING ALGORITHM
//=================================================================
```

Β. Πηγαίος Κώδικας για επικοινωνία με SNMP

Για την επικοινωνία μέσω SNMP υλοποιήθηκε η μέθοδος runSNMP η οποία ορίζει ανρικέιμενα τύπου SNMPObject και SNMPAgent για την επικεινωνία με τις δίαφορες SNMP συσκευές του δικτύου.

```
private void runSNMP()
        {

this.deviceTTableAdapter.Fill(this.netArgusDataSet.DeviceT);
            Application.DoEvents();

            string StationName = "", StationMAC = "",
StationClientsNumbers = "";
            string ClientIP = "", ClientName = "", ClientMAC = "",
ClientSNR = "";
            for (int i = 0; i <
this.netArgusDataSet.Tables["SnmpT"].Rows.Count; i++)
            {

switch(this.netArgusDataSet.Tables["SnmpT"].Rows[i]["description"].To
String().Trim())
                {
                    case "Station Name":
                        StationName =
this.netArgusDataSet.Tables["SnmpT"].Rows[i]["oid"].ToString().Trim()
;
                        break;
                    case "Station MAC":
                        StationMAC =
this.netArgusDataSet.Tables["SnmpT"].Rows[i]["oid"].ToString().Trim()
;
                        break;
                    case "Station Clients Number":
                        StationClientsNumbers =
this.netArgusDataSet.Tables["SnmpT"].Rows[i]["oid"].ToString().Trim()
;
                        break;
                    case "Client IP":
                        ClientIP =
this.netArgusDataSet.Tables["SnmpT"].Rows[i]["oid"].ToString().Trim()
;
                        break;
                    case "Client MAC":
                        ClientMAC =
this.netArgusDataSet.Tables["SnmpT"].Rows[i]["oid"].ToString().Trim()
;
                        break;
                    case "Client Name":
                        ClientName =
this.netArgusDataSet.Tables["SnmpT"].Rows[i]["oid"].ToString().Trim()
;
                        break;
```

```
                    case "Client SNR":
                        ClientSNR =
this.netArgusDataSet.Tables["SnmpT"].Rows[i]["oid"].ToString().Trim()
;
                        break;
                }
            }

            for (int i = 0; i <
this.netArgusDataSet.Tables["DeviceT"].Rows.Count; i++)
            {
                if
(this.netArgusDataSet.Tables["DeviceT"].Rows[i]["parent"].ToString().
Trim() == "" &&
Convert.ToInt32(this.netArgusDataSet.Tables["DeviceT"].Rows[i]["snmp"
]) == 1)
                {
                    try
                    {
                        SNMPAgent myAgent = new
SNMPAgent(this.netArgusDataSet.Tables["DeviceT"].Rows[i]["ip"].ToStri
ng().Trim(),

this.netArgusDataSet.Tables["DeviceT"].Rows[i]["login"].ToString().Tr
im(),

this.netArgusDataSet.Tables["DeviceT"].Rows[i]["password"].ToString()
.Trim());

                        // Get Station Name
                        SNMPObject rstName = new
SNMPObject(StationName);  //"1.3.6.1.2.1.1.5.0"
                        String sName =
rstName.getSimpleValue(myAgent);

                        // Get Station MAC
    SNMPObject rstName = new SNMPObject(StationMAC);
//"1.3.6.1.2.1.2.2.1.6"
    String sName = rstName.getSimpleValue(myAgent);

                        // Get client numbers
                        SNMPObject rstClientNum = new
SNMPObject(StationClientsNumbers); //"1.3.6.1.4.1.11898.2.1.33.3.0"
                        int Cnum =
Convert.ToInt32(rstClientNum.getSimpleValue(myAgent));

                        // Find Clients
                        for (int c = 1; c <= Cnum; c++)
                        {
                            // Client IP
                            SNMPObject myRequest = new
SNMPObject(ClientIP + c.ToString());
//"1.3.6.1.4.1.11898.2.1.33.1.1.3."
                            String cIP =
myRequest.getSimpleValue(myAgent);

                            // Client Mac
```

```
                                // myRequest = new SNMPObject(ClientMAC +
c.ToString());  //"1.3.6.1.4.1.11898.2.1.33.1.1.2."
                                // String cMac =
myRequest.getSimpleValue(myAgent);

                                // Client Name
                                myRequest = new SNMPObject(ClientName +
c.ToString()); //"1.3.6.1.4.1.11898.2.1.33.1.1.5."
                                String cName =
myRequest.getSimpleValue(myAgent);
                                // SNR
                                myRequest = new SNMPObject(ClientSNR +
c.ToString());  // "1.3.6.1.4.1.11898.2.1.33.1.1.20."
                                int cSNR =
Convert.ToInt32(myRequest.getSimpleValue(myAgent));

                                Global.sql("UPDATE DeviceT SET parent =
\'" +
this.netArgusDataSet.Tables["DeviceT"].Rows[i]["ip"].ToString().Trim(
) + "\' where ip=\'" + cIP + "\'");
                                    }
                            }
                            catch
                            {

                            }
                        }// end if
                    } // end for
                }




// SNMP CLASS
using System;
using System.Collections;
using System.Diagnostics;
using System.IO;
using System.Reflection;
using System.Threading;
using Org.Snmp.Snmp_pp;
using RFC1157;

namespace SNMPDll
{


//Κλάση SNMP Object
      public class SNMPObject
      {

            string OID;
            string description = "";
            string myValue = null;
            string fullName = null;
            string myType = null;

            // Constractor 1
```

```
        public SNMPObject(string OID)
        {
            this.OID = OID;
        }

        // Constractor 2
        public SNMPObject(string ID, Mib _mib)
        {
            if (char.IsDigit(ID,0))
            {
                this.OID = ID;
            }
            else
            {
                this.OID = _mib.getOID(ID);
            }

            this.description = _mib.getDescription(OID);
            this.fullName = _mib.getFullName(OID);
            string aaa;

        }

        // Παίρνει τιμή από το SNMP object
        public Hashtable getValue(SNMPAgent myAgent)
        {
            Hashtable ht = myAgent.getValue(this);
            myValue = (string)ht["value"];
            myType = (string)ht["type"];
            return ht;
        }

        // Παίρνει τιμή από το SNMP
        public string getSimpleValue(SNMPAgent myAgent)
        {
            if (myValue == null)
            {
                Hashtable ht = myAgent.getValue(this);
                myValue = (string)ht["value"];
                myType = (string)ht["type"];
            }
            return myValue;
        }

        public string getTypeString()
        {
            if (myType == null)
            {
                throw new Exception("Πρέπει πρώτα να ορίσεις
SNMP agent");
            }
            else return myType;
        }


        public SNMPOIDType getType(SNMPAgent myAgent)
        {
            if (myType == null)
```

```
                {
                        this.getTypeString(myAgent);
                }
                return this.getType();

        }

        public SNMPOIDType getType()
        {
                if (myType == null)
                {
                        throw new Exception("Πρέπει πρώτα να ορίσεις
SNMP agent");
                }
                string aaa = SNMPOIDType.OctetString.ToString();
                if (myType == SNMPOIDType.Counter32.ToString())
return SNMPOIDType.Counter32;
                else if (myType == SNMPOIDType.Gauge32.ToString())
return SNMPOIDType.Gauge32;
                else if (myType == SNMPOIDType.Int.ToString())
return SNMPOIDType.Int;
                else if (myType ==
SNMPOIDType.IpAddress.ToString()) return SNMPOIDType.IpAddress;
                else if (myType ==
SNMPOIDType.OctetString.ToString()) return SNMPOIDType.OctetString;
                else if (myType == SNMPOIDType.Oid.ToString())
return SNMPOIDType.Oid;
                else if (myType ==
SNMPOIDType.TimeTicks.ToString()) return SNMPOIDType.TimeTicks;
                else throw new Exception("ΛΑΘΟΣ τύπος !!!");

        }


        public string getFullName()
        {
                if (fullName == null)
                {
                        throw new Exception("ERROR");
                }
                return fullName;
        }


        public string getFullName(Mib _mib)
        {
                if (fullName == "")
                {
                        fullName = _mib.getFullName(this.OID);
                }
                return fullName;
        }


        public string getOID()
        {
                return OID;
        }
```

```
            public string getDescription()
            {
                  return description;
            }


      }


//Κλάση SNMP Agent

      public class SNMPAgent
      {

            string IPAddress = "";
            string communityRead = "";
            string communityWrite = "";

            public SNMPAgent(string IPAddress, string communityRead,
string communityWrite)
            {
                  this.IPAddress = IPAddress;
                  this.communityRead = communityRead;
                  this.communityWrite = communityWrite;
            }

            public SNMPAgent(string IPAddress)
            {
                  this.IPAddress = IPAddress;
                  this.communityRead = "public";
                  this.communityWrite = "public";
            }


            public Hashtable getValue(string fullName, Mib myMib)
            {
                  return this.getValue(new
SNMPObject(myMib.getOID(fullName)));
            }

            public void setValue(SNMPObject mySNMPObject, SNMPOIDType
myType, int myValue)
            {
                  this.setValue(mySNMPObject, myType,
myValue.ToString());
            }

            public string getIPAddress() {return IPAddress;}

            public string getCommunityRead() {return communityRead;}

            public string getCommunityWrite() {return
communityWrite;}
      }

}
```

Γ. Πηγαίος Κώδικας NetArgus Viewer

Δημιουργεί δυναμικά αντικείμενα στην γραφική διεπαφή τα οποία τα χειρίζεται σε συνεργασία με την βάση δεδομένων.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Net;
using System.Net.NetworkInformation;

namespace NetArgus
{
    public partial class frmMain : Form
    {
        private int offset = 25;
        private System.Windows.Forms.Label[] lblDevices;
        private System.Drawing.Pen myPen;
        private System.Drawing.Graphics formGraphics;

        public frmMain()
        {
            InitializeComponent();
        }

        private void frmMain_Load(object sender, EventArgs e)
        {
            try
            {
                this.deviceTTableAdapter.Connection.ConnectionString
= Global.ChangeDB();

this.deviceTTableAdapter.Fill(this.netArgusDataSet.DeviceT);
            }
            catch (Exception)
            {
                    new frmConfig().ShowDialog(this);
                    frmMain_Load(sender, e);
            }

            try
            {
                if (NetArgus.Properties.Settings.Default.MapLocation
!= "")
                {
                    picNetMap.ImageLocation =
NetArgus.Properties.Settings.Default.MapLocation;
                    picNetMap.Load();
                }
            }
            catch (Exception) { }
```

```
            tsbIP.Checked =
NetArgus.Properties.Settings.Default.chkIP;
            tsbHost.Checked =
NetArgus.Properties.Settings.Default.chkHost;
            tsbMac.Checked =
NetArgus.Properties.Settings.Default.chkMac;
            tmrRefresh.Interval =
Convert.ToInt32(Convert.ToDouble(NetArgus.Properties.Settings.Default
.Refresh) * 1000);

            String strHostName = Dns.GetHostName();
            Text = "NetArgus Viewer - [running on " + strHostName +
"]";

            formGraphics = this.picNetMap.CreateGraphics();
            myPen = new
System.Drawing.Pen(NetArgus.Properties.Settings.Default.Color);
            myPen.Width = 1;

            try
            {
                drawDevices();
                drawLabels();
            }
            catch(Exception){ }
        }

        private void frmMain_FormClosing(object sender,
FormClosingEventArgs e)
        {
            NetArgus.Properties.Settings.Default.Save();

            myPen.Dispose();
            formGraphics.Dispose();
        }

        private void tmrRefresh_Tick(object sender, EventArgs e)
        {

this.deviceTTableAdapter.Fill(this.netArgusDataSet.DeviceT);
            for (int i = 0; i <
this.netArgusDataSet.Tables["DeviceT"].Rows.Count; i++)
            {
                lblDevices[i].Left =
Convert.ToInt32(this.netArgusDataSet.Tables["DeviceT"].Rows[i]["x"].T
oString()) - offset;
                lblDevices[i].Top =
Convert.ToInt32(this.netArgusDataSet.Tables["DeviceT"].Rows[i]["y"].T
oString()) - offset;
            }
            drawDeviceLines(1);
        }

        private int mygetImage(String type)
        {
            if (type == "Router")
                return 0;
            if (type == "Access Point")
```

```
                return 1;
            if (type == "Antenna")
                return 2;
            if (type == "PC")
                return 3;
            if (type == "Laptop")
                return 4;
            if (type == "Printer")
                return 5;
            if (type == "Wireless Client")
                return 6;
            return 0;
        }

        private void drawDevices(){
            // Create new device
            lblDevices = new
System.Windows.Forms.Label[this.netArgusDataSet.Tables["DeviceT"].Row
s.Count];

            for (int i = 0; i <
this.netArgusDataSet.Tables["DeviceT"].Rows.Count; i++)
            {
                //Properties
                lblDevices[i] = new System.Windows.Forms.Label();
                lblDevices[i].Parent = picNetMap;
                lblDevices[i].ImageList = imgLDevice;
                lblDevices[i].ImageIndex =
mygetImage(this.netArgusDataSet.Tables["DeviceT"].Rows[i]["type"].ToS
tring());
               // lblDevices[i].BackColor = Color.Transparent;
                lblDevices[i].BackColor = Color.FromArgb(50,
Color.Green);
                lblDevices[i].ForeColor =
NetArgus.Properties.Settings.Default.Color;
                lblDevices[i].BorderStyle = BorderStyle.Fixed3D;
                lblDevices[i].Width = 50;
                lblDevices[i].Height = 50;
                lblDevices[i].Font = new Font(Font.FontFamily,
lblDevices[i].Font.Size - 3);
                lblDevices[i].TextAlign =
System.Drawing.ContentAlignment.BottomCenter;
                lblDevices[i].Left =
Convert.ToInt32(this.netArgusDataSet.Tables["DeviceT"].Rows[i]["x"].T
oString()) - offset;
                lblDevices[i].Top =
Convert.ToInt32(this.netArgusDataSet.Tables["DeviceT"].Rows[i]["y"].T
oString()) - offset;
                // Events
                lblDevices[i].MouseDown += new
MouseEventHandler(lblDevices_MouseDown);
                lblDevices[i].MouseUp += new
MouseEventHandler(lblDevices_MouseUp);
                lblDevices[i].MouseMove += new
MouseEventHandler(lblDevices_MouseMove);
                lblDevices[i].Visible = true;
            }
        }
```

```
        private void drawDeviceLines(int loadpic)
        {
            if (loadpic == 1)
            {

this.deviceTTableAdapter.Fill(this.netArgusDataSet.DeviceT);
                picNetMap.ImageLocation =
NetArgus.Properties.Settings.Default.MapLocation;
                Application.DoEvents();
            }

            for (int i = 0; i <
this.netArgusDataSet.Tables["DeviceT"].Rows.Count; i++)
            {
                if
(this.netArgusDataSet.Tables["DeviceT"].Rows[i]["parent"].ToString().
Trim() == "")
                {
                    for (int j = 0; j <
this.netArgusDataSet.Tables["DeviceT"].Rows.Count; j++)
                    {
                        if
(this.netArgusDataSet.Tables["DeviceT"].Rows[j]["parent"].ToString().
Trim() ==

this.netArgusDataSet.Tables["DeviceT"].Rows[i]["ip"].ToString().Trim(
))
                        {
                            int x =
Convert.ToInt32(this.netArgusDataSet.Tables["DeviceT"].Rows[j]["x"].T
oString());
                            int y =
Convert.ToInt32(this.netArgusDataSet.Tables["DeviceT"].Rows[j]["y"].T
oString());
                            int px =
Convert.ToInt32(this.netArgusDataSet.Tables["DeviceT"].Rows[i]["x"].T
oString());
                            int py =
Convert.ToInt32(this.netArgusDataSet.Tables["DeviceT"].Rows[i]["y"].T
oString());
                            formGraphics.DrawLine(myPen, x , y , px ,
py );
                        }
                    }
                }
            }
        }

        private void drawLabels()
        {

this.deviceTTableAdapter.Fill(this.netArgusDataSet.DeviceT);
            for (int i = 0; i <
this.netArgusDataSet.Tables["DeviceT"].Rows.Count; i++)
            {
                lblDevices[i].Text = "";
                if (NetArgus.Properties.Settings.Default.chkIP)
```

```
                    lblDevices[i].Text +=
this.netArgusDataSet.Tables["DeviceT"].Rows[i]["ip"].ToString()+"\n";
                if (NetArgus.Properties.Settings.Default.chkHost)
                    lblDevices[i].Text +=
this.netArgusDataSet.Tables["DeviceT"].Rows[i]["name"].ToString() +
"\n";
                if (NetArgus.Properties.Settings.Default.chkMac)
                    lblDevices[i].Text +=
this.netArgusDataSet.Tables["DeviceT"].Rows[i]["mac"].ToString() +
"\n";
                lblDevices[i].Refresh();
            }
        }

         // Δύναμικό event για picNetMap
        private void picNetMap_Paint(object sender,
System.Windows.Forms.PaintEventArgs e)
        {
            drawDeviceLines(0);
        }

        // DRAG DEVICES
        //
============================================================
        private bool isDragging = false;
        private int clickOffsetX, clickOffsetY;
        private int selectedDevice = -1;

        // Start dragging.
        private void lblDevices_MouseDown(System.Object
sender,System.Windows.Forms.MouseEventArgs e)
        {
            clickOffsetX = e.X ;
            clickOffsetY = e.Y;
            // Find which device was pressed
            for (int i = 0; i <
this.netArgusDataSet.Tables["DeviceT"].Rows.Count; i++)
            {
                if (sender.Equals(lblDevices[i]))
                {
                    if (e.Button == MouseButtons.Left)
                    {
                        isDragging = true;
                        selectedDevice = i;
                        lblDevices[i].ImageIndex = 7;
                        break;
                    }
                    else
                    {
                        lblInfoType.Text =
this.netArgusDataSet.Tables["DeviceT"].Rows[i]["type"].ToString();
                        lblInfoIP.Text = "IP: " +
this.netArgusDataSet.Tables["DeviceT"].Rows[i]["ip"].ToString();
                        lblInfoMAC.Text = "MAC: " +
this.netArgusDataSet.Tables["DeviceT"].Rows[i]["mac"].ToString();
                        lblInfoName.Text = "Host Name: " +
this.netArgusDataSet.Tables["DeviceT"].Rows[i]["name"].ToString();
```

```
                        lblInfoPosition.Text = "Position: x=" +
this.netArgusDataSet.Tables["DeviceT"].Rows[i]["x"].ToString() + "
y=" + this.netArgusDataSet.Tables["DeviceT"].Rows[i]["y"].ToString();
                        lblUserData.Text = "";
                        lblCargo.Text = "";

if(this.netArgusDataSet.Tables["DeviceT"].Rows[i]["userdata"].ToStrin
g()!="")
                            lblUserData.Text = "Client User: " +
this.netArgusDataSet.Tables["DeviceT"].Rows[i]["userdata"].ToString()
;
                        if
(this.netArgusDataSet.Tables["DeviceT"].Rows[i]["cargo"].ToString()
!= "")
                            lblCargo.Text = "Cargo:
"+this.netArgusDataSet.Tables["DeviceT"].Rows[i]["cargo"].ToString();
                        pnlInfo.Left =
Convert.ToInt32(this.netArgusDataSet.Tables["DeviceT"].Rows[i]["x"].T
oString())-(pnlInfo.Width/2)+offset;
                        pnlInfo.Top =
Convert.ToInt32(this.netArgusDataSet.Tables["DeviceT"].Rows[i]["y"].T
oString()) - (pnlInfo.Height / 2) + offset;
                        if (pnlInfo.Left < 0)
                            pnlInfo.Left = 0;
                        if (pnlInfo.Left > 800 - pnlInfo.Width)
                            pnlInfo.Left = 800 - pnlInfo.Width;
                        if (pnlInfo.Top < 0)
                            pnlInfo.Top = 0;
                        if (pnlInfo.Top > 600 - pnlInfo.Height/2)
                            pnlInfo.Top = 600 - pnlInfo.Height;
                        pnlInfo.Visible = true;
                    }
                }
            }
        }

        // End dragging.
        private void lblDevices_MouseUp(System.Object
sender,System.Windows.Forms.MouseEventArgs e)
        {
            if (isDragging == true && selectedDevice != -1)
            {
                lblDevices[selectedDevice].ImageIndex =
mygetImage(this.netArgusDataSet.Tables["DeviceT"].Rows[selectedDevice
]["type"].ToString());
                isDragging = false;
                Global.sql("UPDATE DeviceT SET x = " +
(lblDevices[selectedDevice].Left+offset).ToString() + ",y=" +
(lblDevices[selectedDevice].Top+offset).ToString() + "where ip=\'" +
this.netArgusDataSet.Tables["DeviceT"].Rows[selectedDevice]["ip"].ToS
tring() + "\'");
                selectedDevice = -1;
                drawDeviceLines(1);
            }
            picNetMap.Cursor = Cursors.Default;
            pnlInfo.Visible = false;
        }
```

```
        // Move the control (during dragging).
        private void lblDevices_MouseMove(System.Object
sender,System.Windows.Forms.MouseEventArgs e)
        {
            if (isDragging == true && selectedDevice!=-1)
            {
                lblDevices[selectedDevice].Left = e.X +
lblDevices[selectedDevice].Left -clickOffsetX;
                lblDevices[selectedDevice].Top = e.Y +
lblDevices[selectedDevice].Top -clickOffsetY;
                int x = lblDevices[selectedDevice].Left + offset;
                int y = lblDevices[selectedDevice].Top + offset;
                this.stbLable.Text = "X = " + x.ToString() + ", Y = "
+ y.ToString();
            }
        }
        // END DRAG DEVICE

        // MENU
        // Exit Application
        private void exitToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            Application.Exit();
        }

        // Load Map
        private void loadMapToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            dlgLoadMap.InitialDirectory = @".";
            dlgLoadMap.Filter = "All files
(*.*)|*.*|*.jpg|*.jpg|*.bmp|*.bmp";
            dlgLoadMap.FilterIndex = 2;
            dlgLoadMap.RestoreDirectory = true;
            if (dlgLoadMap.ShowDialog() == DialogResult.OK)
            {
                String file = dlgLoadMap.FileName;
                picNetMap.ImageLocation = dlgLoadMap.FileName;
                NetArgus.Properties.Settings.Default.MapLocation =
dlgLoadMap.FileName;
            }
        }

        // Load Device Form
        private void manageNetDevicesToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            for (int i = 0; i <
this.netArgusDataSet.Tables["DeviceT"].Rows.Count; i++)
            {
                lblDevices[i].Dispose();
            }
            new frmNetDevices().ShowDialog(this);
            drawDeviceLines(1);
            drawDevices();
            drawLabels();
        }
```

```
        // Load Configuration Form
        private void configNetArgusToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            new frmConfig().ShowDialog(this);
            try
            {
                this.deviceTTableAdapter.Connection.ConnectionString
= Global.ChangeDB();

this.deviceTTableAdapter.Fill(this.netArgusDataSet.DeviceT);
                tmrRefresh.Interval =
Convert.ToInt32(NetArgus.Properties.Settings.Default.Refresh) * 1000;
            }
            catch (Exception)
            {
                configNetArgusToolStripMenuItem_Click(sender, e);
            }
        }

        private void aboutToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            new frmAboutBox().ShowDialog(this);
        }
        // SPEED BUTTONS
        private void tsbLoadMap_Click(object sender, EventArgs e)
        {
            dlgLoadMap.InitialDirectory = @".";
            dlgLoadMap.Filter = "All files
(*.*)|*.*|*.jpg|*.jpg|*.bmp|*.bmp";
            dlgLoadMap.FilterIndex = 2;
            dlgLoadMap.RestoreDirectory = true;
            if (dlgLoadMap.ShowDialog() == DialogResult.OK)
            {
                String file = dlgLoadMap.FileName;
                picNetMap.ImageLocation = dlgLoadMap.FileName;
                NetArgus.Properties.Settings.Default.MapLocation =
dlgLoadMap.FileName;
            }
        }

        private void tsbSelectColor_Click(object sender, EventArgs e)
        {
            colorDlg.ShowDialog();
            NetArgus.Properties.Settings.Default.Color =
colorDlg.Color;
            for (int i = 0; i <
this.netArgusDataSet.Tables["DeviceT"].Rows.Count; i++)
            {
                lblDevices[i].ForeColor =
NetArgus.Properties.Settings.Default.Color;
            }
            myPen.Color = NetArgus.Properties.Settings.Default.Color;
        }

        private void tsbIP_Click(object sender, EventArgs e)
```

```
        {
            if (tsbIP.CheckState ==
System.Windows.Forms.CheckState.Checked)
                NetArgus.Properties.Settings.Default.chkIP = true;
            else
                NetArgus.Properties.Settings.Default.chkIP = false;

            drawLabels();
        }

        private void tsbHost_Click(object sender, EventArgs e)
        {
            if (tsbHost.CheckState ==
System.Windows.Forms.CheckState.Checked)
                NetArgus.Properties.Settings.Default.chkHost = true;
            else
                NetArgus.Properties.Settings.Default.chkHost = false;
            drawLabels();
        }

        private void tsbMac_Click(object sender, EventArgs e)
        {
            if (tsbMac.CheckState ==
System.Windows.Forms.CheckState.Checked)
                NetArgus.Properties.Settings.Default.chkMac = true;
            else
                NetArgus.Properties.Settings.Default.chkMac = false;
            drawLabels();
        }

        private void tsbStartUpdate_Click(object sender, EventArgs e)
        {
            tsbStartUpdate.Visible = false;
            tsbStopUpdate.Visible = true;
            this.stbLable.Text = "Real Time Update in process ...";
            tmrRefresh.Interval =
Convert.ToInt32(Convert.ToDouble(NetArgus.Properties.Settings.Default
.Refresh) * 1000);
            tmrRefresh.Enabled = true;
        }

        private void tsbStopUpdate_Click(object sender, EventArgs e)
        {
            tsbStartUpdate.Visible = true;
            tsbStopUpdate.Visible = false;
            this.stbLable.Text = "Real Time Update stoped";
            tmrRefresh.Enabled = false;
        }
    }
}
```

Δ. Πηγαίος κώδικας εφαρμογής NetArgus Client

Χρησιμοποιεί επικοινωνία με την ασύρματη κάρτα γραφικών, συλέγει της στάθμη σήματος από τους εντός εμβέλειας σταθμούς εκπομπής και ενημερώνει την βάση δεδομένων.

```
        private void StartScanning()
        {
            _interface = (interfaceList.SelectedItem as
NetworkInterface);
            if (null != _interface)
            {
                lock (networks)
                {
                    networks.Clear();
                }
                ResetData();

                _terminate.Reset();
                _scanThread = new Thread(ScanNetworks);
                _scanThread.IsBackground = true;
                _scanThread.Start();
                _viewUpdateTimer.Start();

                UpdateNetworkScanState(true);
            }
        }

        private void UpdateNetworkScanState(bool enabled)
        {
            if (this.InvokeRequired)
            {
                NetworkScanStateHandler handler = new
NetworkScanStateHandler(UpdateNetworkScanState);
                this.Invoke(handler, new object[] { enabled });
            }
            else
            {
                if (enabled)
                {
                    scanButton.Text =
_localizer.GetString("StopScanning");
                    toolStripMenuStart.Text =
_localizer.GetString("StopScanning");
                    toolStripMenuStart.Visible = false;
                    toolStripMenuStop.Visible = true;
                    lblTransfer.Visible = true;
                    lblActive.Text = "Client Activated";
                    lblActive.ForeColor = Color.Green;
                    pic.Visible = true;
                    notifyIcon.Text = "Argus Client is Active";
                    Settings.Default.Start = true;
                }
                else
                {
```

```
                    scanButton.Text =
_localizer.GetString("StartScanning");
                    toolStripMenuStart.Text =
_localizer.GetString("StartScanning");
                    toolStripMenuStart.Visible = true;
                    toolStripMenuStop.Visible = false;
                    lblTransfer.Visible = false;
                    lblActive.Text = "Client Inactive";
                    lblActive.ForeColor = Color.Red;
                    pic.Visible = false;
                    notifyIcon.Text = "Argus Client is Inactive";
                    Settings.Default.Start = false;
                }
                interfaceList.Enabled = !enabled;
                inactiveNetworkTimeoutComboBox.Enabled = !enabled;
                txtDB.Enabled = !enabled;
            }
        }

        private void StopScanning()
        {
            StopScanning(true);
        }

        private void StopScanning(bool killThread) {
            _viewUpdateTimer.Stop();
            Thread.Sleep(150);

            if (killThread)
            {
                _terminate.Set();
                if (!_scanThread.Join(5000))
                {
                    _scanThread.Abort();
                }
            }

            UpdateNetworkScanState(false);
        }

        private void CheckScanStatus() {
            if (_scanThread.ThreadState ==
System.Threading.ThreadState.Stopped) {
                StopScanning();
            }
        }

        private void scanButton_Click(object sender, EventArgs e) {
            if
(scanButton.Text.Equals(_localizer.GetString("StartScanning")))
            {
                StartScanning();
            }
            else
            {
                StopScanning();
            }
        }
```

```
        private void ResetData() {
            scannerGridView.Rows.Clear();
            // Start the network IDs over again
            ScannerNetwork.ResetNextId();
        }

        private void graphSplitter_SplitterMoving(object sender,
SplitterEventArgs e)
        {
            if (e.Y > e.SplitY)
            {
                _autoResize = false;
            }
        }

        private void ScannerForm_Resize(object sender, EventArgs e)
        {
            if (FormWindowState.Minimized == WindowState)
                Hide();
        }

        private void notifyIcon_MouseDoubleClick(object sender,
MouseEventArgs e)
        {
            Show();
            WindowState = FormWindowState.Normal;
        }

        private void
inactiveNetworkTimeoutComboBox_SelectionChangeCommitted(object
sender, EventArgs e)
        {
            switch (inactiveNetworkTimeoutComboBox.SelectedIndex)
            {
                case 0:
                    Settings.Default.InactiveNetworkTimeout =
TimeSpan.Zero;
                    break;
                case 1:
                    Settings.Default.InactiveNetworkTimeout =
TimeSpan.FromMinutes(1);
                    break;

                case 2:
                    Settings.Default.InactiveNetworkTimeout =
TimeSpan.FromMinutes(5);
                    break;

                case 3:
                    Settings.Default.InactiveNetworkTimeout =
TimeSpan.FromMinutes(15);
                    break;

                case 4:
                    Settings.Default.InactiveNetworkTimeout =
TimeSpan.FromMinutes(30);
                    break;
```

```
                case 5:
                    Settings.Default.InactiveNetworkTimeout =
TimeSpan.FromMinutes(60);
                    break;
            }
            clearInactiveNetworks();
        }

        private void interfaceList_SelectionChangeCommitted(object
sender, EventArgs e)
        {
            Settings.Default.NetCard = interfaceList.Text;
        }

        private void btnExit_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void toolStripMenuRestore_Click(object sender,
EventArgs e)
        {
            Show();
            WindowState = FormWindowState.Normal;
        }

        private void toolStripMenuClose_Click(object sender,
EventArgs e)
        {
            this.Close();
        }

        private void toolStripMenuStart_Click(object sender,
EventArgs e)
        {
            if
(toolStripMenuStart.Text.Equals(_localizer.GetString("StartScanning")
))
            {
                StartScanning();
            }
            else
            {
                StopScanning();
            }
        }

        private void toolStripMenuStop_Click(object sender, EventArgs
e)
        {
            if
(toolStripMenuStart.Text.Equals(_localizer.GetString("StartScanning")
))
            {
                StartScanning();
            }
            else
```

```
            {
                StopScanning();
            }
        }

        private String GetMyIP()
        {
            String IP = "";
            String strHostName = Dns.GetHostName();
            // Find host by name
            IPHostEntry iphostentry = Dns.GetHostByName(strHostName);
            foreach (IPAddress ipaddress in iphostentry.AddressList)
            {
                IP = ipaddress.ToString();
                break;
            }
            return (IP);
        }

        private void txtDB_Leave(object sender, EventArgs e)
        {
            Settings.Default.DBLocation = txtDB.Text;
        }

        private void txtUser_Leave(object sender, EventArgs e)
        {
            Settings.Default.UserData = txtUser.Text;
            Global.sql("UPDATE DeviceT SET userdata=\'" +
txtUser.Text + "\' WHERE ip=\'" + GetMyIP() + "\'");
        }

        private void txtCargo_Leave(object sender, EventArgs e)
        {
            Settings.Default.Cargo = txtCargo.Text;
            Global.sql("UPDATE DeviceT SET cargo=\'" + txtCargo.Text
+ "\' WHERE ip=\'" + GetMyIP() + "\'");
        }

    }

        private void updateView(object source, ElapsedEventArgs e) {

            CheckScanStatus();

            if (!_abort && (_lastDrawnTimeStamp <
_lastScanTimeStamp)) {
                _lastDrawnTimeStamp = DateTime.Now;

                DateTime time = DateTime.Now;

                clearInactiveNetworks();
                lock (networks) {

                    foreach (ScannerNetwork network in networks) {

                        DataGridViewRow row =
FindDataRow(network.MacAddress.ToString());
                        if (null != row) {
```

```
                            String stationMac, Signal;
                            stationMac =
network.MacAddress.ToString();

                            row.Cells["channelColumn"].Value =
network.Channel;
                            row.Cells["securityColumn"].Value =
network.Security;
                            row.Cells["typeColumn"].Value =
network.NetworkType;
                            row.Cells["lastSeenColumn"].Value =
network.LastSeenTimeStamp.ToLongTimeString();
                            row.Cells["speedColumn"].Value =
network.Speed;
                            row.Cells["speedColumn"].ToolTipText =
network.SpeedString;
                            // Check to see how much time has passed
since the network was last seen
                            TimeSpan duration = _lastScanTimeStamp -
network.LastSeenTimeStamp;
                            if (duration.Seconds <= 30) {
                                row.Cells["signalColumn"].Value =
network.Signal.ToString();
                                Signal =
Math.Abs(network.Signal).ToString();
                            }
                            else {
                                // After 30 seconds, set the RSSI
value to unknown
                                row.Cells["signalColumn"].Value =
"0";
                                Signal = "0";
                            }

                            Global.sql("UPDATE SnrT SET snr = \'" +
Signal + "\' where stationMac=\'" + stationMac + "\' and clientIP=\'"
+ clientIP + "\'");
                        }
                        else {
                            // add the new network to the data grid
                            row = new DataGridViewRow();
                            row.CreateCells(scannerGridView,
network.Data);

                            // Ryan: The column names aren't working
here... ugh@!
                            row.Cells[1].Style.BackColor =
network.LineColor;
                            row.Cells[1].Style.SelectionBackColor =
network.LineColor;
                            scannerGridView.Rows.Add(row);
                            updateVerticalHeights();
                            row.Selected = false;

                        }
                    }
                }
```

```
                System.ComponentModel.ListSortDirection sortDir =
System.ComponentModel.ListSortDirection.Ascending;
                if (scannerGridView.SortOrder ==
SortOrder.Descending)
                    sortDir =
System.ComponentModel.ListSortDirection.Descending;
                scannerGridView.Sort(scannerGridView.SortedColumn,
sortDir);
            }
        }
```