Μάιος 2013

Ατομική Διπλωματική Εργασία

ΠΕΙΡΑΜΑΤΙΚΗ ΑΠΟΤΙΜΗΣΗ ΠΛΑΙΣΙΟΥ ΓΕΩΤΟΠΟΘΕΣΙΑΣ ΜΕ ΕΓΓΥΗΣΕΙΣ ΙΔΙΩΤΙΚΟΤΗΤΑΣ ΚΑΙ ΜΕΓΑΛΑ ΔΕΔΟΜΕΝΑ

Πασχάλης Μπέης

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2013

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΕΙΡΑΜΑΤΙΚΗ ΑΠΟΤΙΜΗΣΗ ΠΛΑΙΣΙΟΥ ΓΕΩΤΟΠΟΘΕΣΙΑΣ ΜΕ ΕΓΓΥΗΣΕΙΣ ΙΔΙΩΤΙΚΟΤΗΤΑΣ ΚΑΙ ΜΕΓΑΛΑ ΔΕΔΟΜΕΝΑ

Πασχάλης Μπέης

Επιβλέπων Καθηγητής
Δρ. Δημήτρης Ζεϊναλιπούρ

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2013

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω τον Δρ. Αντρέα Κωνσταντινίδη για την άψογη συνεργασία που είχα μαζί του σε όλους τους τομείς, καθ' όλη την διάρκεια της χρονιάς και την πολύτιμη καθοδήγηση που μου πρόσφερε στην εκπόνηση αυτής της εργασίας. Επίσης θα ήθελα να ευχαριστήσω τους μεταπτυχιακούς φοιτητές κ. Γιώργο Λάρκου, και κ. Κώστα Κωνσταντίνου για τις πολύτιμες τους απόψεις σε τεχνικά ζητήματα που προέκυψαν κατά την έκβαση της διπλωματικής μου εργασίας.

Ακόμη θα ήθελα να ευχαριστήσω τον Δρ. Δημήτρη Ζεϊναλιπούρ, για την ευκαιρία που μου έδωσε να εκπονήσω την Ατομική Διπλωματική Εργασία μαζί του και να έρθω σε επαφή με νέες, ενδιαφέρουσες και περιζήτητες τεχνολογίες, που θα με βοηθήσουν τόσο για την συνέχεια των σπουδών μου σε μεταπτυχιακό επίπεδο, αλλά και σε επαγγελματική ή ακαδημαϊκή καριέρα.

Τέλος θα ήθελα να ευχαριστήσω την οικογένεια μου, που με στηρίζει οικονομικά καθ' όλη την διάρκεια της έκβασης των σπουδών μου, και ευελπιστώ ότι μία μέρα θα είμαι σε θέση να ανταποδώσω την βοήθεια αυτή.

Περίληψη

Τα συστήματα γεωτοποθέτησης που χρησιμοποιούν ως βάση τους τα Ασύρματα Δίκτυα (WiFi) πρόσφατα έτυχαν ιδιαίτερης προσοχής λόγω του ότι το Παγκόσμιο Σύστημα Γεωτοποθέτησης (GPS) δεν είναι διαθέσιμο σε εσωτερικούς χώρους αλλά και λόγω των υψηλών ποσοστών ενέργειας που καταναλώνονται με την χρήση του. Τα ήδη υπάρχοντα συστήματα γεωτοποθέτησης με χρήση Ασύρματων Δικτύων, βασίζονται σε διαδικασίες γεωτοποθέτησης υπολογιζόμενες στην πλευρά του εξυπηρετητή, πράγμα που τις καθιστά εισβολέα στην ιδιωτικότητα των χρηστών.

Ήδη στο εργαστήριο Διαχείρισης Δεδομένων και Συστημάτων [1] (DMSL) υλοποιήθηκε σύστημα που παρέχει υψηλής ακρίβειας γεωτοποθέτηση με Χάρτες Τιμών Ισχύος (radiomaps) βασισμένες στα Ασύρματα Δίκτυα (WiFi). Τέτοιοι χάρτες διατηρούν την ιδιωτικότητα των χρηστών όμως καταναλώνουν αξιοσημείωτη ενέργεια για την μεταφορά τους στις κινητές συσκευές μέσω Ασύρματων Δικτύων (WiFi) ή Κινητών δεδομένων (Mobile Data).

Στην εργασία αυτή, υλοποιήθηκαν και αποτιμήθηκαν δύο καινοτόμες οικογένειες αλγόριθμων που παρέχουν γεωτοποθέτηση με υψηλή ακρίβεια, χαμηλή ενεργειακή κατΦανάλωση και διατήρηση της ιδιωτικότητας των χρηστών στις έξυπνες κινητές συσκευές, επονομαζόμενες VectorMap και Temporal VectorMap [1]. Ο αλγόριθμος VM σχεδιάστηκε για γεωτοποθετήσεις ενός στιγμιότυπου, ενώ ο αλγόριθμος TVM για συνεχόμενες γεωτοποθετήσεις. Και οι δύο αλγόριθμοι επιτρέπουν στον χρήστη να γεωτοποθετηθεί χωρίς να χρειαστεί να μεταφέρει ολόκληρο τον Χάρτη Τιμών Ισχύος από τον εξυπηρετητή. Επιπλέων εξασφαλίζουν ότι η ταυτότητα των χρηστών είναι δυσδιάκριτη ανάμεσα σε τουλάχιστον k-1 άλλους χρήστες. Η αποτίμηση των τεχνικών αυτών, χωρίστηκε σε δύο μέρη. Στο πρώτο μέρος, χρησιμοποιήθηκε προσομοιωτής (simulator). Στο δεύτερο μέρος, έγινε πραγματική υλοποίηση με χρήση βάσης δεδομένων HBase, εξυπηρετητή Apache Tomcat και πραγματικές συσκευές Android. Για την δημιουργία περιβάλλοντος μεγάλων δεδομένων χρησιμοποιήθηκαν υπάρχουσες μετρήσεις στο κτίριο Πληροφορικής του Πανεπιστημίου Κύπρου, του ερευνητικού κέντρου «ΚΟΙΟΣ», αλλά και δημιουργήθηκαν δεδομένα τα οποία εξάχθηκαν από τον οργανισμό CrawDad [2] . Τα δεδομένα αυτά πολλαπλασιάστηκαν ούτως ώστε να δημιουργηθούν ρεαλιστικά κτίρια, τα οποία διασκορπίστηκαν σε όλες τις πόλεις του κόσμου. Τα αποτελέσματα πιστοποίησαν ότι οι αλγόριθμοι VM και TVM μπορούν να προσφέρουν υψηλής ακρίβειας γεωτοποθέτηση, με εγγύηση K ανωνυμίας, και περίπου 80% λιγότερη ενέργεια από ανταγωνιστικές προσεγγίσεις

[1] Data Management and Systems Laboratory, http://www.dmsl.cs.ucy.ac.cy
[2] A Community Resource for Archiving Wireless Data At Dartmouth, http://www.crawdad.org

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Νέα εποχή με τις έξυπνες κινητές συσκευές

Η χρονιά 2011 ήταν ιδιαίτερα σημαντική για το πεδίο της Πληροφορικής, καθώς ο αριθμός των έξυπνων κινητών συσκευών (Smartphones), ξεπέρασε για πρώτη φορά στην ιστορία τον αριθμό όλων των ειδών Προσωπικών Υπολογιστών σε συνδυασμό. (δηλαδή Προσωπικοί Υπολογιστές, Φορητοί Υπολογιστές, Υπολογιστές Δικτύου-NetBooks μαζί), σηματοδοτώντας την έναρξη της εποχής μετά τον Προσωπικό Υπολογιστή [3]. Οι ισχυρές υπολογιστικές ικανότητες σε αυτές τις συσκευές, σε συνδυασμό με τον αυξανόμενο αριθμό των αισθητήρων τους, εισήγαγαν μία άνευ προηγουμένου γκάμα από εφαρμογές (στο Google Play υπάρχουν περισσότερες από 750,000 εφαρμογές με περισσότερες από 48 δισεκατομμύρια εγκαταστάσεις[4]).

Ένα εξαιρετικά σημαντικό πρόβλημα που δεν είχε επιβληθεί μέχρι σήμερα, ήταν αυτό της υψηλής ακρίβειας γεωτοποθέτησης, με χαμηλή ενεργειακή κατανάλωση και διατήρηση της ανωνυμίας, χωρίς την χρήση του Παγκόσμιου Συστήματος Γεωτοποθέτησης (GPS) ή αντίστοιχων προσεγγίσεων (GLONASS). Ένα τέτοιο σύστημα θα επέτρεπε στους χρήστες να πλοηγηθούν και να αλληλοεπιδράσουν (π.χ. με παιχνίδια ή εφαρμογές) σε εσωτερικούς δημόσιους χώρους (π.χ. αεροδρόμια, εμπορικά κέντρα, βιβλιοθήκες κ.α) χωρίς ανησυχίες περί ιδιωτικότητας ή νομοθετικούς φραγμούς, που τώρα υπάρχουν σε υπηρεσίες γεωτοποθέτησης υπολογιζόμενες στην πλευρά του εξυπηρετητή (π.χ. Χάρτες Εσωτερικών Χώρων Google).

Παρότι στην εργασία αυτή η προσοχή στρέφεται ιδιαίτερα στους εσωτερικούς χώρους, οι αλγόριθμοι αυτοί μπορούν εξίσου να εφαρμοστούν σε αστικούς εξωτερικούς χώρους, όπου τα Ασύρματα Δίκτυα (WiFi) έχουν πυκνή παρουσία.

[3] 3 Feb. 2012: Canalys Press Release, http://goo.gl/qg4oB
[4] 15 May 2013: Google I/O, http://goo.gl/6XgXz

Τεχνολογία / πλευρά	Ακρίβεια	Ενέργεια	Ιδιωτικότητα
A-GPS / πελάτη)	≈1m / ακριβές	Κακή	**Καλή**
Cell_ID DB / εξυπηρετητή	≈1000m / τραχύ	**Καλή**	Κακή
Wifi_ID DB / εξυπηρετητή	≈200m / τραχύ	**Καλή**	Κακή
Radiomap / εξυπηρετητή	≈2-4m / ακριβές	**Καλή**	Κακή
Radiomap / πελάτη	≈2-4m / ακριβές	Κακή	**Καλή**
Vectormap / πελάτη	**≈2-4m / ακριβές**	**Καλή**	**Καλή**

Πίνακας 1: *Τεχνολογίες γεωτοποθέτησης σε Έξυπνες Κινητές Συσκευές*

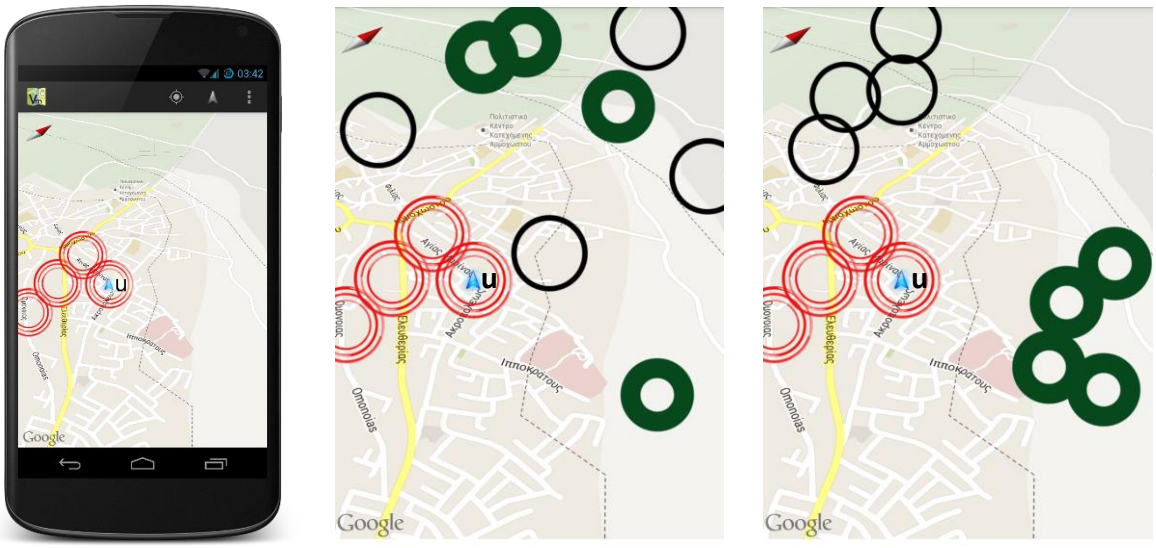
Ο πίνακας 1 παρουσιάζει τις πτυχές που εξερευνήθηκαν σε αυτή την εργασία, καθώς και τα βασικά χαρακτηριστικά της οικογένειας αλγορίθμων Vectormap.

1.2 Ήδη υπάρχουσες υπηρεσίες γεωτοποθέτησης

Ευρέως διαδεδομένα Λειτουργικά Συστήματα σε έξυπνες κινητές συσκευές προσφέρουν ήδη ενσωματωμένη Διεπαφή Προγραμματισμού Εφαρμογών (API) που μπορεί να συνδυάσει δυναμικά τις διαθέσιμες τεχνικές γεωτοποθέτησης. Για παράδειγμα στο λειτουργικό σύστημα Android, παρέχονται οι ακόλουθες δύο επιλογές:

i. ACCESS_FINE_LOCATION: καθορίζει υψηλής ακρίβειας, υψηλής ενεργειακής κατανάλωσης και διατήρησης της ιδιωτικότητας, διαδικασία γεωτοποθέτησης, χρησιμοποιώντας το υποβοηθούμενο Παγκόσμιο Σύστημα Γεωτοποθέτησης (A-GPS)
ii. ACCESS_COARSE_LOCATION: καθορίζει χαμηλής ακρίβειας (200-1000 μέτρα), χαμηλής ενεργειακής κατανάλωσης, και καταπάτησης της ιδιωτικότητας των χρηστών διαδικασία γεωτοποθέτησης που βασίζεται στις πληροφορίες γεωτοποθέτησης της Google, χρησιμοποιώντας Διεπαφή Προγραμματισμού Εφαρμογών (API) τεχνολογίας Ιστού 2.0.

Σε αντίθεση με τα προβλήματα ιδιωτικότητας στις προσεγγίσεις που είναι υλοποιημένες στην πλευρά του εξυπηρετητή, η εφαρμογή Airplace[2] παρέχει υψηλής ακρίβειας και διατήρησης της ιδιωτικότητας γεωτοποθέτηση χρησιμοποιώντας σημεία πρόσβασης Ασύρματων Δικτύων (WiFi). Το Airplace εκμεταλλεύεται τα αποτυπώματα Ισχύων Λαμβανόμενου Σήματος (RSS). Τα αποτυπώματα αυτά είναι μετρήσεις ισχύος, καταγραμμένες από Σημεία Πρόσβασης που βρίσκονται κοντά στον χρήστη, τα οποία είναι αποθηκευμένα σε Χάρτες Τιμών Ισχύος (radiomaps) που καλύπτουν μια περιοχή ενδιαφέροντος. Οι Χάρτες Τιμών Ισχύος πρέπει να



Εικόνα 2: *Γεωτοποθέτηση χρησιμοποιώντας την οικογένεια αλγορίθμων Vectormap. Εικόνα 1: Ο χρήστης μετακινείται σε ένα εσωτερικό χώρο, και χρειάζεται να γνωρίζει την τοποθεσία του χωρίς την χρήση GPS. Εικόνα 2: Ο εξυπηρετητής S, μετά από 4 διαδοχικές γεωτοποθετήσεις με την χρήση του Vectormap αλγόριθμου, μπορεί ξεκάθαρα να βρει την τοποθεσία του χρήστη Εικόνα 3: Ο εξυπηρετητής μετά από 4 διαδοχικές γεωτοποθετήσεις, δεν μπορεί να ξεχωρίσει ποια είναι η πραγματική πορεία που ακολουθεί ο χρήστης (είναι δυσδιάκριτη μεταξύ k-1 άλλων τροχιών)*

μεταφερθούν στις έξυπνες κινητές συσκευές πριν από την διαδικασία γεωτοποθέτησης. Οι χάρτες αυτοί ενδέχεται να είναι πολύ μεγάλοι, για παράδειγμα ο ιστιότοπος WiGLE.net είχε περίπου 1.3 δισεκατομμύρια μοναδικά καταγραμμένων αποτυπωμάτων από εθελοντές που εκτελούσαν διαδικασίες γεωγραφικού εντοπισμού (wardriving) από τον Φεβρουάριο του 2012. Αυτό έχει ως αποτέλεσμα η προσέγγιση που ακολουθείται στο Airplace να οδηγείται στην σπατάλη πολύτιμης ενέργειας, η οποία είναι και περιορισμένη στις έξυπνες κινητές συσκευές, καθώς και εύρους ζώνης δικτύου.

1.3 Αποτίμηση δύο νέων αλγορίθμων

Θα αποτιμηθούν δύο καινοτόμες οικογένειες αλγόριθμων για γεωτοποθέτηση υψηλής ακρίβειας, χαμηλής ενεργειακής κατανάλωσης, και διατήρησης της ιδιωτικότητας των χρηστών σε έξυπνες κινητές συσκευές, επονομαζόμενες Vectormap (VM) και Temporal Vectormap (TVM) αντίστοιχα. Και οι δύο οικογένειες αλγόριθμων τρέχουν στην πλευρά του πελάτη, επιτρέποντας του να γεωτοποθετηθεί χωρίς να χρειάζεται να μεταφέρει ολόκληρο τον Χάρτη Τιμών Ισχύος από τον εξυπηρετητή. Επιπρόσθετα και οι δύο οικογένειες αλγόριθμων εγγυούνται ότι η ταυτότητα ενός χρήστη θα είναι δυσδιάκριτη από τον εξυπηρετητή για

τουλάχιστον k-1 άλλους χρήστες (δηλαδή, διασφαλίζεται k-ανωνυμία [3]), όπου k είναι παράμετρος καθορισμένη από τον χρήστη.

Ο Vectormap αλγόριθμος είναι μια βελτιωμένη έκδοση από προηγούμενη δουλειά [4] στο εργαστήριο Διαχείρισης Δεδομένων και Συστημάτων (DMSL), που παρέχει k-ανωνυμία για γεωτοποθέτηση ενός στιγμιότυπου, χρησιμοποιώντας μια δομή, επονομαζόμενη: πίνακας k-ανωνυμίας Bloom (κAB) [5]. Στην περίπτωση όπου ένας χρήστης χρειαζόταν συνεχή γεωτοποθέτηση (π.χ., όταν ο χρήστης κινείται), ο kAB πίνακας από μόνος του, δεν ήταν επαρκής για να διατηρήσει την ιδιωτικότητα του χρήστη. Για να γίνει αυτό κατανοητό, ας θεωρήσουμε ότι υπάρχει κάποιος χρήστης στην Εικόνα 2 (αριστερά) που κινείται σε κάποιο εσωτερικό χώρο. Ο χρήστης αποσκοπεί στο να αντλήσει την τοποθεσία του με την βοήθεια ενός εξυπηρετητή γεωτοποθέτησης που διατηρεί τον Χάρτη Τιμών Ισχύος του εσωτερικού χώρου. Αν ο χρήστης προκαλέσει τέσσερις διαδοχικές και ανεξάρτητες εκτελέσεις του αλγόριθμου Vectormap, τότε ο εξυπηρετητής θα μπορεί ξεκάθαρα να εξάγει την πραγματική πορεία του (όπως φαίνονται οι διαδοχικοί κύκλοι στην Εικόνα 2 - κέντρο).

Στην δεύτερη οικογένεια αλγορίθμων, την Temporal Vector Map (TVM), που αποτελεί βελτίωση της πρώτης οικογένειας αλγορίθμων, ο χρήστης προκαλεί διαδοχικές αλλά υπό όρους αλληλένδετες εκτελέσεις του Vectormap αλγορίθμου. Ειδικότερα ο χρήστης αφήνει τον εξυπηρετητή να νομίζει ότι μπορεί να είναι σε οποιαδήποτε από τις k τροχιές που διακινούνται στον χώρο (όπως φαίνετε στην Εικόνα 2 δεξιά). Καθώς και οι δύο οικογένειες αλγόριθμων, Vectormap και Temporal Vectormap, μεταφέρουν μόνο ένα μέρος των Χαρτών Τιμών Ισχύος από τον εξυπηρετητή στον πελάτη, υπερτερούν κατά της προσέγγισης όπου οι Χάρτες Τιμών Ισχύος μεταφέρονται εξ' ολοκλήρου στις έξυπνες κινητές συσκευές. Τα σημεία υπεροχής της οικογένειας αλγορίθμων Vectormap αφορούν την κατανάλωση ενέργειας και τον χρόνο εκτέλεσης, ενώ παράλληλα διατηρούν όλα τα υπόλοιπα πλεονεκτήματα της προσέγγισης Airplace, όπως η γεωτοποθέτηση υψηλής ακρίβειας και η διατήρηση της ιδιωτικότητας του χρήστη. Περιληπτικά, οι σημαντικότερες συνεισφορές είναι:

- Αποτίμηση της οικογένειας αλγόριθμων Temporal Vectormap (TVM), η οποία προσφέρει τα ίδια πλεονεκτήματα με την οικογένεια αλγορίθμων Vectormap για συνεχόμενες γεωτοποθετήσεις, χρησιμοποιώντας την ιδέα της k-1 ανωνυμίας για k ψεύτικες τροχιές που κινούνται στον χώρο, με χρήση ενός προσομοιωτή και ρεαλιστικών δεδομένων
- Πειραματική πιστοποίηση της απόδοσης των αλγορίθμων, με εκτενή πειραματική μελέτη, σε προσομοίωση η οποία χρησιμοποιεί δεδομένα τα οποία συλλέχτηκαν τοπικά στο Πανεπιστήμιο Κύπρου.
- Υλοποίηση ενός ολοκληρωμένου συστήματος γεωτοποθέτησης με χρήστη υποδομών μεγάλων δεδομένων και κινητών συσκευών Android

Κεφάλαιο 2

Σχετική Εργασία

2.1 Τεχνικές γεωτοποθέτησης

Η φιλοσοφία γύρω από τις τεχνικές γεωτοποθέτησης ποικίλει, καθώς αξιοποιούνται πολλές τεχνολογίες. Το υποβοηθούμενο Παγκόσμιο Σύστημα Γεωτοποθέτησης (Assisted-GPS), είναι προφανώς καθολικά διαθέσιμο, όμως χαρακτηρίζεται «ακριβό» από πλευράς κατανάλωσης ενέργειας, καθώς επίσης και αρνητικά επηρεαζόμενο από το περιβάλλον (π.χ., συννεφιασμένες μέρες, δάση, στα κέντρα πόλεων, κ.α.). Εκτός από το Παγκόσμιο Σύστημα Γεωτοποθέτησης, η κοινότητα γεωτοποθέτησης [6] εισηγήθηκε πολυάριθμες ιδιόκτητες (proprietary) λύσεις οι οποίες περιλαμβάνουν: υπέρυθρες ακτίνες, Bluetooth, οπτική ή ακουστική ανάλυση, Συχνότητες Ισχύος Αναγνώρισης (RFID), Αδρανείς Μονάδες Μέτρησης, Υπέρ-Ευρεία Ζώνη, Δίκτυα Ασύρματων Αισθητήρων, Ασύρματα Τοπικά Δίκτυα Περιοχής (WLAN) και άλλα· συμπεριλαμβάνοντας τους συνδυασμούς τους σε υβριδικά συστήματα [7]. Οι περισσότερες από αυτές τις τεχνολογίες αν και μπορούν να παραδώσουν υψηλής ακρίβειας γεωτοποθεσία, ως επί το πλείστο απαιτούν την εγκατάσταση και την ρύθμιση ακριβού εξοπλισμού, όπως προσαρμοσμένους πομπούς ή κεραίες, τα οποία είναι εξειδικευμένα για γεωτοποθετήσεις. Αυτές οι τεχνολογίες είναι χρονοβόρες, και ως συνήθως έχουν υψηλά κόστη εγκατάστασης, καθώς επίσης και την ανάγκη να μεταφέρουν οι χρήστες τον επιπλέον εξειδικευμένο εξοπλισμό ούτως ώστε να είναι σε θέση να γεωτοποθετηθούν.

Το σύστημα Airplace [2] εκμεταλλεύεται τις τιμές *Ισχύος Λαμβανόμενου Σήματος* (RSS), που εξάγονται μέσω παθητικής σάρωσης των εκπομπών των πακέτων, τα οποία μεταδίδονται από γειτονικά σημεία πρόσβασης ως μέρος της καθιερωμένης λειτουργίας των Ασύρματων Δικτύων. Για την αντιμετώπιση της πρόκλησης για τις συνθήκες διάδοσης του σήματος σε εσωτερικούς χώρους εξ' αιτίας των πολλών μονοπατιών, αντανακλάσεων και των διαθλάσεων, τα αποτυπώματα Ισχύος Λαμβανόμενου Σήματος (RSS), συλλέγονται εκ των προτέρων στα προκαθορισμένα σημεία αναφοράς. Κάθε αποτύπωμα Ισχύος Λαμβανόμενου Σήματος

CLIENT REQUEST

```
{
  "homeMobileCountryCode": 310,
  "homeMobileNetworkCode": 410,
  "radioType": "gsm",
  "carrier": "Vodafone",
  "cellTowers": [
    {
      "cellId": 42,
      "locationAreaCode": 415,
      "mobileCountryCode": 310,
      "mobileNetworkCode": 410,
      "age": 0,
      "signalStrength": -60,
      "timingAdvance": 5555
    }
    { ... more CellTowers ... }
  ],
  "wifiAccessPoints": [
    {
      ... continued
      "macAddress": "01:23:45:67:89:AB",
      "signalStrength": -65,
      "age": 0,
      "channel": 11,
      "signalToNoiseRatio": 40
    }
    { ... more APs ... }
  ]
}
```

LOCALIZATION SERVER RESPONSE

```
{
    "location": {
    "latitude": 51.0,
    "longitude": -0.1,
    },
    "accuracy": 1200.4,
}
```

Πίνακας 3: *Αίτηση για γεωτοποθέτηση χρησιμοποιώντας το API της Google, και η απάντηση σε μορφή JSON, Το παράδειγμα πάρθηκε από: http://goo.gl/7Ib65*

συσχετίζετε με την αντίστοιχη τοποθεσία αναφοράς και είναι αποθηκεμένο στον Χάρτη Τιμών Ισχύος (Radiomap) ο οποίος καλύπτει μια ολόκληρη περιοχή ενδιαφέροντος.

Ουσιαστικά ο Χάρτης Τιμών Ισχύος είναι μια χαρτογράφηση από τον πολυδιάστατο χώρο των τιμών Ισχύος Λαμβανόμενου Σήματος (RSS), στις φυσικές διαστάσεις.

Δημοφιλής Εργαλειοθήκες Ανάπτυξης Λογισμικού (SDK) για έξυπνες κινητές συσκευές, παρέχουν στις μέρες μας πρόσβαση σε Διεπαφές Προγραμματισμού Εφαρμογών (API) για γεωτοποθέτηση, ούτως ώστε οι προγραμματιστές να μπορούν να εξάγουν τις συντεταγμένες μιας έξυπνης κινητής συσκευής με ένα εύκολο τρόπο, όπως αυτός περιγράφεται κατά την εισαγωγή.

Το κύριο ζήτημα με αυτές τις προσεγγίσεις είναι ότι αποκαλύπτουν την ταυτότητα του χρήστη στην υπηρεσία. Ένα παράδειγμα είναι η αλληλεπίδραση με την Διεπαφή Προγραμματισμού Εφαρμογών γεωτοποθέτησης της Google το οποίο παρουσιάζετε στον πίνακα 3. Όπως μπορούμε να δούμε, ο χρήστης αποκαλύπτει τα σημεία πρόσβασης Ασύρματων Δικτύων και τους Πύργους Κινητής Τηλεφωνίας (Cell Towers) ως μέρος της επερώτησης για γεωτοποθέτηση.

2.2 Διαμοιρασμός δεδομένων με διατήρηση της ιδιωτικότητας

Στις σχεσιακές βάσεις δεδομένων, η k-ανωνυμία ήταν μακρά ερευνημένο πρόβλημα με τις ρίζες του στην διατήρηση της ιδιωτικότητας στον διαμοιρασμό των δεδομένων ιατρικών φακέλων και την έλευση της βάσης δεδομένων «Ιπποκράτης» από την IBM το 2002 [8]. Η ιδιωτικότητα όσο αφορά την τοποθεσία, συνήθως αναφέρετε στο σενάριο όπου κάποιος ιδιοκτήτης δεδομένων θέλει να δημοσιεύσει μέρος των δεδομένων ή να επιτρέψει χωρικές επερωτήσεις στην δική του κινητή βάση δεδομένων.

Εικόνα 4: *Η απόκρυψη (cloaking), κατά την οποία ο εξυπηρετητής γνωρίζει μια ευρύτερη περιοχή (όπως σκιαγραφείτε με γαλάζιο) στην οποία είναι ο χρήστης, χωρίς ωστόσο να μπορεί να γνωρίζει την πραγματική του τοποθεσία*

Για να πετύχει διατήρηση της ιδιωτικότητας, ο ιδιοκτήτης δεδομένων πρέπει πρώτα να «εξυγιάνει» τα δεδομένα που θα μοιραστεί, ούτως ώστε κανείς να μην μπορεί να αντιστοιχήσει κάποια συγκεκριμένη εγγραφή με τα αντίστοιχα δεδομένα ή να μπορεί να εξάγει ευαίσθητες πληροφορίες μέσα από αυτά.

Οι τεχνικές διατήρησης της ιδιωτικότητας για υπηρεσίες γεωτοποθέτησης είναι βασισμένες σε κάποιες από τις παρακάτω έννοιες:

ι) ψεύτικες τοποθεσίες, ιι) χωρική απόκρυψη (cloaking), ιιι) μετασχηματισμοί χώρου και ιν) k-ανωνυμία.

Στην τεχνική των ψεύτικων τοποθεσιών, χρησιμοποιείται ένα σύνολο από ψεύτικες τοποθεσίες για κάθε χρήστη, ούτως ώστε να προστατευθεί η πραγματική του τοποθεσία [9] [10].

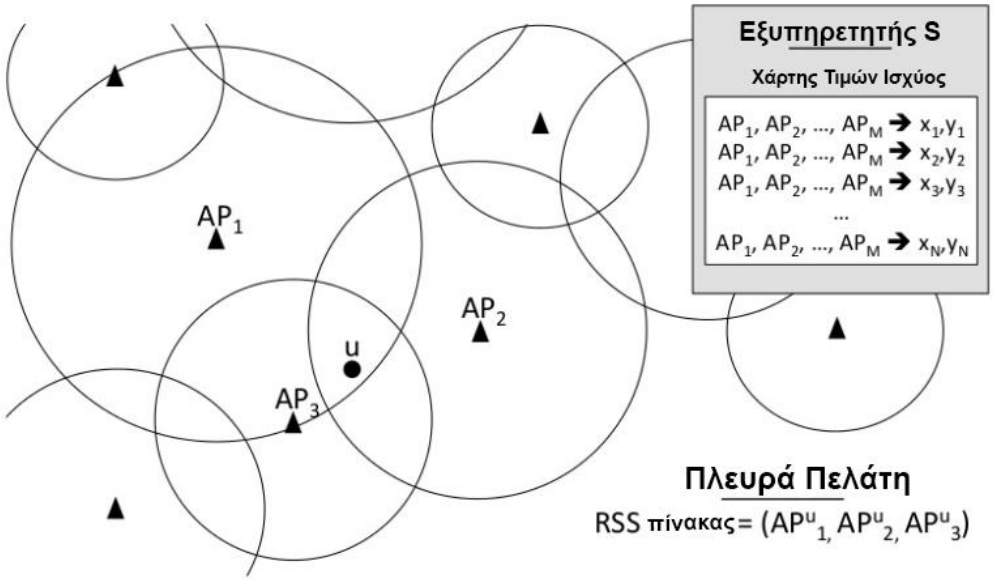
Στην τεχνική της χωρικής απόκρυψης (cloaking), η βασική ιδέα είναι να καταστεί ασαφής η ακριβής τοποθεσία του χρήστη, σε μία ευρύτερη περιοχή η οποία ικανοποιεί της απαιτήσεις για ιδιωτικότητα του χρήστη [11] [12].

Στην τεχνική των μετασχηματισμών χώρου, οι τοποθεσίες των χρηστών μετατρέπονται σε μία άλλη διάσταση στην οποία οι ακριβείς [13] [14] ή οι κατά προσέγγιση [15] χωρικές σχέσεις διασφαλίζονται.

Και τέλος, η τεχνική της k-ανωνυμίας, εγγυάται ότι οι επερωτήσεις του χρήστη είναι δυσδιάκριτες μεταξύ τουλάχιστον k-1 άλλους χρήστες [3] [16].

Για την ιδιωτικότητα της τοποθεσίας των χρηστών, η k-χωρική ανωνυμία επιτυγχάνετε με την συσκότιση της τοποθεσίας του χρήστη ο οποίος πραγματοποιεί επερωτήσεις, έχοντας ως αποτέλεσμα να μην μπορεί να αναγνωριστεί η πραγματική τοποθεσία, με πιθανότητα

Σήμανση που χρησιμοποιήθηκε για αυτή την εργασία	
Σήμανση	Περιγραφή
$u \in U$	Χρήστης έξυπνης κινητής συσκευής που αιτείτε γεωτοποθέτηση (U όλοι οι χρήστες)
A	Γεωγραφική περιοχή για την οποία υπάρχει Χάρτης Τιμών Ισχύος
s	Εξυπηρετητής που έχει Χάρτη Τιμών Ισχύος
k	Η επιλογή του u για k-ανωνυμία
(x_i, y_y)	Τοποθεσία μέσα σε περιοχή A (π.χ., latitude, longitude)
AP_i	Σημείο Πρόσβασης Ασύρματου Δικτύου μέσα στο A με μοναδικό αναγνωριστικό i
C	Όλα τα διαθέσιμα APs, $C=\{AP_1, AP_2, \dots, AP_N\}$
RSS	Ισχύς Λαμβανόμενου Σήματος από τα Ασύρματα Δίκτυα (π.χ., RSSi)
AP_i^u	RSS ενός AP_i μετρημένο από τον u



Σχήμα 5: *Το μοντέλο συστήματος*

ψηλότερη από 1/k. Αυτό μπορεί να επιτευχθεί χρησιμοποιώντας k ψεύτικες τοποθεσίες, υποθέτοντας ότι οι τοποθεσίες του χρήστη έχουν ομοιόμορφες πιθανότητες μέσα στον χώρο. Ομοίως στην ανωνυμία τροχιών, οι προσεγγίσεις k-ανωνυμίας εγγυούνται ότι τουλάχιστον k τροχιές χρήστη θα είναι δυσδιάκριτες μεταξύ τους. Οι προσεγγίσεις k-ανωνυμίας για αυτό τον σκοπό περιλαμβάνουν τις υλοποιήσεις Always-Walk-with-Other [17] και Never-Walk-Alone (NWA) [18], οι οποίες συλλέγουν k τροχιές βασισμένες στην χωρική τους ομοιότητα. Η παραλλαγή Wait-4-Me (W4M) [19] είναι επίσης προσέγγιση η οποία βασίζεται σε χωροχρονική επεξεργασία.

Όλες οι πιο πάνω προσεγγίσεις σχετίζονται με την διατήρηση της ιδιωτικότητας στον διαμοιρασμό δεδομένων, εν αντίθεση με την γεωτοποθέτηση που υπολογίζετε στην πλευρά του εξυπηρετητή (Server-Side). Τα σενάρια μας σχετίζονται με τις περιπτώσεις όπου ένας χρήστης κάποιας έξυπνης κινητής συσκευής κινείτε στον χώρο και χρειάζεται γεωτοποθετηθεί

μόνο για κάποιο στιγμιότυπο ή και συνεχόμενα. Η γεωτοποθέτηση με γνώμονα την ανάγκη του χρήστη για ιδιωτικότητα, θα επιτρέπει στον χρήστη να γεωτοποθετηθεί, χωρίς να αποκαλύψει την τοποθεσία του στον εξυπηρετητή, ή πιο λεπτομερώς αυτό θα αποτρέπει στον εξυπηρετητή να αναγνωρίσει την πραγματική θέση του χρήστη με πιθανότητα ψηλότερη από την παράμετρο k που θα μπορεί να καθοριστεί από τον χρήστη.

Για στιγμιότυπη γεωτοποθέτηση, με γνώμονα την ιδιωτικότητα των χρηστών, θα αναλυθεί ο αλγόριθμος Vectormap, ο οποίος είναι αποτελεσματικός για την αίτηση ενός μερικού Χάρτη Τιμών Ισχύος (Radiomap) από τον εξυπηρετητή, ο οποίος θα περιλαμβάνει τις απαραίτητες πληροφορίες για το Σημείο Πρόσβασης Ασύρματου Δικτύου (WiFi AP), που γειτνιάζει με τον χρήστη, αλλά και σε k-1 ψεύτικα Σημεία Πρόσβασης.

Αυτό επιτυγχάνεται με την χρήση φίλτρων Bloom, τα οποία εγγυώνται ότι η μέγιστη πιθανότητα για τον εξυπηρετητή όσο αφορά την πραγματική θέση του χρήστη θα είναι 1/k.

Επιπρόσθετα, η συνεχόμενη γεωτοποθέτηση με γνώμονα την ιδιωτικότητα των χρηστών είναι ακόμη μεγαλύτερη πρόκληση, καθώς διαδοχικές αιτήσεις από τον ίδιο χρήστη μπορούν να αποκαλύψουν τις ψεύτικες τοποθεσίες του χρήστη και να αυξήσουν την πιθανότητα του εξυπηρετητή για την πραγματική τοποθεσία του χρήστη από 1/k σε 1, όπως αυτό παρουσιάστηκε κατά την εισαγωγή. Κατά την αποτίμηση της νέας οικογένειας αλγορίθμων Temporal Vectormap, θα αποδειχθεί πως η πιθανότητα 1/k στην πλευρά του εξυπηρετητή, όσο αφορά την πραγματική τοποθεσία του χρήστη, θα παραμείνει αναλλοίωτη καθ’ όλη την διάρκεια των συνεχόμενων γεωτοποθετήσεων.

Κεφάλαιο 3

Διατύπωση του Προβλήματος και Μοντέλο Συστήματος

3.1 Διατύπωση Προβλήματος

Υποθέτουμε ότι μια περιοχή A καλύπτεται από ένα σύνολο από Σημεία Πρόσβασης Ασύρματων Δικτύων (Wifi APs) $C = \{ AP_1, AP_2, \dots , AP_N\}$. Αυτή η περιοχή δεν είναι απαραίτητα συνεχόμενη, και δεν υποθέτουμε κάποια συγκεκριμένη διανομή των Σημείων Πρόσβασης στο σύνολο C. Κάθε Σημείο Πρόσβασης AP_i του συνόλου C έχει ένα μοναδικό αναγνωριστικό (το BSSID/MAC address της συσκευής δικτύου) που μεταδίδεται και είναι δημόσια διαθέσιμο. Στον πραγματικό κόσμο, η περιοχή A μπορεί να συμπεριλάβει όλες τις περιοχές από εσωτερικά κτίρια σε οποιαδήποτε γωνιά του πλανήτη. Τα προβλήματα επεκτασιμότητας θα αναλυθούν περαιτέρω σε επόμενο κεφάλαιο, αφού επεξηγηθεί η αρχιτεκτονική του εξυπηρετητή, που εξειδικεύεται σε σύνολα δεδομένων τεράστιου όγκου.
Ένας χρήστης έξυπνων κινητών συσκευών μέσα σε κάποια περιοχή A μπορεί να χρησιμοποιήσει την κεραία Ασύρματου Δικτύου της συσκευής του για να μετρήσει την Ισχύ Λαμβανόμενου Σήματος (RSS) και να πάρει το αναγνωριστικό (BSSID) κάποιου σημείου πρόσβασης AP_i που ανήκει στο σύνολο C. Κάθε χρήστης U μπορεί επίσης να επικοινωνήσει με τον εξυπηρετητή S μέσω Ασύρματου Δικτύου (WiFi) ή σύνδεσης Κινητών δεδομένων (Mobile Data). Θέτουμε την ιδιωτικότητα ενός χρήστη u, ως p_u, που αντιστοιχεί στο ανώτατο όριο πρόβλεψης που μπορεί να υποθέσει ο εξυπηρετητής S για την τοποθεσία του χρήστη U. Υποθέτουμε ότι ολόκληρη η διαδικασία γεωτοποθέτησης για τον χρήστη u στην συσκευή του, καταναλώνει ενέργεια E_u και πόρους δικτύου R_u.

Το πρόβλημα θέτετε ως εξής:

Παρέχουμε στιγμιότυπη ή συνεχόμενη γεωτοποθέτηση σε εσωτερικούς χώρους για κάθε χρήστη U, καθώς κινείτε μέσα σε κάποια περιοχή A, χρησιμοποιώντας τις τιμές Ισχύος Λαμβανόμενου Σήματος και τα αναγνωριστικά των Σημείων Πρόσβασης που τον γειτνιάζουν. Το σύστημα εγγυάται ιδιωτικότητα p_u, ενώ παράλληλα μειώνετε η ενέργεια E_u και οι πόροι δικτύου R_u.

Για να λάβει χώρα η γεωτοποθέτηση στο πιο πάνω μοντέλο συστήματος, ο εξυπηρετητής S πρέπει να έχει τον Χάρτη Τιμών Ισχύος της περιοχής A. Ένας Χάρτης Τιμών Ισχύος είναι ένας δισδιάστατος πίνακας ο οποίος εκφράζει την σχέση μεταξύ των τοποθεσιών (latitude, longitude, σε παγκόσμιο σύστημα συντεταγμένων), και τις τιμές Ισχύος Λαμβανόμενου Σήματος (RSS), για κάθε AP που ανήκει στο σύνολο C και μετρήθηκε στις διακριτές τοποθεσίες (x,y) όπως αυτό περιγράφετε στο [2]. Ας υποθέσουμε ότι ο εξυπηρετητής διατηρεί ένα πίνακα δύο διαστάσεων MATRIX [N] [M], ο οποίος καταγράφει τις τιμές Ισχύος Λαμβανόμενου Σήματος M, των Σημείων Πρόσβασης APs, σε N διακριτές τοποθεσίες στον πλανήτη (x,y). Για παράδειγμα ο Χάρτης Τιμών Ισχύος MATRIX μπορεί να είναι στην ακόλουθη μορφή:

Χάρτης Τιμών Ισχύος (MATRIX)

```
AP_1, AP_2, ... AP_M => x_1, y_1
AP_1, AP_2, ... AP_M => x_2, y_2
AP_1, AP_2, ... AP_M => x_3, y_3
                . . .
AP_1, AP_2, ... AP_M => x_N, y_N
```

Ο MATRIX ουσιαστικά συνθέτεται με την περισυλλογή πολλών πινάκων με τιμές RSS:

```
AP_1, AP_2, ... AP_w => xi, yi
```

Στις μετρήσεις αυτές το $w << M$. Η καταγραφή τους έγινε από χρήστες οι οποίοι εκτελούσαν διαδικασίες γεωγραφικού εντοπισμού (wardriving), δηλαδή τις διαδικασίες όπου κάποιο άτομο αναζητεί ασύρματα δίκτυα καθώς κινείτε, χρησιμοποιώντας κάποιον φορητό υπολογιστή, PDA ή έξυπνη κινητή συσκευή. Επιπρόσθετα, ο MATRIX είναι εξαιρετικά μεγάλος εν συγκρίσει με το N, καθώς η διάσταση M είναι συνήθως μικρότερη και μπορεί να αναπαρασταθεί αποδοτικά με δομές γειτονικών πινάκων. Για ευκολίες παρουσίασης, ας υποθέσουμε ότι τα περισσότερα σημεία στον πίνακα MATRIX είναι μηδενικά (δηλαδή -110dB για τιμή RSS).

3.2 Ο Αλγόριθμος Vectormap (VM)

Σε αυτό το υποκεφάλαιο θα παρουσιαστεί ο αλγόριθμος Vectormap (VM), για υψηλής ακρίβειας, χαμηλής ενεργειακής κατανάλωσης, και διατήρησης της ιδιωτικότητας των χρηστών, διαδικασία γεωτοποθέτησης, σε στιγμιότυπες εκτελέσεις από τις έξυπνες κινητές συσκευές.

3.2.1. Τα φίλτρα Bloom k-ανωνυμίας (kAB)

Ο αλγόριθμος Vectormap είναι μία αποκεντρωμένη προσέγγιση βασισμένη στους Χάρτες Τιμών Ισχύος, κατά την οποία ο χρήστης αποστέλλει τους πίνακες τιμών Ισχύος Λαμβανόμενου Σήματος (RSS) στον εξυπηρετητή και λαμβάνει μερικό Χάρτη Τιμών Ισχύος για να μπορέσει να γεωτοποθετηθεί. Το κύριο χαρακτηριστικό του αλγόριθμου Vectormap είναι ότι αντί να αποστέλλει πίνακα τιμών με τις τιμές Ισχύος Λαμβανόμενου Σήματος (RSS), τις μετατρέπει σε ένα φίλτρο bloom k-ανωνυμίας για να επιτύχει αποκρυπτογράφηση και k-ανωνυμία. Τα φίλτρα bloom [5] είναι δομές δεδομένων αποτελεσματικές από πλευράς χωρικών πιθανοτήτων και χρησιμοποιούνται για την απάντηση επερωτήσεων. Η ιδέα είναι η δέσμευση ενός πίνακα από b δυαδικά ψηφία, ο οποίος αρχικά περιείχε σε όλες τις θέσεις του την τιμή 0. Ακολούθως χρησιμοποιούνται h ανεξάρτητες μεταξύ τους συναρτήσεις κατακερματισμού, για την διαμοίραση ενός στοιχείου σε h τοποθεσίες στον πίνακα, με μια τυχαία και ομοιόμορφη διανομή. Για την δημιουργία φίλτρων Bloom για κάποιο αντικείμενο, τροφοδοτούμε το αντικείμενο σε κάθε μία από της h συναρτήσεις κατακερματισμού για πάρουμε σαν αποτέλεσμα h θέσεις στον πίνακα δυαδικών ψηφίων και να τις θέσουμε σε 1. Για τον έλεγχο κατά πόσο κάποιο αντικείμενο ανήκει σε κάποιο σύνολο, πρέπει να συγκρίνουμε τον πίνακα της επερώτησης με τον πίνακα που περιέχει τους δυαδικούς πίνακες φίλτρων bloom όλων των Σημείων Πρόσβασης (AP), ολόκληρου του Χάρτη Τιμών Ισχύος (radiomap).
Αν έστω και μία μη μηδενική θέση στον δυαδικό πίνακα της επερώτησης, δεν ταιριάζει με μια μη μηδενική θέση στον πίνακα συνόλου δυαδικών πινάκων, τότε το αντικείμενο της επερώτησης σίγουρα δεν υπάρχει στο σύνολο. Αν όλα τα μη μηδενικά σημεία ταιριάζουν, τότε το αντικείμενο μπορεί να θεωρηθεί μέλος του συνόλου, αφού τα φίλτρα bloom δεν αποτρέπουν τις ψευδο-θετικές εκτιμήσεις (false-positives).
Το σημαντικό χαρακτηριστικό των φίλτρων bloom, είναι ότι υπάρχει ένας καθαρός συμβιβασμός μεταξύ του b (του αριθμού των δυαδικών ψηφίων), και της πιθανότητας των ψευδο-θετικών εκτιμήσεων. Δεδομένου h βέλτιστων συναρτήσεων κατακερματισμού, b δυαδικά ψηφία για φίλτρα bloom, και μ ως τον αριθμό των αντικειμένων, μπορούμε να υπολογίσουμε τον αριθμό των ψευδο-θετικών εκτιμήσεων που παράγονται από τα φίλτρα bloom, όπως αυτό θα επεξηγηθεί εκτενέστερα στο επόμενο υποκεφάλαιο. Ο χρήστης μπορεί να καθορίσει τον πλεονασμό των φίλτρων bloom που θα είναι ο αριθμός k από αναγνωριστικά των Σημείων Πρόσβασης, τα οποία θα χαρτογραφούνται σε κάποιο φίλτρο. Αυτό επιτυγχάνεται με την επιλογή του αριθμού δυαδικών ψηφίων b, που θα χρησιμοποιηθεί για την κωδικοποίηση των φίλτρων. Υποθέτουμε ότι ο χρήστης έχει μια καλή εκτίμηση του αριθμού M από όλα τα αναγνωριστικά των Σημείων Πρόσβασης στο σύστημα και του αριθμού h των συναρτήσεων κατακερματισμού (π.χ., αυτά μπορούν να μεταφερθούν από τον εξυπηρετητή

κατά την διαδικασία της αρχικοποίησης). Έπειτα χρησιμοποιεί μ=1 για να δημιουργήσει το φίλτρο bloom, αφού είναι μόνο ένα το αναγνωριστικό Σημείου Πρόσβασης το οποίο επιθυμεί αποκρύψει. Με αυτό τον τρόπο, μπορούμε να εξασφαλίσουμε k-χωρική-ανωνυμία, μιας και ο εξυπηρετητής δεν θα μπορεί να διακρίνει τα Σημεία Πρόσβασης Ασύρματων Δικτύων που ακούει πραγματικά ο χρήστης.

Η ψευδο-θετική εκτίμηση (fpr), δίνετε κατά προσέγγιση από την παρακάτω εξίσωση:

$$fpr = (1 - e^{-h\mu/b})^h \quad (1)$$

που σημαίνει δεδομένου του h, μ και την καθορισμένη τιμή k από τον χρήστη, μπορούμε να υπολογίσουμε τον αριθμό b των δυαδικών ψηφίων που μπορούμε να χρησιμοποιήσουμε στα φίλτρα bloom.

3.2.2. Περίγραμμα λειτουργίας του αλγορίθμου VectorMap (VM)

Υπάρχουν h προκαθορισμένες συναρτήσεις κατακερματισμού (συνιστώμενες για φίλτρα bloom [20] [21]), οι οποίες είναι γνωστές και στους πελάτες, αλλά και στον εξυπηρετητή. Ο εξυπηρετητής κρατά τον ενημερωμένο Χάρτη Τιμών Ισχύος, και μία λίστα από δυαδικούς πίνακες φίλτρων bloom, από όλα τα M αναγνωριστικά των AP που υπάρχουν στον Χάρτη Τιμών Ισχύος. Ο χρήστης πρώτα επιλέγει το αναγνωριστικό του AP το οποίο βρίσκετε πιο κοντά του ούτως, ώστε να αργήσει να αιτηθεί νέο μερικό Χάρτη Τιμών Ισχύος όταν τεθεί εκτός εμβέλειας του προηγούμενο σημείου. Ακολούθως δημιουργεί τον kAB πίνακα-φίλτρο χρησιμοποιώντας τις h συναρτήσεις κατακερματισμού. Τότε, ο χρήστης αποστέλλει τον πίνακα-φίλτρο kAB στον εξυπηρετητή. Ο εξυπηρετητής ταιριάζει τον εισερχόμενο δυαδικό πίνακα με k δυαδικούς πίνακες bloom, από όλα τα αναγνωριστικά των AP, αναλόγως με την επιλεγμένη ψευδο-θετική εκτίμηση της πράξης (1). Ο εξυπηρετητής αναγνωρίζει όλες τις γραμμές n << N από τους Χάρτες Τιμών Ισχύος που έχουν μη-μηδενικές τιμές Ισχύος Λαμβανόμενου Σήματος (RSS), για τα AP που ταίριαξαν, και αφού συνθέσει μερικό Χάρτη Τιμών Ισχύος (Partial Radiomap), τον επιστρέφει στον χρήστη. Ως εκ τούτου, ο χρήστης μπορεί να γεωτοποθετηθεί χωρίς να αποκαλύπτει την πραγματική του τοποθεσία, καθώς διατηρεί απόκρυψη (cloaking) αλλά και k-ανωνυμία.

Είναι σημαντικό να σημειωθεί ότι μόνο ένα αναγνωριστικό AP(δηλαδή μ=1), από τα AP που γειτνιάζει ο χρήστης χρησιμοποιείται για την παραγωγή του φίλτρου bloom. Αν χρησιμοποιείτο περισσότερο του ενός αναγνωριστικού AP, τότε ο εξυπηρετητής θα ήταν σε θέση να αναγνωρίσει τις ψευδο-θετικές εκτιμήσεις (δηλαδή τα ψεύτικα Σημεία Πρόσβασης), που παράγονται από το φίλτρο bloom, αφού θα γνώριζε τον αριθμό αναγνωριστικών που

Αλγόριθμος 1: Vectormap (VM)
Είσοδος
• V: Πίνακας τιμών Ισχύος Λαμβανόμενου Σήματος
• h: Συναρτήσεις κατακερματισμού
Έξοδος: μερικός Χάρτης Τιμών Ισχύος
Βήμα 0 – Εγκαθίδρυση: Καθορισμός k. Υπολογισμός b χρησιμοποιώντας την πράξη (3)
Βήμα 1 - u δημιουργεί kAB : ο χρήστης u επιλέγει ένα AP_{id}^{u}, δηλαδή μ=1, από τον πίνακα του με τις τιμές Ισχύος Λαμβανόμενου Σήματος για να δημιουργήσει το kAB (δηλαδή το φίλτρο bloom) μεγέθους b, με καθορισμένη την τιμή k από αυτόν
Βήμα 2 – u προωθεί το kAB στον s: Ο χρήστης u προωθεί τον πίνακα kAB στον εξυπηρετητή s
Βήμα 3 – s εντοπίζει ψευδο-θετικές εκτιμήσεις: ο εξυπηρετητής εντοπίζει τα k AP_{id}s εκ των οποίων τα φίλτρα bloom τους ταίριαξαν με τον εισερχόμενο πίνακα kAB. Ανακτώνται οι γραμμές του πίνακα MATRIX, όπου οι τιμές Ισχύος Λαμβανόμενου Σήματος δεν είναι μηδενικές
Βήμα 4 – s προωθεί τον μερικό Χάρτη Τιμών Ισχύος στον u: ο εξυπηρετητής s, αποστέλλει τις γραμμές οι οποίες ταυτοποιήθηκαν, δηλαδή τον μερικό Χάρτη Τιμών Ισχύος, στον χρήστη u
Βήμα 5 – ο χρήστης u γεωτοποθετείται: Ο χρήστης u, εκτελεί την διαδικασία γεωτοποθέτησης τοπικά, χρησιμοποιώντας τον μερικό Χάρτη Τιμών Ισχύος που έλαβε.

Περίγραμμα 6: *λειτουργία του αλγόριθμου Vectormap*

χρησιμοποιήθηκε για την δημιουργία του φίλτρου, και οι τυχαίες ψευδο-θετικές εκτιμήσεις ως επί το πλείστο δεν θα επικαλύπτονταν.

Το σύνολο της απάντησης από τον εξυπηρετητή θα μπορούσε να μειωθεί επιπλέων, με τον συμβιβασμό για την ποσότητα ανωνυμίας που θα έχει τελικά ο χρήστης. Για να το πετύχει αυτό ο χρήστης, μαζί με το φίλτρο bloom θα πρέπει να προωθήσει στον εξυπηρετητή και κάποιες πληροφορίες Ισχύος Λαμβανόμενου Σήματος (RSS).

3.2.3. Παράδειγμα εκτέλεσης του αλγορίθμου Vectormap (VM)

Όπως φαίνεται στο περίγραμμα 6, ας υποθέσουμε ότι ο χρήστης u, βρίσκετε στην τοποθεσία (x,y) όπου τα σήματα από τρία διαφορετικά Σημεία Πρόσβασης μπορούν να ληφθούν.

Αυτά είναι AP_1, AP_2, AP_3, και τιμές Ισχύος Λαμβανόμενου Σήματος (RSS) τους είναι 20%, 50% και 70% αντίστοιχα. Ως εκ τούτου ο πίνακας Ισχύος Λαμβανόμενου Σήματος του χρήστη είναι V= { AP1:20%, AP_2:50%, AP_3:70%}. Υποθέτουμε ότι το M = 100 APs, h=3 οι προκαθορισμένες συναρτήσεις κατακερματισμού. Η επιθυμία του χρήστη για χωρική ανωνυμία είναι k=3.

Ο χρήστης αρχικά υπολογίζει το απαιτούμενο μέγεθος b του κAB φίλτρου που θα εξασφαλίσει k-1 ψευδο-θετικές εκτιμήσεις στην πλευρά του εξυπηρετητή. Με τις τιμές k και M, η τιμή των ψευδο-θετικών εκτιμήσεων καθορίζετε με την πιο κάτω εξίσωση:

$$fpr = k/M \quad (2)$$

η οποία έχει ως αποτέλεσμα fpr=0.03.

kAB φίλτρο του χρήστη =

0	0	1	0	1	0	0

Τα Σημεία Πρόσβασης Ασύρματων Δικτύων που ταίριαξαν στην πλευρά του εξυπηρετητή είναι: AP_2, AP_{13}, AP_{65}

Τοποθεσία	AP_1	AP_2	AP_3	...	AP_{13}	...	AP_{65}	...	AP_M
x_1,y_1	-110	-20	-110	...	-110	...	-20	...	-110
x_2,y_2	-110	-110	-110	...	-110	...	-110	...	-110
x_3,y_3	-110	-110	-110	...	-30	...	-80	...	-110
x_4,y_5	-10	-110	-110	...	-46	...	-110	...	-110
x_5,y_5	-110	-110	-110	...	-110	...	-110	...	-12
...	...	...	...	...	...	...	...	...	...
x_{10},y_{10}	-10	-110	-60	...	-100	...	-110	...	-110
x_{11},y_{11}	-25	-50	-73	...	-110	...	-110	...	-110
x_{12},y_{12}	-50	-65	-20	...	-30	...	-110	...	-110
...	...	...	...	...	...	...	...	...	...
x_{50},y_{50}	-72	-110	-16	...	-110	...	-110	...	-110
x_{51},y_{51}	-83	-110	-110	...	-110	...	-110	...	-110
x_{52},y_{52}	-110	-70	-110	...	-19	...	-12	...	-110
x_{53},y_{53}	-110	-110	-86	...	-110	...	-56	...	-110
x_{54},y_{54}	-110	-110	-110	...	-110	...	-110	...	-43
...	...	...	...	...	...	...	...	...	...

Πίνακας 7: *Ένας χάρτης με Μ Σημεία Πρόσβασης που περιέχει τις τιμές Ισχύος Λαμβανόμενου Σήματος για Ν (x,y) τοποθεσίες. Οι σκιασμένες γραμμές περιέχουν πληροφορίες για τα Σημεία Πρόσβασης που αποκωδικοποιήθηκαν από το φίλτρο kAB που απεστάλη στον εξυπηρετητή. Θα χρησιμοποιηθούν για την δημιουργία ενός μερικού Χάρτη Τιμών Ισχύος.*

Χρησιμοποιώντας τις εξισώσεις (1) και (2) προκύπτει η εξίσωση:

$$b = \frac{-h\mu}{ln(1 - \sqrt[h]{fpr})} \quad (3)$$

Το αποτέλεσμα που υπολογίζει ο χρήστης είναι περίπου b=8.1, οπόταν ο χρήστης θέτει b=8 για να εξασφαλίσει ότι θα υπάρξουν τουλάχιστον k-1=2 ψευδο-θετικές εκτιμήσεις. Τότε ο χρήστης επιλέγει ένα AP_{id} από τον πίνακα τιμών Ισχύος Λαμβανόμενου Σήματος (RSS), π.χ. AP_3 και το τροφοδοτεί στις προκαθορισμένες συναρτήσεις κατακερματισμού για να παράξουν το kAB φίλτρο μεγέθους b=8, με κάποια δυαδικά ψηφία με την τιμή 1, π.χ. με τρείς συναρτήσεις κατακερματισμου που επιστρέφουν ένα ακέραιο αριθμό εντός των ορίων του πίνακα μεγέθους b=8, και την $1^η$ να επιστρέφει την τιμή 1, την $2^η$ την τιμή 4 και την $3^η$ την τιμή 7, το bloom φίλτρο θα ήταν: {0,1,0,0,1,0,0,1}.

Ακολούθως στέλνει το φίλτρο στον εξυπηρετητή. Σημειώστε ότι η επικοινωνία μπορεί να διεξαχθεί μέσω ενός διαδικτυακού πρωτοκόλλου (IP) με υπηρεσίες ανωνυμίας σε περίπτωση ενός παγκόσμιου εξυπηρετητή για Χάρτες Τιμών Ισχύος, ούτως ώστε η χώρα ή η ήπειρος της αίτησης να μην αποκαλύπτετε.

Στην πλευρά του εξυπηρετητή (όπως φαίνετε στον Πίνακα 7) τα αναγνωριστικά των Σημείων Πρόσβασης που ταιριάζουν με το φίλτρο kAB που μεταφέρθηκε, αναγνωρίστηκαν. Σύμφωνα με το επιλεχθέν fpr, θα ταιριάξουν τουλάχιστον k=3 αναγνωριστικά Σημείων Πρόσβασης στον εξυπηρετητή. Ένα από αυτά τα αναγνωριστικά των σημείων πρόσβασης, π.χ. το AP_3, είναι το

πραγματικό Σημείο Πρόσβασης το οποίο ο χρήστης έχει στον πίνακα Ισχύος Λαμβανόμενου Σήματος (RSS), ενώ τα υπόλοιπα δύο αναγνωριστικά, AP_{13} και $AP_{65,}$ είναι ψεύτικα. Τώρα ο εξυπηρετητής ανακτά για κάθε Σημείο Πρόσβασης που ταίριαξε, τις γραμμές που περιέχουν μη μηδενικές τιμές Ισχύος Λαμβανόμενου Σήματος (όπως σημειώνετε με σκίαση στον πίνακα 7). Οι ανακτηθείς γραμμές αποστέλλονται στον χρήστη, ο οποίος τώρα μπορεί να αναλύσει τα δεδομένα και να δημιουργήσει ένα μερικό Χάρτη Τιμών Ισχύος και ακολούθως να γεωτοποθετηθεί χρησιμοποιώντας κάποιον γνωστό αλγόριθμο (π.χ. KNN, WKNN, MSME, WMSME) όπως περιγράφετε στο [22].

Ο Vectormap επιτυγχάνει k-χωρική ανωνυμία, επιτρέποντας στον εξυπηρετητή να γνωρίζει μόνο με μια πιθανότητα 1/k το ότι ο χρήστης βρίσκετε σε κάποιο από τα Σημεία Πρόσβασης τα οποία ταίριαξαν. Αυτό επιτυγχάνεται εξ' αιτίας των ψεύτικων αναγνωριστικών Σημείων Πρόσβασης. Επίσης, επιτυγχάνετε και απόκρυψη (cloaking), μιας και ο εξυπηρετητής δεν γνωρίζει σε ποιο σημείο των 3 περιοχών βρίσκετε ο χρήστης (υπάρχει ίση πιθανότητα να βρίσκετε σε οποιοδήποτε σημείο μέσα σε αυτές τις περιοχές).

Στην προσέγγιση του Vectormap αλγόριθμου, ο χρήστης μπορεί να επιλέξει την ποσότητα των πληροφοριών των τιμών Ισχύος Λαμβανόμενου Σήματος (RSS) που αποστέλλονται στον εξυπηρετητή. Όσες περισσότερες πληροφορίες έχει ο εξυπηρετητής, τόσο μικρότερη είναι η απόκρυψη (cloaking) του χρήστη, όμως τόσο μικρότερο είναι το μέγεθος του πίνακα MATRIX. Ως εκ τούτου υπάρχει συμβιβασμός μεταξύ της απόκρυψης και της ιδιωτικότητας του χρήστη. Ο χρήστης μπορεί να επιλέξει να στήλει μια εμβέλεια στην οποία μία ή περισσότερες από τις τιμές Ισχύος Λαμβανόμενου Σήματος είναι ψευδής, ή μια ή και περισσότερες πραγματικές τιμές Ισχύος Λαμβανόμενου Σήματος, και να υποδηλώσει κατά πόσο οι τιμές Ισχύος Λαμβανόμενου Σήματος ανταποκρίνονται στο αναγνωριστικό AP μέσα από το φίλτρο bloom.

Εδώ υπενθυμίζετε ότι η ποσότητα των τιμών Ισχύος Λαμβανόμενου Σήματος που αποστέλλονται στον εξυπηρετητή δεν επηρεάζουν την εγγύηση της k-χωρικής ανωνυμίας. Ακόμα και αν ο χρήστης αποστέλλει πληροφορίες Ισχύος Λαμβανόμενου Σήματος για να μειώσει το μέγεθος του μερικού Χάρτη Τιμών Ισχύος, ο εξυπηρετητής θα μπορεί μόνο να περιορίσει τις αποκρυμμένες περιοχές, αλλά όχι να πάρει 3 συγκεκριμένες τοποθεσίες

3.3 Ο αλγόριθμος Temporal VectorMap (TVM)

Σε αυτό το υποκεφάλαιο θα αποτιμηθεί ο αλγόριθμος Temporal Vectormap (TVM) [1]. Ο αλγόριθμος TVM ανήκει στην οικογένεια αλγορίθμων Temporal Vectormap, μαζί με τον αλγόριθμο Random Temporal Vectormap. Ο TVM αλγόριθμος παρέχει υψηλής ακρίβειας, χαμηλής ενεργειακής κατανάλωσης, και διατήρησης της ιδιωτικότητας διαδικασία

γεωτοποθέτησης που μπορεί να εκτελεστεί σε συνεχόμενες γεωτοποθετήσεις στις έξυπνες κινητές συσκευές. Πριν την περιγραφή του αλγορίθμου, ξεκαθαρίζετε ότι κάποιος χρήστης u, είναι σε θέση να γεωτοποθετηθεί έως ότου κινείτε μέσα στην εμβέλεια του Σημείου Πρόσβασης που χρησιμοποιήθηκε για να παραχθεί το kAB φίλτρο. Αυτό οφείλετε στο γεγονός ότι ο εξυπηρετητής S επιστρέφει όλες τις γραμμές που περιέχουν τις μη αρνητικές τιμές Ισχύος Λαμβανόμενου Σήματος καθώς επίσης και τις συντεταγμένες (x,y) όλων των τοποθεσιών που υπάγονται σε αυτή την ακτίνα μετάδοσης. Όταν ο χρήστης μεταφερθεί έξω από την εμβέλεια του Σημείου Πρόσβασης, τότε απαιτείτε ένας ενημερωμένος μερικός Χάρτης Τιμών Ισχύος ούτως ώστε να μπορεί να εκτελεστεί εκ νέου η διαδικασία της γεωτοποθέτησης. Με την προσέγγιση του αλγορίθμου Vectormap είναι πάντα δυνατόν να ληφθεί ένας ενημερωμένος μερικός Χάρτης Τιμών Ισχύος, όμως οι νέες ψεύτικες τοποθεσίες είναι χωρικά πολύ μακριά από τις αρχικές ψεύτικες τοποθεσίες. Σε αυτές τις περιπτώσεις, ακόμα και αν ο χρήστης δεν προδίδει την πραγματική του τοποθεσία, ο εξυπηρετητής μπορεί εύκολα να περιορίσει τις υποψήφιες τοποθεσίες του χρήστη και να ανακαλέσει την k-ανωνυμία του χρήστη όπως αυτό αναλύθηκε πιο πριν.

3.3.1. Περίγραμμα λειτουργίας του αλγορίθμου Temporal VectorMap (TVM)

Ο αλγόριθμος Temporal Vectormap (TVM) είναι σχεδιασμένος για να επιτρέπει στον χρήστη να παραπλανάει τον εξυπηρετητή, όχι μόνο όσο αφορά την τρέχων τοποθεσία του, αλλά και για την χωροχρονική διαδρομή που ακολουθεί. Στον Temporal Vectormap αλγόριθμο ο χρήστης χρησιμοποιεί τον Vectormap αλγόριθμο μόνο την πρώτη φορά, κατά την εκκίνηση της διαδικασίας γεωτοποθέτησης, για να πάρει τον πρώτο μερικό Χάρτη Τιμών Ισχύος. Έτσι ο χρήστης μπορεί εύκολα να αναγνωρίσει όλους τους γείτονες των πραγματικών αλλά και των ψεύτικων σημείων πρόσβασης. Τότε, κάθε φορά που ο χρήστης τίθεται εκτός εμβέλειας, και χρειάζεται ένα νέο μερικό Χάρτη Τιμών Ισχύος από τον εξυπηρετητή, επιλέγει τον «κατάλληλο» γείτονα από τα ψεύτικα σημεία πρόσβασης ούτως ώστε να είναι σε θέση να δημιουργήσει ρεαλιστικά μοτίβα μετακίνησης, παρόμοια με το πραγματικό. Με αυτό τον τρόπο ο Temporal Vectormap αλγόριθμος παρεμποδίζει τον εξυπηρετητή να συμπεράνει το πραγματικό μονοπάτι και έτσι διατηρεί την k-ανωνυμία με την έννοια του Never Walk Alone [18].

Προκειμένου ο χρήστης να βρει τον νέο «κατάλληλο» γείτονα ενός ψεύτικου Σημείου Πρόσβασης, χρειάζεται αρχικά να υπολογίσει την Ευκλείδεια απόσταση μεταξύ της αρχικής του θέσης l = (x,y) και της θέσης l'= (x',y') ακριβώς πριν βγει εκτός εμβέλειας, με την πιο κάτω εξίσωση:

$$d_{l,l'} = \sqrt{(x - x')^2 + (y - y')^2} \quad (4)$$

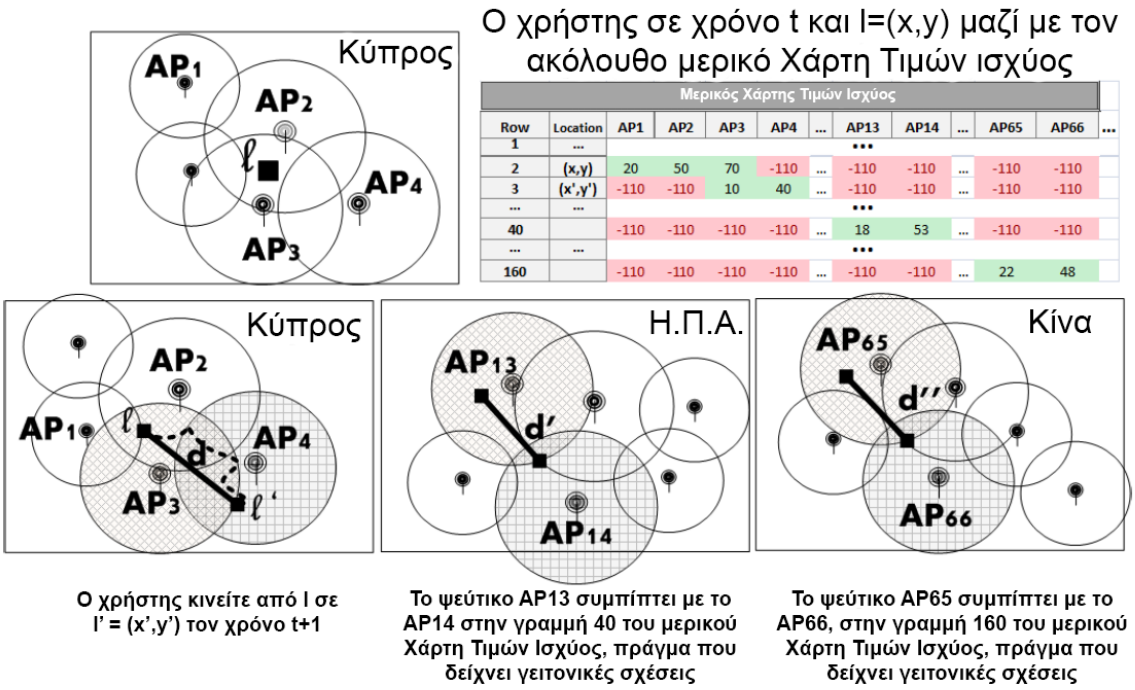
Αλγόριθμος 2: Temporal Vectormap (TVM)
Είσοδος
• V: Πίνακας τιμών Ισχύος Λαμβανόμενου Σήματος
• μερικός Χάρτης Τιμών Ισχύος
Έξοδος: ενημερωμένος μερικός Χάρτης Τιμών Ισχύος
Βήμα 1 - u αναγνωρίζει τους δικούς του γείτονες: ο u χρησιμοποιεί τον μερικό Χάρτη Τιμών Ισχύος, που αρχικά έλαβε με τον αλγόριθμο Vectormap, για να εντοπίσει όλα τα γειτονικά APs στα k-1 ψεύτικα AP (π.χ. AP_y και AP_z) που επιστράφηκαν από τον εξυπηρετητή s, όταν AP_x^u είχε χρησιμοποιηθεί. Έτσι αναγνωρίζει τις επικαλύψεις ανάμεσα στα ψεύτικα APs, και σε άλλα άγνωστα APs στον μερικό Χάρτη Τιμών Ισχύος (είναι οι γραμμές οι οποίες ένα ψεύτικο AP και ένα ή περισσότερα άλλα APs έχουν μη μηδενικές τιμές RSS)
Βήμα 2 – u αναγνωρίζει τους ψεύτικους γείτονες: ο u εντοπίζει ένα γειτονικό AP, για κάθε ψεύτικο AP (π.χ. AP_y' και AP_z'), στα οποία υπάρχει όμοιο μοτίβο ίχνους με αυτό που ακολουθήθηκε από τον χρήστη καθώς μετακινόταν από το l= (x,y) στο l'= (x',y') και μέσα στην εμβέλεια του νέου AP_x'^{u}
Βήμα 3 – u προωθεί τα APs στον s: ο u προωθεί στον s, τα τρία APs, π.χ. το νέο πραγματικό AP_x'^{u}, και τους δύο ψεύτικους γείτονες AP_y' και AP_z', για να διατηρήσει την k-ανωνυμία του
Βήμα 4 – s προωθεί τον ενημερωμένο μερικό Χάρτη Τιμών Ισχύος στον u: ο εξυπηρετητής s, αποστέλλει ένα ενημερωμένο μερικό Χάρτη Τιμών Ισχύος με όλες τις γραμμές που περιέχουν μη μηδενικές τιμές RSS για οποιοδήποτε από τα τρία APs
Βήμα 5 – ο χρήστης u γεωτοποθετείται: Ο χρήστης u εκτελεί την διαδικασία γεωτοποθέτησης τοπικά χρησιμοποιώντας τον ενημερωμένο μερικό Χάρτη Τιμών Ισχύος που έλαβε.

Περίγραμμα 8: *Ο αλγόριθμος Temporal Vectormap.*

Επίσης για σκοπούς αποτίμησης της χρονικής απόδοσης του αλγορίθμου Temporal VectorMap υλοποιήθηκε και ο Random Temporal Vectormap, κατά τον οποίο η επιλογή των νέων ψεύτικων σημείων πρόσβασης γινόταν με τυχαίο τρόπο.

Αναλυτικά η διαδικασία έχει ως ακολούθως:

1. Ο χρήστης πρέπει να αναγνωρίσει ένα γειτονικό Σημείο Πρόσβασης για κάθε ψεύτικο Σημείο Πρόσβασης το οποίο ταιριάζει με την πραγματική τροιχειά που ακολουθήθηκε από τον χρήστη. Για να γίνει αυτό, εντοπίζετε υποσυνόλο από γραμμές V που ανήκουν στον πίνακα MATRIX (συμπεριλαμβανομένου των συντεταγμένων τους) που έχουν τιμές Ισχύος Λαμβανόμενου Σήματος για τα ψεύτικα Σημεία Πρόσβασης (π.χ. AP_y) κοντινές με τις τιμές Ισχύος Λαμβανόμενου Σήματος του πραγματικού σημείου πρόσβασης όταν ο χρήστης ήταν στην αρχική τοποθεσία στο σημείο l.
2. Παράλληλα, εντοπίζετε υποσύνολο γραμμών U που ανήκουν στον πίνακα MATRIX (συμπεριλαμβανομένου και των συντεταγμένων τους), όπου το ψεύτικο Σημείο Πρόσβασης AP_y και τουλάχιστον ένα από τα γειτονικά του Σημεία Πρόσβασης (π.χ. $AP_{y'}$) έχουν τιμές Ισχύος Λαμβανόμενου Σήματος κοντινές με τις τιμές του πραγματικού Σημείου Πρόσβασης και του πραγματικού γειτονικού του, στην τοποθεσία l', στην χρονική στιγμή που ο χρήστης αιτήθηκε τον ενημερωμένο μερικό Χάρτη Τιμών Ισχύος.



Εικόνα 9: *Παράδειγμα εκτέλεσης του αλγορίθμου TVM*

3. Υπολογισμός της Ευκλείδιας απόστασης μεταξύ των αποστάσεων d' όλων των συνδιασμών μεταξύ των συντεταγμένων στους πίνακες U και V.
4. Τελικά, το $AP_{y'}$ που θα επιλεγεί ως το γειτονικό για το ψεύτικο Σημείο Πρόσβασης AP_y θα είναι αυτό που υπολογίστηκε να έχει την κοντινότερη απόσταση d', σε σχέση με την πραγματική απόσταση d.

Αυτή η διαδικασία επαναλαμβάνετε για όλα τα ψεύτικα Σημεία Πρόσβασης. Ο χρήστης τότε αποστέλνει το νέο πραγματικό Σημείο Πρόσβασης, καθώς επίσης και τα νέα ψεύτικα Σημεία Πρόσβασης στον εξυπηρετητή. Με αυτό τον τρόπο, ο χρήστης παραπλανεί τον εξυπηρετητή όσο αφορά την πραγματική του τοποθεσία αφού το αποτέλεσμα δείχνει παρόμοιες τροχιές για k τοποθεσίες.

3.3.2. Παράδειγμα εκτέλεσης του αλγορίθμου Temporal VectorMap (TVM)

Όπως φαίνεται στην εικόνα 9 (πάνω αριστερά), ο χρήστης σε κάποια χρονική στιγμή και στην τοποθεσία l= (x,y) αιτείται από τον εξυπηρετητή μερικό Χάρτη Τιμών Ισχύος χρησιμοποιώντας τον Vectormap αλγόριθμο, όπως αυτό επεξηγήθηκε πιο πριν. Σε αυτή την τοποθεσία ο χρήστης είχε ένα πίνακα τιμών Ισχύος Λαμβανόμενου Σήματος V = {AP_1: 20, AP_2:50, AP_3: 70} και χρησιμοποίησε το AP_3 για να δημιουργήσει το kAB φίλτρο. Ο εξυπηρετητής επέστρεψε ένα μερικό Χάρτη Τιμών Ισχύος (εικόνα 9 – πάνω δεξιά) συμπεριλαμβάνοντας όλες τις τοποθεσίες που καλύφθηκαν από την ακτίνα του AP_3 και τα δύο

ψεύτικα Σημεία Πρόσβασης, π.χ. AP_{13} και AP_{65}. Τότε ο χρήστης μετακινείτε στην τοποθεσία l'= (x',y') η οποία είναι ακριβώς πριν μετακινηθεί εκτός της εμβέλειας του Σημείου Πρόσβασης AP_3: 10 και εντός εμβέλειας του γειτονικού Σημείου Πρόσβασης AP_4: 40 (εικόνα 9, κάτω δεξιά). Ως εκ τούτου ο χρήστης αιτείτε από τον εξυπηρετητή ένα ενημερωμένο μερικό Χάρτη Τιμών Ισχύος. Για να γίνει αυτό, χωρίς ο χρήστης να αποκαλύψει την τοποθεσία του, προηγουμένως υπολογίζει την Ευκλείδεια απόσταση d, από το σημείο l στο σημείο l', χρησιμοποιώντας την πράξη (4).

Τότε ο χρήστης εντοπίζει όλους τους γείτονες για κάθε ψεύτικο Σημείο Πρόσβασης, με την αναγνώριση επικαλύψεων στον μερικό Χάρτη Τιμών Ισχύος, οι οποίες είναι μη αρνητικές τιμές Ισχύος Λαμβανόμενου Σήματος για τα ψεύτικα Σημεία Πρόσβασης, αλλά και τα υπόλοιπα Σημεία Πρόσβασης σε κάθε γραμμή. Για παράδειγμα στην εικόνα 9, η γραμμή 40 δείχνει ότι το ψεύτικο AP_{13} έχει επικάλυψη με το AP_{14}, ως εκ τούτου το AP_{14} του είναι γειτονικό. Ομοίως η γραμμή 160 δείχνει ότι το AP_{65} και το AP_{66} είναι γειτονικά. Ωστόσο ένα ψεύτικο σημείο πρόσβασης μπορεί να έχει περισσότερους από ένα γείτονες.

Ως εκ τούτου ο χρήστης πρέπει να επιλέξει τον γείτονα που θα μπορούσε να δημιουργήσει τροχιά πιο κοντινή με την πραγματική τροχιά του χρήστη, πράγμα που θα καθιστά αδύνατες τις προσπάθειες του εξυπηρετητή για εντοπισμό της πραγματικής τοποθεσίας του χρήστη.

Ο χρήστης για να το πετύχει αυτό, θα πρέπει για κάθε ψεύτικο σημείο πρόσβασης να βρει δύο τοποθεσίες, που να έχουν παρόμοιες τιμές Ισχύος Λαμβανόμενου Σήματος με τις πραγματικές τιμές στο σημείο l, για την πρώτη τοποθεσία και στο σημείο l' για την δεύτερη τοποθεσία. Επίσης σε αυτά τα σημεία, θα πρέπει να ληφθεί υπόψιν και η ευκλείδεια απόσταση d', η οποία να είναι όσο πιο κοντινή γίνετε στην πραγματική απόσταση d. Τελικά ο χρήστης προωθεί το νέο Σημείο Πρόσβασης AP_4 και τους νέους γείτονες των δύο ψεύτικων σημείων πρόσβασης (π.χ. AP_{14} και AP_{66}) στον εξυπηρετητή και λαμβάνει ένα ενημερωμένο μερικό Χάρτη Τιμών Ισχύος. Έτσι διατηρεί την k-ανωνυμία του, ενώ παράλληλα γεωτοποθετείτε με χαμηλότερη κατανάλωση ενέργειας και λιγότερο εύρος ζώνης για το δίκτυο σε σχέση με την αρχική υλοποίηση του Airplace.

Κεφάλαιο 4

Τεχνολογικό Υπόβαθρο

Για την διεκπεραίωση αυτής της εργασίας χρησιμοποιήθηκαν αρκετές σύγχρονες και περιζήτητες τεχνολογίες. Ο εξυπηρετητής απαρτίζετε από: κατανεμημένη βάση δεδομένων, κατανεμημένο σύστημα αρχείων και εξυπηρετητή ιστού για την αλληλεπίδραση με τους πελάτες. Στην εφαρμογή πελάτη χρησιμοποιήθηκε το εργαλείο ανάπτυξης λογισμικού του Λειτουργικού Συστήματος Android και η Διεπαφή Προγραμματισμού Εφαρμογών (API) χαρτών Google.

4.1 Τεχνολογίες Εξυπηρετητή

Το σύνολο του εξυπηρετητή αποτελείται από τρία στρώματα. Αναλυτικά οι τεχνολογίες που χρησιμοποιήθηκαν είναι: το κατανεμημένο σύστημα αρχείων Hadoop, το μοντέλο χαρτογράφησης/μείωσης Hadoop, η κατανεμημένη βάση δεδομένων Hbase, μαζί με την Διεπαφή Προγραμματισμού Εφαρμογών (API) που παρέχει και ο Εξυπηρετητής Ιστού Apache Tomcat.

4.1.1 Κατανεμημένο Σύστημα Αρχείων Hadoop

Το Apache Hadoop στοχεύει για αξιοπιστία, ευελιξία στην επεκτασιμότητα, και διασκορπισμένο υπολογισμό (distributed computing). Είναι Δωρεάν και Ανοιχτού Πηγαίου Κώδικα Λογισμικό (FOSS) του οργανισμού Apache, και είναι ένα πλαίσιο το οποίο επιτρέπει την κατανεμημένη επεξεργασία τεράστιων συνόλων δεδομένων σε συμπλέγματα υπολογιστών, χρησιμοποιώντας απλά προγραμματιστικά μοντέλα. Είναι σχεδιασμένο για να επεκτείνετε από μια υπολογιστική μηχανή σε χιλιάδες μηχανές, εκ των οποίων η κάθε μια προσφέρει στο σύμπλεγμα την τοπική της υπολογιστική ισχύ και την αποθηκευτική της μονάδα. Επιπρόσθετα αντί να βασίζετε στο ότι το υλικό θα προσφέρει υψηλή διαθεσιμότητα, το πλαίσιο καθαυτό είναι σχεδιασμένο για να εντοπίζει και να χειρίζεται σφάλματα σε επίπεδο εφαρμογής. Έτσι μπορεί να παραδώσει υψηλή διαθεσιμότητα υπηρεσίας πάνω από ένα σύμπλεγμα υπολογιστών, στο οποίο κάθε υπολογιστική μονάδα μπορεί να είναι επιρρεπείς σε σφάλματα, διατηρώντας την έτσι σε χαμηλό κόστος. Για να το πετύχει αυτό, δημιουργεί αριθμό αντιγράφων για κάθε αρχείο, με παράμετρο η οποία καθορίζετε από τον διαχειριστή του συστήματος.

Από το σύστημα Apache Hadoop, χρησιμοποιήθηκε η λειτουργική μονάδα Hadoop Distributed File System (HDFS), η οποία είναι ένα διασκορπισμένο σύστημα αρχείων το οποίο παρέχει υψηλή απόδοση σε πρόσβαση στα δεδομένα εφαρμογής. Το HDFS είναι υλοποίηση Δωρεάν και Ανοιχτού Πηγαίου Κώδικα (FOSS) βασισμένη στο Google File System [23] το οποίο δημοσιεύτηκε το 2003. Είναι υλοποιημένο σε γλώσσα προγραμματισμού Java. Οι εφαρμογές Hadoop, γράφουν τα δεδομένα τους μια φορά, αλλά τα διαβάζουν πολλαπλές φορές (write once, read many semantics), και απαιτούν οι ταχύτητες ανάγνωσης να είναι πολύ υψηλές. Ακόμη οι υπολογισμοί γίνονται κοντά στα δεδομένα, ούτως ώστε να μειώνετε το εύρος ζώνης στο δίκτυο, και να αυξάνεται η ολική απόδοση του συστήματος.

Επίσης υποστηρίζετε εξισορρόπηση φόρτου εργασίας στο συμπλέγμα. Αναλυτικά δημιουργούνται νέα αντίγραφα από δεδομένα αυτόματα, και μεταφέρονται σε κάποιο νέο κόμβο αν αυξηθεί σημαντικά ο αριθμός αιτήσεων για κάποιο συγκεκριμένο αρχείο. Με αυτό τον τρόπο αποσυμφορίζεται ο αρχικός κόμβος.

4.1.2 Μοντέλο Χαρτογράφησης/Μείωσης

Το μοντέλο Χαρτογράφησης/Μείωσης (Map/Reduce) είναι για επεξεργασία τεράστιου όγκου δεδομένων. Ως επί το πλείστον, τα περισσότερα προγράμματα Hadoop είναι γραμμένα με το μοντέλο Χαρτογράφησης/Μείωσης. Σε αυτό το μοντέλο η είσοδος είναι σπασμένη σε πολλά

μικρά κομμάτια, τα οποία διασκορπίζονται για να επεξεργαστούν ανεξάρτητα και παράλληλα. Το πρώτο μέρος του μοντέλου ονομάζετε χαρτογράφηση (map). Τα αποτελέσματα αυτών των ανεξάρτητων και παράλληλων εκτελέσεων, σε δεύτερη φάση συλλέγονται σε ομάδες και επεξεργάζονται ομαδικά. Η δεύτερη φάση του μοντέλου ονομάζετε μείωση (Reduce).

Κατά την χαρτογράφηση (map), στο Hadoop, τα δεδομένα σπάζουν σε πολλά κομμάτια τα οποία ονομάζονται FileSplits. Κατά την διαδικασία κατακερματισμού των δεδομένων δεν λαμβάνετε υπόψιν το περιεχόμενο των αρχείων. Για κάθε αρχείο, τρέχει και μια υποεργασία χαρτογράφησης (map task). Κάθε υποεργασία χαρτογράφησης, καταναλώνει ζευγάρια κλειδιών-τιμής (key-value pairs) από το κατακερματισμένο αρχείο που της ανατέθηκε, και παράγει ενδιάμεσα ζευγάρια κλειδιών-τιμής. Ακολουθούμενης της διαδικασίας χαρτογράφησης, το πλαίσιο του Hadoop, ταξινομεί τα ενδιάμεσα ζευγάρια κλειδιών-τιμών, και παράγει σύνολα από πλειάδες ούτως ώστε όλες οι τιμές που σχετίζονται με κάποιο συγκεκριμένο κλειδί, να εμφανίζονται μαζί. Επίσης μοιράζει το σύνολο των πλειάδων σε αριθμό από τεμάχια τα οποία αντιστοιχούν στον αριθμό των υποεργασιών μείωσης (reduce tasks).

Κατά την μείωση (reduce) στο Hadoop, καταναλώνονται οι πλειάδες οι οποίες ταξινομήθηκαν και ομαδοποιήθηκαν κατά την διαδικασία χαρτογράφησης. Κάθε μειωτής (reducer), καταναλώνει τις πλειάδες που του ανατέθηκαν. Για να γίνει μείωση, χρειάζεται μια μέθοδος καθορισμένη από τον χρήστη, η οποία μετουσιώνει τις πλειάδες σε έξοδο ζευγαριών κλειδιών-τιμών. Το πλαίσιο Hadoop αφού διαμοιράσει τις υποεργασίες μείωσης στο σύμπλεγμα των εργατών, φροντίζει να στείλει τα κατάλληλα ενδιάμεσα δεδομένα στον κατάλληλο μειωτή (reducer).

4.1.3 Αρχιτεκτονική συστημάτων Hadoop και HDFS

Το όλο σύστημα τρέχει εργασίες (jobs), και διαμοιράζει υποεργασίες (tasks), οι οποίες είναι κομμάτια από την εργασία. Αποθηκεύει τα δεδομένα με παράλληλη και διασκορπισμένη τάση. Οι εργασίες ανήκουν στο μοντέλο Χαρτογράφησης/Μείωσης (Map/Reduce), και η αρχιτεκτονική στην οποία τρέχουν εργασίες είναι της μορφής ενός κυρίου κόμβου και πολλών υπηρετών (master, slave).

Υπάρχει μια μηχανή, ο κύριος κόμβος (master) στον οποίο τρέχει η διεργασία ονοματοδωσίας (NameNode), η οποία είναι υπεύθυνη για τον χώρο ονομάτων, και για τον έλεγχο της πρόσβασης στα αρχεία από τις εφαρμογές πελάτες της. Κρατάει στην μνήμη της το δέντρο με όλα τα αρχεία του συστήματος, καθώς και σε ποιο σημείο στο σύμπλεγμα βρίσκονται τα φυσικά αρχεία, χωρίς να κρατάει καμία πληροφορία για το περιεχόμενο τους. Είναι υπεύθυνη

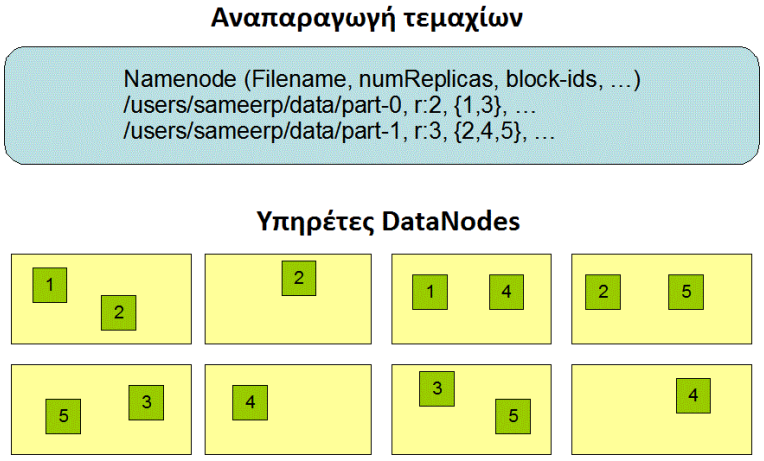
τις διαδικασίες του συστήματος αρχείων όπως άνοιγμα αρχείου, κλείσιμο αρχείου, αλλαγή ονόματος, διαγραφή κ.α.. Οι εφαρμογές πελάτες, επικοινωνούν πρώτα με τον NameNode για να εντοπίσουν κάποιο αρχείο, πριν το επεξεργασθούν. Από τον NameNode καταγράφονται οι οποιεσδήποτε αλλαγές στα αρχεία, καθώς και ο παράγοντας αντιγράφων (replicas) των αρχείων. Επιπλέων εκτελεί τις απαραίτητες ενέργειες που αφορούν τα αντίγραφα των αρχείων και γνωρίζει την κατάσταση των υπηρετών (slaves) ανά πάσα στιγμή όσο αφορά τα δεδομένα τους, αφού λαμβάνει περιοδικά τα καρδιοχτύπια (heartbeats) τους.

Είναι μοναδικό σημείο αποτυχίας (single point of failure), όμως υπάρχει η επιλογή για την διεργασία εναλλακτικής ονοματοδωσίας (SecondaryNameNode). Με την χρήση του SecondaryNameNode, σε διαφορετική μηχανή, σε περίπτωση τερματισμού του κυρίως κόμβου, μπορεί ο δευτερεύων να αναλάβει και να συνεχίσει τις διαδικασίες χωρίς να επηρεαστεί το όλο σύστημα.

Επίσης στον κύριο κόμβο τρέχει η διεργασία JobTracker που χειρίζεται τις εργασίες. Ο JobTracker σπάζει μια εργασία σε πολλές υποεργασίες τις οποίες διαμοιράζει σε πολλές μηχανές υπηρετών (slaves), οι οποίοι με την σειρά τους εξυπηρετούν τον κύριο τους στην αντίστοιχη διεργασία τους, ονομαζόμενη TaskTracker. Ο κύκλος ζωής της εκτέλεσης εργασιών είναι ο εξής:

1. Ο χρήστης εισάγει εργασίες (jobs) στον κύριο κόμβο (JobTracker) οι οποίες αφού μπουν σε ουρά εξυπηρετούνται κατά την σειρά άφιξης (FCFS).
2. Ο κύριος κόμβος σπάζει τις εργασίες σε πολλές υποεργασίες, του μοντέλου Χαρτογράφησης/Μείωσης και τις παραδίδει στους υπηρέτες (TaskTrackers). Η επιλογή των υπηρετών γίνετε ιδανικά, φροντίζοντας να σταλεί η υποεργασία «κοντά» στα δεδομένα (data locality semantics). Για να γίνει αυτό στέλνει την υποεργασία στους κόμβους όπου υπάρχουν τα δεδομένα, ή τουλάχιστον στους κόμβους που είναι στην ίδια σχάρα (rack) στο σύμπλεγμα του εξυπηρετητή. Για τον εντοπισμό των κόμβων που είναι κοντά στα δεδομένα, γίνετε επικοινωνία του JobTracker με τον NameNode.
3. Οι υπηρέτες εκτελούν την χαρτογράφηση, ετοιμάζουν τα ενδιάμεσα αποτελέσματα, και έπειτα εκτελούν και την Μείωση. Στην επικοινωνία με τον πελάτη για την μεταφορά των δεδομένων, μεσολαβεί μόνο την πρώτη φορά ο NameNode, και τις υπόλοιπες φορές γίνετε απευθείας με τους DataNodes.

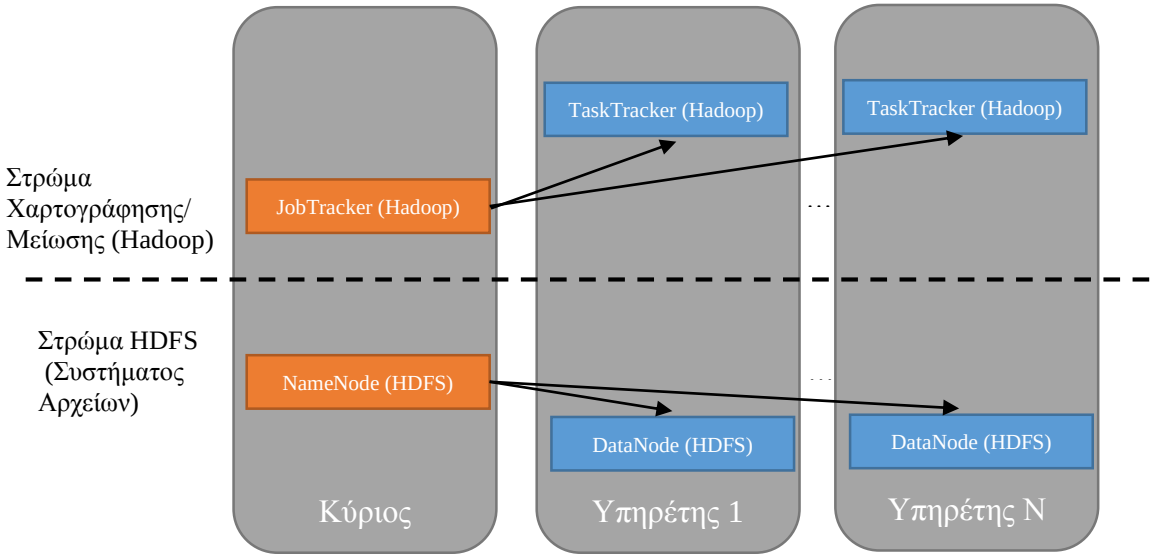
   Οι υπηρέτες (slaves) είναι οι κόμβοι δεδομένων και τρέχουν την διεργασία δεδομένων (DataNode), με την οποία χειρίζονται την τοπική αποθηκευτική μονάδα και κατά συνέπεια και τα αποθηκευμένα δεδομένα. Αυτή η λειτουργία μπορεί να έρθει σε αντιστοιχία με την λειτουργία NameNode του κυρίου κόμβου. Ένα λειτουργικό σύστημα HDFS έχει περισσότερο από ένα DataNode, όπου όλα τα δεδομένα έχουν



Σχήμα 10: *Ο κύριος κόμβος (NameNode), που περιέχει τις διευθύνσεις αρχείων, και η κατανομή των αρχείων και των αντιγράφων στους υπηρέτες DataNodes.*

αντίγραφα ασφαλείας στο σύμπλεγμα. Οι πελάτες αφού βρουν την φυσική τοποθεσία των δεδομένων μέσω του NameNode, επικοινωνούν με τον DataNode, έχοντας έτσι απευθείας πρόσβαση στα δεδομένα. Οι κόμβοι δεδομένων (DataNodes) συνεννοούνται μεταξύ τους, ούτως ώστε να δημιουργήσουν τα αντίγραφα των δεδομένων τα οποία καθορίζονται από τον διαχειριστή του συστήματος HDFS. Επειδή τα δεδομένα αντιγράφονται σε διαφορετικές αποθηκευτικές μονάδες, σε διαφορετικούς εξυπηρετητές, δεν υπάρχει ανάγκη για χρήση επιπλέων τεχνολογιών (π.χ. RAID), για αποθήκευση πολλαπλών αντιγράφων στον ίδιο εξυπηρετητή. Επιπρόσθετα ο υποκατάλογος στον οποίον θα αποθηκευτούν τα αρχεία, επιλέγετε με χρήση ευρετικού, ούτως ώστε ο συνολικός αριθμός των ήδη υπάρχοντων αρχείων στον συγκεκριμένο υποκατάλογο να είναι ο βέλτιστος για σκοπούς χρονικής απόδοσης.

Η διαδικασία δημιουργίας αντιγράφων γίνετε αποδοτικά χρησιμοποιώντας διασωλήνωση (pipeline). Για παράδειγμα όταν κάποιος χρήστης γράψει δεδομένα στο σύστημα HDFS, τα δεδομένα ξεκινούν να αποθηκεύονται σε κάποιο DataNode σε μικρές μερίδες. Όταν αποθηκευτεί στον πρώτο DataNode η μερίδα δεδομένων, τότε συνεχίζει στον δεύτερο DataNode για να δημιουργήσει το πρώτο αντίγραφο, κ.ο.κ μέχρι να εγγραφούν όλα τα προκαθορισμένα αντίγραφα, και εν τέλει μέσω διασωλήνωσης να αποθηκευτεί ολόκληρο το αρχείο και μαζί με τα αντίγραφά του. Επειδή κάποιος κόμβος δεδομένων καθώς λαμβάνει δεδομένα από τον προηγούμενο κόμβο, ταυτόχρονα μπορεί να προωθεί τα δεδομένα που έγραψε στον επόμενο κόμβο, η διαδικασία δημιουργίας αντιγράφων επιταχύνετε κατά πολύ. Στους υπηρέτες εκτελείται επίσης και η διεργασία εκτέλεσης υποεργασιών (TaskTracker), κατά την οποία εκτελούν τις υποεργασίες μόλις τις δεχθούν από τον κύριο κόμβο. Οι



Σχήμα 11: *Η διαχώριση του στρώματος Χαρτογράφησης/Μείωσης (Hadoop) με το στρώμα Κατανεμημένου Συστήματος Αρχείων (HDFS).*

υποεργασίες αυτές μπορεί να είναι χαρτογράφησης ή μείωσης. Επιπρόσθετα οι TaskTrackers χειρίζονται την κίνηση δεδομένων μεταξύ των φάσεων χαρτογράφησης και μείωσης.

4.1.4 Μη σχεσιακή, κατανεμημένη βάση δεδομένων HBase

Η βάση δεδομένων HBase είναι μη-σχεσιακή (NoSQL), κατανεμημένη (distributed) βάση δεδομένων, Δωρεάν και Ανοιχτού Πηγαίου Κώδικα, υλοποιημένη σε γλώσσα προγραμματισμού Java. Είναι βασισμένη στην δημοσίευση της Google για την ιδιόκτητη της βάση δεδομένων BigTable [24]. Η υλοποίησή της αποτελεί μέρος του έργου Hadoop, και τρέχει στην κορυφή ενός Hadoop κατανεμημένου συστήματος (HDFS), παρέχοντας όμοιες δυνατότητες με την βάση δεδομένων BigTable. Η διαφορά ωστόσο της HBase με το Hadoop/HDFS είναι ότι το Hadoop/HDFS παρέχει ένα γενικής χρήσης διάσπαρτο σύστημα αποθήκευσης τεράστιων δεδομένων, χωρίς να παρέχει γρήγορη αναζήτηση για συγκεκριμένα αρχεία, εν αντίθεση με την HBase, που το παρέχει αυτό, αφού δημιουργεί υπαρκτά στο HDFS αρχεία επονομαζόμενα StoreFiles, για τα οποία δημιουργεί ευρετήρια.
Επειδή η HBase υπολείπετε πολλά από τα χαρακτηριστικά των σχεσιακών Βάσεων δεδομένων όπως στήλες με τύπους, δευτερεύων δείκτες, πυροδοτήσεις (triggers) και προχωρημένες γλώσσες επερωτήσεων (SQL), χαρακτηρίζετε περισσότερο από τον όρο σαν «βάση Αποθήκης» (StoreBase) αντί «βάση δεδομένων» (Database).

	Βάση δεδομένων RDBMS	Βάση δεδομένων HBase
Διάταξη δεδομένων	Γραμμές	Οικογένεια Στήλης
Συναλλαγές	Πολλαπλών Γραμμών (ACID)	Μιας γραμμής μόνο
Γλώσσα επερωτήσεων	SQL	Get/Put/Scan/Delete
Ασφάλεια	Πιστοποίηση/Εξουσιοδότηση	Ακόμη δεν υλοποιήθηκε
Ευρετήρια	Τυχαίες στήλες	Πρωτεύων κλειδί μόνο
Ανάγνωση/Γραφή όρια διακίνησης	1000 επερωτήσεις/δευτερόλεπτο	1,000,000 επερωτήσεις/δευτερόλεπτο

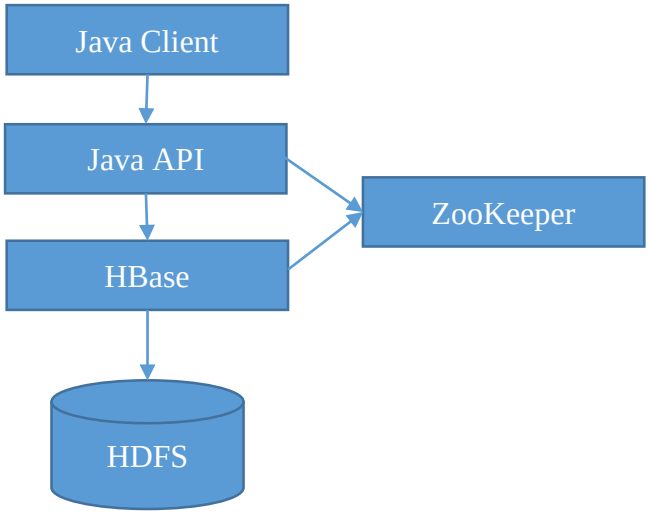
Πίνακας 12: *Σύγκριση HBase με RDBMS βάση δεδομένων*

Ο κύριος σκοπός της είναι η παροχή τεράστιων ποσοτήτων σποραδικών δεδομένων, με ανεκτικότητα σε λάθη. Τα κύρια χαρακτηριστικά της που κληρονομούνται από την βάση δεδομένων BigTable είναι η συμπίεση, η λειτουργία εντός κύριας μνήμης, και η χρήση των φίλτρων bloom ανά στήλες, όπως αυτό περιγράφετε στην δημοσίευση του BigTable [24]. Υπάρχει ισχυρή συνέπεια στα δεδομένα όσο αφορά την ανάγνωση ή την γραφή, και ο τρόπος προσανατολισμού είναι με βάση τις στήλες αντί τις γραμμές, εν αντίθεση με τις σχεσιακές βάσεις. Τα δεδομένα που μπορεί να περιέχει είναι ημι-δομημένα (semi-structured). Επιπρόσθετα παρέχει διαχείριση και παρακολούθηση μέσω δυναμικής ιστοσελίδας, για όλους τους κόμβους που αποτελούν το σύμπλεγμα του συστήματος HBase.

Κατά την αύξηση των δεδομένων στην HBase, οι πίνακες σπάζουν σε περιοχές, οι οποίες φιλοξενούνται από τους υπηρέτες, και η διεργασία που είναι υπεύθυνη για κάθε περιοχή ονομάζετε RegionServer. Υπάρχει αυτόματη εφεδρική λειτουργεία μεταξύ των RegionServer σε περίπτωση σφάλματος σε κάποια μηχανή του συμπλέγματος.

Επίσης λόγω του της τοποθέτησης της HBase στην κορυφή ενός κατανεμημένου συστήματος Hadoop, κληρονομούνται όλα τα πλεονεκτήματα του Hadoop, που επεξηγήθηκαν στο προηγούμενο υποκεφάλαιο, όπως για παράδειγμα αυτό της επεκτασιμότητας. Αν η βάση δεδομένων HBase επεκταθεί για παράδειγμα από 10 σε 20 υπηρέτες (RegionServers), τότε θα διπλασιαστεί η χωρητικότητα της βάσης δεδομένων, αλλά και η υπολογιστική της ισχύ. Αυτό την καθιστά πολύ ισχυρή έναντι των σχεσιακών Βάσεων δεδομένων, οι οποίες από ένα σημείο και έπειτα υστερούν ή χρειάζονται εξειδικευμένο υλικό για να διατηρήσουν την απόδοσή τους.

Η χρήση της HBase δεν ταιριάζει σε όλα τα προβλήματα. Πρέπει να γίνετε όταν υπάρχει ανάγκη για τυχαία ανάγνωση ή γράψιμο δεδομένων σε πραγματικό χρόνο, σε πίνακες τεράστιων συνόλων δεδομένων. Το μέγεθος αυτών των πινάκων κυμαίνεται στα δισεκατομμύρια γραμμές και εκατομμύρια στήλες. Ακόμη η βάση δεδομένων HBase όπως και το HDFS γενικότερα, με την χρήση λιγότερων από πέντε υπηρετών, δεν έχει και τα



Σχήμα 13: *Αρχιτεκτονική Στρωμάτων Υψηλού επιπέδου*

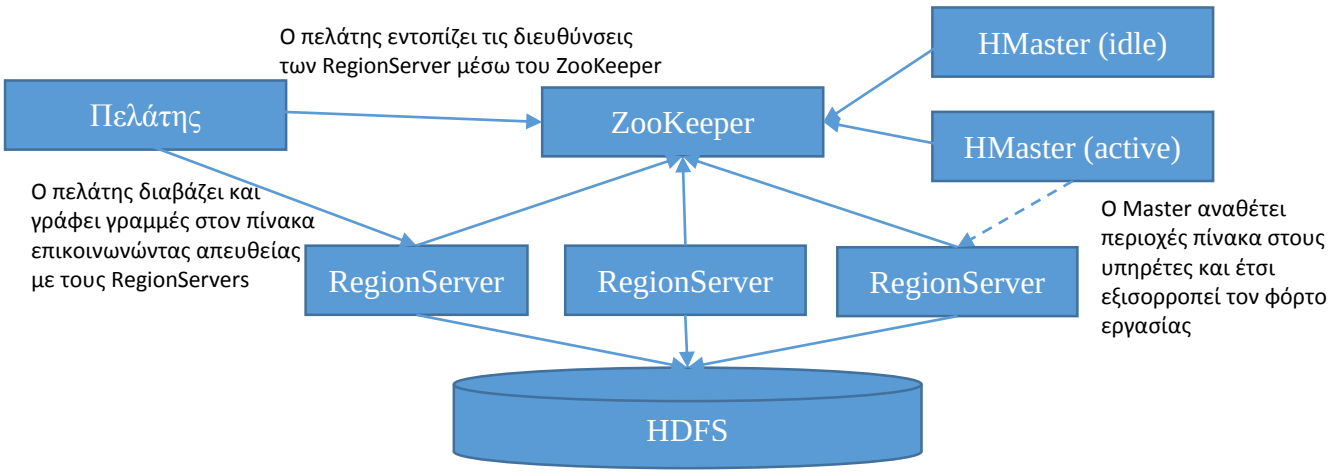
καλύτερα αποτελέσματα. Αυτό οφείλετε σε διάφορους παράγοντες. Ένας παράγοντας είναι η δημιουργία αντιγράφων που είναι προκαθορισμένη σε τρία. Με λιγότερο από τρείς κόμβους, σημαίνει όλα τα δεδομένα είναι σε όλους τους κόμβους, επομένως δεν υπάρχει διάσπαση στον όγκο των δεδομένων. Ένας δεύτερος παράγοντας είναι ο χρόνος για δημιουργία νημάτων, στους υπηρέτες, για την εκτέλεση των υποεργασιών Χαρτογράφησης/Μείωσης.

Η πρόσβαση στην βάση δεδομένων HBase γίνετε με την χρήση της Διεπαφής Προγραμματισμού Εφαρμογών (API) σε γλώσσα προγραμματισμού JAVA, ή μπορούν να χρησιμοποιηθούν άλλες, ημιτελής Διεπαφές Προγραμματισμού Εφαρμογών (API) όπως το REST, Avro ή Thrift σε γλώσσα PHP. Η HBase είναι ευρέως χρησιμοποιημένη σε οδηγημένες από δεδομένα (data driven) εφαρμογές, όπως για παράδειγμα το StumpleUpon[5], το Twitter, και η υπηρεσία μηνυμάτων του Facebook[6].

4.1.5 Αρχιτεκτονική συστήματος βάσης δεδομένων HBase

Ως φυσικό επακόλουθο της υλοποίησης της HBase, στην κορυφή του συστήματος Hadoop/HDFS, συναντάμε και πάλι την αρχιτεκτονική του κυρίου/υπηρετών (master/slaves). Στον κύριο κόμβο του συστήματος HBase, ο οποίος αντιστοιχεί με τον κύριο κόμβο του συστήματος Hadoop/HDFS (δηλαδή είναι η ίδια φυσική μηχανή), τρέχει και η διεργασία του κυρίου της HBase, επονομαζόμενη HMaster. Η HMaster είναι υπεύθυνη για την παρακολούθηση όλων των αντίστοιχων διεργασιών στο επίπεδο των υπηρετών που αφορούν την HBase, και ονομάζονται RegionServers. Χειρίζεται όλες τις αλλαγές στα μέτα-δεδομένα

[5] StumpleUpon HBase Presentation: http://goo.gl/EuVuY
[6] Facebook: “Why our ‘next-gen’ comms ditched MySQL.”: http://goo.gl/2ZTey



Σχήμα 14: *Αρχιτεκτονική Συμπλέγματος HBase*

της βάσης δεδομένων, εντοπίζει και επιλύει προβλήματα που προκύπτουν στους υπηρέτες, και αναθέτει περιοχές του πίνακα της βάσης δεδομένων σε αυτούς.

Επίσης το σύστημα της HBase μπορεί να εκκινήσει σε περιβάλλον πολλών κυρίων κόμβων, όπου ο πρώτος που θα του ανατεθεί από τον ZooKeeper (η διεργασία που είναι υπεύθυνη για τις διασυνδέσεις μεταξύ διάσπαρτων φυσικά κόμβων στο HBase/HDFS) θα είναι ο κύριος κόμβος, και οι υπόλοιποι θα παραμείνουν απλά αδρανείς. Έτσι σε περιπτώσεις αποτυχίας του κύριου κόμβου είναι ιδιαίτερα χρήσιμο, αφού κάποιος άλλος κύριος κόμβος μπορεί να μεταβεί σε ενεργό κατάσταση και έτσι η βάση δεδομένων να συνεχίσει να λειτουργεί κανονικά. Επιπλέον, κατά την αποτυχία του κυρίου κόμβου και στην περίπτωση ύπαρξης μόνο ενός κυρίου κόμβου, για κάποιο χρονικό διάστημα οι πελάτες της βάσης δεδομένων μπορούν αν συνεχίσουν τα εξυπηρετούνται κανονικά, μιας και με την βοήθεια της διεργασίας ZooKeeper επικοινωνούν απευθείας με τους υπηρέτες, όπως γίνετε και στο σύστημα HDFS.

Οι υπηρέτες στο σύστημα HBase, είναι οι ίδιοι φυσικοί κόμβοι με τους υπηρέτες του συστήματος Hadoop/HDFS, και τρέχουν την διεργασία RegionServer, η οποία είναι υπεύθυνη για την εξυπηρέτηση και την διαχείριση των περιοχών του πίνακα που τους έχει ανατεθεί. Εξυπηρετούν για ανάγνωση και γραφή δεδομένων, μέσω των βασικών διεργασιών που παρέχει η HBase και θα επεξηγηθούν λεπτομερώς στην συνέχεια.

Το τελευταίο σώμα το οποίο είναι απαραίτητο για την λειτουργεία του συστήματος HBase, είναι ο προαναφερθής ZooKeeper, ο οποίος είναι ενσωματωμένος στην HBase και συγκεκριμένα στον κύριο κόμβο. Η βασική λειτουργία του είναι η παροχή υψηλής απόδοσης συντονιστικής υπηρεσίας σε διαδεδομένες εφαρμογές. Εκτελεί κοινές λειτουργίες όπως ονοματοδοσία, διαχείριση ρυθμίσεων, συγχρονισμός μεταξύ των κόμβων.

Table	Πίνακας της HBase
-Region	Περιοχή του πίνακα
--Store	Αποθήκη ανά οικογένεια στήλης για κάθε περιοχή του πίνακα
---MemStore	Μνήμη για κάθε αποθήκη (Store) κάθε περιοχής του πίνακα
---StoreFile	Αρχείο για κάθε αποθήκη (Store) κάθε περιοχής του πίνακα
----Block	Κομμάτι εσωτερικά του Αρχείου Αποθήκης (StoreFile) για κάθε αποθήκη (Store)

Πίνακας 15: *Ιεραρχία δεδομένων από το ψηλότερο (πίνακας) στο χαμηλότερο επίπεδο (block αρχείων)*

4.1.6 Μοντέλο δεδομένων στην HBase

Σε υψηλό επίπεδο, οι εφαρμογές αποθηκεύουν δεδομένα σε πίνακες στην βάση δεδομένων HBase. Οι πίνακες στην HBase αποτελούνται από γραμμές και στήλες, με την διαφορά ότι οι στήλες ανήκουν σε κάποια οικογένεια στηλών (column family).

Οι οικογένειες στηλών πρέπει να οριστούν κατά την δημιουργία του σχήματος της βάσης δεδομένων, εν αντίθεση με τις ίδιες τις στήλες οι οποίες μπορεί να δημιουργηθούν δυναμικά, ακόμα και όταν η βάση δεδομένων είναι διαθέσιμη και τρέχει. Η οικογένεια στήλης είναι μονάδα ρύθμισης της απόδοσης, γι’ αυτό και τα ονόματα οικογενειών στήλης πρέπει να διατηρούν μικρό μήκος.

Κάθε κελί του πίνακα υποστηρίζει εκδόσεις και το περιεχόμενο του είναι μια σειρά από δυφιολέξεις (bytes) χωρίς ερμηνεία (uninterpreted). Τα κλειδιά των γραμμών, είναι επίσης πίνακες από δυφιολέξεις, έτσι σχεδόν όλοι οι τύποι ή δομές δεδομένων θα μπορούσαν να είναι κλειδί. Οι γραμμές είναι ταξινομημένες με βάση το κλειδί, και η ταξινόμηση γίνετε ανά δυφιολέξη (byte-ordered). Η κάθε γραμμή εσωτερικά είναι επίσης ταξινομημένη, με βάση την οικογένεια στήλης, ακολουθούμενη από την στήλη, και τέλος την χρονοσφραγίδα έκδοσης της.

Η πρόσβαση στα δεδομένα της γραμμής γίνεται με βάση το κλειδί της. Οι βασικές λειτουργίες δεδομένων που υπάρχουν είναι: Get, Put, Scan και Delete. Η Get επιστρέφει τα χαρακτηριστικά μιας συγκεκριμένης γραμμής, ενώ η Put μπορεί να προσθέσει νέα δεδομένα σε κάποιο πίνακα, ή αν το κλειδί προ-υπήρχε μπορεί να τα ενημερώσει.

Η εντολή Scan επιτρέπει την σάρωση σε πολλαπλές γραμμές που έχουν συγκεκριμένα χαρακτηριστικά. Τρέχει σε όλους τους διάσπαρτους κόμβους υπηρετών, σειριακά με βάση τις περιοχές (region) των πινάκων. Ενώ για λίγα δεδομένα η απόδοση της είναι ικανοποιητική, καθώς αυτά αυξάνονται η απόδοση της μειώνετε εκθετικά. Τέλος η Delete διαγράφει γραμμές από τον πίνακα.

Η ιεραρχία των δεδομένων, από πίνακα της βάσης, μέχρι τα αρχεία που αποτελούν το σύνολο των δεδομένων παρουσιάζετε στον πίνακα 15.

Ανά διαστήματα και όταν αυτό καθίσταται αναγκαίο, στην βάση δεδομένων HBase εκτελούνται διαδικασίες συμπίεσης, που χωρίζονται σε σημαντικές και ασήμαντες. Κατά τις ασήμαντες συμπιέσεις συνήθως μικρά γειτονικά Αρχεία Αποθήκης (StoreFiles) ενώνονται, ενώ υπάρχουν περιπτώσεις όπου οι μικρές συμπιέσεις καταλήγουν σε μεγάλες αφού υπάρχει η πιθανότητα ένωσης όλων των Αρχείων Αποθήκης (StoreFiles) μιας Αποθήκης (Store).

Κατά τις σημαντικές συμπιέσεις ληγμένα κελιά ή διαγραφές, προκαλούν την διαγραφή των δεδομένων από τις φυσικές τους τοποθεσίες, ενώ μετά από κάθε σημαντική συμπίεση, οι Αποθήκες (stores) περιέχουν μόνο ένα Αρχείο Αποθήκης (StoreFile), πράγμα που βελτιώνει κατά πολύ την απόδοση.

Οι πίνακες στην βάση δεδομένων HBase, μπορούν να εισαχθούν σαν είσοδος και έξοδος για τις υποεργασίες χαρτογράφησης/μείωσης που τρέχουν στο σύστημα hadoop.

4.1.7 Εξυπηρετητής Ιστού Apache Tomcat

Ο εξυπηρετητής ιστού Apache Tomcat, είναι λογισμικό Ανοιχτού και Δωρεάν Πηγαίου Κώδικα (FOSS) και αποτελεί ανοιχτή υλοποίηση του εξυπηρετητή Java κα της τεχνολογίας Java Servlets. Ένα από τα πλεονεκτήματα του είναι η γλώσσα προγραμματισμού Java που χρησιμοποιείτε για την σύνταξη των Java Servlets. Εξαιτίας του ότι η Διεπαφή Προγραμματισμού Εφαρμογών (API) της βάσης δεδομένων HBase, χρησιμοποιεί την γλώσσα Java, η χρήση του Apache Tomcat υπερτερεί έναντι άλλων προσεγγίσεων. Επιπρόσθετα οι Java Servlets διατηρούνται στην μνήμη και δεν χρειάζεται η επαναφόρτωση τους σε διαδοχικές κλήσεις. Ακόμη ένα άλλο σημαντικό πλεονέκτημα είναι το ότι ένα στιγμιότυπο ενός Servlet μπορεί να ικανοποιήσει παράλληλες αιτήσεις. Επίσης είναι διαθέσιμη και η τεχνολογία των JavaServer Pages (JSP), η οποία σε συνδυασμό με τους Java Servlets επιτρέπει το κτίσιμο ιστοσελίδων, δυναμικού περιεχομένου.

4.2 Τεχνολογίες Πελάτη

Για την εφαρμογή του πελάτη χρησιμοποιήθηκαν: το Εργαλείο Ανάπτυξης Λογισμικού (SDK) του Λειτουργικού Συστήματος Android[7], η γλώσσας προγραμματισμού JAVA, η Διεπαφή Προγραμματισμού Εφαρμογών (API) χαρτών Google έκδοση 2η, και βιβλιοθήκες συμβατότητας[8] με προηγούμενες εκδόσεις Android.

[7] Android Open Source Project (AOSP): http://source.android.com/
[8] Sherlock Library: http://goo.gl/xqSxq

4.2.1 Ανάπτυξη λογισμικού στην πλατφόρμα Android

Η πλατφόρμα ανάπτυξης λογισμικού Android, είναι μια Δωρεάν και Ανοιχτού Πηγαίου Κώδικα υλοποίηση, που χρησιμοποιεί πυρήνα Linux. Το Εργαλείο Ανάπτυξης Λογισμικού (SDK), που χρησιμοποιείτε για την πλατφόρμα Android, είναι σε γλώσσα προγραμματισμού Java, και παρέχει την δυνατότητα στους προγραμματιστές να αλληλοεπιδράσουν με τον πυρήνα, ούτως ώστε να εξασφαλίσουν πρόσβαση στους πόρους που χρειάζονται, με απώτερο σκοπό την δημιουργία μιας ολοκληρωμένης εφαρμογής. Μερικοί από τους πόρους στις έξυπνες κινητές συσκευές είναι: η οθόνη αφής, η εσωτερική και εξωτερική μνήμη, τα ηχεία, ο δέκτης Ασύρματων Δικτύων, ο δέκτης Παγκόσμιου Συστήματος Συντεταγμένων, και μια σειρά από αισθητήρες όπως γυροσκόπιο, πυξίδα, βαρόμετρο.

4.2.2 Γλώσσα προγραμματισμού Java

Είναι μια αντικειμενοστρεφής γλώσσα προγραμματισμού, που υλοποιήθηκε από την εταιρεία Sun, και κατά το 2010 εξαγοράστηκε από την εταιρεία Oracle. Τα σημαντικά πλεονεκτήματα της γλώσσας προγραμματισμού Java είναι η λειτουργεία της ανεξαρτήτως Λειτουργικού Συστήματος και πλατφόρμας, χωρίς να πρέπει να μεταγλωττίζετε ο κώδικας εκ νέου. Αυτό επιτυγχάνετε με την χρήση της εικονικής μηχανής (Virtual Machine) της γλώσσας Java. Επίσης οι πολλές βιβλιοθήκες καθώς και τα ολοκληρωμένα περιβάλλοντα ανάπτυξης λογισμικού για την γλώσσα Java, με τεχνολογίες όπως αυτόματη συμπλήρωση κώδικα, επιταχύνουν κατά πολύ την συγγραφή κώδικα, κάνοντας έτσι πολύ πιο εύκολο το έργο των προγραμματιστών.

4.2.3 Διεπαφή Προγραμματισμού Εφαρμογών χαρτών Google

Χρησιμοποιώντας την Διεπαφή Προγραμματισμού Εφαρμογών (API) χαρτών Google, μπορούν να εισαχθούν χάρτες σε τρισδιάστατη μορφή βασισμένοι στα δεδομένα των χαρτών Google σε κάποια εφαρμογή στην πλατφόρμα Android. Η διεπαφή χειρίζεται αυτόματα την πρόσβαση στους εξυπηρετητές για την διάθεση των χαρτών, την εμφάνιση τους, αλλά και την αλληλεπίδραση που τυγχάνουν από τον χρήστη. Χρησιμοποιώντας την διεπαφή μπορούν να εισαχθούν στοιχεία στον χάρτη όπως σημεία, σχήματα, εικόνες, ή και επίπεδα. Ακόμη μπορούν να εξαχθούν και να παρουσιαστούν πληροφορίες που αφορούν σημεία στον χάρτη, όπως για παράδειγμα η κοντινότερη πόλη, ή ακόμη να γίνουν μαθηματικές πράξεις όπως η απόσταση μεταξύ δύο σημείων. Τέλος οι χάρτες μεταφέρονται τεμαχισμένοι (σε tiles) στην συσκευή,

και υποστηρίζετε «κρύπτη», έτσι μπορούν να φορτωθούν γρηγορότερα, εξοικονομώντας παράλληλα εύρος ζώνης.

4.2.4 Βιβλιοθήκες συμβατότητας Android

Λόγω των μεγάλων αλλαγών που έτυχαν στην πλατφόρμα του Λειτουργικού Συστήματος Android, οι εκδόσεις του χωρίστηκαν σε δύο μεγάλες κατηγορίες: πριν την έκδοση Ice Cream Sandwich (ICS) και μετά. Πριν την έκδοση ICS το λειτουργικό ήταν χωρισμένο σε δύο μέρη, για την υποστήριξη των έξυπνων κινητών, αλλά και των ταμπλέτων (tablets). Μετά την έκδοση ICS, με ενιαίο λειτουργικό υποστηρίζονται τα έξυπνα κινητά (smartphones και phablets), οι ταμπλέτες (tablets), οι τηλεοράσεις (Google TVs) καθώς και άλλες ανερχόμενες συσκευές (πχ, Google Glass). Για να παρέχετε όμως συμβατότητα με τις συσκευές πριν την έλευση της έκδοσης ICS, υπάρχουν βιβλιοθήκες υποστήριξης, ούτως ώστε να επιτυγχάνετε προς τα πίσω συμβατότητα. Η συμβατότητα αυτή αφορά την πλοήγηση στην εφαρμογή, καθώς και βασικά στοιχεία που καθορίζουν τον τρόπο εμφάνισης ανάλογα με τα διαφορετικά μεγέθη και τύπους οθονών, όλων των έξυπνων συσκευών.

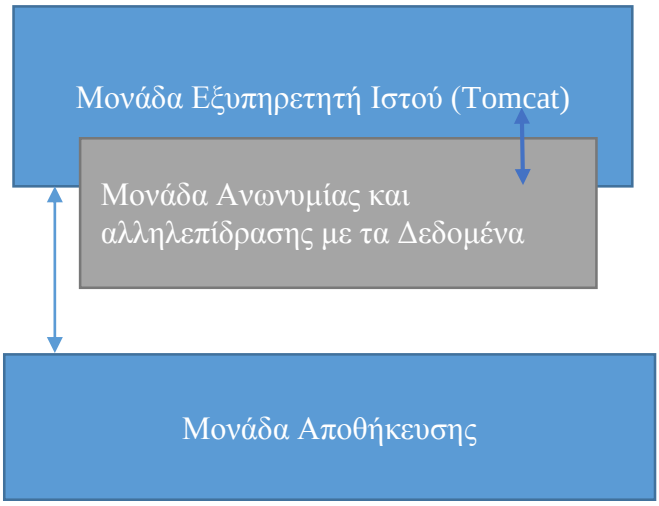
Κεφάλαιο 5

Υλοποίηση του Συστήματος

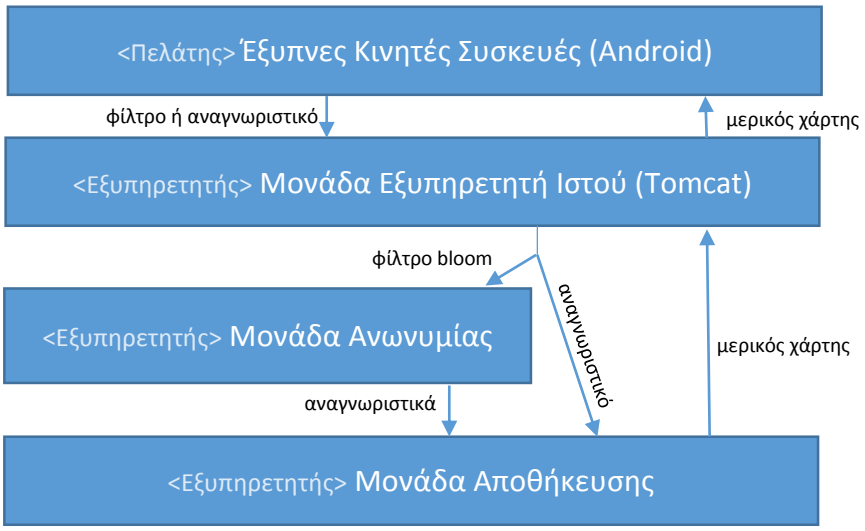
Η υλοποίηση του εξυπηρετητή, χωρίζετε σε τρία στρώματα τα οποία αλληλοεπιδρούν μεταξύ τους ούτως ώστε να αποτελέσουν μια Διεπαφή Προγραμματισμού Εφαρμογών (API) Ιστού 2.0 (Web2.0), για να παρέχουν σε τρίτους παραγόμενους Χάρτες Τιμών Ισχύος ανάλογα με τις επιλογές ανωνυμίας που θα τεθούν. Η υλοποίηση του πελάτη είναι στην πλατφόρμα ανοιχτού πηγαίου κώδικα Android, και μέρος της βασίστηκε σε προηγούμενες εκδόσεις της εφαρμογής Airplace [2]. Τα επίπεδα του συστήματος, όπως είναι χωρισμένα σε πελάτη και εξυπηρετητή, παρουσιάζονται γραφικά στο σχήμα 10.

5.1 Υλοποίηση Εξυπηρετητή

Τα τρία στρώματα που συνθέτουν τον εξυπηρετητή είναι: η μονάδα Ιστού, η μονάδα Ανωνυμίας και Αλληλεπίδρασης με τα δεδομένα, και η μονάδα Αποθήκευσης.
Η μονάδα Αποθήκευσης αποτελείται από το Κατανεμημένου Σύστημα Αρχείων (Distributed File System) Hadoop, το οποίο αποτελεί ανοιχτή υλοποίηση του Google File System (GFS) [23]. Στην κορυφή του Συστήματος Αρχείων Hadoop, είναι εγκατεστημένη η κατανεμημένη βάση δεδομένων HBase, η οποία εξειδικεύεται για τεράστιους όγκους συνόλων δεδομένων (datasets) και αποτελεί την ανοιχτή υλοποίηση του BigTable [24] της Google.
Η μονάδα Ανωνυμίας και Αλληλεπίδρασης με τα δεδομένα, αποτελεί το ενδιάμεσο στρώμα, μεταξύ της Μονάδας Αποθήκευσης και του εξυπηρετητή Ιστού. Σε αυτό το στρώμα



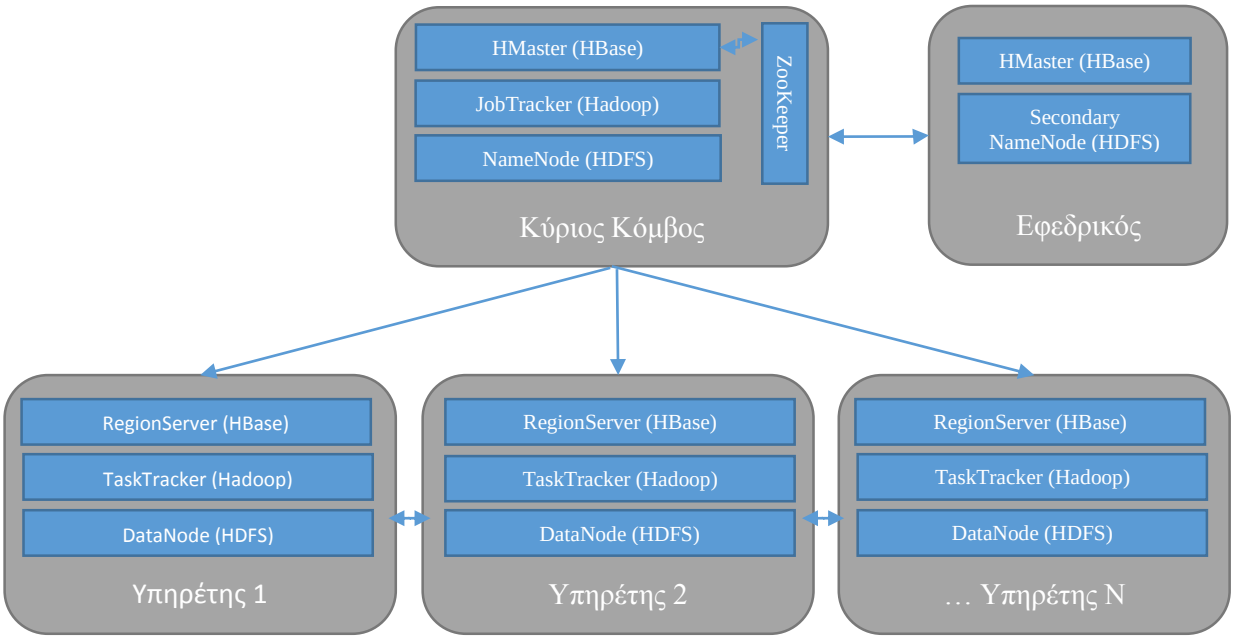
Σχήμα 16: *Τα τρία στρώματα που συνθέτουν τον εξυπηρετητή*



Σχήμα 17: *Αλληλεπίδραση μεταξύ των στρωμάτων του συστήματος σε πελάτη και εξυπηρετητή*

χρησιμοποιείται η Διεπαφή Προγραμματισμού Εφαρμογών (API) της βάσης δεδομένων HBase σε γλώσσα προγραμματισμού Java.

Η μονάδα Ιστού αποτελείται από εξυπηρετητή ιστού Apache Tomcat. Η επιλογή χρήσης της τεχνολογίας Java Servlets αποδείχθηκε η πιο αποδοτική, εν συγκρίση με άλλους εξυπηρετητές ιστού που χρησιμοποιήθηκαν αρχικά, όπως για παράδειγμα τον επίσης Ανοιχτού και Δωρεάν Πηγαίου Κώδικα (FOSS) Apache PHP.



Σχήμα 18: *Κόμβοι Συμπλέγματος*

5.1.1 Αρχιτεκτονική συμπλέγματος του συστήματός

Η αρχιτεκτονική του συμπλέγματος από κόμβους, αποτελείται από μια μηχανή, τον κύριο κόμβο, μια μηχανή τον εφεδρικό κόμβο, και τρεις μηχανές υπηρέτες.

Ο κύριος κόμβος, για το HDFS τρέχει την διεργασία NameNode, η οποία γνωρίζει την φυσική τοποθεσία των αρχείων χωρίς να γνωρίζει κάτι περισσότερο σχετικά με το περιεχόμενο τους. Για το πλαίσιο Hadoop υπάρχει ο JobTracker, ο οποίος θα διαμοιράζει υποεργασίες Χαρτογράφησης και Μείωσης στους Υπηρέτες που βρίσκονται κοντά στα δεδομένα. Για την HBase υπάρχει η διεργασία HMaster η οποία περιλαμβάνει και το πλαίσιο του ZooKeeper. Η HBase είναι υπεύθυνη για τον διαχωρισμό των περιοχών του πίνακα της βάσης δεδομένων στους υπηρέτες, την αναδιάρθρωση των περιοχών, αλλά και την ίση κατανομή και εξισορρόπηση του φόρτου εργασίας.

Ο εφεδρικός κόμβος τρέχει την διεργασία SecondaryNameNode, η οποία είναι η εφεδρική διεργασία της NameNode για το HDFS, και θα τεθεί αυτόματα σε λειτουργία σε περίπτωση σφάλματος του κυρίου κόμβου. Επίσης τρέχει και η κύρια λειτουργεία της βάσης δεδομένων HBase η οποία είναι ανενεργή, εκτός και αν προκύψει κάποιο πρόβλημα στον κύριο κόμβο, οπόταν ο ZooKeeper θα την ενεργοποιήσει.

Οι υπηρέτες τρέχουν όλοι τις ίδιες διεργασίες. Στο πιο χαμηλό στρώμα, για το HDFS τρέχει η διεργασία DataNode, η οποία είναι υπεύθυνη για την τροφοδότηση πελατών με τα δεδομένα, ή για το γράψιμο νέων δεδομένων. Είναι ο χώρος που βρίσκονται φυσικά τα δεδομένα. Για το Hadoop τρέχει η διεργασία TaskTracker, η οποία είναι υπεύθυνη για την εκτέλεση

υποεργασιών που της αναθέτει η διεργασία JobTracker. Οι υποεργασίες αυτές είναι Χαρτογράφησης ή Μείωσης. Στο πιο πάνω επίπεδο, για την HBase, τρέχει η διεργασία RegionServer η οποία είναι υπεύθυνη για την διαχείριση των περιοχών του πίνακα της βάσης δεδομένων που της αναθέτει η αντίστοιχη κύρια διεργασία της HBase, η HMaster.
Στο σύμπλεγμα οι μηχανές διαθέτουν 2.7 GB κύριας μνήμης, και μοιράζονται 4 φυσικούς επεξεργαστές. Το λειτουργικό σύστημα είναι το GNU Linux/CentOS, στο οποίο τρέχει το Hadoop/HDFS και η HBase.

5.1.2 Μονάδα Ανωνυμίας και αλληλεπίδρασης με τα δεδομένα

Η μονάδα Ανωνυμίας και Αλληλεπίδρασης με τα δεδομένα αποτελεί μέρος του κώδικα των Java Servlets στον εξυπηρετητή Ιστού Apache Tomcat. Η μονάδα αυτή κάνει χρήση της Διεπαφής Προγραμματισμού Εφαρμογών (API) σε γλώσσα JAVA της βάσης δεδομένων HBase. Η τεκμηρίωση για την διεπαφή αυτή, είναι διαθέσιμη στο διαδίκτυο[9].
Κατά την αρχικοποίηση του Java Servlet, ανοίγονται συνδέσεις προς τους πίνακες της κατανεμημένης βάση δεδομένων HBase, οι οποίες παραμένουν ανοιχτές μέχρι τον τερματισμό του εξυπηρετητή Ιστού. Οποιεσδήποτε αλληλεπιδράσεις με την Μονάδα Αποθήκευσης γίνονται μέσω αυτών των συνδέσεων, χωρίς να χρειαστεί να ανοίξει κάποια νέα σύνδεση.
Για την λήψη δεδομένων από την κατανεμημένη βάση δεδομένων HBase, αποκτάτε σαρωτής στα δεδομένα στον οποίο θέτονται συγκεκριμένα φίλτρα, ούτως ώστε να εξαχθεί το επιθυμητό αποτέλεσμα.
Ακόμη χρησιμοποιούνται μέθοδοι Χαρτογράφησης και Μείωσης (Map/Reduce), οι οποίες αποτελούν το μοντέλο εξαγωγής δεδομένων από βάσεις δεδομένων για μεγάλα δεδομένα, όπως την HBase. Το μοντέλο αυτό, κατά την αύξηση του όγκου δεδομένων, είναι πολύ πιο αποδοτικό από τις παραδοσιακές σχεσιακές βάσεις δεδομένων (RDBMS).

5.1.3 Εξυπηρετητής Ιστού Apache Tomcat

Ο εξυπηρετητής ιστού Apache Tomcat, κρίθηκε ως ο καταλληλότερος μεταξύ άλλων εξυπηρετητών, όπως για παράδειγμα τον Apache PHP. Ο εξυπηρετητής PHP υστερούσε σε χρόνο και κατανάλωση μνήμης έναντι του εξυπηρετητή Tomcat, και ο λόγος ήταν ότι δέσμευε μνήμη για κάθε μια αλληλεπίδραση με την μονάδα Ανωνυμίας και Αλληλεπίδρασης με τα δεδομένα. Οι αλληλεπιδράσεις αυτές, γίνονταν μέσω ενός Αρχείου Java (Java Archive - jar), το οποίο χρησιμοποιούσε την Διεπαφή Προγραμματισμού Εφαρμογών της Αποθηκευτικής

[9] HBase Java API: http://hbase.apache.org/apidocs

Μονάδας. Σε κάποιες πρόχειρες μετρήσεις μεταξύ του εξυπηρετητή Apache PHP και Apache Tomcat, ο Tomcat ήταν 5 φορές πιο γρήγορος για την λήψη 15 Χαρτών Τιμών Ισχύος, έναντι του PHP για την λήψη μόλις ενός Χάρτη Τιμών Ισχύος.

Επίσης η επιλογή της γλώσσας προγραμματισμού Java, ήταν η σοφότερη μιας και οι Διεπαφές Προγραμματισμού Εφαρμογών της βάσης δεδομένων HBase για άλλες γλώσσες (πχ, PHP) χαρακτηρίζονταν ως ασταθής (unstable) και ημιτελής.

Χρησιμοποιώντας Java η μονάδα Ανωνυμίας και Αλληλεπίδρασης με τα δεδομένα αποτελεί μέρος του σώματος ενός Java Servlet, χωρίς να χρειάζεται να δεσμεύετε μνήμη για κάθε ξεχωριστή εκτέλεση, εν αντίθεση με τα Αρχεία Java (jar). Έτσι κατά την αρχικοποίηση του Servlet δεσμεύονται οι απαραίτητοι πόροι, οι οποίοι επαναχρησιμοποιούνται και ανακυκλώνονται αποδοτικά, μόνο και όταν αυτό είναι αναγκαίο.

Ο εξυπηρετητής ιστού λειτουργεί με βάση τα πρότυπα Ιστού 2.0. Μπορούν να τεθούν παράμετροι για τον τύπο της επερώτησης (παράμετρος «mac»), το φίλτρο που θα τεθεί (παράμετρος «filter»), και επιπρόσθετες πληροφορίες (παράμετρος «extras») που αφορούν μορφή εξόδου που θα έχει ο Χάρτης Τιμών Ισχύος. Ο τρόπος με τον οποίον εισάγονται τα φίλτρα είναι η μέθοδος GET του πρωτοκόλλου HTTP. Ο τύπος της επερώτησης μπορεί να είναι «mac» ή «bloom». Για τους δύο αυτούς τύπους, χρησιμοποιείται το πεδίο φίλτρου «filter», το οποίο περιέχει ένα ή περισσότερα αναγνωριστικά «mac» (δηλαδή BSSID ενός Σημείου Πρόσβασης), διαχωρισμένα με κόμμα, για την πρώτη περίπτωση, και φίλτρο «bloom» για την δεύτερη περίπτωση.

Παραδείγματα χρήσης του «mac» είναι:

`type=mac&filter=bssid1` ή `type=mac&filter=bssid1,bssid2,bssid3`

Η περίπτωση χρήσης ενός «mac» χρησιμοποιείται όταν ο χρήστης ενδιαφέρετε μόνο για απόκρυψη (cloaking) ακριβής τοποθεσίας. Δηλαδή αφήνει τον εξυπηρετητή να γνωρίζει μια ευρύτερη τοποθεσία στην οποία βρίσκετε, χωρίς να μπορεί να ξέρει το ακριβές σημείο (π.χ., ο εξυπηρετητής γνωρίζει ότι κάποιος χρήστης βρίσκετε στο κτίριο Πληροφορικής στο Πανεπιστήμιο Κύπρου, όμως δεν γνωρίζει σε ποιο σημείο ακριβώς στο κτίριο είναι, ανά πάσα στιγμή). Η περίπτωση χρήστης πολλών χαρακτηριστικών «mac» χρησιμοποιείτε όταν ο χρήστης θέλει να διοχετεύσει στον εξυπηρετητή Ιστού πολλά αναγνωριστικά Σημείων Πρόσβασης, των οποίων οι μερικοί Χάρτες Τιμών Ισχύος θα εξαχθούν από την Μονάδα Αποθήκευσης. Συγκεκριμένα χρησιμοποιείτε στην οικογένεια αλγορίθμων Temporal VectorMap (TVM), για όλες τις γεωτοποθετήσεις πλην της πρώτης.

Παράδειγμα χρήστης του «bloom» είναι:

`type=bloom&filter=010110`

Η περίπτωση χρήσης «bloom», χρησιμοποιείται για αίτηση χαρτών από τον εξυπηρετητή με κάποιο φίλτρο τύπου bloom. Συγκεκριμένα χρησιμοποιείται στον αλγόριθμο VectorMap

(VM), κάθε φορά που εκτελείται η διαδικασία γεωτοποθέτησης, αλλά και κατά την πρώτη γεωτοποθέτηση στον αλγόριθμο Temporal VectorMap (TVM).

Επιπρόσθετα, υπάρχει τρίτη παράμετρος στην αίτηση, με το όνομα «extras», η οποία διοχετεύει επιπλέον πληροφορίες που αφορούν την μορφή απάντησης που θα λάβει ο πελάτης από τον εξυπηρετητή.

Στην περίπτωση τύπου «mac» ή «bloom», με την παράμετρο extras=1 υποδεικνύεται στον εξυπηρετητή να παράξει Χάρτη Τιμών Ισχύος σε μορφή «ζεύξης κλειδιών-τιμής» (key-value pairs).

Μια γραμμή του χάρτη αυτής της μορφής είναι:

```
x:y:h, mac1=rss1, mac2=rss2, mac3=rss3
```

όπου το κλειδί αποτελείται από τις συντεταγμένες και το αζιμούθιο (heading) της κινητής συσκευής, και ακολουθούν τα Σημεία Πρόσβασης με τις Λαμβανόμενες Τιμές Ισχύος (RSS). Τα θετικά αυτής της προσέγγισης είναι ότι δεν συμπεριλαμβάνονται στον Χάρτη Τιμών Ισχύος αρκετές τιμές οι οποίες είναι μηδενικές (δηλαδή NaN = -110), και έτσι σε μεγάλους και συνενωμένους Χάρτες Τιμών Ισχύος, το μέγεθος παραμένει σχετικά μικρό. Τα αρνητικά είναι ότι υπάρχει επανάληψη του ίδιου αναγνωριστικού (BSSID) ενός σημείο πρόσβασης, για όλα τα σημεία στα οποία το Σημείο Πρόσβασης είχε μη μηδενική τιμή. Αυτό γίνετε εμφανές όταν οι Χάρτες Τιμών Ισχύος είναι μικροί έως κανονικοί σε μέγεθος. Επίσης αυτή η μορφή θα πρέπει να τύχει μεγάλης επεξεργασίας από τις κινητές συσκευές μέχρι να φτάσει στην μορφή κατακερματισμένου χάρτη (hashmap) ούτως ώστε να μπορεί να χρησιμοποιηθεί για διαδικασίες γεωτοποθέτησης.

Στην περίπτωση τύπου «mac» ή «bloom», με την παράμετρο extras=2, υποδεικνύεται στον εξυπηρετητή να συνθέσει τον Χάρτη Τιμών Ισχύος σε μορφή ίδια με αυτή που χρησιμοποιείτο στην εφαρμογή Airplace. Ο παραγόμενος χάρτης περιλαμβάνει όλες τις γραμμές των Σημείων Πρόσβασης που έχουν μη μηδενικές τιμές.

Ένα παράδειγμα χρήσης της τιμής extras=2 είναι:

```
# X, Y, HEADING, mac1, mac2, mac3
x:y:h,-107.2,-92.6,-110
```

όπου η πρώτη γραμμή περιέχει όλα τα αναγνωριστικά Σημείων Πρόσβασης, και οι υπόλοιπες γραμμές τις συντεταγμένες μαζί με το αζιμούθιο, ακολουθούμενες από ένα πίνακα τιμών Ισχύος Λαμβανόμενου Σήματος (RSS).

Το θετικό αυτό της προσέγγισης είναι ότι παράγετε ο χάρτης σε μορφή που δεν χρειάζεστε περαιτέρω επεξεργασία από τις κινητές συσκευές, πέραν μιας απλής ανάγνωσης. Επίσης για μικρούς Χάρτες Τιμών Ισχύος, το μέγεθος παραμένει αρκετά μικρό, αφού δεν υπάρχει επανάληψη στα αναγνωριστικά των Σημείων Πρόσβασης, εν αντίθεση με την μέθοδο 1. Παρόλα αυτά, σε μεγαλύτερους Χάρτες Τιμών Ισχύος, ή σε πολλούς συνενωμένους Χάρτες Τιμών Ισχύος (π.χ. όταν χρησιμοποιούνται πολλά αναγνωριστικά «mac», διαχωρισμένα

μεταξύ τους με κόμμα, ή όταν κάποιο φίλτρο bloom είχε πολλές ψευδο-θετικές εκτιμήσεις) το μέγεθος αυξάνετε εκθετικά. Η αιτία είναι οι πολλαπλές μηδενικές τιμές (πχ, NaN = -110), οι οποίες τώρα αποτελούν μέρος στον Χάρτη Τιμών Ισχύος.

Για την διατήρηση του χαμηλού μεγέθους σε παραγόμενους Χάρτες Τιμών Ισχύος, αλλά παράλληλα και την διατήρηση του πλεονεκτήματος της προσέγγισης Airplace, όπου οι κινητές συσκευές δεν έπρεπε να σπαταλήσουν πολύτιμη ενέργεια και χρόνο για δημιουργία του χάρτη κατακερματισμού (hashmap), επινοήθηκε μια νέα μορφή παραγόμενων χαρτών, η οποία ενεργοποιείτε με την χρήση της παραμέτρου extras=3. Στην μορφή αυτή, αντί να δημιουργηθεί ένας χάρτης ο οποίος να περιλαμβάνει τιμές πολλών αναγνωριστικών Σημείων Πρόσβασης, δημιουργείτε αντικείμενο JavaScript (JSon), το οποίο περιέχει τους χάρτες αυτούς, διαχωρισμένους μεταξύ τους. Αυτή η μορφή χρησιμοποιείτε σε περιπτώσεις πολλαπλών αναγνωριστικών Σημείων Πρόσβασης με την χρήση του «mac», ή πολλαπλές αντιστοιχήσεις σε αναγνωριστικά Σημείων Πρόσβασης με την χρήστη του «bloom».

Πιο αναλυτικά, αφού ανακτηθούν τα δεδομένα για N αναγνωριστικά Σημείων Πρόσβασης από την μονάδα Αποθήκευσης, αντί να επεξεργαστούν και να παραχθεί ένας ενιαίος Χάρτης Τιμών Ισχύος και κατά συνέπεια αρκετά μεγάλος (ειδικά αν αυτό γινόταν με την χρήστη της μεθόδου extras=2), διαχωρίζονται και αποθηκεύονται σε N λίστες, ανάλογα με το αναγνωριστικό που προορίζονται. Στην συνέχεια δημιουργούνται N χάρτες ο οποίοι έχουν την πιο κάτω μορφή:

```
# X, Y, HEADING, mac1, mac2, mac3, mac4
x:y:h -107.2,-92.6,,,
```

Η μορφή αυτή είναι πανομοιότυπη με την μορφή Χαρτών Τιμών Ισχύος που χρησιμοποιείτε στο Airplace, με την διαφορά ότι οι μηδενικές τιμές κωδικοποιούνται τώρα με την κενή λέξη, για εξοικονόμηση περισσότερου χώρου.

Το διαχωρισμένο αντικείμενο απάντησης χωρίζετε σε δύο μέρη. Το πρώτο μέρος (η επικεφαλίδα) περιέχει τον αριθμό των αναγνωριστικών Σημείων Πρόσβασης που αντιστοίχησαν στην επερώτηση και μια εκ των τοποθεσιών τους. Στο δεύτερο μέρος ακολουθούν οι N χάρτες που δημιουργήθηκαν. Μεταξύ των δύο μερών του αντικειμένου, υπάρχει αντιστοίχηση, ούτως ώστε να μπορούν να αγνοηθούν συγκεκριμένοι χάρτες, στην περίπτωση όπου οι έξυπνες κινητές συσκευές δεν τους χρειάζονται, ή όταν πρέπει να εκτελεστούν εξειδικευμένες λειτουργίες (π.χ. σε περίπτωση αντιστοιχίας Σημείων Πρόσβασης σε δύο περιοχές στην Λευκωσία, μία περιοχή στο Παρίσι, και μία περιοχή στο Τόκυο, και επιθυμητή ανωνυμία χρήστη=3, να διασφαλιστεί ότι δεν θα επιλεγούν και τα δύο σημεία που βρίσκονται στην Λευκωσία).

Με την προσέγγιση extras=3 εκμεταλλευόμαστε τα θετικά των προσεγγίσεων extras=1 και extras=2, ενώ παράλληλα αποφεύγουμε τα αρνητικά τους. Το τελικό αποτέλεσμα περιλαμβάνει πολλούς και μικρούς σε μέγεθος Χάρτες Τιμών Ισχύος, όπου δεν υπάρχει

επανάληψη των αναγνωριστικών των Σημείων Πρόσβασης, καθώς επίσης και περιορισμένες μηδενικές τιμές, αφού αποφεύγετε η συνένωσή τους. Τέλος οι κινητές συσκευές δεν θα πρέπει να επεξεργαστούν τον χάρτη για να παράξουν τον χάρτη κατακερματισμού (hashmap) που είναι απαραίτητος για γεωτοποθεσία.

Ένα παράδειγμα της διαχωρισμένης μορφής αντικειμένου για δύο σημεία είναι το εξής:

```
{ 'code': '1', 'matches': [
                              { 'mac':'bssid1', 'loc':'lat1,lon1' },
                              { 'mac':'bssid2', 'loc':'lat2,lon2' }
                           ],
'rmaps': [
            { μερικός χάρτης1 },
            { μερικός χάρτης2 }
          ]
}
```

Επιπρόσθετα, επιλύθηκε το πρόβλημα επιπλέων φόρτου εργασίας που υπήρχε μεταξύ των αλγορίθμων Temporal VectorMap και Random Temporal Vectormap. Ο Temporal Vectormap απαιτούσε περισσότερο χρόνο για εντοπισμό ρεαλιστικών γειτονικών σημείων πρόσβασης στις ψεύτικες τοποθεσίες, ενώ με τους μικρότερους σε μέγεθος και ξεχωριστούς Χάρτες Τιμών Ισχύος ο χρόνος αυτός είναι μηδενικός.

Στο κεφάλαιο 6, υπάρχει πειραματική αποτίμηση στα μεγέθη των μερικών Χαρτών Τιμών Ισχύος, μεταξύ των τριών εκδόσεων.

Κλείνοντας την υλοποίηση στον Apache Tomcat, το πρότυπο για εισαγωγή παραμέτρων μέσω ενός Ενιαίου Εντοπιστή Πόρων (URL) είναι το εξής:

protocol:server_url:port/path/?type=*τύπος_επερώτησης*&filter=*φίλτρο*&extras=*έξτρα*

Ένα URL πραγματικής μεταφοράς παραμέτρων είναι

http://vectormap1.in.cs.ucy.ac.cy:8080/vm/Server?type=bloom&filter=01100&extras=3

Στο πιο πάνω παράδειγμα, ενημερώνουμε τον εξυπηρετητή ότι θα χρησιμοποιηθεί η μέθοδος εύρεσης μερικού Χάρτη Τιμών Ισχύος χρησιμοποιώντας το φίλτρο bloom «01100», με επιπρόσθετες πληροφορίες τον κωδικό τρία (3), για να λάβουμε τον παραγόμενο χάρτη σε διαχωρισμένη μορφή.

5.1.4 Παραγωγή ρεαλιστικών χαρτών για σκοπούς αποτίμησης αλγορίθμων

Για την ορθή αποτίμηση της απόδοσης της βάσης δεδομένων HBase χρειάζεται μεγάλος όγκος συνόλων δεδομένων (datasets). Αφού το μέγεθος των πραγματικών χαρτών δεν ήταν αρκετό, δημιουργήθηκαν ρεαλιστικοί χάρτες, οι οποίοι εισάχθηκαν στην μονάδα Αποθήκευσης, δηλαδή το σύμπλεγμα που αποτελεί την HBase. Για την δημιουργία των χαρτών αυτών, εξάχθηκαν μετρήσεις Ισχύος Λαμβανόμενου Σήματος (RSS), μαζί με αποτυπώματα του Παγκόσμιου Συστήματος Γεωτοποθέτησης (GPS), από τον οργανισμό Crawdad. Η καταγραφή

αυτών των μετρήσεων στον οργανισμό Crawdad έγινε μέσω διαδικασιών γεωγραφικού εντοπισμού (wardriving) με τους αισθητήρες Παγκόσμιου Συστήματος Γεωτοποθέτησης (GPS), Ασύρματου Δίκτυου (WiFi), και Δικτύου Κινητής Τηλεφωνίας (GSM). Η μορφή μερικών αρχείων που έπειτα από επεξεργασία μετατράπηκαν σε Χάρτες Τιμών Ισχύος ήταν η πιο κάτω:

```
GPS line: latitude, longitude
Wifi Line: Τιμές Ισχύος Λαμβανόμενου Σήματος
…
Wifi Line: Τιμές Ισχύος Λαμβανόμενου Σήματος
GSM Line : Τιμές Δικτύων Κινητής Τηλεφωνίας
…
```

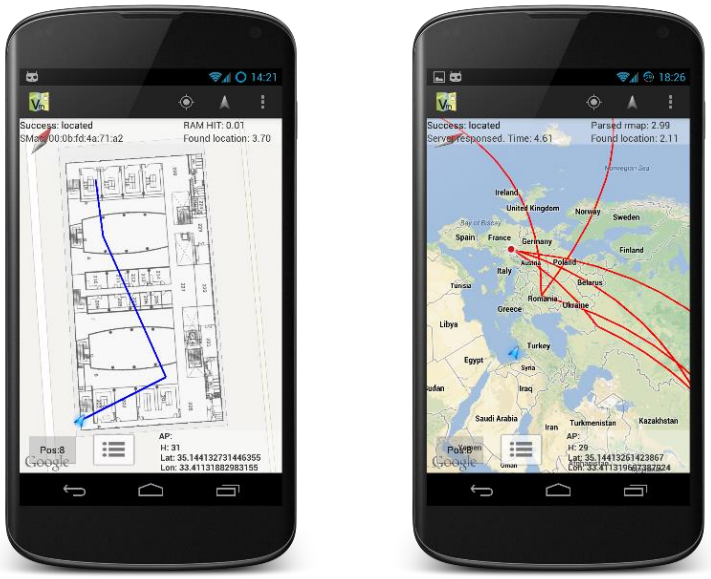
Για την παραγωγή Χάρτη Τιμών Ισχύος, γινόταν αντιστοίχηση της πρώτης εμφανιζόμενης τιμής γεωτοποθεσίας (latitude, longitude) σύμφωνα με τις μετρήσεις του Παγκόσμιου Συστήματος Συντεταγμένων με τις ακολουθούμενες μετρήσεις των Σημείων Πρόσβασης Ασύρματων Δικτύων.

Επειδή οι επιπλέων χάρτες που εξάχθηκαν δεν ήταν αρκετοί, ούτως ώστε να αποτιμηθεί η βάση δεδομένων HBase, αλλά και για σκοπούς αποτίμησης των αλγορίθμων ανωνυμίας, χρειάστηκε η αντιγραφή τους σε σημεία διάσπαρτα ανά τον πλανήτη. Συγκεκριμένα, τα εξαγόμενα κτίρια από τον οργανισμό Crawdad «αντιγράφηκαν» στο κέντρο όλων των πόλεων του πλανήτη. Για τον εντοπισμό των γεωγραφικών τοποθεσιών στις πόλεις, εξάχθηκαν δεδομένα από την ιστοσελίδα TimeGenie[10], η οποία περιέχει συντεταγμένες του κέντρου 7,500 πόλεων στον πλανήτη. Ακολούθως, απαλείφοντας τα απαραίτητα δεκαδικά ψηφία στις τιμές γεωτοποθεσίας των πραγματικών ή ρεαλιστικών κτιρίων, και την άθροιση τους με τις τιμές του κέντρου της κάθε πόλης, επιτεύχθηκε η διατήρηση των πραγματικών αποστάσεων. Επίσης, για τους χάρτες των αντιγραμμένων κτιρίων, παράχθηκαν μοναδικά τεχνητά αναγνωριστικά Σημείων Πρόσβασης (BSSID), με σήμανση, ούτως ώστε σε μελλοντικό στάδιο να μπορούν να ξεχωρίσουν από τα πραγματικά κτίρια. Το σύνολο δεδομένων μετά την «κλωνοποίηση» Χαρτών Τιμών Ισχύος σε 7,500 πόλεις του πλανήτη, ανέρχεται στα 80 GB.

5.2 Υλοποίηση Πελάτη

Η υλοποίηση του πελάτη αποτελείτε από εφαρμογή στο Λειτουργικό Σύστημα Android. Στην εφαρμογή αυτή, χρησιμοποιήθηκαν μέρη από κώδικα, προηγούμενων εκδόσεων της εφαρμογής Airplace. Έγιναν αλλαγές ούτως ώστε το τελικό αποτέλεσμα να περιλαμβάνει τις τελευταίες τροποποιήσεις που εισηγήθηκαν στο προηγούμενο σύστημα γεωτοποθεσίας, που ήταν η χρήση πυξίδας για τον καθορισμό του χάρτη κατακερματισμού Χαρτών Τιμών Ισχύος

[10] TimeGenie: World Cities Geolocations, www.timegenie.com



Εικόνα 19: *Η εφαρμογή γεωτοποθεσίας Airplace με την χρήση του αλγορίθμου Vectormap.*

που θα χρησιμοποιείτο, και η χρήση συντεταγμένων σύμφωνα με το Παγκόσμιο Σύστημα Συντεταγμένων (Latitude, Longitude), αντί της χρήσης τοπικού συστήματος συντεταγμένων. Επίσης έγινε χρήση της εγγενής (native) Διεπαφής Προγραμματισμού Εφαρμογών (API) των χαρτών Google, έκδοση δεύτερη[11] που προσφέρει αισθητικά καλύτερη απόδοση σε σύγκριση με την αντίστοιχη εφαρμογή Ιστού, για τις έξυπνες κινητές συσκευές.

5.2.1 Ροή στην εφαρμογή

Για την υλοποίηση της ροής στην εφαρμογή, χρησιμοποιήθηκαν Υπηρεσίες[12], Ασύγχρονες Εργασίες[13], καθώς και Αναμεταδόσεις (Broadcasts)[14], οι οποίες αποτελούν μέρος του Εργαλείου Ανάπτυξης Λογισμικού (SDK) της πλατφόρμας Android.

Μια διαδικασία γεωτοποθέτησης ξεκινά με την ανάγνωση των πληροφοριών για τα Σημεία Πρόσβασης Ασύρματων Δικτύων που γειτνιάζουν με τον χρήστη. Ακολούθως, όταν οι τιμές αυτές είναι έτοιμες από το Λειτουργικό Σύστημα, με αναμετάδοση ενημερώνετε η εφαρμογή, η οποία με την σειρά της ενημερώνει τις πιο πρόσφατες τιμές των γειτονικών Σημείων Πρόσβασης. Έπειτα, αποφασίζετε ποιος Χάρτης Τιμών Ισχύος θα πρέπει να χρησιμοποιηθεί. Αν δεν υπάρχει ο χάρτης αυτός τοπικά, σε κάποια κρύπτη στην έξυπνη κινητή συσκευή, τότε ανάλογα με τον αλγόριθμο ανωνυμίας η εφαρμογή θα επικοινωνήσει με τον εξυπηρετητή για

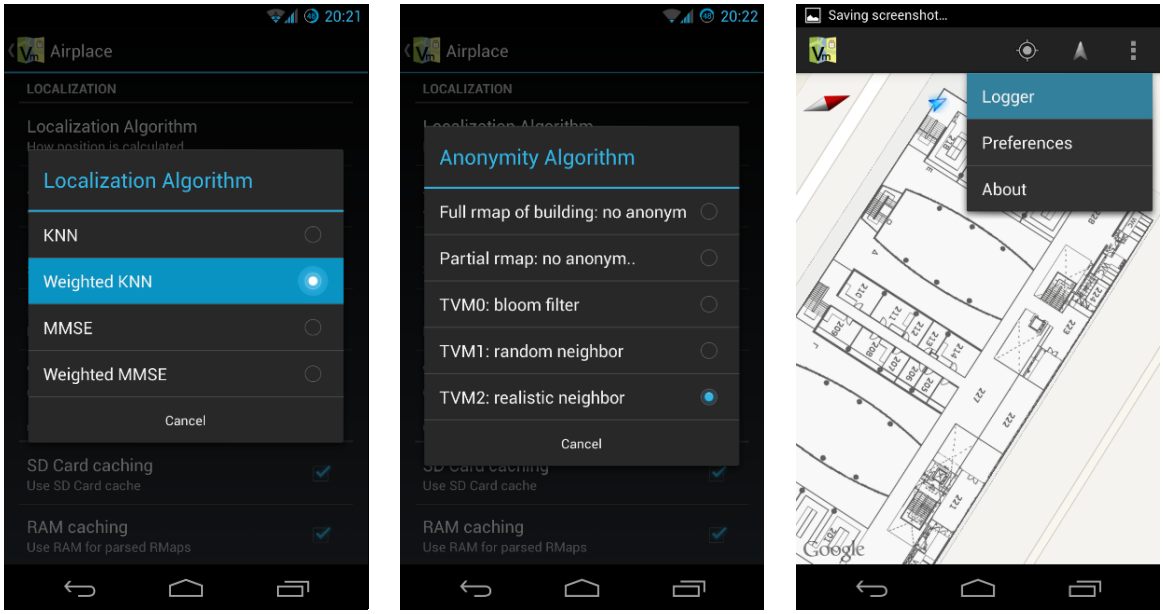
[11] Google maps Android API v2, https://developers.google.com/maps/documentation/android/
[12] Android Services: http://goo.gl/LACxr
[13] Android Asynchronous Tasks, http://goo.gl/Kz3Qh
[14] Android Broadcasts. http://goo.gl/IjCkM

Εικόνα 20: *Ο μετετρεμμένος Χάρτης Τιμών Ισχύος του πρώτου ορόφου στο κτίριο Πληροφορικής Πανεπιστημίου Κύπρου, από τοπικό σύστημα συντεταγμένων, στο Παγκόσμιο Σύστημα Συντεταγμένων, όπως εμφανίζετε σε Χάρτες Google*

να λάβει τον νέο ενημερωμένο μερικό Χάρτη Τιμών Ισχύος. Μετά την λήψη του νέου χάρτη, αναλύετε και αποθηκεύετε σε προσωρινή μνήμη και στην συνέχεια χρησιμοποιείται για γεωτοποθέτηση.
Η υπηρεσίες που αποτελούν μέρος της ροής διακόπτονται προσωρινά όταν η εφαρμογή περιέλθει σε οποιαδήποτε άλλη κατάσταση εκτός της ενεργής.

5.2.2 Προσθήκες στην εφαρμογή

Οι αλγόριθμοι γεωτοποθεσίας KNN, KNN με βάρη, MSME και MSME με βάρη, ενημερώθηκαν, ούτως ώστε να λειτουργούν με βάση το αζιμούθιο(heading). Δηλαδή η επιλογή του χάρτη κατακερματισμού για να χρησιμοποιηθεί ως Χάρτης Τιμών Ισχύος για γεωτοποθέτηση, λαμβάνει υπόψιν τις τιμές του αζιμούθιου. Έτσι θα επιτυγχάνετε μεγαλύτερη ακρίβεια στις τιμές Ισχύος Λαμβανόμενου Σήματος των Σημείων Πρόσβασης που γειτνιάζουν τον χρήστη, αφού η στάση του σώματος του χρήστη αποτελεί εμπόδιο μεταξύ των γειτονικών κεραιών εκπομπής των Ασύρματων Δικτύων, και της κεραίας λήψης Ασύρματων Δικτύων στην κινητή του συσκευή.
Για την γεωτοποθέτηση σύμφωνα με το Παγκόσμιο Σύστημα Συντεταγμένων, μετατράπηκαν οι Χάρτες Τιμών Ισχύος από προηγούμενες μετρήσεις στο κτίριο Πληροφορικής του Πανεπιστημίου Κύπρου, από τοπικό σύστημα συντεταγμένων (που ήταν σε μέτρα), στο Παγκόσμιο Σύστημα Συντεταγμένων. Για να επιτευχθεί η μετατροπή των σημείων, χρησιμοποιήθηκαν οι παράμετροι μήκος και πλάτος του κτηρίου Πληροφορικής στο



Εικόνες 21: *Επιλογές για αλγορίθμους γεωτοποθεσίας και ανωνυμίας, και η μετάβαση στον Logger*

Πανεπιστήμιο Κύπρου, καθώς και τρείς από τις άκριες του κτιρίου, όπως αυτές εξάγονταν από τους χάρτες Google. Έπειτα, για κάθε μετακίνηση σε μέτρα στους δύο άξονες του τοπικού συστήματος συντεταγμένων, πραγματοποιούνταν τέσσερις μετακινήσεις, δύο για κάθε άξονα, σε υπολογιζόμενη κλίμακα, στο Παγκόσμιο Σύστημα Συντεταγμένων. Τα αποτελέσματα επικυρώθηκαν με ψηφιακό χάρτη, στον οποίο τοποθετήθηκαν όλα τα μετετρεμμένα σημεία.

Επίσης για καλύτερη απόδοση της εφαρμογής, επαναχρησιμοποίηση δεδομένων, εξοικονόμηση πόρων δικτύου, αλλά και αποσυμφόρηση του εξυπηρετητή υλοποιήθηκε κρύπτη (cache) δύο επιπέδων. Το πρώτο επίπεδο κρύπτης είναι στην κύρια μνήμη, και το δεύτερο στην κάρτα μνήμης (sd card) της έξυπνης κινητής συσκευής.

Όταν κάποιος Χάρτης Τιμών Ισχύος μεταφερθεί από τον εξυπηρετητή Ιστού στην έξυπνη κινητή συσκευή, τότε αυτός αποθηκεύετε στην κάρτα μνήμης της συσκευής, αν αυτό είναι επιθυμητό μέσω των ρυθμίσεων. Στην συνέχεια, ο χάρτης αναλύετε και δημιουργούνται οι κατάλληλοι χάρτες κατακερματισμού (hashmaps) που χρησιμοποιούνται για γεωτοποθέτηση. Ακολούθως, ανάλογα με τις προτιμήσεις του χρήστη, ο αναλυμένος χάρτης αποθηκεύετε στην κρύπτη της κύριας μνήμης. Για την κρύπτη στην κύρια μνήμη χρησιμοποιείται ο αλγόριθμος Λιγότερο Πρόσφατα Χρησιμοποιούμενος (LRU), και ο αριθμός των υποδοχών της κρύπτης μπορεί να τεθεί δυναμικά από τον χρήστη.

Κατά την επιλογή του Χάρτη Τιμών Ισχύος για γεωτοποθέτηση, πρώτα ερευνάτε κατά πόσο κάποια από τα πιο κοντινά γειτνιάζουσα και εισακουόμενα Σημεία Πρόσβασης Ασύρματων Δικτύων, υπάρχουν στο πρώτο επίπεδο κρύπτης. Σε περίπτωση αποτυχίας, ερευνάτε κατά πόσο υπάρχουν στον δεύτερο επίπεδο κρύπτης. Αν και πάλι δεν εντοπιστεί ο Χάρτης Τιμών Ισχύος, τότε ανάλογα με τον αλγόριθμο ανωνυμίας, η κινητή συσκευή επικοινωνεί με τον εξυπηρετητή

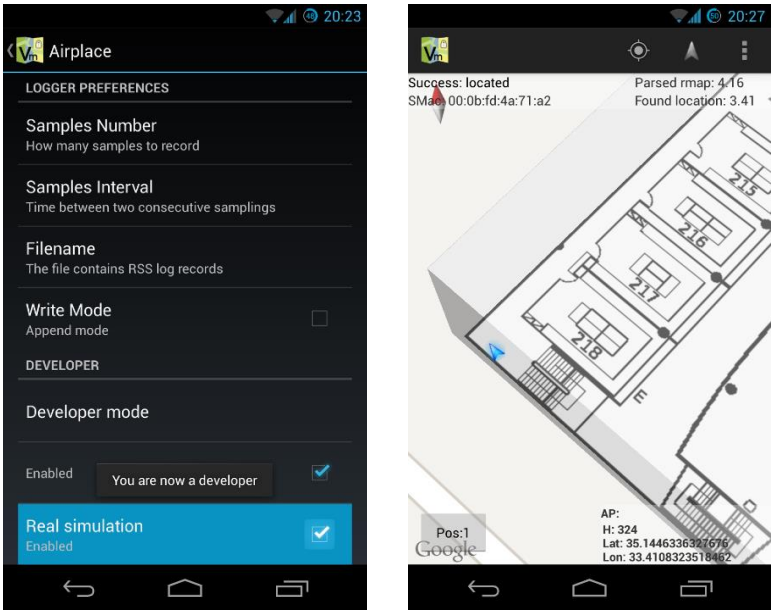
για την λήψη του ενημερωμένου χάρτη, με βάση το κοντινότερο Σημείο Πρόσβασης. Σε περίπτωση που ο χάρτης εντοπιστεί σε οποιαδήποτε από τις κρύπτες, τότε αφού δεν προκλήθηκε επικοινωνία με τον εξυπηρετητή, ο χρήστης δεν έδωσε απολύτως καμία πληροφορία όσο αφορά την φυσική του τοποθεσία.

Οι αλγόριθμοι ανωνυμίας που υλοποιήθηκαν όπως φαίνονται στην εικόνα 21 (μέση) είναι:
1) με την χρήση μερικού Χάρτη Τιμών Ισχύος (pRMap) 2) ο αλγόριθμος Vectormap (TVM0) και 3) οι αλγόριθμοι Random Temporal Vectormap (TVM1) και Temporal Vectormap (TVM2).

Ο αλγόριθμος με την χρήση μερικού Χάρτη Τιμών Ισχύος (pRMap), για να αιτηθεί νέο μερικό Χάρτη Τιμών Ισχύος, επικοινωνεί με τον εξυπηρετητή τροφοδοτώντας τον μόνο με το αναγνωριστικό του Σημείου Πρόσβασης Ασύρματου Δικτύου. Αυτό έχει ως αποτέλεσμα να προδίδει την τοποθεσία του, διατηρώντας ωστόσο απόκρυψη (cloaking). Δηλαδή ο εξυπηρετητής γνωρίζει πως ο χρήστης βρίσκετε εντός του μερικού Χάρτη Τιμών Ισχύος, δεν γνωρίζει όμως την ακριβή του τοποθεσία. Επίσης η ανταλλαγή μηνυμάτων στο δίκτυο είναι μικρότερη, μιας και ο χρήστης λαμβάνει μόνο ένα μερικό Χάρτη Τιμών Ισχύος, και όχι περισσότερους, πράγμα που συμβαίνει στις προσεγγίσεις της οικογένειας αλγορίθμων Vectormap και Temporal Vectormap.

Ο αλγόριθμος Vectormap (TVM0 ή VM), είναι η υλοποίηση σε πραγματικές κινητές συσκευές του αλγόριθμου όπως αυτός προτάθηκε στο [4], με κάποιες επιπρόσθετες βελτιώσεις. Ο χρήστης, κάθε φορά που χρειάζεται να αιτηθεί νέο μερικό χάρτη από τον εξυπηρετητή, δημιουργεί ένα φίλτρο bloom για το κοντινότερο Σημείο Πρόσβασης Ασύρματου Δικτύου που ακούει, κατ' αντίστοιχο τρόπο με τον εξυπηρετητή. Ακολούθως αποστέλλει το φίλτρο bloom στον εξυπηρετητή, και τότε λαμβάνει ένα διαχωρισμένο αντικείμενο μορφής JSon, όπως αυτό είχε περιγραφεί σε προηγούμενο κεφάλαιο. Το διαχωρισμένο αντικείμενο, περιέχει τον μερικό Χάρτη Τιμών Ισχύος για την πραγματική τοποθεσία του χρήστη, καθώς επίσης και ψευδο-θετικές εκτιμήσεις για περιοχές διάσπαρτες στον πλανήτη, ούτως ώστε να ξεγελαστεί ο εξυπηρετητής για την πραγματική τοποθεσία του χρήστη. Αν ο χρήστης επαναλάβει τον αλγόριθμο Vectormap, τότε η διαδικασία θα είναι ακριβώς η ίδια, με την μόνη διαφορά ότι οι ψευδο-θετικές εκτιμήσεις δεν θα έχουν κάποια χωρική σύνδεση μεταξύ τους, και αυτό θα έχει ως αποτέλεσμα η πραγματική θέση του χρήστη να αποκαλύπτεται.

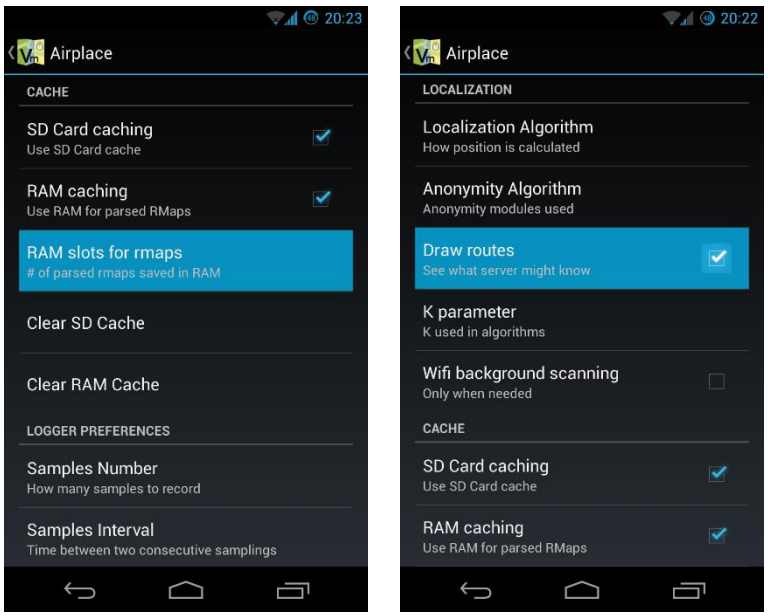
Ο αλγόριθμος Random Temporal Vectormap (TVM1), όταν χρειαστεί να επικοινωνήσει με τον εξυπηρετητή για την λήψη Χάρτη Τιμών Ισχύος νέας εμβέλειας, μόνο κατά την πρώτη φορά, εκτελεί την διαδικασία του Vectormap. Δηλαδή δημιουργεί φίλτρο bloom, το οποίο αποστέλλει στον εξυπηρετητή για να το ταυτοποιήσει με τον αληθινό Χάρτη Τιμών Ισχύος, αλλά και τις ψευδο-θετικές εκτιμήσεις. Ακολούθως, από το διαχωρισμένο αντικείμενο χαρτών



Εικόνα 22: *Ενεργοποίηση της πραγματικής προσομοίωσης και Προχωρημένες επιλογές προγραμματιστών*

που λαμβάνει, αναλύει και αποθηκεύει τον αληθινό Χάρτη Τιμών Ισχύος, αλλά και τον αριθμό από τις ψευδο-θετικές εκτιμήσεις χαρτών που επιθυμεί ο χρήστης. Στην συνέχεια, όταν βγει εκτός εμβέλειας ο χρήστης και επιβάλλετε επικοινωνία με τον εξυπηρετητή για την λήψη ενός νέου Χάρτη Τιμών Ισχύος, τότε ως νέο αληθινό Σημείο Πρόσβασης Ασύρματων Δικτύων, επιλέγετε το εκείνο που βρίσκετε ποιο κοντά του, και για τα νέα ψεύτικα σημεία πρόσβασης επιλέγει κάποια τυχαία σημεία που μπορούν να εξαχθούν από τους Χάρτες Τιμών Ισχύος των ψευδο-θετικών εκτιμήσεων. Η διαδικασία αυτή επαναλαμβάνετε για κάθε νέα αίτηση από τον εξυπηρετητή.

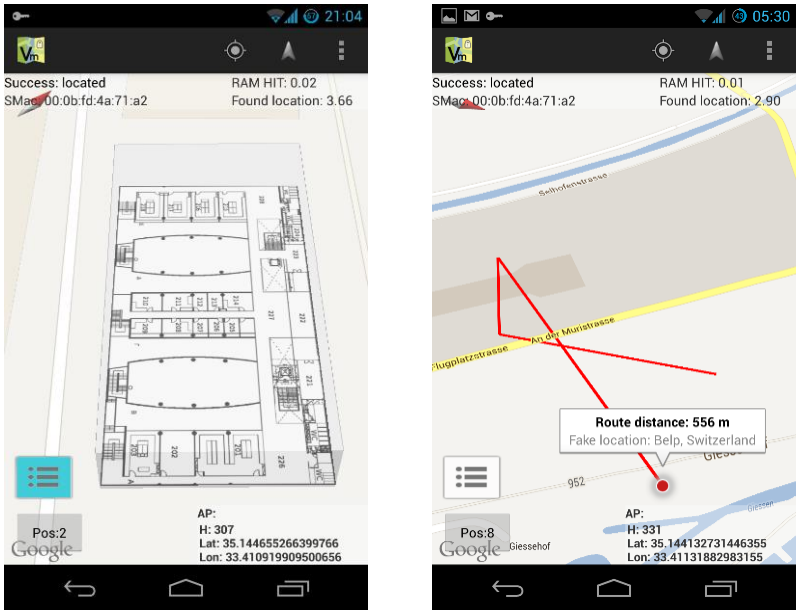
Ο αλγόριθμος Temporal Vectormap (TVM2), λειτουργεί σε παρόμοιο μοτίβο με τον Random Temporal Vectormap. Δηλαδή κατά την πρώτη επικοινωνία με τον εξυπηρετητή χρησιμοποιείται η μέθοδος Vectormap, ενώ κατά τις επόμενες αιτήσεις, χρησιμοποιούνται οι Χάρτες Τιμών ισχύος των εκάστοτε ψεύτικων Σημείων Πρόσβασης, για τον υπολογισμό των νέων ψεύτικων σημείων. Η διαφορά του Random Temporal Vectormap, με τον Temporal Vectormap, είναι ότι ο πρώτος διαλέγει τυχαία τους νέους ψεύτικους γείτονές του, ενώ ο τελευταίος επιλέγει ρεαλιστικούς γείτονες, όπως περιεγράφηκε σε προηγούμενο κεφάλαιο. Το αποτέλεσμα είναι οι ψεύτικες αποστάσεις που πιστεύει ο εξυπηρετητής ότι καλύφθηκαν χρησιμοποιώντας τον Temporal Vectormap να είναι πιο ρεαλιστικές και όμοιες με την πραγματική απόσταση που διανύθηκε. Πράγμα που καθιστά αδύνατη την προσπάθεια του εξυπηρετητή να ξεχωρίσει ποια είναι η πραγματική τοποθεσία και πορεία του χρήστη από τις ψευδο-θετικές εκτιμήσεις.



Εικόνα 23: *Επιλογή για τις θέσεις μνήμης των αναλυμένων (parsed) Χαρτών Τιμών Ισχύος, ενεργοποίησης/απενεργοποίησης/εκκαθάρισης κρυπτών, και επιλογή για σχεδίαση των διαδρομών που μπορεί να εξάγει ο εξυπηρετητής με βάση τις πληροφορίες που τον τροφοδοτεί ο πελάτης*

Για την ευκολότερη αποτίμηση των αλγορίθμων αλλά και για εξοικονόμηση χρόνου, υλοποιήθηκε μέθοδος επονομαζόμενη «Πραγματική Προσομοίωση», η οποία μπορεί να ενεργοποιηθεί κάτω από τις προχωρημένες επιλογές προγραμματιστών στην εφαρμογή. Κατά την λειτουργεία «Πραγματικής Προσημείωσης» ο χρήστης εξάγει πληροφορίες από τα σημεία πρόσβασης που θα μπορούσε να ακούσει, αν βρισκόταν σε 8 υπαρκτά και διαφορετικά σημεία στο κτίριο Πληροφορικής στο Πανεπιστήμιο Κύπρου. Ακολούθως οι τιμές αυτές, μπορούν να χρησιμοποιηθούν σε συνδυασμό με όλες τις επιλογές όπως τα δύο επίπεδα κρύπτης, αλγόριθμους γεωτοποθέτησης, αλγόριθμους ανωνυμίας, για συγκρίσεις των επιμέρους συνδυασμών.

Επίσης στην εφαρμογή διαχειρίζονται ειδικές περιπτώσεις, όπως για παράδειγμα αλλαγή στην τοπολογία του δικτύου (π.χ. η πρόσθεση ενός νέου σημείου πρόσβασης ασύρματου δικτύου σε κάποιο κτίριο). Αυτό θα έχει ως αποτέλεσμα ο εξυπηρετητής να μην περιέχει πληροφορίες για το νεοεισαχθέν Σημείο πρόσβασης. Τότε στην έξυπνη κινητή συσκευή θα προστεθεί σημαία για το συγκεκριμένο σημείο πρόσβασης, και θα επαναληφθεί η διαδικασία επιλογής Σημείου Πρόσβασης για την εξεύρεση του μερικού Χάρτη Τιμών Ισχύος, αγνοώντας το σημείο αυτό.



Εικόνα 24: *Επιλογή αποκάλυψης υποθέσεων του εξυπηρετητή, και αποκάλυψη μιας εκ των ψεύτικων διαδρομών*

Επιπρόσθετα υπάρχουν επιλογές για δυναμική ενεργοποίηση ή απενεργοποίηση όλων των επιπλέων λειτουργιών της εφαρμογής, διαγραφή των κρυπτών, ενεργοποίηση προχωρημένων επιλογών προγραμματιστών, ενσωμάτωση επιλογών για χαρτογράφηση (Logger) και γεωτοποθέτηση (Tracker) που προϋπήρχαν, ούτως ώστε να επιτευχθεί ευκολότερη διεξαγωγή πειραμάτων. Οι χρόνοι όλων των επικοινωνιών καθώς και των διεργασιών που τρέχουν στο παρασκήνιο καταγράφονται, και προβάλλονται όταν η επιλογή προχωρημένων προγραμματιστών για επιπλέων πληροφορίες είναι ενεργοποιημένη.

5.2.3 Γραφική αναπαράσταση της ιδιωτικότητας του χρήστη

Ο χρήστης, κατά την επικοινωνία του με τον εξυπηρετητή, αποκαλύπτει πληροφορίες όσο αφορά την πραγματική του τοποθεσία. Όταν γίνει αυτό, στην γραφική διεπαφή εμφανίζετε μια νέα επιλογή για την παρατήρηση των τοποθεσιών αλλά και των διαδρομών που μπορεί να υποθέσει ο εξυπηρετητής. Επομένως ανά πάσα στιγμή οι πελάτες χρησιμοποιώντας τις έξυπνες κινητές συσκευές τους, μπορούν να δουν γραφικά τα συμπεράσματα που μπορεί να εξάγει ο εξυπηρετητής με βάση τις πληροφορίες που του παρείχαν, και να τις συγκρίνουν από πλευράς μοτίβου κίνησης, φυσικής τοποθεσίας αλλά και της συνολικής απόστασης της διαδρομής.

Κεφάλαιο 6

Πειραματική Αξιολόγηση

Σε αυτό το κεφάλαιο θα παρουσιαστεί η πειραματική αξιολόγηση των αλγορίθμων Vectormap, και Temporal Vectormap, όπως αυτοί προτάθηκαν στο [1], καθώς επίσης και πειράματα με συνδυασμούς χρήσης ή όχι κρυπτών, περιορισμούς στις κρύπτες, μετρήσεις ενεργειακής κατανάλωσης και άλλα. Αρχικά θα παρουσιαστεί η μεθοδολογία που ακολουθήθηκε και ακολούθως τα αποτελέσματα.

6.1 Μεθοδολογία για την πειραματική αποτίμηση

Δεδομένα Χαρτών Τιμών Ισχύος: Χρησιμοποιήθηκαν τα ακόλουθα σύνολα δεδομένων για τους Χάρτες Τιμών Ισχύος στις προσομοιώσεις με γνώμονα τα ίχνη (trace-driven):

i) **Χάρτης Τιμών Ισχύος κτηρίου Πληροφορικής Πανεπιστημίου Κύπρου:** Ο χάρτης αυτός παράχθηκε από την περισυλλογή δεδομένων στο κτήριο Πληροφορικής του Πανεπιστημίου Κύπρου. Χρησιμοποιήθηκαν τρείς κινητές συσκευές με λειτουργικό σύστημα Android (HTC Hero, HTC Desire και Samsung Nexus S), για την περισυλλογή 30 τιμών-αποτυπωμάτων Ισχύος Λαμβανόμενου Σήματος (RSS) σε 1500 ξεχωριστές τοποθεσίες στους τέσσερις ορόφους του κτιρίου (συνολικά 45,000 μετρήσεις τιμών Ισχύος Λαμβανόμενου Σήματος). Υπάρχουν 120 σημεία πρόσβασης Ασύρματων Δικτύων εγκατεστημένα στους τέσσερις ορόφους του κτιρίου, συμπεριλαμβανομένων και των σημείων πρόσβασης Ασύρματων Δικτύων γειτονικών κτιρίων, τα οποία μπορούν μερικός να εισακουστούν σε κάποια σημεία του κτιρίου. Κατά μέσω όρο, 10.6 σημεία πρόσβασης Ασύρματων Δικτύων εντοπίζονται ανά φυσική τοποθεσία. Τα δεδομένα περισυλλέχτηκαν περπατώντας σε διαδρομή που αποτελείται από 2900 τοποθεσίες.

ii) **Χάρτης Τιμών Ισχύος στο «ΚΟΙΟΣ»:** Τα δεδομένα αυτά περισυλλέχτηκαν μέσα σε ένα τυπικό περιβάλλον γραφείου, στο ερευνητικό κέντρο «ΚΟΙΟΣ» του Πανεπιστημίου Κύπρου. Ο συνολικός χώρος καλύπτει έκταση 560 τετραγωνικών μέτρων, και αποτελείται από πολλά ιδιωτικά γραφεία, εργαστήρια, αίθουσα διάσκεψης, και διαδρόμους. Στο κτίριο «ΚΟΙΟΣ» υπάρχουν 9 εγκατεστημένα σημεία πρόσβασης Ασύρματων Δικτύων τα οποία χρησιμοποιούν το πρότυπο IEEE 802.11b/g για να παρέχουν πλήρη κάλυψη σε όλη την έκταση του κτιρίου. Επιπρόσθετα υπάρχουν πολλά γειτονικά σημεία πρόσβασης Ασύρματων Δικτύων, τα οποία μπορούν μερικώς να εντοπιστούν σε διαφορετικές περιοχές του κτιρίου. Για το συγκεκριμένο σύνολο δεδομένων, χρησιμοποιήθηκαν τρεις συσκευές που έτρεχαν λειτουργικό σύστημα Android (HTC Desire, HTF Flyer, και Samsung Nexus S), οι οποίες περισύλλεξαν τιμές Ισχύος Λαμβανόμενου Σήματος (RSS), σε 105 διαφορετικές τοποθεσίες. Οι διαδοχικές αυτές τοποθεσίες είχαν 2-3 μέτρα απόσταση μεταξύ τους. Στο σύνολο καταγράφηκαν 2100 σημεία-αποτυπώματα, τα οποία αναλογούν σε 20 αποτυπώματα ανά φυσική τοποθεσία. Η καταγραφή τιμών Ισχύος Λαμβανόμενου Σήματος έγινε με τον ρυθμό ενός δείγματος ανά δευτερόλεπτο.

Οι δύο πιο πάνω χάρτες, μετατράπηκαν από το τοπικό σύστημα συντεταγμένων (σε μέτρα), στο Παγκόσμιο Σύστημα Συντεταγμένων, με την μέθοδο που αναλύθηκε σε προηγούμενο κεφάλαιο.

iii) **Παραγωγή συνόλου δεδομένων μεγάλης κλίμακας:** Για την αποτίμηση της επεκτασιμότητας των δύο αλγορίθμων, σε μεγαλύτερα δεδομένα, σχεδιάστηκαν δύο σύνολα δεδομένων με τον συνδυασμό και επέκταση των δύο συνόλων πραγματικών δεδομένων στο κτίριο Πληροφορικής και το ερευνητικό κέντρο «ΚΟΙΟΣ» του Πανεπιστημίου Κύπρου. Συγκεκριμένα το πρώτο από τα δύο, θα μπορούσε να προσομοιώσει μετρήσεις που αφορούσαν κάποιο εργοστάσιο μεσαίας κλίμακας, και δημιουργήθηκε συνθέτοντας τα δυο προαναφερθέντα σύνολα δεδομένων. Το δεύτερο σύνολο δεδομένων, το οποίο θα μπορούσε να προσομοιώσει ένα μεγάλο εμπορικό κέντρο, δημιουργήθηκε επεκτείνοντας το πρώτο σύνολο δεδομένων τέσσερις φορές. Δηλαδή το εμπορικό κέντρο θα είχε τετραπλάσιο μέγεθος από το συνδυασμένο κτίριο της Πληροφορικής μαζί με του ερευνητικού κέντρου «ΚΟΙΟΣ».

iv) **Παραγωγή Χαρτών Τιμών Ισχύος από τον οργανισμό Crawdad:** όπως περιεγράφηκε στο προηγούμενο κεφάλαιο, χρησιμοποιώντας δεδομένα από τον οργανισμό Crawdad, και συνδυάζοντας διαδοχικές μετρήσεις των αισθητήρων Παγκόσμιου Συστήματος Γεωτοποθέτησης (GPS), και του δέκτη Ασύρματων Δικτύων (WiFi), παράχθηκαν Χάρτες Τιμών Ισχύος για τέσσερις περιοχές στις Ηνωμένες Πολιτείες Αμερικής: στην περιοχή Downtown και στην περιοχή Ravenna της πόλης Seattle στην πολιτεία Ουάσιγκτον, στην

πόλη Kirkland της πολιτείας Ουάσιγκτον, και στο Πανεπιστήμιο Dartmouth, στην πόλη Ανόβερο της πολιτείας New Hampshire.

v) **Παραγωγή ρεαλιστικών χαρτών από τα δεδομένα που εξάχθηκαν από τον οργανισμό Crawdad:** όπως περιεγράφηκε σε προηγούμενο κεφάλαιο, ένα από τα κτίρια που εξάχθηκαν (το κτίριο του πανεπιστημίου Darthmouth), «αντιγράφηκε» στο κέντρο σχεδόν όλων των πόλεων του πλανήτη (7,500 πόλεις στο σύνολο)

6.2 Αλγόριθμοι και μετρικές

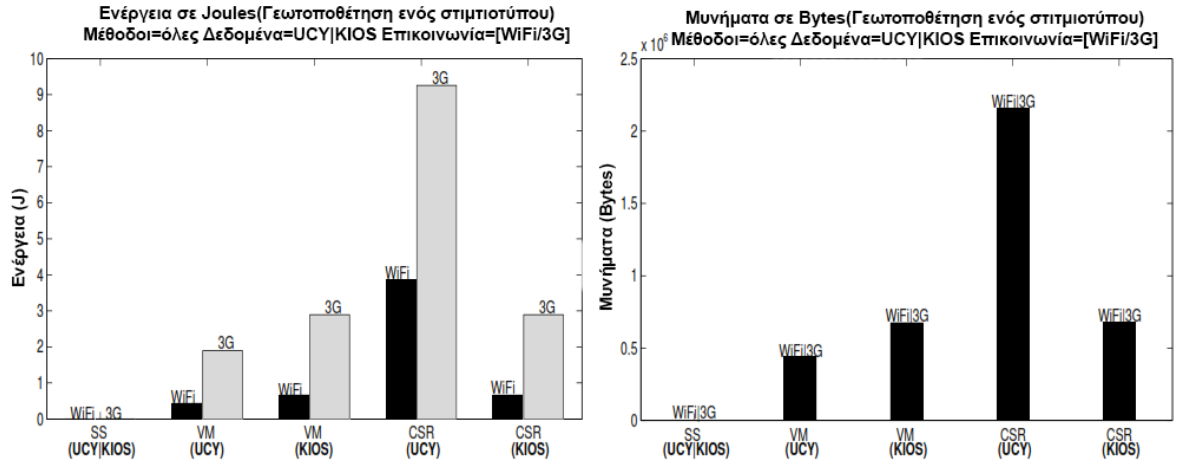
Η πειραματική αποτίμηση είχε ως στόχο την αξιολόγηση των αλγορίθμων Vectormap και Temporal Vectormap σε σύγκριση με εναλλακτικές προσεγγίσεις που παρουσιάστηκαν κατά την εισαγωγή.

Ειδικότερα, συγκρίθηκαν οι αλγόριθμοι έναντι των:

i) Υπολογισμός στην πλευρά του Εξυπηρετητή (SS - ServerSide): όπως αυτός συνοψίζετε στο σχήμα 25 (π.χ. με την χρήση αναγνωριστικού του πύργου Κινητής Τηλεφωνίας, ή αναγνωριστικού του Σημείου Πρόσβασης Ασύρματου Δικτύου). Σε αυτή την προσέγγιση υπάρχει καλύτερη απόδοση από τους αλγόριθμους Vectormap και Temporal Vectormap, επεμβαίνουν όμως στην ιδιωτικότητα των χρηστών

ii) Υπολογισμός στην πλευρά της έξυπνης κινητής συσκευής με την χρήση Χάρτη Τιμών Ισχύος (CSR - Client-Side Radiomap): είναι η προσέγγιση που χρησιμοποιήθηκε στο σύστημα εσωτερικής γεωτοποθέτησης Airplace [2]. Οι αλγόριθμοι, Vectormap και Temporal Vectormap, έχουν την ίδια ακρίβεια γεωτοποθεσίας, καθώς και τα ίδια επίπεδα ιδιωτικότητας, όμως επιτυγχάνοντας λιγότερη κατανάλωση ενέργειας, σε σύγκριση με την προσέγγιση CSR.

Δεν θα υπάρξει σύγκριση με τεχνολογίες οι οποίες κάνουν χρήση της κεραίας Παγκόσμιου Συστήματος Γεωτοποθέτησης (GPS), αφού αυτή η τεχνολογία κρίνετε ως ακατάλληλη για γεωτοποθεσία σε εσωτερικούς χώρους. Σε αυτή την προσέγγιση ωστόσο θα αναμέναμε σαφώς μεγαλύτερη κατανάλωση ενέργειας. Όλες οι μετρήσεις έγιναν ανα μέσο όρο 10 διαδοχικών εκτελέσεων.



Σχήμα 25: *Πειραματική σειρά 1: Σύγκριση του VM, SS και CSR προσεγγίσεων με σύνολο δεδομένων τους Χάρτες Τιμών Ισχύος UCY και KIOS για σενάριο στιγμιότυπης γεωτοποθεσίας.*

6.3 Δίκτυο και ενεργειακό μοντέλο

Το ενεργειακό προφίλ που χρησιμοποιήθηκε, εξάχθηκε με την βοήθεια της εφαρμογής PowerTutor[15], και παρουσιάζετε στον πίνακα 26.

Η επικοινωνία μεταξύ της έξυπνης κινητής συσκευής και του εξυπηρετητή, πραγματοποιείτο χρησιμοποιώντας το από το πρωτόκολλο WiFi 802.11b, ή συνδέσεις Κινητών δεδομένων (Mobile Data) 3G, οι οποίες είχαν κατερχόμενη ζεύξη (downlink) για το πρωτόκολλο TCP της τάξεως 1022kbps και 560kbps αντίστοιχα.

6.4 Αποτελέσματα πειραμάτων

Τα πειραματικά αποτελέσματα χωρίστηκαν σε τρείς πειραματικές σειρές. Η πειραματική σειρά 1, έχει ως στόχο την αποτίμηση του αλγορίθμου Vectormap σε γεωτοποθέτηση ενός στιγμιότυπου, η πειραματική σειρά 2 έχει ως στόχο την αποτίμηση του αλγορίθμου Temporal Vectormap σε σενάριο συνεχόμενης γεωτοποθέτησης (γεωτοποθέτηση πολλαπλών διαδοχικών στιγμιότυπων), και τέλος η τρίτη πειραματική σειρά, έχει ως στόχο την παρουσίαση αποτελεσμάτων στο πραγματικό σύστημα. Οι δύο πρώτες πειραματικές σειρές, ήταν υλοποιημένες σε προσομοιωτή, τόσο για τον εξυπηρετητή αλλά και για τον πελάτη. Στην τρίτη πειραματική σειρά μεταφέρθηκε η λειτουργικότητα από προσομοιωτή για τον

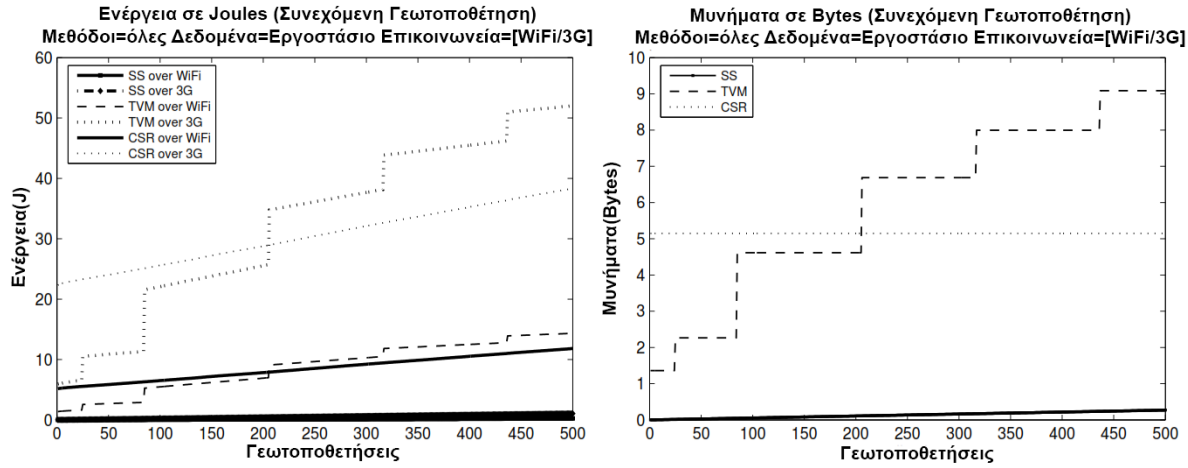
[15] PowerTutor, 2012, http://powertutor.org

Ενεργειακό προφίλ μιας τυπικής έξυπνης κινητής συσκευής	
Βασική Λειτουργία	**Ισχύς (mW=mJ/s)**
Επεξεργαστής – ελάχιστη χρήση (μόνο το ΛΣ)	35mW
Επεξεργαστής – κανονική χρήση (ελαφριά επεξεργασία)	175mW
Επεξεργαστής με βαριά χρήση	469mW
Ασύρματο Δίκτυο Ανενεργό (Συνδεδεμένο)	34mW
Γεωτοποθεσία με Ασύρματο Δίκτυο	**125mW**
Ασύρματο Δίκτυο με Βαριά Χρήση	400mW
Γεωτοποθεσία με 3G	**300mW**
Απασχολημένο 3G	900mW
Παγκόσμιο Σύστημα Συντεταγμένων Ενεργοποιημένο	**275mW**
Οθόνη σε οικονομική κατάσταση	300mW
Οθόνη με μέγιστη φωτεινότητα	676mW

Πίνακας 26: *Ενεργειακά προφίλ στις έξυπνες κινητές συσκευές*

εξυπηρετητή και τον πελάτη, σε πραγματικές υποδομές εξυπηρετητή (εξυπηρετητής ιστού Apache Tomcat, και κατανεμημένη βάση δεδομένων HBase), και πελάτη (εφαρμογή στην πλατφόρμα Android). Αναλυτικά, οι τρεις πειραματικές σειρές:

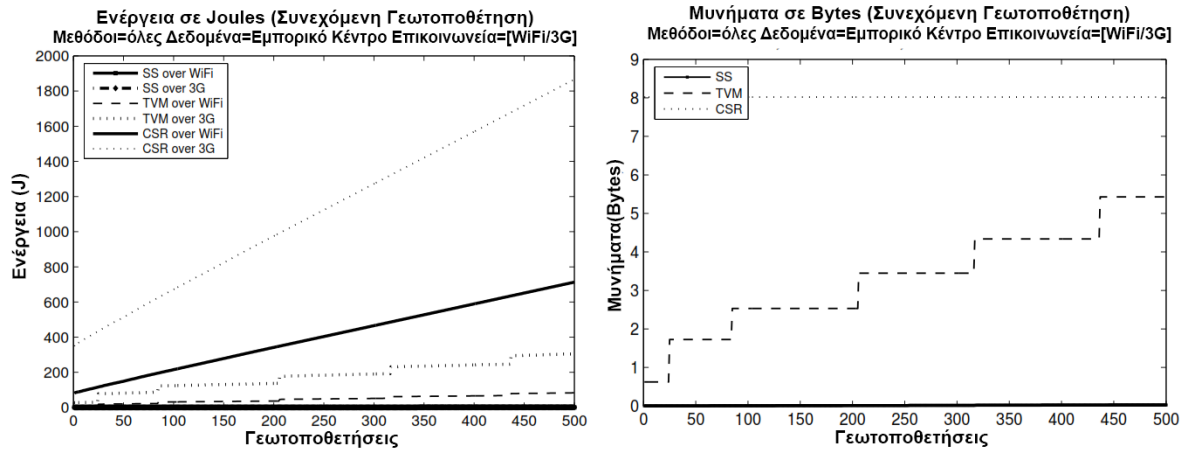
1) Πειραματική σειρά 1, γεωτοποθέτηση ενός στιγμιότυπου: Στο σχήμα 25, παρουσιάζονται τα αποτελέσματα όλων των προσεγγίσεων όσο αφορά την κατανάλωση ενέργειας και τον αριθμό μηνυμάτων που απαιτούνται για γεωτοποθέτηση ενός στιγμιότυπου, χρησιμοποιώντας τα σύνολα δεδομένων (Χάρτες Τιμών Ισχύος) που περιεγράφηκαν πριν, του κτιρίου Πληροφορικής και του ερευνητικού κέντρου «ΚΟΙΟΣ», του Πανεπιστημίου Κύπρου. Στο σύνολο δεδομένων του Πανεπιστημίου Κύπρου, ο αλγόριθμος Vectormap αυξάνει την απόδοσή του σε σύγκριση με τον CSR αλγόριθμο κατά 80% όσο αφορά την ενέργεια, και χρησιμοποιεί 80% λιγότερους πόρους δικτύου. Στο σύνολο δεδομένων του ερευνητικού κέντρου «ΚΟΙΟΣ», ο αλγόριθμος Vectormap παρέχει 10% λιγότερη κατανάλωση ενέργειας και χρησιμοποιεί 15% λιγότερους πόρους δικτύου. Σε αυτό το σύνολο δεδομένων, το οποίο είναι αρκετά μικρότερο από το σύνολο δεδομένων του Πανεπιστημίου Κύπρου, η βελτίωση του αλγόριθμου Vectormap σε σύγκριση με τον αλγόριθμο CSR δεν είναι και τόσο φανερή. Αυτό οφείλετε στο γεγονός ότι η διαφορά στο μέγεθος του μερικού Χάρτη Τιμών Ισχύος που μεταφέρθηκε κατά την προσέγγιση Vectormap, και το μέγεθος ολόκληρου του Χάρτη Τιμών Ισχύος που μεταφέρθηκε κατά την προσέγγιση CSR είναι μικρή.
2) Πειραματική σειρά 2, συνεχόμενη γεωτοποθέτηση: Στην δεύτερη πειραματική σειρά, συγκρίνουμε τον αλγόριθμο Temporal Vectormap με τον αλγόριθμο SS (Server-Side) και



Σχήμα 27: *Σύγκριση του TVM, SS και του CSR αλγορίθμων, σε σύνολο δεδομένων μεγέθους ενός εργοστασίου, σε σενάριο συνεχόμενης γεωτοποθεσίας*

CSR για επικοινωνία μέσω Ασύρματων Δικτύων, και Κινητών δεδομένων (Mobile Data – 3G), στους Χάρτες Τιμών ισχύος που αναφέρθηκαν πριν και θα μπορούσαν να αντιπροσωπεύσουν ένα εργοστάσιο, ή κάποιο εμπορικό κέντρο. Οι προσεγγίσεις αποτιμώνται με 500 συνεχόμενες γεωτοποθετήσεις, και αναλύεται η κατανάλωση ενέργειας και ο αριθμός μηνυμάτων που ανταλλάσσονται μεταξύ πελάτη και συστήματος εξυπηρετητή. Σε όλες τις περιπτώσεις, ο χρήστης έξυπνων κινητών συσκευών ακολουθούσε μια τυχαία διαδρομή η οποία περιείχε πέντε αλλαγές εμβέλειας (δηλαδή ο χρήστης έβγαινε εκτός εμβέλειας του προηγούμενο σημείου πρόσβασης Ασύρματου Δικτύου, και θα έπρεπε να αιτηθεί νέο Χάρτη Τιμών Ισχύος για να μπορεί να γεωτοποθετηθεί και πάλι). Οι περιπτώσεις όπου ο χρήστης μεταφέρετε εκτός εμβέλειας του προηγούμενου σημείου πρόσβασης φαίνονται καθαρά στα σημεία στο σχήμα 27 και 28, περίπου κατά την γεωτοποθέτηση στα σημεία 25, 85, 200, 310 και 450.

Στο σχήμα 27, παρατηρούμε ότι ο αλγόριθμος Temporal Vectormap υπερτερεί έναντι του αλγορίθμου CSR, με 50% λιγότερη κατανάλωση ενέργειας, και 75% λιγότερη κατανάλωση σε πόρους δικτύου στην αρχή, και για τους δύο τύπους επικοινωνίας (με Ασύρματες Συνδέσεις ή Κινητά δεδομένα). Αυτή η καλύτερη επίδοση ωστόσο, μειώνετε καθώς ο χρήστης συνεχίζει να γεωτοποθετείται. Ο CSR αλγόριθμος υπερτερεί τελικά του Temporal Vectormap αλγορίθμου, μετά από την τρίτη έξοδο από την εμβέλεια του Σημείου Πρόσβασης (περίπου μετά την 200^{η} γεωτοποθέτηση), αφού το συνολικό μέγεθος των μερικών Χαρτών Τιμών Ισχύος που μεταφέρθηκαν κατά τον Temporal Vectormap αλγόριθμο, είναι υψηλότερο από ολόκληρο τον χάρτη που μεταφέρθηκε μια φορά κατά την χρήση του CSR αλγορίθμου. Επίσης, όσο αφορά την απόδοση και την κατανάλωση ενέργειας, ο αλγόριθμος Temporal



Σχήμα 28: *Σύγκριση του TVM, SS και του CSR αλγορίθμων, σε σύνολο δεδομένων μεγέθους ενός εμπορικού κέντρου, σε σενάριο συνεχόμενης γεωτοποθεσίας*

Vectormap επιβαρύνετε περισσότερο, αφού ο μερικός Χάρτης Τιμών Ισχύος, στην ουσία είναι πολλοί χάρτες (ο πραγματικός και οι ψευδο-θετικές εκτιμήσεις) συνενωμένοι σε ένα χάρτη.
Η απόδοση του Temporal Vectormap αλγορίθμου χειροτερεύει ακόμη περισσότερο όταν ακολουθήσουν περισσότερες περιπτώσεις όπου ο χρήστης βγαίνει εκτός εμβέλειας του προηγούμενο Σημείου Πρόσβασης Ασύρματων Δικτύων, και έτσι περισσότεροι μερικοί Χάρτες Τιμών Ισχύος πρέπει να μεταφερθούν. Στο σχήμα 28, η προσέγγιση Temporal Vectormap παρέχει 75% και 50% λιγότερη κατανάλωση ενέργειας για τις επικοινωνίες μέσω Ασύρματων Δικτύων, και Κινητών δεδομένων αντίστοιχα, και λιγότερους πόρους δικτύου της κλίμακας 20-90% σε σύγκριση με τον αλγόριθμο CSR κατά την διάρκεια των 500 συνεχόμενων γεωτοποθετήσεων. Αυτό οφείλετε στο γεγονός ότι η προσέγγιση Temporal Vectormap μεταφέρει ένα μέρος του Χάρτη Τιμών Ισχύος για το εμπορικό κέντρο, σε 500 διαδοχικές γεωτοποθετήσεις, το οποίο είναι πολύ μικρότερο σε σύγκριση με ολόκληρο τον Χάρτη Τιμών Ισχύος που χρειάζεται να μεταφέρει η προσέγγιση CSR.
Εν κατακλείδι, είναι φανερό ότι ο αλγόριθμος Temporal Vectormap εξοικονομεί περισσότερη ενέργεια και πόρους δικτύου καθώς αυξάνετε το μέγεθος του Χάρτη Τιμών Ισχύος. Επιπρόσθετα, για να ελαφρύνουμε το μειονέκτημα του Temporal Vectormap, για τους Χάρτες Τιμών Ισχύος μικρού μεγέθους μπορούμε να χρησιμοποιήσουμε κρύπτες, οι οποίες να τους αποθηκεύουν, ούτως ώστε στην χειρότερη περίπτωση να έχουμε την ίδια απόδοση με τον αλγόριθμο CSR. Το πρόβλημα αυτό επηρεάζει αρνητικά στον τομέα δικτύου, λόγω της μεταφοράς δεδομένων, αλλά και τον πελάτη αφού η εκτέλεση του αλγορίθμου Temporal Vectormap γίνετε εξαιρετικά χρονοβόρα. Η αιτία που η εκτέλεση του αλγορίθμου γίνετε χρονοβόρα είναι ότι για κάθε φορά που πρέπει να υπολογιστεί νέος ψεύτικος γείτονας, οι υπολογισμοί γίνονται με δεδομένο εισόδου ένα μεγάλο Χάρτη Τιμών Ισχύος (δηλαδή με N

χάρτες συνενωμένους). Ωστόσο το πρόβλημα αυτό, επιλύθηκε πλήρως με την εισαγωγή του αντικειμένου διαχωρισμένης μορφής χαρτών.

Και στις δύο περιπτώσεις ο αλγόριθμος SS, υπερτερεί έναντι των υπόλοιπων προσεγγίσεων για όλες τις περιπτώσεις, και όλες τις μετρικές. Ωστόσο όμως, η προσέγγιση SS, δεν μπορεί να θεωρηθεί λύση για το προτεινόμενο πρόβλημα, αφού παραβιάζει την ιδιωτικότητα των χρηστών. Οι προσεγγίσεις Vectormap και Temporal Vectormap, εγγυώνται ότι για στιγμιότυπη ή συνεχόμενες γεωτοποθετήσεις αντίστοιχα, ο χρήστης δεν θα αποκαλύψει την πραγματική του τοποθεσία.

3) Πειραματική σειρά 3, πειράματα στην πραγματική υλοποίηση πελάτη-εξυπηρετητή:

Τα πειράματα αυτά έχουν ως στόχο να παρουσιάσουν αποτελέσματα σε πραγματικά δεδομένα, με την χρήση μιας πραγματικής υλοποίησης από πλευράς πελάτη και εξυπηρετητή. Κατά το πρώτο πείραμα της σειράς 3, θα αποτιμηθούν οι εκδόσεις των Χαρτών Τιμών Ισχύος στις τρείς διαφορετικές μορφές: η πρώτη μορφή είναι η μορφή με ζευγάρια κλειδιών-τιμών (key-value pairs), η δεύτερη μορφή είναι η μορφή που συναντάμε στην υλοποίηση του Airplace με επιπρόσθετο στοιχείο την χρήση της πυξίδας, και η τρίτη μορφή είναι η μορφή των διαχωρισμένων Χαρτών Τιμών Ισχύος, η οποία διαχώριση γίνετε στον εξυπηρετητή κατά την αλληλεπίδραση του με την βάση δεδομένων. Η πρώτη μορφή, έχει το μειονέκτημα ότι επαναλαμβάνει για κάθε τιμή Ισχύος Λαμβανόμενου Σήματος, το αναγνωριστικό του Σημείου Πρόσβασης Ασύρματων Δικτύων. Επομένως θα μπορούσε να εξοικονομηθεί ο χώρος αυτός. Η δεύτερη μορφή δεν υποφέρει από αυτό το πρόβλημα. Καθώς όμως αυξάνετε το μέγεθος των Χαρτών Τιμών Ισχύος, επειδή περιλαμβάνονται οι μηδενικές τιμές, το μέγεθος αυξάνετε εκθετικά. Η τρίτη μορφή απολαμβάνει τα πλεονεκτήματα των δύο άλλων προσεγγίσεων, ενώ παράλληλα απαλλάσσετε από τα μειονεκτήματά τους. Σε αυτό το πείραμα, χρησιμοποιήθηκαν τρεις διαφορετικές περιπτώσεις: 1) Η περίπτωση όπου ο Χάρτης Τιμών Ισχύος είναι πολύ μικρός, και αφορά μόνο ένα μερικό χάρτη. Σε αυτή την περίπτωση η δεύτερη έκδοση ισοβαθμεί με την τρίτη, και υπερτερούν έναντι της πρώτης. Επίσης είναι πολύ σπάνια περίπτωση. 2) Η περίπτωση αυτή είναι και η πιο ευρέως εμφανιζόμενη. Συναντάτε όταν ένας Χάρτης Τιμών Ισχύος παράγετε από κάποιο φίλτρο bloom ή πολλαπλά αναγνωριστικά mac. Συγκεκριμένα ο εξυπηρετητής είχε 4 ψευδο-θετικές εκτιμήσεις. Δηλαδή το αποτέλεσμα θα πρέπει να περιέχει 5 μερικούς Χάρτες Τιμών Ισχύος, αν λάβουμε υπόψιν και τον αληθινό. Κατά την πρώτη έκδοση, λόγο της επανάληψης των αναγνωριστικών των σημείων πρόσβασης Ασύρματων δικτύων, το μέγεθος της ανέρχεται στα 650 kb. Η δεύτερη προσέγγιση, λόγω του ότι οι 5 χάρτες τιμών ισχύος είναι σχετικά μικροί, αλλά και του ότι κατά την σύμπτυξη τους υπάρχει τεράστια επανάληψη των κενών τιμών (αυτό προκύπτει γιατί στην περίπτωση μιας θετικής τιμής στον πραγματικό χάρτη τιμών ισχύος, όλα τα σημεία πρόσβασης στις ψευδο-

	Έκδοση 1	Έκδοση 2	Έκδοση 3
Μικρός Μερικός Χάρτης (Σπάνιο)	18kb	10kb	**10kb**
Χάρτης Φίλτρου Bloom (Συνήθεις Περίπτωση)	650kb	2.42mb	**270kb**
Μεγάλος Μερικός Χάρτης (Σπάνιο)	1.1mb	7.8mb	**1.2mb**

Πίνακας 29: *Σύγκριση των μεγεθών ανάλογα με την έκδοση του Χάρτη Τιμών Ισχύος*

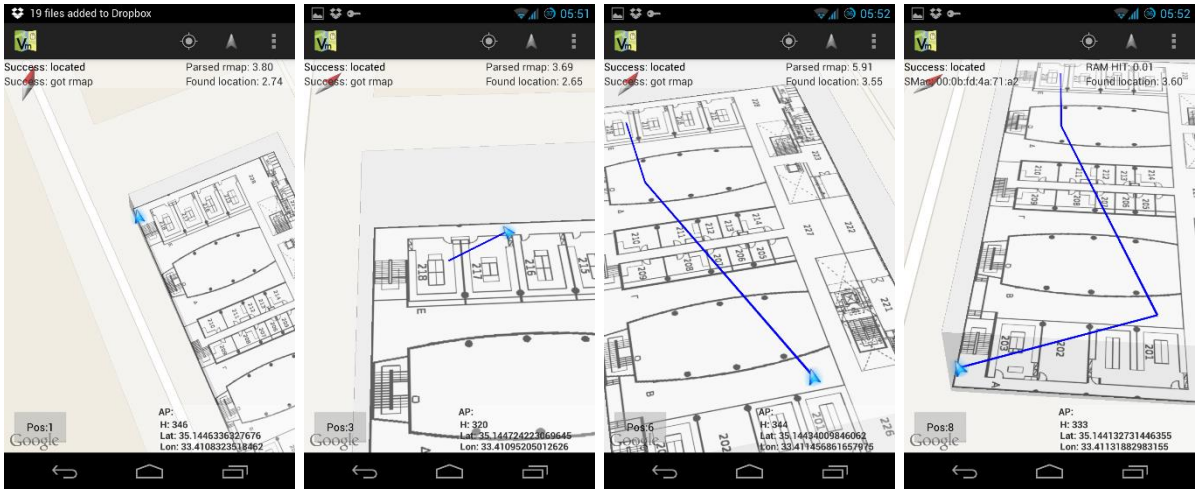
θετικές εκτιμήσεις θα πρέπει να έχουν μηδενική τιμή) ανεβαίνει εκθετικά και το μέγεθος της και φτάνει τα 2.42mb.

Η τρίτη προσέγγιση, όπου διαχωρίζει στο αποτέλεσμα σε 5 ξεχωριστούς μερικούς Χάρτες Τιμών Ισχύος, εκμεταλλεύεται τα θετικά στοιχεία και των δύο μορφών. Δηλαδή δεν υπάρχει η επανάληψη του αναγνωριστικού Σημείου Πρόσβασης της πρώτης προσέγγισης, αλλά ούτε και η μεγάλη επανάληψη μηδενικών τιμών της δεύτερης προσέγγισης. Το μέγεθος είναι μόλις 270kb.

3) Η τρίτη περίπτωση, είναι επίσης σπάνια, και αφορά παρόμοια περίπτωση με την δεύτερη, το μόνο που σε αυτή την περίπτωση, ένας από τους μερικούς χάρτες των σημείων πρόσβασης ψευδο-θετικών εκτιμήσεων, έχει τεράστιο μέγεθος. Γι' αυτό και οι μηδενικές τιμές, ακόμα και στην περίπτωση της τρίτης έκδοσης είναι πολλές, αφορούν όμως μόνο την περίπτωση του συγκεκριμένου μερικού χάρτη. Η πρώτη περίπτωση έχει το μικρότερο μέγεθος που ανέρχεται στα 1.1mb, η δεύτερη περίπτωση αφού συντίθενται 5 χάρτες, ένας εκ των οποίων είναι σχετικά πολύ μεγάλος, πάσχει από μεγάλη επανάληψη μηδενικών τιμών και το μέγεθος της εκθετικά ανεβαίνει στα 7.8mb, και η τρίτη περίπτωση, το μέγεθος κυμαίνεται πολύ κοντά στην πρώτη, και είναι 1.2mb.

Επιπρόσθετα πλεονεκτήματα της τρίτης έκδοσης, όπως προανέφερα σε προηγούμενο κεφάλαιο, είναι στις κινητές συσκευές, όπου οι χάρτες δεν χρειάζονται κάποια επιπρόσθετη επεξεργασία από ότι στην έκδοση του Airplace, καθώς επίσης μπορούν γρηγορότερα να επιλεγούν ή να απορριφθούν κάποιοι από αυτούς. Ακόμη επιλύθηκε το πρόβλημα όπου ο αλγόριθμος Temporal Vectormap λόγω επιπρόσθετων υπολογισμών υστερούσε έναντι του Vectormap στην περίπτωση που εξερχόταν εκτός εμβέλειας πολλές φορές. Τώρα ο χρόνος αυτός είναι αμελητέος, αφού οι αντί να γίνονται προσπελάσεις για εξαγωγή στοιχείων σε ένα μεγάλο μερικό χάρτη, γίνονται σε πολλούς μικρούς χάρτες.

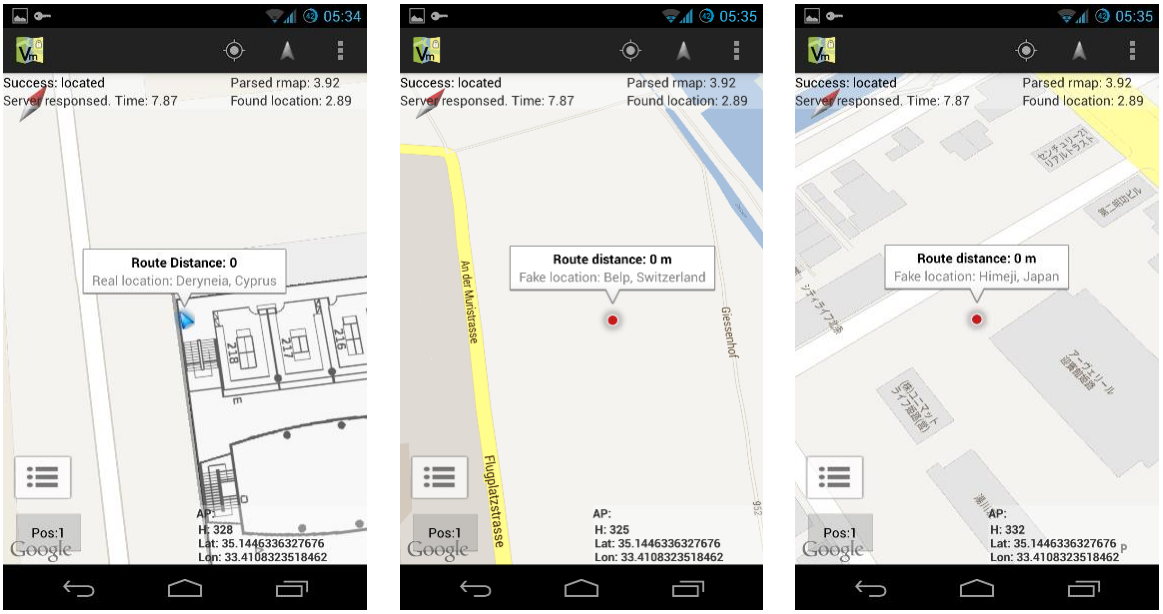
Το δεύτερο πείραμα της τρίτης σειράς, περιλαμβάνει συγκρίσεις μεταξύ των αλγορίθμων: μερικός Χάρτης Τιμών Ισχύος (pRMap), Vectormap (VM), Random Temporal Vectormap (rTVM), και Temporal Vectormap (TVM), χρησιμοποιώντας την επιλογή «Πραγματικής Προσημείωσης» της εφαρμογής Airplace. Η πραγματική προσημείωση περιλαμβάνει 8 διαφορετικά σημεία στο κτίριο πληροφορικής στο Πανεπιστήμιο Κύπρου, και συνολικά ο χρήστης αιτείτε 4 φορές μερικούς Χάρτες Τιμών Ισχύος.



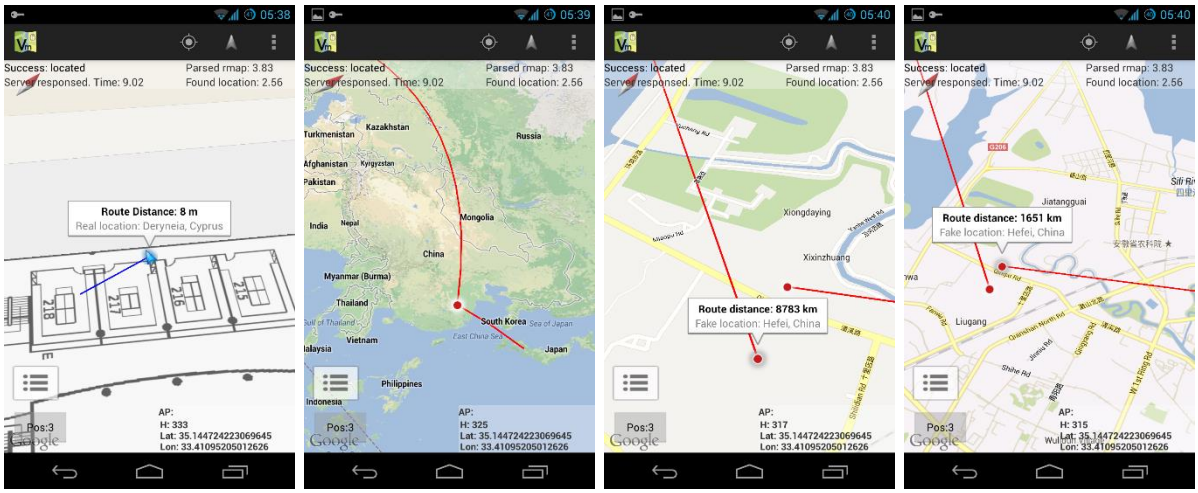
Εικόνα 30: *Γεωτοποθέτηση σε 8 σημεία χρησιμοποιώντας τον αλγόριθμο Μερικού Χάρτη Τιμών Ισχύος. Ο εξυπηρετητής μπορεί να γνωρίζει την πορεία του χρήστη, αν ενώσει σημεία των τεσσάρων εμβελειών σημείων πρόσβασης που άλλαξε ο χρήστης, όπως αυτό φαίνετε με την μπλε γραμμή.*

Κατά την χρήση του αλγόριθμου με μερικό Χάρτη Τιμών Ισχύος (pRMap), ο χρήστης στέλνει στον εξυπηρετητή το πιο κοντινό του σημείο πρόσβασης Ασύρματων Δικτύων, και τότε ο εξυπηρετητής απαντά με τον μερικό Χάρτη Τιμών Ισχύος του συγκεκριμένου σημείου πρόσβασης. Τότε ο εξυπηρετητής μπορεί να γνωρίζει την ευρύτερη περιοχή που κινείτε ο χρήστης, η οποία είναι η ακτίνα εμβέλειας των σημείων πρόσβασης στο κτίριο Πληροφορικής στο Πανεπιστήμιο Κύπρου.

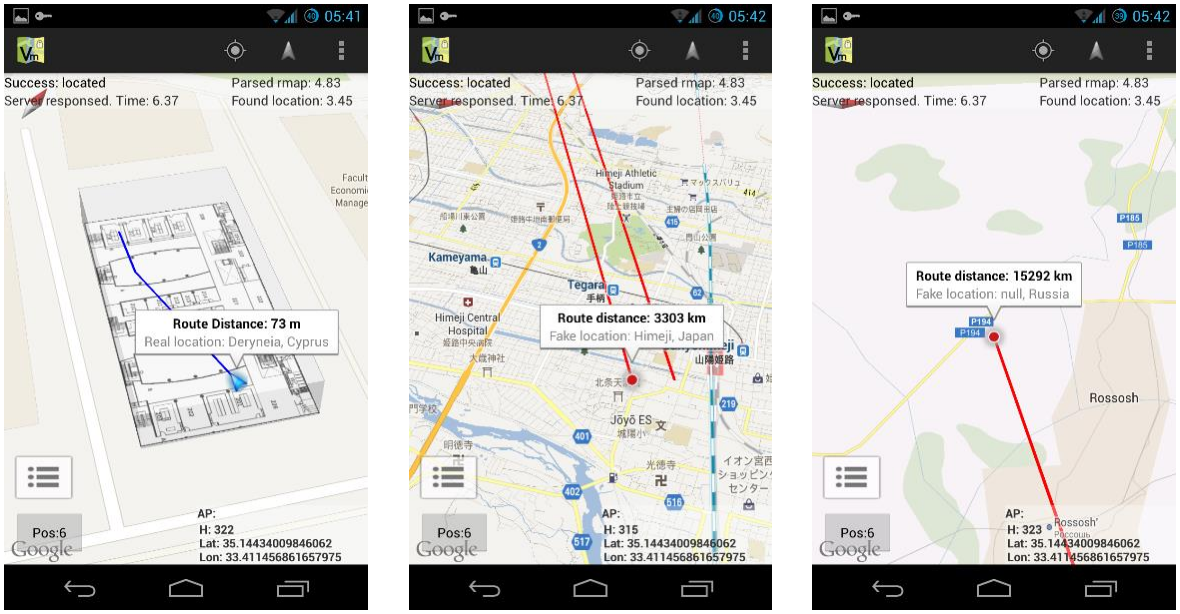
Κατά την χρήση του αλγορίθμου Vectormap, μπορούμε να δούμε γραφικά, κατά τις τέσσερις εξόδους του χρήστη από την εμβέλεια των Σημείων Πρόσβασης, ότι τα νέα ψεύτικα σημεία πρόσβασης, δεν έχουν καμία χωρική σχέση μεταξύ τους. Επομένως ο εξυπηρετητής μπορεί πολύ εύκολα να απομονώσει τις ψεύτικες διαδρομές, και να ξεχωρίσει την πραγματική διαδρομή του χρήστη.



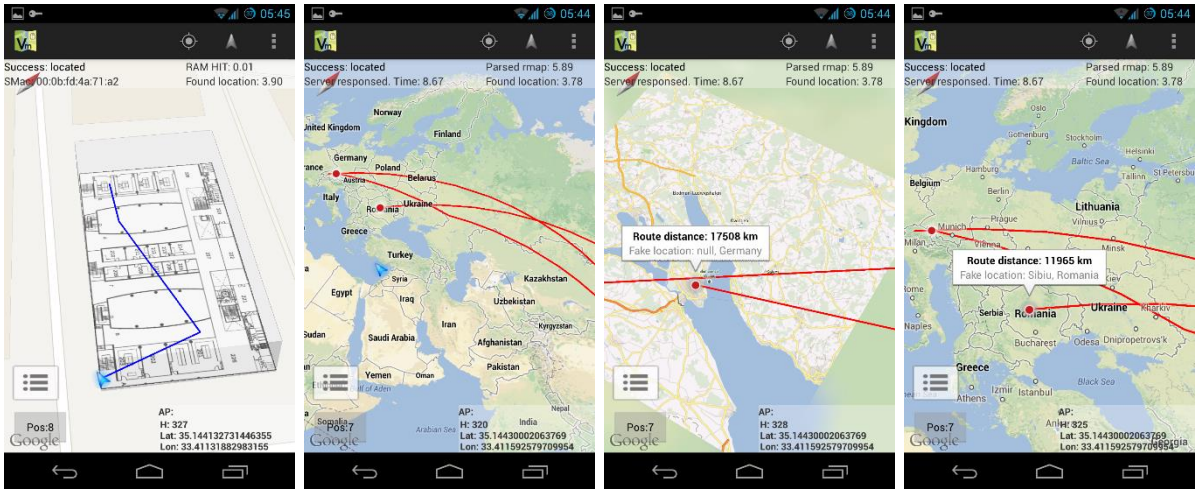
Εικόνες 31: *Πρώτη διαδικασία γεωτοποθέτησης χρησιμοποιώντας τον αλγόριθμο Vectormap. Ο εξυπηρετητής δεν μπορεί να ξεχωρίσει ακόμη την πραγματική τοποθεσία του χρήστη*



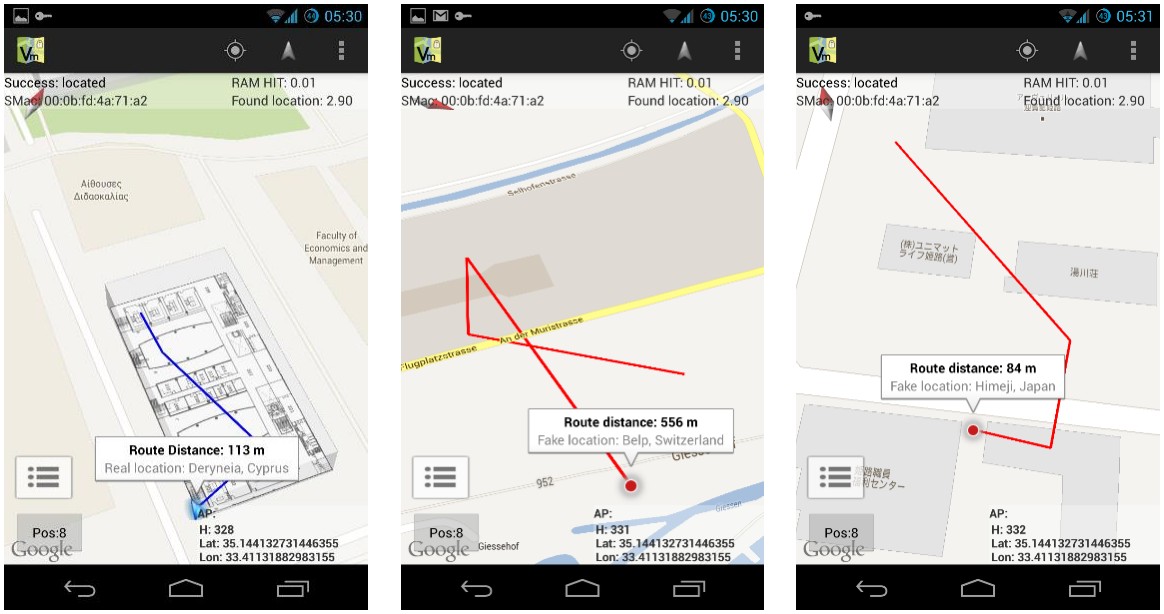
Εικόνες 32: *Δεύτερη διαδικασία γεωτοποθέτησης με τον αλγόριθμο Vectormap, κατά την πρώτη έξοδο από την εμβέλεια του σημείου πρόσβασης. Εικόνα 1: φαίνετε μια λογική πορεία του χρήστη. Εικόνα 2: φαίνετε στον παγκόσμιο χάρτη μια πολύ μεγάλη και παράλογη πορεία. Εικόνα 3: Η 1η ψεύτικη πορεία από την Ελβετία κατέληξε στην Κίνα Εικόνα 4: Η 2η ψεύτικη πορεία από Ιαπωνία, κατέληξε Κίνα επίσης.*



Εικόνες 33: *Τρίτη διαδικασία γεωτοποθέτησης με τον αλγόριθμο Vectormap, κατά την δεύτερη έξοδο από την εμβέλεια του σημείου πρόσβασης. Εικόνα 1: φαίνετε μια λογική πορεία του χρήστη (73 μέτρων). Εικόνα 2: Η 1η ψεύτικη πορεία από την Κίνα κατέληξε στην Ιαπωνία Εικόνα 3: Η 2η ψεύτικη πορεία από Κίνα, κατέληξε Ρωσία*



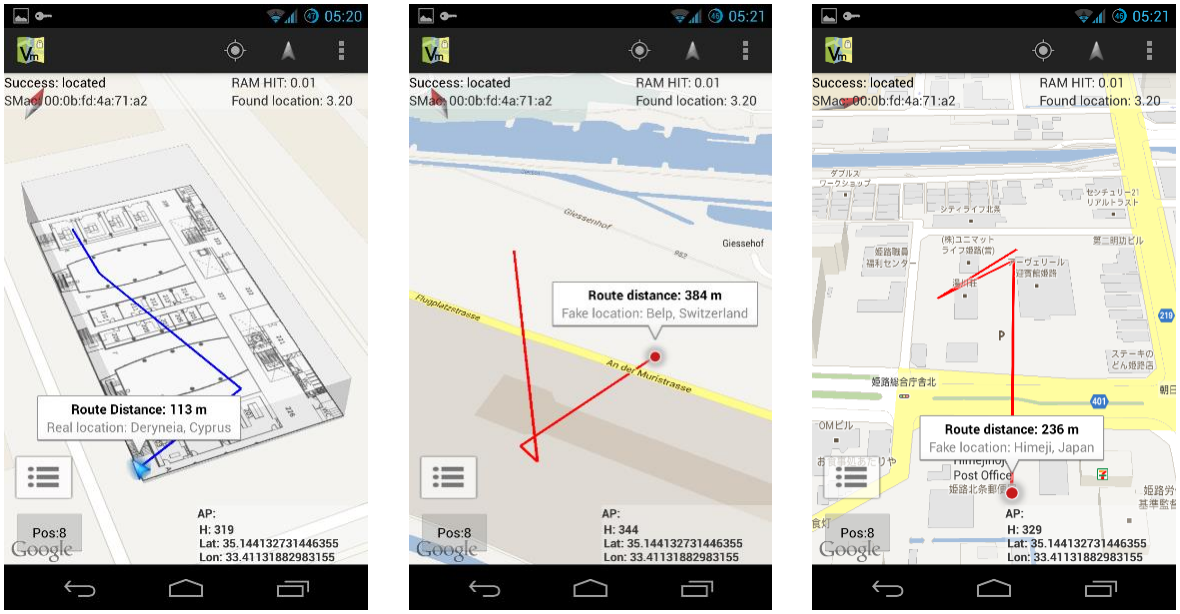
Εικόνες 34: *Τέταρτη διαδικασία γεωτοποθέτησης με τον αλγόριθμο Vectormap, κατά την τρίτη έξοδο από την εμβέλεια του σημείου πρόσβασης. Εικόνα 1: φαίνετε μια λογική πορεία του χρήστη. Εικόνα 2: φαίνονται στον παγκόσμιο χάρτη όλες οι πορείες, και έτσι ξεκάθαρα μπορεί να αντιληφθεί ο εξυπηρετητής ότι ο χρήστης βρίσκετε στην Κύπρο. Εικόνα 3: Η 1η ψεύτικη πορεία από την Ιαπωνία κατέληξε στην Γερμανία Εικόνα 4: Η 2η ψεύτικη πορεία από Ρωσία, κατέληξε Ρουμανία.*



Εικόνες 35: *Γεωτοποθέτηση σε 8 σημεία χρησιμοποιώντας τον αλγόριθμο Random Vectormap, και εναλλαγή σε 4 εμβέλειες των σημείων Πρόσβασης Ασύρματων Δικτύων. Εικόνα 1: η πραγματική διαδρομή στην Κύπρο, συνολικής απόστασης 113 μέτρων. Εικόνα 2: Η 1η ψεύτικη διαδρομή στην Ελβετία, συνολικής απόστασης 556 μέτρων. Εδώ υπάρχει μια διαφορά περίπου 400 μέτρων από την πραγματική πορεία. Εικόνα 3: Η 2η ψεύτικη διαδρομή στην Ιαπωνία, συνολικής απόστασης 84 μέτρων.*

Ακολουθεί η ίδια προσημείωση, τώρα με την χρήση του αλγορίθμου Random Temporal Vectormap. Ο αλγόριθμος Random Temporal Vectormap, μπορεί να συγχύσει σε μεγάλο βαθμό τον εξυπηρετητή, μιας και οι νέες ψεύτικες τοποθεσίες, είναι τυχαίοι πραγματικοί γείτονες των προηγούμενων ψεύτικων τοποθεσιών. Έτσι, όπως φαίνεται και γραφικά, δεν υπάρχει κάποια τεράστια και φανερή αλλαγή, στις διαδρομές που είναι πιθανόν να εκτελεί ο χρήστης. Η απόσταση στην 1η ψεύτικη πορεία ωστόσο, έχει μια απόκλιση 400 μέτρων από την κανονική. Η 2η ψεύτικη πορεία έχει σχετικά πολύ κοντινή απόσταση με την πραγματική. Οι τιμές αυτές όμως θα μπορούσαν να ήταν πολύ πιο διαφορετικές, μιας και οι νέοι ψεύτικοι γείτονες, επιλέγονται με τυχαία βάση.

Ο αλγόριθμος Temporal Vectormap, για τον υπολογισμό των νέων ψεύτικων γειτόνων, χρησιμοποιεί τις τιμές εισόδου και εξόδου για την Ισχύ Λαμβανόμενου Σήματος, καθώς και την ευκλείδεια απόσταση που συνολικά διανύθηκε. Έπειτα επιλέγει τον συνδυασμό που ταιριάζει καλύτερα ως ο επόμενος γείτονας για κάθε ψευδο-θετική εκτίμηση του εξυπηρετητή. Η διαδρομές ωστόσο έχουν κάποια διαφορά μεταξύ τους, αυτό όμως είναι λογικό, γιατί κάθε φορά που ο χρήστης βγαίνει εκτός εμβέλειας, ενδέχεται να υπάρχει απόκλιση 50-100 μέτρων, αφού τα σημεία που χρησιμοποιούνται για την εναποθέτηση της «κουκίδας» στον χάρτη,



Εικόνες 36: *Γεωτοποθέτηση σε 8 σημεία χρησιμοποιώντας τον αλγόριθμο Temporal Vectormap, και εναλλαγή σε 4 εμβέλειες των σημείων Πρόσβασης Ασύρματων Δικτύων. Εικόνα 1: η πραγματική διαδρομή στην Κύπρο, συνολικής απόστασης 113 μέτρων. Εικόνα 2: Η 1η ψεύτικη διαδρομή στην Ελβετία, συνολικής απόστασης 384 μέτρων. Εικόνα 3: Η 2η ψεύτικη διαδρομή στην Ιαπωνία, συνολικής απόστασης 236 μέτρων.*

παράγονται τυχαία από τον εξυπηρετητή, και δεν αποτελούν το κεντρικό σημείο του μερικού Χάρτη Τιμών Ισχύος.

Τα πιο πάνω πειράματα, για τις οικογένειες αλγορίθμων Vectormap και Temporal Vectormap, υπάρχουν σε μορφή video, στον παρακάτω σύνδεσμο:

http://www.cs.ucy.ac.cy/~pmpeis01/vectormap/video

Το τρίτο πείραμα, της τρίτης σειράς, περιλαμβάνει χρόνους που χρειάζονται για την λήψη του Χάρτη Τιμών Ισχύος από τον εξυπηρετητή, για την ανάλυσή του, καθώς και για τον χρόνο εξεύρεσης της τοποθεσίας του χρήστη.

Για την λήψη του Χάρτη Τιμών Ισχύος από τον εξυπηρετητή, απαιτείτε χρόνος γύρω στα 5-9 δευτερόλεπτα.

Για την ανάλυση του Χάρτη Τιμών Ισχύος, που περιλαμβάνει την δημιουργία χαρτών κατακερματισμού που χρησιμοποιούνται για γεωτοποθέτηση, απαιτείτε χρόνος γύρω στα 4-5 δευτερόλεπτα. Με την χρήση της κρύπτης στην κύρια μνήμη, ο χρόνος αυτός σε περίπτωση επιτυχίας (cache hit) είναι μηδενικός.

Για την εξεύρεση γεωτοποθεσίας χρησιμοποιώντας τους αλγόριθμους KNN, WKNN, MSME, WMSME απαιτούνται οι χρόνοι που παρουσιάζονται στον πίνακα 39. Οι αλγόριθμοι MSME με ή χωρίς βάρη είναι κατά πολύ πιο γρήγοροι από τους αλγορίθμους KNN με ή χωρίς βάρη, ωστόσο οι αλγόριθμοι KNN έχουν σαφώς μεγαλύτερη ακρίβεια.

Χρόνοι απόκρισης πελάτη για λήψη χαρτών	
7.87s	7.58s
9.02s	6.52s
6.37s	4.68s
8.67s	4.77s

Πίνακας 37: *Χρόνοι που απαιτούνται για λήψη Χάρτη Τιμών Ισχύος χρησιμοποιώντας τις οικογένειες αλγορίθμων Vectormap, και Temporal Vectormap σε δύο εκτελέσεις της «Πραγματικής Προσομοίωσης».*

Χρόνοι απόκρισης πελάτη για ανάλυση χαρτών
3.92s
3.83s
4.83s
5.89s

Πίνακας 38: *Χρόνοι που απαιτούνται για την ανάλυση (parsing) των χαρτών στις έξυπνες κινητές συσκευές.*

Χρόνοι απόκρισης πελάτη για εξεύρεση γεωτοποθεσίας			
KNN	**WKNN**	**MSME**	**WMSME**
3.20s	2.89s	0.04s	0.03s
2.91s	2.56s	0.02s	0.02s
3.29s	2.74s	0.01s	0.02s
3.74s	3.78	0.02s	0.01s

Πίνακας 39: *Χρόνοι που απαιτούνται για την εξεύρεση τοποθεσίας σε δευτερόλεπτα*

Ενεργειακή κατανάλωση των αλγορίθμων σε mW	
Αλγόριθμος	**Κατανάλωση Ενέργειας**
Partial RMap	743
VM	813
Random TVM	805
TVM	825
SD Cache	480

Πίνακας 40: *Κατανάλωση ενέργειας καταγραμμένη με την εφαρμογή PowerTutor, ανάμεσα στους αλγορίθμους ανωνυμίας: μερικός χάρτης (pRMap), Vectormap (VM), Random Temporal Vectormap (rTVM), και Temporal Vectormap (TVM)*

Το τέταρτο πείραμα της τρίτης σειράς, περιλαμβάνει πέντε ενεργειακές μετρήσεις μεταξύ των τεσσάρων αλγορίθμων (pRmap, VM, rTVM και TVM) χρησιμοποιώντας την εφαρμογή PowerTutor. Στις τέσσερις πρώτες μετρήσεις αυτού του πειράματος η «κρύπτη» 1^{ου} επιπέδου ήταν ενεργοποιημένη (RAM cache), ενώ η «κρύπτη» δεύτερου επιπέδου(SD cache) ήταν απενεργοποιημένη. Όπως υποδεικνύουν οι μετρήσεις, οι αλγόριθμοι δεν έχουν μεγάλη διαφορά μεταξύ τους όσο αφορά την κατανάλωση ενέργειας. Την χαμηλότερη κατανάλωση ενέργειας έχει ο αλγόριθμος με μερικό Χάρτη Τιμών Ισχύος (pRMap). Σε αυτό τον αλγόριθμο καταναλώνονται 743 milliwatt, όμως δεν διατηρείτε k-ανωνυμία όσο αφορά την πραγματική τοποθεσία του χρήστη. Ακολουθούν με πολύ μικρή διαφορά μεταξύ τους, οι αλγόριθμοι Vectormap, Random Temporal Vectomap, και Temporal Vectormap με κατανάλωση ενέργειας μεταξύ 805-825 milliwatt. Και οι τρείς αυτοί αλγόριθμοι, διατηρούν k-ανωνυμία, ωστόσο σε συνεχόμενες γεωτοποθετήσεις μόνοι οι αλγόριθμοι rTVM και TVM διατηρούν το ποσοστό ανωνυμίας στο 1/k. Ωστόσο, αυτό που αξίζει να σημειωθεί, είναι η επίλυση των προβλημάτων που προέκυψαν κατά την προσομοίωση, που ήθελαν την οικογένεια αλγορίθμων Temporal Vectormap να είναι ακριβότερη σε κατανάλωση από την οικογένεια αλγορίθμων Vectormap. Τελειώνοντας με το πείραμα 4, η πέμπτη μέτρηση η οποία αναγράφετε στον πίνακα 50 ως SD

Cache, ήταν η περίπτωση όπου όλοι οι απαραίτητοι μερικοί χάρτες προ-υπήρχαν στην «κρύπτη» δευτέρου επιπέδου. Έτσι ο χρήστης δεν χρειάστηκε ποτέ να επικοινωνήσει με τον εξυπηρετητή. Η κατανάλωση ενέργειας είναι σαφώς λιγότερη και ανέρχεται στα 480 milliwatt.

Κεφάλαιο 7

Συμπεράσματα και Μελλοντική δουλειά

7.1 Συμπεράσματα

Σε αυτή την Ατομική Διπλωματική Εργασία, αποτιμήθηκαν δύο πρωτοποριακές οικογένειες αλγόριθμων που προσφέρουν υψηλής ακρίβειας, χαμηλής ενεργειακής κατανάλωσης, και διατήρησης της ανωνυμίας γεωτοποθέτηση, σε έξυπνες κινητές συσκευές, επονομαζόμενες Vectormap και Temporal Vectormap αντίστοιχα. Η οικογένεια αλγορίθμων Vectormap σχεδιάστηκε για στιγμιότυπες γεωτοποθετήσεις, ενώ η οικογένεια αλγορίθμων Temporal Vectormap μπορεί να εκτελέσει και συνεχόμενες γεωτοποθετήσεις χωρίς να προδίδει την θέση του χρήστη. Και οι δύο οικογένειες αλγόριθμων εγγυώνται ότι η τοποθεσία του χρήστη είναι δυσδιάκριτη από τον εξυπηρετητή, μεταξύ k-1 άλλων χρηστών, όπου η παράμετρος k μπορεί να καθοριστεί από τον χρήστη. Οι τεχνικές αποτιμήθηκαν με προσομοίωση χρησιμοποιώντας πραγματικά και ρεαλιστικά δεδομένα, και έδειξαν ότι μπορούν να προσφέρουν υψηλής ακρίβειας γεωτοποθέτηση με εγγυήσεις k-ανωνυμίας, και 80% περίπου λιγότερη κατανάλωση ενέργειας από ανταγωνιστικές προσεγγίσεις.

Πέραν της προσομοίωσης, οι οικογένειες αλγόριθμων αποτιμήθηκαν με πραγματική υλοποίηση του συστήματος, που περιλαμβάνει την υποδομή του εξυπηρετητή και του πελάτη. Ο εξυπηρετητής αποτελείται από την κατανεμημένη βάση δεδομένων HBase, στην κορυφή του κατανεμημένου συστήματος Hadoop/HDFS κάνοντας χρήση του μοντέλου χαρτογράφησης/μείωσης. Για την αλληλεπίδραση με τους πελάτες χρησιμοποιείτε ο εξυπηρετητής Ιστού Apache Tomcat, ο οποίος ενθυλακώνει την Διεπαφή Προγραμματισμού Εφαρμογών (API) της HBase για την εξαγωγή των δεδομένων. Ο πελάτης αποτελείται από την εφαρμογή Airplace, που επιτρέπει στους χρήστες να γεωτοποθετηθούν με διατήρηση ή όχι της ανωνυμίας, καθώς και ζωντανή παρακολούθηση των συμπερασμάτων που μπορεί να εξάγει ο εξυπηρετητής για την πραγματική τους θέση, ανάλογα με τα δεδομένα που τον τροφοδοτούν.

7.2 Μελλοντική δουλεία

Σε αυτό το έργο, μπορούν να υπάρξουν βελτιώσεις και προσθήκες σε πολλές πτυχές της υλοποίησης. Ξεκινώντας από την υποδομή συμπλέγματος του εξυπηρετητή, θα ήταν αποτελεσματικότερο το λειτουργικό σύστημα να μετατραπεί από CentOS σε Ubuntu Server για περισσότερη σταθερότητα, καλύτερη απόδοση, και σαφώς λιγότερη κατανάλωση πόρων κυρίας μνήμης (π.χ., δεν θα υπάρχει γραφική διεπαφή που τώρα δεσμεύει αχρείαστα μνήμη). Επιπρόσθετα να προ-αποφασιστεί το υλικό (hardware) που μπορεί να δοθεί σε αυτό το έργο, για να επαναρυθμιστούν οι ρυθμίσεις του Hadoop/HDFS και HBase ούτως ώστε να επιτευχθεί η μέγιστη δυνατή απόδοση του συμπλέγματος (π.χ., μια από τις ρυθμίσεις αφορά τον αριθμό των υποεργασιών (Task threads) που μπορούν να τρέχουν ανά πάσα στιγμή σε κάποιο υπηρέτη. Ο αριθμός αυτός εξαρτάτε από τον αριθμό επεξεργαστών, καθώς και την κύρια μνήμη).

Όσο αφορά την βάση δεδομένων HBase, θα πρέπει να υπάρξει καλύτερη εκμετάλλευση των ιδιοτήτων της. Η HBase είναι πολύ γρήγορη στις διαδικασίες εξαγωγής συγκεκριμένου κλειδιού (με την χρήση της εντολής get), αλλά όχι και τόσο γρήγορη σε εντολές όπως η σάρωση όλων των εγγραφών. Θα μπορούσε να γίνει προσπάθεια για ομαδοποίηση των δεδομένων των Χαρτών Τιμών Ισχύος χωρικά, εάν αυτό είναι εφικτό. Αναλυτικότερα, με κάποιο δεύτερο πίνακα ευρετηρίων να διαχωρίζονται οι τοποθεσίες σε κομμάτια τα οποία αντιστοιχούν σε γειτονικές περιοχές στον χάρτη. Με βάση τα κομμάτια χαρτών, να εντοπίζετε το πρωτεύον κλειδί της γραμμής που περιέχει τον συγκεκριμένο χάρτη, και να σερβίρετε στους πελάτες.

Για παράδειγμα θα μπορούσε να γίνει ομαδοποίηση των παρακάτω τοποθεσιών όπως φαίνετε στον πιο κάτω πίνακα:

Πρωτεύον Κλειδί	
35.1448*	33.4111*
35.14483160428558	**33.4111**0521029802
35.14482085173739	**33.4111**1670735697
35.14480854013891	**33.4111**272598478
35.14489695802178	**33.4111**3877011103
35.14488545274341	**33.4111**49913654235

Ο τρόπος διαχωρισμού φυσικά ενδέχετε να τροποποιηθεί, αφού μελετηθεί σαφώς περισσότερο. Ένα άλλο πλεονέκτημα αυτής της προσέγγισης θα είναι ότι οι Χάρτες Τιμών Ισχύος θα δημιουργούνται χωρικά, και όχι ανάλογα με το σημείο πρόσβασης που ακούει ο χρήστης. Έτσι οι αλγόριθμοι γεωτοποθέτησης (π.χ. KNN) που θα εφαρμόζονται, θα έχουν καλύτερη ακρίβεια στα αποτελέσματά τους, και η απόσταση διαχωρισμού (η οποία καθορίζει

και την απόκρυψη - cloaking), θα μπορεί να επεκταθεί ή να μειωθεί από τους χρήστες δυναμικά σε μεταγενέστερα στάδια εξέλιξης αυτής της ιδέας.

Μια άλλη βελτίωση που μπορεί να επιτευχθεί στην πλευρά του εξυπηρετητή είναι η σοφότερη επιλογή του σημείου για τους Χάρτες Τιμών Ισχύος που αντιπροσωπεύει το κέντρο τους, στην μορφή διαχωρισμένου αντικειμένου. Στην παρούσα φάση, το σημείο αυτό επιλέγετε τυχαία. Θα ήταν σαφώς καλύτερο για τους υπολογισμούς των αποστάσεων στην πλευρά του πελάτη, το σημείο αυτό να ήταν το κεντρικό στον μερικό Χάρτη Τιμών Ισχύος. Αν γίνει αυτό στην πλευρά του πελάτη θα είναι χρονοβόρο και θα επιβαρύνει επιπλέων τον επεξεργαστή των κινητών συσκευών. Μια εύκολη και χωρίς κόστος απόδοσης υλοποίηση στον εξυπηρετητή θα μπορούσε να είναι η εξής: κατά την διαχώριση των χαρτών για την παραγωγή του διαχωρισμένου αντικειμένου, ο εξυπηρετητής θα μπορούσε να κρατάει στην μνήμη του πληροφορίες για τα πιο κοντινά και απομακρυσμένα σημεία του Παγκόσμιου Συστήματος Συντεταγμένων (latitude και longitude). Ακολούθως, αντί για τυχαία τιμή για την τοποθεσία στην επικεφαλίδα της μορφής διαχωρισμένου αντικειμένου, να χρησιμοποιείτε ο μέσος όρος των κοντινών και απομακρυσμένων σημείων κάθε Χάρτη Τιμών Ισχύος. Η μορφή του δηλαδή θα παρέμενε ως ακολούθως:

```
{ 'code': '1', 'matches': [
                                { 'mac':'bssid1', 'loc':'mean_Lat1,mean_Lon1'
                                },
                                { 'mac':'bssid2', 'loc':'mean_Lat2,mean_Lon2'
                                }
                            ],
'rmaps': [
            { μερικός χάρτης1 },
            { μερικός χάρτης2 }
          ]
}
```

Η οικογένεια αλγορίθμων Temporal Vectormap, μπορεί να έχει μια επιπλέον προσθήκη. Μετά την παρατήρηση της πορείας που δημιουργείτε στο πραγματικό και τα ψευδο-θετικά σημεία, εξάγετε το συμπέρασμα ότι δεν ακολουθείτε το ίδιο μοτίβο, όσο αφορά τον προσανατολισμό της γης. Για παράδειγμα, στην περίπτωση όπου ο χρήστης στην πραγματική μετακίνηση κινείτε αρχικά ανατολικά και έπειτα δυτικά, δεν υπάρχει καμία εγγύηση ότι οι ψεύτικες πορείες θα έχουν το ίδιο μοτίβο. Αφού λοιπόν οι συντεταγμένες είναι γνωστές στην εφαρμογή πελάτη καθ' όλη την διάρκεια της διαδικασίας γεωτοποθέτησης, θα μπορούσαν να ληφθούν υπόψιν όταν ο χρήστης αιτείται νέο μερικό Χάρτη Τιμών Ισχύος. Με αυτό τον τρόπο μπορεί να δοθεί βάρος στις συγκεκριμένες κατευθύνσεις, με αποτέλεσμα το μοτίβο μετακίνησης να έχει περισσότερες ομοιότητες, όσο αφορά τον προσανατολισμό της γης, στις τροχιές μετακίνησης. Συνεχίζοντας, η εφαρμογή πελάτη Airplace μπορεί επίσης να τύχει σημαντικών βελτιώσεων, σε θέματα που αφορούν την αρχική υλοποίηση Airplace [2], αλλά και τις νέες τροποποιήσεις. Για παράδειγμα οι χάρτες κατακερματισμού που περιέχουν τις τιμές Ισχύος Λαμβανόμενου

Σήματος (RSS), είναι σε μορφή συμβολοσειρών (string). Έτσι σπαταλιέται αρκετά περισσότερος χώρος (ειδικά αν λάβουμε υπόψιν ότι το πρώτο επίπεδο κρύπτης μπορεί να φυλάει ανά πάσα στιγμή περισσότερους του ενός αναλυμένους (parsed) Χάρτες Τιμών Ισχύος), αλλά και χρόνος για την μετατροπή τους κάθε φορά σε ακέραιους αριθμούς.
Όταν τελειώσει η βασική σειρά αλλαγών που πρέπει να γίνουν στο εφαρμογή Airplace, θα μπορούσαν να γίνουν παράλληλοι οι αλγόριθμοι γεωτοποθεσίας (KNN, WKNN, MSME, WMSME). Ακόμη και σε έξυπνες κινητές συσκευές με μονοπύρηνο επεξεργαστή, ο παράλληλος κώδικας μπορεί να αποδώσει ελαφρώς καλύτερα από τον σειριακό.

Επιπλέων πληροφορίες καθώς επίσης και μελλοντική δουλειά για αυτή την εργασία, θα αναρτώνται στον πιο κάτω σύνδεσμο:
http://www.cs.ucy.ac.cy/~pmpeis01/vectormap/

Βιβλιογραφία

[1] A. Konstantinidis, G. Chatzimilioudis, P. Mpeis, and D. Zeinalipour-yazti, "Smartphone Localization with VectorMap," under review, 2013.

[2] C. Laoudias, G. Constantinou, M. Constantinides, S. Nicolaou, D. Zeinalipour-Yazti, and C. G. Panayiotou, "The Airplace Indoor Positioning Platform for Android Smartphones," *2012 IEEE 13th International Conference on Mobile Data Management*, pp. 312–315, Jul. 2012.

[3] L. Sweeney, "k-anonymity: a model for protecting privacy," *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, vol. 10, no. 5, pp. 557–570, 2002.

[4] A. Konstantinidis and G. Chatzimilioudis, "Towards Planet-Scale Localization on Smartphones with a Partial Radiomap," pp. 1–6, 2012.

[5] B. Bloom, "Space/time trade-offs in hash coding with allowable errors.," *Communications of ACM,*, vol. 13, no. 7, 1970.

[6] I. N. Y. Gu, A. Lo, "A survey of indoor positioning systems for wireless personal networks," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 1, pp. 13–32, 2009.

[7] D. Zeinalipour-Yazti. C.-L. Li, C. Laoudias, G. Larkou, G. Chatzimilioudis and P. C., "Hybrid indoor positioning on multi-sensor android smartphones," *IPIN*, 2012.

[8] Y. X. R. Agrawal, J. Kiernan, R. Srikant, "Hippocratic databases," *VLDB*, 2002.

[9] T. S. H. Kido, Y. Yanagisawa, "An anonymous communication technique using dummies for location-based services," *IEEE ICP*, 2005.

[10] H. L. M. Yiu, C. Jensen, X. Huang, "Spacetwist: Managing the trade-offs among location privacy, query performance, and query accuracy in mobile services," *ICDE*, 2008.

[11] M. G. and D. Grunwald, "Anonymous usage of location-based services through spatial and temporal cloaking.," *MobiSys*, 2013.

[12] C.-Y. C. and X. L. M. F. Mokbel, "Spatial cloaking for anonymous location-based services in mobile peer-to-peer environments," *Geoinformatica*, 2011.

[13] K. T. G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, "Private queries in location based services: Anonymizers are not necessary," *ACM SIGMOD*, 2008.

[14] P. K. M. Yiu, G. Ghinita, C. Jensen, "Outsourcing search services on private spatial data," *ICDE*, 2009.

[15] A. K. and C. Shahabi, "Blind evaluation of nearest neighbor queries using space transformation to preserve location privacy," *SSTD*, 2007.

[16] P. Samarati, “Protecting respondents’ identities in microdata release.,” *IEEE TKDE*, vol. 23, no. 6, p. 2001.

[17] M. A. M. Nergiz and Y. Saygin, “Towards trajectory anonymization: a generalization-based approach,” *SPRINGL*, 2008.

[18] F. B. O. Abul and M. Nanni, “Never walk alone: Uncertainty for anonymity in moving objects databases,” *ICDE*, 2008.

[19] Inf. Systems, “Anonymization of moving objects databases by clustering and perturbation,” vol. 35, no. 8, pp. 884–910.

[20] B. Jenkins, “Hash Functions for Hash Table Lookup”, 1997.

[21] A. Appleby, “MurmurHash.” [Online]. Available: https://sites.google.com/site/murmurhash/.

[22] C. G. P. C. Laoudias, P. Kemppi, “Localization using radial basis function networks and signal strength fingerprints in WLAN,” *Globecom*, 2009.

[23] S. Ghemawat, H. Gobioff, and S.-T. Leung, “The Google file system,” *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, p. 29, Dec. 2003.

[24] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, “Bigtable : A Distributed Storage System for Structured Data.”, 2006

Παράρτημα Α

Κώδικας για δημιουργία μερικού χάρτη με χρήση φίλτρου bloom

```
/**
 * Runs the algorithms and prints matched macs or builds Partial Radiomap
 * @return JSon object filled with result
 * @throws IOException
 */
public boolean runMethod() throws IOException {

    boolean foundData = false;

    // MAC Addresses matched with the bloom filter
    ArrayList<String> macMatched = new ArrayList<String>();

    // Match bloomfilter with MAC Addresses
    matchMACSwithBloomfilter(macMatched);

    //Print only macs that matched
    if (method == 0){
        String message = "";
        for (String mac : macMatched){

            //Print a new line
            if (foundData){
                message += "\n";
            }
            foundData = true;
            message += mac;
        }
        //If result found
        if (foundData){
            //Print JSON Object Header
            Serve.printString(writer,
                    JSonObj.printHeader(Base.CODE_BLOOM_MAC_ONLY));
            Serve.printString(writer, message);

            //Print ending of JSon Object
            Serve.printString(writer, JSonObj.printEnding());
        }
    }
    // Print radiomap
    else if (method == 1){

        foundData = printPartialRadiomapMacValuePairs(macMatched);
    }
    else if (method == 2){

        foundData = printPartialRadiomapVersion2(macMatched);
    }
    else if (method == 3){

        foundData = printPartialRadiomapSmartVersion(macMatched);
    }
    return foundData;
}
```

```
/**
     * Match bloomfilter with MAC Addresses
     *
     * @param macMatched MAC Addresses Matched
     * @throws IOException
     */
    private void matchMACSwithBloomfilter(ArrayList<String> macMatched)
            throws IOException {

        // Scanner to HBase
        Scan scan = new Scan();

        // Scanner to our table
        ResultScanner scanner = Base.htableBloomFilter.getScanner(scan);

        String key, value;
        for (Result result : scanner){
            for (KeyValue kv : result.raw()){

                // Get key and value
                key = Bytes.toString(kv.getRow());
                value = Bytes.toString(kv.getValue());

                if (value.equals(bloomFilter.toString())){
                    macMatched.add(key);
                }

            }
        }

        scanner.close();
    }


/**
     * @param macAddressInput
     * @return
     * @throws IOException
     */
    public boolean printPartialRadiomapSmartVersion(
            ArrayList<String> macAddressInput) throws IOException {
        boolean foundData = false;

        // Scan to HBase
        Scan scan = new Scan();

        // One of the filters must pass
        FilterList filters = new FilterList(FilterList.Operator.MUST_PASS_ONE);

        //Create filters for all matched MAC Addresses
        for (String macAddress : macAddressInput){

            // If the MACaddress: WR pair is found, then dont include it
            // This pair wont found because WR will never be as a value in our tables
            // So if a MAC exists, will be included, and if dont, wont
            SingleColumnValueFilter boolMatchFilter = new SingleColumnValueFilter(
                    MapperBulkLoadRadiomap.SRV_COL_FAM,
                    Bytes.toBytes(macAddress), CompareOp.NOT_EQUAL,
                    Bytes.toBytes("WR")); // a dummy value

            //If mac dont exists, wont be included
            // (except if that line is covered by another matched MAC)
            boolMatchFilter.setFilterIfMissing(true);

            //Add filter to list
            filters.addFilter(boolMatchFilter);

        }

        // Set the filter list to scanner
        scan.setFilter(filters);
```

```
// Scanner to our table
ResultScanner scanner = Base.htableMacAddresses.getScanner(scan);

// Contains all coordinates for a row
ArrayList<Coordinate> rowCoordinates = new ArrayList<Coordinate>();
// Contains all macs for a row
ArrayList<String> rowMacs = new ArrayList<String>();

// Contains coordinates to be written on each produced radiomap
        ArrayList<ArrayList<Coordinate>> rmapsCoordinates = new
ArrayList<ArrayList<Coordinate>>();
ArrayList<ArrayList<String>> rmapsUniqueMacs = new ArrayList<ArrayList<String>>();

// create tables to store each radiomap its coordinates
for (int i = 0; i < macAddressInput.size(); i++){
    rmapsCoordinates.add(new ArrayList<Coordinate>());
    rmapsUniqueMacs.add(new ArrayList<String>());
}

// In what radiomap to insert each data
// (it may has to inserted in more than one rmap)
ArrayList<Integer> whereToInsert = new ArrayList<Integer>();

ArrayList<String> macsCopy = new ArrayList<String>();

ArrayList<String> macsNCoords = new ArrayList<String>();
macsCopy.addAll(macAddressInput);// copy macs to new arraylist
macsNCoords.addAll(macAddressInput);//add macs values for now

//Lat, Lon, Heading values, mac
String x = null, y = null, h = null, mac = null;
String[] keyData = null;

boolean keyFlag;
for (Result result : scanner){

    //Clear indices
    whereToInsert.clear();
    //Clear coordinates of a row
    rowCoordinates.clear();
    rowMacs.clear();

    // Flag to avoid multiple inserts of key (x,y,h)
    keyFlag = true;
    // Found results in HBase
    foundData = true;

    // Process each row
    for (KeyValue kv : result.raw()){

        // Save key data of row
        if (keyFlag){
            keyData = Bytes.toString(kv.getRow()).split(":");

            //TODO REMOVE THIS WHEN ALL RMAPS ARE LAT, LON
            if (keyData.length == 4){
                x = keyData[1];
                y = keyData[2];
                h = keyData[3];
            }
            else if (keyData.length == 3){
                x = keyData[0];
                y = keyData[1];
                h = keyData[2];
            }

            keyFlag = false;
        }

        // Get MAC Address
            mac = Bytes.toString(kv.getQualifier());

            // Find in which radiomaps this row has to entered
            if (updateWhereToInsertRow(whereToInsert, macAddressInput,
                    mac)){
                //Print header macs
                saveHeaderMacs(macAddressInput, macsCopy, macsNCoords,
                        mac, x, y);
            }
            //add mac addresses of a row
            rowMacs.add(mac);
            //Save new coordinate
            rowCoordinates.add(new Coordinate(x, y, h, mac, Bytes
                    .toString(kv.getValue())));
        }// Row processed

        //Save data to arraylists
        saveDataToEachRadiomap(rowMacs, rowCoordinates, rmapsUniqueMacs,
                rmapsCoordinates, whereToInsert);

    } // Scanner to whole hbase

    printHeaderAndMacsWithCoords(macAddressInput, macsNCoords);

    // Print radiomaps
    Coordinate.printRadiomaps(rmapsUniqueMacs, rmapsCoordinates, writer);
    writer.print("\n]\n}");
    scanner.close();
    return foundData;
}

/**
 * @param macsNCoords
 */
private void printHeaderAndMacsWithCoords(ArrayList<String> macsMatched,
        ArrayList<String> macsNCoords) {

    //Print JSon header + Json array of bloom matches
    writer.print("{ 'code': '" + Base.CODE_BLOOM_BUILD_RADIOMAP
            + "', 'matches': [\n");

    for (int i = 0; i < macsNCoords.size(); i++){

        writer.print("{ 'mac':'" + macsMatched.get(i) + "' , 'loc':'"
                + macsNCoords.get(i) + "' } ");

        if (i < macsNCoords.size() - 1){
            writer.print(",\n");
        }
    }
    // Close JSONArray
    writer.print("\n],\n");
    // Print JSON Array for partial rmaps
    writer.print("'rmaps': [\n");
}



/**
 * @param matchedMacs
 * @param macs
 * @param macsNcoords
 * @param mac
 * @param x
 * @param y
 */
private void saveHeaderMacs(final ArrayList<String> matchedMacs,
        ArrayList<String> macsCopy, ArrayList<String> macsNcoords,
        String mac, String x, String y) {
```

```
        for (int i = 0; i < macsCopy.size(); i++){
            if (macsCopy.get(i).equals(mac)){

                int index = findMacIndex(matchedMacs, mac);
                    macsNcoords.set(index, x + "," + y);
                    macsCopy.remove(i); //remove mac
                    break; }
        } }

    /**
     * Returns the index of the mac, in matched macs table
     *
     * @param matchedMacs
     * @param mac
     * @return
     */
    private int findMacIndex(ArrayList<String> matchedMacs, String mac) {

        for (int i = 0; i < matchedMacs.size(); i++){
            if (matchedMacs.get(i).equals(mac)) return i; }
        return -1;
    }


/** Save all data to appropriate radiomaps
     * @param rowMacs All MAC Addresses of a row
     * @param rowCoordinates All Coordinates of a row
     * @param rmapsUniqueMacs All radiomaps with all unique MAC Addresses tables
     * @param rmapsCoordinates All radiomaps with all coordinates
     * @param whereToInsert the indices of rows to insert rowMacs and rowCoordinates
     */
    private void saveDataToEachRadiomap(ArrayList<String> rowMacs,
            ArrayList<Coordinate> rowCoordinates,
            ArrayList<ArrayList<String>> rmapsUniqueMacs,
            ArrayList<ArrayList<Coordinate>> rmapsCoordinates,
            ArrayList<Integer> whereToInsert) {

        //Iterate over all indices
        for (int i = 0; i < whereToInsert.size(); i++){
            int index = whereToInsert.get(i);
            //Add unique macs to i-index radiomap
            Coordinate.saveUniqueMacs(rmapsUniqueMacs.get(index), rowMacs);
            //Add coordinates to i-index radiomap
            rmapsCoordinates.get(index).addAll(rowCoordinates);

        }
    }

    /** Updates where the indices of radiomaps that this row has to be inserted
     * @param whereToInsert indices
     * @param macAddressInput Matched MAC Addresses
     * @param mac new mac's row
     */
    private boolean updateWhereToInsertRow(ArrayList<Integer> whereToInsert,
            ArrayList<String> macAddressInput, String mac) {
        boolean found = false;

        //Iterate over all matched macs
        for (int i = 0; i < macAddressInput.size(); i++){
            // MAC matches
            if (macAddressInput.get(i).equals(mac)){
                //save its index
                whereToInsert.add(i);
                found = true; //Found where to add line
            }
        }
        return found;
    }
```

Παράρτημα Β

Κώδικας φίλτρων bloom που εκτελείτε σε εξυπηρετητή και πελάτη

```java
public class Bloomfilter {

    private int     bloomFilterSize;
    private Murmur2 murmur2;

    /**
     * Constructor
     *
     * kAnonymity is fixed/forced to 3, and doing selective MAC on client,
     * solves our problems. Hashfunctions num is 3.
     *
     * @param tOTAL_MAC_ADDRESSES
     * @param kAnonymity
     * @param hashFunctionsNum
     */
    public Bloomfilter(int bloomSize) {
        this.bloomFilterSize = bloomSize;

    }

    /**
     * Generate the bloomfilter
     *
     * @param macAddress
     *            Number of MAC Addresses
     * @param hashFunctionsNum
     *            Number of Hashfunctions used
     * @return
     */
    public String generateBloomFilter(String macAddress) {
        boolean[] result = new boolean[bloomFilterSize];

        try{

            murmur2 = new Murmur2();
            JenkinsHash jh = new JenkinsHash();
            int infiniteCnt = 0;
            int seed = 0;
            int pos1 = (int) ((jh.hash(macAddress.getBytes())) % bloomFilterSize);
            int pos2 = pos1;

            while (pos2 == pos1 || infiniteCnt > 1000){
                pos2 = getMurmur(macAddress, seed += 143);
                infiniteCnt++;
            }


            // Flip 1st bit (MD5)
            result[pos1] = true;
            //          result[pos3] = true;

            result[pos2] = true;

        }
        catch (Exception e){
            e.printStackTrace();
        }
        return bitToString(result);
    }

    /**     */
    private int getMurmur(String macAddress, int seed) {
        int r = murmur2.hash(macAddress, seed);
        if (r <= 0){
            r *= -1;
        }
        return r % bloomFilterSize;
    }

    /**
     * Converts bits array to a String
     *
     * @param bits
     * @return
     */
    private String bitToString(boolean[] bits) {
        String res = "";
        for (int i = 0; i < bits.length; i++){
            if (bits[i] == true) res += "1";
            else res += "0";
        }
        return res;
    }

    /**
     * Hashfunction 1
     *
     * @param macAddress
     * @return
     * @throws NoSuchAlgorithmException
     */
    @SuppressWarnings("unused")
    private int hashfunction1(String macString)
            throws NoSuchAlgorithmException {
        int resOfH2 = MD5(macString);
        if (resOfH2 < 0){
            resOfH2 *= (-1);
        }
        // H2
        return resOfH2 % bloomFilterSize;
    }

    /**
     * Use the hash function of MD5 and then use the result with hashCode
     *
     * @param macAddress
     * @return a value
     * @throws NoSuchAlgorithmException
     */
    private int MD5(String macAddress) throws NoSuchAlgorithmException {
        MessageDigest m = MessageDigest.getInstance("MD5");
        m.update(macAddress.getBytes(), 0, macAddress.length());
        String str = new BigInteger(1, m.digest()).toString(16);


        return str.hashCode();
    }


}
```

Παράρτημα Γ

Κώδικας εφαρμογής του αλγορίθμου Temporal Vectormap

```
public class TVMalgorithms {

    /**
     * @author paschalis
     *
     */
    public static class AsyncTaskTVMalgorithms extends
            AsyncTask<Void, Void, String> {

        private App app;
        private String mac;
        private static int realRandomIndex;

        @Override
        protected void onPreExecute() {
            super.onPreExecute();
        }

        public AsyncTaskTVMalgorithms(App app, String mac) {
            this.app = app;
            this.mac = mac;
        }

        @Override
        protected String doInBackground(Void... params) {

            if (app.tvm1pRequests != null){
                replaceFaultyMac();
            }
            else{
                ArrayList<String> newFakes;

                switch (app.anonymityAlgorithm) {
                    case TVM1:
                        newFakes = new ArrayList<String>();


                        // Calculate a new random fake range for all fake mac addresses
                        for (int i = 0; i < app.cacheData.currentFakeRmaps
                                .size(); i++){

                            newFakes
                            .add(TVMalgorithms.chooseARandomNeighbor(
                             app.cacheData.currentFakeRmaps.get(i),
                            app.currentFakeMatchedMacs.get(i)));
                        }

                        buildRandomFakesAndRealsOrder(newFakes);

                        break;
                    case TVM2:

                        newFakes = TVMalgorithms.chooseMAC_TBMA(
                                app,
                                    (int) Heading.azimuth);
                        buildRandomFakesAndRealsOrder(newFakes);
                        break;

                    default:
                        break;
                }
            }

            return app.tvm1pRequests;
        }

        private void replaceFaultyMac() {

            String[] previousRequest = app.tvm1pRequests.split(",");

            // Replace faulty mac
            previousRequest[realRandomIndex] = mac;

            String result = "";
            int size = previousRequest.length;
            for (int i = 0; i < size - 1; i++){
                result += previousRequest[i] + ",";
            }

            result += previousRequest[size - 1];

            app.tvm1pRequests = result;
        }


        @Override
        protected void onPostExecute(String result) {
            super.onPostExecute(result);
            app.setBgProgressOff();

            String[] getParams = { App.URL_TYPE + "mac" + App.URL_FILTER
                    + result + App.URL_EXTRAS + App.URL_EXTRAS_V3 };

            // Get partial radiomap
            new AsyncTaskHttpExecutor(
                    app,
                        App.getUrl(),
                        AsyncTaskType.getMultiMacs,
                        getParams,
                        mac).execute();
        }

        /**
         * @param newFakes
         * @return a string with new fake macs + real mac in a random order
         */
        private
                void buildRandomFakesAndRealsOrder(
                        ArrayList<String> newFakes) {

            realRandomIndex = App.random.nextInt(newFakes.size());

            // Add real mac in a random index
            newFakes.add(realRandomIndex, mac);

            String result = "";
            int size = newFakes.size();

            for (int i = 0; i < size - 1; i++){
                result += newFakes.get(i) + ",";
            }

            result += newFakes.get(size - 1);
            app.tvm1pRequests = result;
        }
```

```
    /**
     * @param newFakes
     * @return a string with new fake macs + real mac in a random order
     */
    private void buildRandomFakesAndRealsOrderTvmPlus(
            ArrayList<String> newFakes) {

        String[] previousRequest = app.tvm1pRequests.split(",");
        int newFakesCnt = 0;
        // Replace previous mac with new one and fakes with new fakes
        for (int i = 0; i < previousRequest.length; i++){
            if (previousRequest[i].equals(app.previousMac)){
                // RM
                Log.e(TAG, "Replaced previous mac");
                previousRequest[i] = mac;
            }
            else{
                previousRequest[i] = newFakes.get(newFakesCnt++);
            }
        }
        String result = "";
        int size = previousRequest.length;
        for (int i = 0; i < size - 1; i++){
            result += previousRequest[i] + ",";
        }

        result += previousRequest[size - 1];
        app.tvm1pRequests = result;
    }
}

/**
 * @return a new fake random range
 */
public static String chooseARandomNeighbor(
        RadioMap fakeRadiomap, String fakeMac) {

    // Random heading
    int randomHeading = App.random.nextInt(App.MAX_DEGREES);

    // Hashmap for the random heading
    HashMap<String, ArrayList<String>> randomHashmap;
    List<String> keys;

    randomHashmap = fakeRadiomap.getlocationRssHashMap(0);
    keys = new ArrayList<String>(randomHashmap.keySet());
    randomHashmap = fakeRadiomap.getlocationRssHashMap(1);
    keys = new ArrayList<String>(randomHashmap.keySet());
    randomHashmap = fakeRadiomap.getlocationRssHashMap(2);
    keys = new ArrayList<String>(randomHashmap.keySet());
    randomHashmap = fakeRadiomap.getlocationRssHashMap(3);
    keys = new ArrayList<String>(randomHashmap.keySet());

    randomHashmap = fakeRadiomap.getlocationRssHashMap(randomHeading);

    // Get mac addresses
    ArrayList<String> fakeMacs = fakeRadiomap.getmacAddresses();

    // Find fake mac index to avoid it
    int fakeMacIndex = findMacIndex(fakeMac, fakeMacs);

    // Get random hashmap entry
    keys = new ArrayList<String>(randomHashmap.keySet());

    int randomKey = App.random.nextInt(keys.size());

    String randomKeyS = keys.get(randomKey);

    ArrayList<String> rssValues = randomHashmap.get(randomKeyS);

    // Find a not nan rss value
    for (int i = 0; i < rssValues.size(); i++){
        if (fakeMacIndex == i) continue; // ignore previous fake mac

        // If its not a NaN value - FUTURE CHECK THIS:
        // parse it and compare it as integer
        if (!rssValues.get(i).equals(App.NanValueUsed)){ return fakeMacs
                .get(i); // Found new fake mac
        }
    }
    return null; // failed to found new fake mac
}

/**
 * @return next real and N fake MAC Addresses based on Euclidean distance
 */
public static ArrayList<String> chooseMAC_TBMA(App app, int heading) {

    ArrayList<String> result = new ArrayList<String>();
    // Entered position
    LatLng enteredPosition = app.previousCoordinates;
    // Exited position
    LatLng exitedPosition = app.currentCoordinates;
    // Fake radiomaps
    ArrayList<RadioMap> fakeRmaps = app.cacheData.currentFakeRmaps;
    // Fake MAC Addresses
    ArrayList<String> fakeMacs = app.currentFakeMatchedMacs;
    // Index to user for heading
    int headingIndex = RadioMap.findTableIndex(heading);

    // Entered RSS data
    ArrayList<LogRecord> enteredLogRecord = app.enteredScanListGet();
    // Exited RSS data
    ArrayList<LogRecord> exitedLogRecord = app.exitedScanListGet();


    // Distance user covered
    double realDistanceCovered = Algorithms.distance(
            enteredPosition,
                exitedPosition);

    // Find RSS Values of previous mac when exited and entered an AP
    float prevMAC_enteredRSS = getRss(enteredLogRecord, app.previousMac);
    float prevMAC_exitedRSS = getRss(exitedLogRecord, app.previousMac);

    // Find RSS values of current mac when exited and entered an AP
    float curMAC_enteredRSS = getRss(
            enteredLogRecord,
                app.cacheData.currentMac);
    float curMAC_exitedRSS = getRss(
            exitedLogRecord,
                app.cacheData.currentMac);

    // New fake macs from new neighbors
    ArrayList<String> newFakeMacs = new ArrayList<String>();
    ArrayList<Integer> newFakeMacColumns = new ArrayList<Integer>();

    int currentFakeColumn;

    // For all fake macs
    for (int m = 0; m < fakeMacs.size(); m++){

        float threshold = App.EXIT_RSS_THRESHOLD;
        // Fake MAC Address
        String fakeMac = fakeMacs.get(m);
        // Fake Radiomap (of fake mac)
        RadioMap fakeRmap = fakeRmaps.get(m);
        // Fake HashMap of RMap
        HashMap<String, ArrayList<String>> fakeHmap = fakeRmap
                .getlocationRssHashMap(headingIndex);
        ArrayList<String> fakeRmapMacs = fakeRmap.getmacAddresses();
```

```
ArrayList<ArrayList<String>> enteredRows = new ArrayList<ArrayList<String>>();
ArrayList<LatLng> enteredRowsPositions = new ArrayList<LatLng>();
ArrayList<PossibleNeighbor> exitedRows = new ArrayList<PossibleNeighbor>();

// Get MAC Address column, of fake MAC Address
currentFakeColumn = getMacColumnWithMacs(fakeMac, fakeRmapMacs);
// Find possible entered and exited positions of current fake AP
float closestEntered = Float.MAX_VALUE;
float closestExited = Float.MAX_VALUE;
float tmpRes;
boolean foundClosedEntered = false, foundClosedExited = false;

String firstRowKeyStr = fakeHmap.keySet().iterator().next();
// Split key: lat" "lon
String[] firstRowKeyStrTbl = firstRowKeyStr.split(" ");

// Save parsed key value
LatLng firstRowKey = new LatLng(
        Double.parseDouble(firstRowKeyStrTbl[0]),
            Double.parseDouble(firstRowKeyStrTbl[1]));

for (Iterator i = fakeHmap.entrySet().iterator(); i.hasNext()
        && (!foundClosedEntered || !foundClosedExited); i
        .remove()){

    Map.Entry pairs = (Map.Entry) i.next();

    // Get rows of radiomap (Key + values)
    String rowKeyString = (String) pairs.getKey();
    // Split key: lat" "lon
    String[] rowKeyTbl = rowKeyString.split(" ");

    // Save parsed key value
    LatLng rowKey = new LatLng(
            Double.parseDouble(rowKeyTbl[0]),
                Double.parseDouble(rowKeyTbl[1]));
    ArrayList<String> rowRss = (ArrayList<String>) pairs.getValue();

    // If is a NaN value, continue
    if (rowRss.get(currentFakeColumn).equals(App.NanValueUsed))
        continue;

    float curFakeColumnRSS = Float.parseFloat(rowRss.get(currentFakeColumn));

    // If still searching for closest Entered Point
    if (!foundClosedEntered){

        // Save all rows of possible entered(to AP) positions
        tmpRes = curFakeColumnRSS - prevMAC_enteredRSS;
        if (tmpRes < 0) tmpRes *= -1;

        // Find closest RSS value of fake AP to real starting position
        // Do this because we cant find
        if (tmpRes < closestEntered){
            enteredRows.clear();
            enteredRowsPositions.clear();
            enteredRows.add(rowRss);
            enteredRowsPositions.add(rowKey);
            closestEntered = tmpRes;

            if (closestEntered == 0){
                foundClosedEntered = true;
            }
            continue;
        }
        else if (tmpRes == closestEntered){
            enteredRows.add(rowRss);
            enteredRowsPositions.add(rowKey);
            continue;
        }
    }

    // If still waiting for Closest Exit Point
    if (!foundClosedExited){
        // If row isnt on enteredRows, it may be on exitedRows
        // Save all rows of possible exited positions
        tmpRes = curFakeColumnRSS - prevMAC_exitedRSS;
        if (tmpRes < 0) tmpRes *= -1;

        // If difference closer than previous result - threshold
        if (tmpRes < (closestExited - threshold)){
            exitedRows.clear();
            closestExited = tmpRes;
            // Get columns of possible new fake neighbors (of that row)
            PossibleNeighbor p = findMatchingRssNeighbors(
                    rowRss,
                        rowKey,
                        curMAC exitedRSS,
                        currentFakeColumn,
                        newFakeMacColumns);

            // Save possible row, and possible neighbors on that row
            exitedRows.add(p);
            if (closestExited == 0){
                foundClosedExited = true;
            }
        }
        else if (tmpRes >= closestExited
                && tmpRes <= closestExited + threshold){
            // Get columns of possible new fake neighbors (of that row)
            PossibleNeighbor p = findMatchingRssNeighbors(
                    rowRss,
                        rowKey,
                        curMAC_exitedRSS,
                        currentFakeColumn,
                        newFakeMacColumns);

            // Save possible row, and possible neighbors on that row
            exitedRows.add(p);
            // ADVICE cut threshold in half here!
            if (exitedRows.size() > 5){
                threshold /= 5;
            }
        }

    }

}// End of Partial Radiomap loop

// Workaround for isolated AP
if (exitedRows.size() == 0){
    PossibleNeighbor p = findMatchingRssNeighbors(
            enteredRows.get(0),
                firstRowKey,
                curMAC_exitedRSS,
                currentFakeColumn,
                newFakeMacColumns);

    exitedRows.add(p);
}

int foundEntered = -1;
int foundExited = -1;
double closestResult = Double.MAX_VALUE;
boolean exactMatch = false;

// Find the best-fit route
for (int i = 0; i < enteredRows.size() && !exactMatch; i++){
    for (int j = 0; j < exitedRows.size(); j++){
        enteredPosition = enteredRowsPositions.get(i);
        exitedPosition = exitedRows.get(j).coordinates;

        double fakeDistance = Algorithms.distance(
```

```
    enteredPosition, exitedPosition);
                // Calculate result
                double distanceResult = realDistanceCovered - fakeDistance;
                if (distanceResult < 0) distanceResult *= -1;
                // Save closest result (Best solution)
                if (distanceResult < closestResult){
                    closestResult = distanceResult;
                    foundEntered = i;
                    foundExited = j;

                    if (closestResult == 0){
                        exactMatch = true;
                        break;
                    }
                }
            }
        }

        if (foundEntered == -1 || foundExited == -1){
            Log.e(TAG, "Failed to found entered or exited position");
        }

        PossibleNeighbor found = exitedRows.get(foundExited);
        String MACfound = fakeRmapMacs.get(found.macIndex);

        // Failed to find random neighbor - Isolated AP
        if (MACfound == null){
            Log.e(TAG, "Failed to choose neighbor based on RSS values");
            // TODO pick up random here
        }

        // Save new fake MAC Address found
        result.add(MACfound);

        newFakeMacs.add(MACfound);
        newFakeMacColumns.add(found.macIndex);
    }
    return result;
}

/**
 * Find the RSS value of the given mac in an ArrayList LocRecord
 *
 * @param exitedLogRecord
 * @param previousMac
 * @return
 */
private static
        int getRss(ArrayList<LogRecord> exitedLogRecord, String mac) {

    // Initialize NaNValue
    int rssValue = Integer.parseInt(App.NanValueUsed);

    for (int i = 0; i < exitedLogRecord.size(); i++){
        if (exitedLogRecord.get(i).getBssid().equals(mac)){
            rssValue = exitedLogRecord.get(i).getRss();
            break;
        }
    }
    return rssValue;
}
```

```
/**
 * Finds next possible fake neighbor Access Point. Search all the neighbors
 * to find the closest RSS value to the real new AP neighbor
 * @param rowTbl
 * @param curMAC_exitedRSS
 *             real RSS value from real neighbor
 * @param currentFakeColumn
 *             Column of MAC of current fake AP
 * @return
 */
private static PossibleNeighbor findMatchingRssNeighbors(
        ArrayList<String> rowRss, LatLng coordinates,
        float curMAC_exitedRSS, int currentFakeColumn,
        ArrayList<Integer> newFakeMacColumns) {

    PossibleNeighbor result = new PossibleNeighbor();
    float closest = Float.MAX_VALUE;
    float tmp;
    int index = -1;
    boolean toContinue;

    for (int i = 0; i < rowRss.size(); i++){
        // Continue on current fake AP column, or a NaN column
        if (rowRss.get(i).equals(App.NanValueUsed)) continue;
        if (i == currentFakeColumn) continue;

        toContinue = false;
        // Continue on columns of already found fake neighbors
        for (int j = 0; j < newFakeMacColumns.size(); j++){
            if (i == newFakeMacColumns.get(j)){
                toContinue = true;
                break;
            }
        }
        if (toContinue) continue;

        tmp = Float.parseFloat(rowRss.get(i)) - curMAC_exitedRSS;
        if (tmp < 0) tmp *= -1;

        if (tmp < closest){
            index = i;
            closest = tmp;

            if (closest == 0){
                // ADVICE
                break;
            }
        }
    }

    result.coordinates = coordinates;
    result.close = closest;
    result.macIndex = index;
    result.rowRss = rowRss;
    return result;
}

/**
 * @param mac
 * @param macArrayList
 * @return
 */
private static int getMacColumnWithMacs(
        String mac, ArrayList<String> macArrayList) {
    for (int i = 0; i < macArrayList.size(); i++){
        if (macArrayList.get(i).equals(mac)) return i;
    }
    // not found
    return -1;
}
```

```
    /**
     * @param mac
     *             to find its index
     * @return logRecord of recorded MAC+RSS value pairs
     *
     *
     */
    private static
            int getMacColumn(String mac, ArrayList<LogRecord> logRecord) {


        for (int i = 0; i < logRecord.size(); i++){
            if (logRecord.get(i).getBssid().equals(mac)) return i;
        }
        // not found
        return -1;
    }



    /**
     * @param fakeMac
     * @param fakeMacs
     * @return the index of the fake mac given as input, from all the fake macs
     */
    private static
            int findMacIndex(String fakeMac, ArrayList<String> fakeMacs) {


        for (int i = 0; i < fakeMacs.size(); i++){
            if (fakeMac.equals(fakeMacs.get(i))) return i;
        }

        // not found
        return -1;
    }


    /**
     * @author paschalis
     *
     */
    public static class PossibleNeighbor {

        LatLng coordinates;

        ArrayList<String> rowRss; // FIXME change in rss?

        /**
         */
        int macIndex;

        float close;

    }

}
```

Παράρτημα Δ

Διαχείριση των αναμεταδόσεων στην εφαρμογή Airplace

```
/**
* @author Paschalis
*/
public class MultiBroadcastReceiver extends BroadcastReceiver {

    App app;

    @Override
    public void onReceive(Context context, Intent intent) {

        app = (App) context.getApplicationContext();

        // If another background process run, dont put another one
        if (app.isInBgProgress()) return;

        String action = intent.getAction();

        if (action.equals(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION)){
            // Process the wifi results
            processWifiResults();
        }
        // Get Partial radiomap to localize user
        else if (action.equals(App.BROADCAST_RECEIVER_GET_PARTIAL_RMAP)){

            // Get partial radiomap
            // From Ram Cache, SD cache, or download from Server
            getPartialRadiomap(context);
        }
        else if (action.equals(App.BROADCAST_PARSE_RMAP_AND_FIND_LOCATION)){

            // Parse radiomap and localize user
            parseRmapAndlocalizeUser(context);
        }
    }

    /**
     * Parse radiomap (if not already parsed) and localize user
     */
    private void parseRmapAndlocalizeUser(Context context) {
        app = (App) context.getApplicationContext();

        // Parse radiomap, and then localize
        AsyncTaskParseRmap asyncTaskParseRmap = new AsyncTaskParseRmap(app);
        asyncTaskParseRmap.execute();

    }


    /**
     * Gets partial readiomap 1 try: RAM cache 2 try: SD Card cache try:
     * download from server
     */
    private void getPartialRadiomap(Context context) {

        // Didnt find any AP nearby
        if (app.currentScanListIsEmpty()){

            app.textViewMessage1.setText("No Access Point Found");
            Log.i(TAG, "No Access Point Found");
            // Handle error
            handleRadiomapGetError();

            return;
        }
        else{

            // Find strongest listening MACS - FUTURE: SMART BUFFER
            // (limit to 3 to enable smart buffer)
            ArrayList<String> strongestMacs = getStrongestMacs(app
                    .currentScanListGet());
            app.rmapCache.fillCacheData(strongestMacs);
            // Save previous MAC address
            app.previousMac = app.cacheData.currentMac;
            // Update strongest used mac
            app.textViewMessage2
                    .setText("SMac: " + app.cacheData.currentMac);
            Log.i(TAG, "SMAC chosen: " + app.cacheData.currentMac);
            Log.i(TAG, "Cache? " + app.cacheData.type.toString());
            app.textViewMessage1.setText("Cache: "
                    + app.cacheData.type.toString());

            switch (app.cacheData.type) {
                case RAM:
                    // RAM Cache
                case SD:
                    // SD Cache
                    // Send broadcast to localize user
                    Intent intent = new Intent();
                    intent.setAction(App.BROADCAST_PARSE_RMAP_AND_FIND_LOCATION);
                    context.sendBroadcast(intent);

                    break;
                case MISS:
                    // Missed or caches disabled
                    // Get radiomap form server
                    // Save exiting RSS values (which are the previous of
                    // current)
                    app.exitedScanListAddAll(app.previousScanListGet());
                    app.enteredScanListAddAll(app.currentScanListGet());
                    // Save previous location
                    app.previousCoordinates = app.currentCoordinates;
                    app.locationHistory.add(app.previousCoordinates);
                    // Download and cache file to Download RAM and SD Card
                    app.rmapCache.cacheRadiomapFile(app.cacheData);
                    break;
                default:
                    break;
            }
        }
    }

    /**
     *
     */
    private void handleRadiomapGetError() {

        MainActivity.menuItemTrackMe.setChecked(false);
        MainActivity.menuItemTrackMe.setIcon(MainActivity.MENU_TRACKME_OFF);
    }

    /**
     * Returns the strongest macs MAX value used is: 5
     *
     * @param latestScanList
     * @return
     */
    private ArrayList<String> getStrongestMacs(
            ArrayList<LogRecord> latestScanList) {
        // Copy the latest scan in a new arraylist
        ArrayList<LogRecord> ls2 = new ArrayList<LogRecord>();
        ls2.addAll(latestScanList);
        ArrayList<String> result = new ArrayList<String>();

        final int MAX_MACS = 5;

        // Get 5 strongest macs
        for (int i = 0; i < MAX_MACS && ls2.size() > 0; i++){
            result.add(popStrongestListeningMac(ls2));
        }
        return result;
    }
```

```
/**
 * @param latestScanList
 * @return
 */
private String getStrongestListeningMac(ArrayList<LogRecord> logList) {
    // Minimum possible RSS value
    // int strongestRss = Integer.parseInt(App.NanValueUsed);
    // String bssid = null;
    LogRecord strongestLr = new LogRecord(
            "MAC_ERROR",
                Integer.parseInt(App.NanValueUsed));
    // Find strongest log record
    for (LogRecord logRecord : logList){
        // If found new strongest RSS
        if (logRecord.getRss() > strongestLr.getRss()){
            strongestLr = logRecord;
        }
    }
    return strongestLr.getBssid();
}

/**
 * @param latestScanList
 * @return
 */
private String popStrongestListeningMac(ArrayList<LogRecord> logList) {
    // Minimum possible RSS value
    // int strongestRss = Integer.parseInt(App.NanValueUsed);
    // String bssid = null;
    LogRecord strongestLr = new LogRecord(
            "MAC_ERROR",
                Integer.parseInt(App.NanValueUsed));
    // Find strongest log record
    for (LogRecord logRecord : logList){
        if (app.rmapCache.isProblematicMac(logRecord.getBssid()))
            continue;
        // If found new strongest RSS
        if (logRecord.getRss() > strongestLr.getRss()){
            strongestLr = logRecord;
        }
    }
    // pop strongest log record
    logList.remove(strongestLr);
    return strongestLr.getBssid();
}

/**
 * Fake results on 2 positions at ucy (in lab, and near the parking place
 * (South))
 */
public void pretendFakePosition(int num, App app) {

    this.app = app;
    // Clear previous results
    app.currentScanListClear();
    // Add new fake results
    app.currentScanListAddAll(PretendPositions.getPosition(num));
    // Get partial radiomap with fake positions!
    Intent intent = new Intent();
    intent.setAction(App.BROADCAST_RECEIVER_GET_PARTIAL_RMAP);
    app.sendBroadcast(intent);
}

/**
 *
 */
private synchronized void processWifiResults() {
    app.setBgProgressOn();
    List<ScanResult> wifiList = app.lightWifiManager.getScanResults();
    try{
    app.scanAPs.setText("AP: " + wifiList.size());
```

```
            }
            catch(NullPointerException e){
                Log.e(TAG, "wifiList was null");
            }
            // Show strongest mac
            app.textViewMessage1.setText("Smac:"
             + getStrongestListeningMac(app.currentScanListGet()));
            app.currentScanListClear();
            LogRecord lr = null;
            // If we receive results, add them to latest scan list
            if (wifiList != null && !wifiList.isEmpty()){
                for (int i = 0; i < wifiList.size(); i++){

                    if (isntProblematic(wifiList.get(i).BSSID)){
                        lr = new LogRecord(
                                wifiList.get(i).BSSID,
                                    wifiList.get(i).level);
                        app.currentScanListAdd(lr);
                    }
                }
            }
            app.setBgProgressOff();
            // If user is in trackme mode
            // get partial rmap and then localize
            if (app.isTrackmeEnabled() || app.isFindmeEnabled()){
                Intent intent = new Intent();
                intent.setAction(App.BROADCAST_RECEIVER_GET_PARTIAL_RMAP);
                app.sendBroadcast(intent);

                // If findme was running, disable it and stop service
                if (app.isFindmeEnabled() && !app.alwaysScan){
                    // Stop Wifi Service
                    app.stopService(new Intent(app, LocalizationService.class));
                }
            }
        }

        /**
         * Returns true if Access Point isnt problematic Problematic: there is not
         * data for it in Servers database
         *
         * @param bssid
         * @return true if mac isnt problematic
         */
        private boolean isntProblematic(String mac) {
            for (int i = 0; i < app.problematicAPs.size(); i++){
                // If mac is problematic
                if (mac.equals(app.problematicAPs.get(i))){ return false; }
            }
            return true;
        }
    }
```