Ατομική Διπλωματική Εργασία

ΤΟΠΙΚΟΠΟΙΗΣΗ ΕΞΥΠΝΩΝ ΚΙΝΗΤΩΝ ΣΥΣΚΕΥΩΝ ΧΩΡΙΣ ΤΗ ΧΡΗΣΗ RADIOMAP

Σιλουανός Νικολάου

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2012

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Τοπικοποίηση Έξυπνων Κινητών Συσκευών χωρίς τη χρήση RadioMap

Σιλουανός Νικολάου

Επιβλέπων Καθηγητής

Δημήτρης Ζεϊναλιπούρ

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

Μάιος 2012

Ευχαριστίες

Θα ήθελα καταρχάς να ευχαριστήσω τον επιβλέποντα καθηγητή μου κο Δημήτριο Ζεϊναλιπούρ ο οποίος μου ανάθεσε ένα θέμα εξαιρετικά ενδιαφέρον και ο οποίος κατά κάποιον τρόπο συνέβαλε στη διεύρυνση των γνώσεων μου όσον αφορά κυρίως στον προγραμματισμό κινητών συσκευών αλλά και την ανάπτυξη νέων μεθόδων για τοπικοποίηση σε εσωτερικούς χώρους χωρίς την χρήση GPS. Θεωρώ ότι ο συγκεκριμένος καθηγητής δίνει τη δυνατότητα σε φοιτητές να αναπτύξουν εφαρμογές, σε κινητές συσκευές, οι οποίες στις μέρες μας έχουν μεγάλο ενδιαφέρον.

Επίσης, θα ηθέλα να ευχαριστήσω τους συμφοιτητές μου και φίλους Χρίστο, Κώστα και Μάριο από τους οποίους είχα την υποστήριξη, σε τυχόν δυσκολίες μου, σε όλη την διάρκεια της προσπάθειάς μου για την εκπόνηση των σκοπών και στόχων της παρούσας διπλωματικής μου εργασίας. Θεωρώ ότι η βοήθειά τους, σε εμψυχωτικό και πρακτικό επίπεδο, ήταν μεγάλη. Ακόμη οφείλω να ευχαριστήσω και τα παιδιά τα οποία έδωσαν σε μένα είτε μικρή είτε μεγάλη βοήθεια και δεν έχω την δυνατότητα να τους αναφέρω στο παρόν κείμενο.

Ακόμη, θα ήθελα να ευχαριστήσω τον Χρίστο Λαουδιά καθώς και τον Ανδρέα Κωνσταντινίδη οι οποίοι με βοηθήσαν αρκετά στην υλοποίηση του προσομοιωτή που κλήθηκα να υλοποιήσω.

Τέλος, θα ήθελα να ευχαριστήσω τον Θεό που με αξίωσε, στο τέλος της ημέρας, να τελειώσω την διπλωματική μου εργασία με επιτυχία και να προσφέρω με τον δικό μου τρόπο στην κοινότητα του τμήματος Πληροφορικής του Πανεπιστημίου. Η βοήθεια Του ήταν πολύ σημαντική στις προσωπικές φάσεις απελπισίας ή και δυσκολιών και μου έδωσε την ψυχολογική “ένεση” να συνεχίσω αψηφώντας τις δυσκολίες που αντιμετώπισα.

Αναγνώριση

1. "A Platform for the Evaluation of Fingerprint Positioning Algorithms on Android Smartphones", C. Laoudias, G. Constantinou, M. Constantinides, S. Nicolaou, D. Zeinalipour- Yazti, C. G. Panayiotou, International Conference on Indoor Positioning and Indoor Navigation (IPIN), 2011.

2. *"Demo: The Airplace Indoor Positioning Platform",* C. Laoudias, G. Constantinou, M. Constantinides, S. Nicolaou, D. Zeinalipour-Yazti and C. G. Panayiotou*,* Demo at the *10th ACM International Conference on Mobile Systems, Applications and Services* (MobiSys'12)

3. *"The Airplace Indoor Positioning Platform for Android Smartphones",* C. Laoudias, G. Constantinou, M. Constantinides, S. Nicolaou, D. Zeinalipour-Yazti, C. G. Panayiotou, Demo at the *13th IEEE International Conference on Mobile Data Management* (MDM'12)

4. *"Towards Planet-Scale Localization on Smartphones with a Partial Radiomap",* A. Konstantinidis, G. Chatzimilioudis, C. Laoudias, S. Nicolaou and D. Zeinalipour-Yazti, The 4th ACM International Workshop on Hot Topics in Planet-Scale Measurement (HotPlanet'12)

Περίληψη

Ο γενικότερος τομέας με τον οποίο ασχολήθηκα στην διπλωματική μου εργασία ήταν η τοπικοποίηση σε εσωτερικούς χώρους χωρίς την χρήση GPS αλλά με την χρήση πληροφοριών από τα Access Points.

Στην εργασία[8], αναπτύχθηκε μια εφαρμογή για την αποκεντρικοποιημένη (Decentralized) τοπικοποίηση μέσα σε εσωτερικούς χώρους. Αυτή η μέθοδος είχε το μειονέκτημα ότι η τοπικοποίηση ήταν πολύ αργή και αυτό καθυστερούσε την εφαρμογή να βρεί την τοποθεσία του χρήστη λόγω του μεγάλου αρχείου - radiomap - για το οποίο στην κάθε προσπάθεια τοπικοποίησης από την εφαρμογή θα έπρεπε να το διαπεράσει. Η δεύτερη μέθοδος η οποία χρησιμοποιείται, στις μέρες μας, είναι η κεντρικοποιημένη μέθοδος (Centralized) κατά την οποία ο χρήστης αποστέλλει συνεχώς δεδομένα από το περιβάλλον του στον server και ο server με την σειρά του αποστέλλει στον χρήστη την τοποθεσία του. Το πλεονέκτημα αυτής της μεθόδου είναι η πολύ γρήγορη η διαδικάσία τοπικοποίησης. Παρόλα ταύτα αυτή η μέθοδος δεν μας παρέχει ασφάλεια στα δεδομένα καθώς ο server ξέρει ανά πάσα στιγμή που θα βρίσκεται ο χρήστης.

Για την επίλυση των προβλημάτων που έχει η κάθε μέθοδος, σε αυτή την διπλωματική εργασία, αναπτύχθηκε ο αλγόριθμος Bloom ο οποίος συνδυάζει ουσιαστικά τα θετικά των δύο μεθόδων και μας παρέχει ασφάλεια στα δεδομένα μας αλλά και αρκετά γρήγορη διαδικασία ανεύρεσης της τοποθεσίας του χρήστη. Στόχος της διπλωματικής εργασίας εν τέλει είναι η μελέτη και η σύγκριση της κάθε προσέγγισης για να δούμε στο τέλος κατά πόσο η Bloom προσέγγιση θα έχει τα αναμενώμενα αποτελέσματα.

Ο Bloom αλγόριθμος, περιοδικά, στέλλει στον server συμπιεσμένες πληροφορίες, υπό μορφή ένος bloom vector, οι οποίες βάσει αυτού του bloom vector, ο server αποφασίζει ποιο μέρος του radio-map θα αποστείλει στο χρήστη. Τελικά του αποστέλλει ένα μικρό μέρος του radio-map, βάσει του bloom vector, το οποίο ο client το χρησιμοποιεί για να κάνει τοπικοποίηση. Με αυτό τον τρόπο και ο χρήστης έχει ασφάλεια στα δεδομένα και η τοπικοποίηση γίνεται αρκετά γρήγορα λόγω του περιορισμένου μεγέθους radio-map.

Συνολικά αξιολογήθηκαν οι τρείς προσεγγίσεις με δύο διαφορετικά data-sets το UCY Dataset και το ΚΟΙΟΣ Dataset. Αξιογήθηκαν με βάση τον συνολικό χρόνο τοποικοποίησης, τη κατανάλωση ενέργειας και τα μηνύματα αποστολής και λήψης από και προς τον χρήστη. Σύμφωνα με τα πειράματα που πραγματοποιήθηκαν πάνω στον προσομοιωτή, παρατηρήσαμε ότι για το UCY Dataset, ο Bloom αλγόριθμος βελτιώνεται κατά 80% όσο αφορά τον χρόνο, έχει 83% πιο λίγη κατανάλωση ενέργειας και τέλος 80% πιο λίγη χρήση του δικτύου σε σύγκριση με την Decentralized προσέγγιση. Στα ίδια επίπεδα κινήθηκε και η χρήση του KIOS Dataset, όπου ο Bloom αλγόριθμος βελτιώθηκε κατά 60% όσο αφορά τον χρόνο, έπειτα παρατηρήθηκε 60% πιο λίγη χρήση της κατανάλωσης ενέργειας και τέλος 80% πιο λίγη χρήση του δικτύου. Όσον αφορά την Centralized προσέγγιση είναι πολύ καλύτερη, από την Bloom, και για τα δύο data-sets. Παρόλαταύτα η Bloom μας δίνει την ασφάλεια για τα δεδομένα του χρήστη σε αντίθεση με την Centralized μέθοδο.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Βασικό υπόβαθρο

Σύμφωνα με πρόσφατες στατιστικές [17] οι άνθρω[16]ποι περνούν το 80 με 90% της ώρας τους μέσα σε εσωτερικούς χώρους συμπεριλαμβανομένων και των καταστημάτων, εμπορικών κέντρων, πανεπιστημίων ή ακόμη και αεροδρομίων. Επίσης, το 70% των κλήσεων από κινητό τηλέφωνο αλλά και το 80% της χρήσης διαδικτύου συμβαίνουν σε κλειστούς χώρους. [16] Το πιο πάνω στατιστικό συμπέρασμα είχε ως αποτελέσμα το έντονο ενδιαφέρον όσον αφορά τις εφαρμογές indoor Location-Based Services (LBS) και location-aware. Για παράδειγμα τη καθοδήγηση και τη τοπικοποίηση στο εσωτερικό ενός κτηρίου. Για να χρησιμοποιήσουμε τέτοιου είδους εφαρμογές, οι οποίες να λειτουργούν με ακρίβεια και σε κανονικό ρυθμό σε εσωτερικούς χώρους είναι προφανές ότι η τοπικοποίηση δεν θα γίνεται με την χρήση του GPS. Αυτό γιατί αφενός η χρήση του GPS δύσκολα μπορεί να λειτουργήσει εσωτερικά λόγω της μη κάλυψής του, αφετέρου η συνεχής χρήση του αποτελεί πρόβλημα στην κινητή συσκευή όσον αφορά την ψηλή κατανάλωση ενέργειας. Αυτό είχε ως αποτέλεσμα τα τελευταία 15 χρόνια να υπάρχει μια έντονη μελέτη προτάσσοντας διάφορα συστήματα τοπικοποίησης σε εσωτερικούς χώρους όπως υπέρυθρες, Bleutooth, Radio-frequency identification (RFID), Ultra-wideband (UWB), video cameras, ultrasound, Wireless Sensor Networks (WSN) και Wireless Local Area Networks (WLAN)[4].

Οι πιο πολλές από αυτές τις τεχνολογίες μπορούν να παρέχουν ψηλή ακρίβεια τοπικοποίησης, ωστόσο συνήθως χρειάζονται ακριβό εξοπλισμό για να μπορέσουν να δουλέψουν με ακρίβεια και να έχουν ένα καλό αποτέλεσμα. Απαιτείται, δηλαδή, βασικός εξοπλισμός όπως τα transmitters και οι αντένες τα οποία βοηθούν στη τοπικοποίηση. Αυτά όμως εκτός του ότι για την εγκατάσταση τους χρειάζεται αρκετός χρόνος, επίσης, κοστίζουν και ακριβά προκειμένου να εγκατασταθούν. Συνεπώς, αρκετά συστήματα τοπικοποίησης καταλήγουν να κάνουν χρήση WLAN's οφείλοντας την χρήση τους, στην έντονη παρουσία τους, σε εσωτερικούς χώρους. Η τοποθεσία μπορεί να προσδιοριστεί σε επίπεδο δωματίου κάνοντας χρήση των RSS σημάτων τα οποία εξάγονται, τελικά, από τα γειτονικά Access Points που μπορεί η συσκευή μας να αναγνωρίσει. Έτσι, κινητές συσκεύες οι οποίες διατίθενται στην αγορά, όπως smartphones και ταμπλέτες μπορούν να χρησιμοποιηθούν χωρίς καμιά αλλαγή στο υλικό τους γιατί συνήθως, τέτοιου είδους συσκευές, είναι εξοπλισμένες με σύστημα αναγνώρισης Access Points για πρόσβαση ασύρματα στο διαδίκτυο, και έτσι με μόνο μια εφαρμογή οι συσκευές έχουν την δυνατότητα να συλλέξουν τέτοιου είδους πληροφορίες.

Το τι είναι οι RSS τιμές θα το δούμε και πιο κάτω αναλύοντας με λεπτομέρεια την εφαρμογή που πρωτάθηκε το Καλοκαίρι του 2011 και υλοποιήθηκε από μια ομάδα φοιτητών, η οποία χρησιμοποιεί ακριβώς, αυτή τη μέθοδο της τοπικοποίησης σε εσωτερικούς χώρους. Η προσέγγιση έχει δύο φάσεις: την φάση κατά την οποία συλλέγονται όλα τα RSS Fingerprints για κάθε σημείο, ξεχωριστά της εσωτερικής τοποθεσίας που θέλουμε να πλοηγηθούμε (όπου αυτή η διαδικασία γίνεται με τη χρήση εμπορικής κινητής συσκευής). Έπειτα έχουμε τη δεύτερη φάση, κατά την οποία, γίνεται η πλοήγηση στον χώρο, μπορεί να υπολογιστεί η τοποθεσία του χρησιμοποιώντας τα παρόντα RSS Fingerprints, τα οποία συλλέγει ο χρήστης εκείνη την στιγμή, βρίσκοντας την καλύτερη αντιστοίχηση στο radio-map, το οποίο έχει τα δεδομένα που συλλέγηκαν στην πρώτη φάση για κάθε τοποθεσία στον χώρο.

Η εφαρμογή που υλοποιήθηκε το Καλοκαίρι του 2011 έχει το όνομα Airplace[1]. Πρόκειται για μια πλατφόρμα, η οποία θα επεξηγηθεί με σαφήνια παρακάτω, την οποία ο χρήστης μπορεί να εγκαταστήσει στο Android smartphone του και να την χρησιμοποιήσει. Περιλαμβάνει την δημιουργία του radio-map και έχει την δυνατότητα άμεσης τοπικοποίησης όπως επίσης δίνει και την δυνατότητα της χρήσης διαφορετικών αλγορίθμων για υπολογισμό της τοποθεσίας του χρήστη.

Για να μπορέσουμε να έχουμε ένα βασικό υπόβαθρο για την συγκεκριμένη διπλωματική εργασία σημαντικό είναι να επεξηγήσουμε κάποιους βασικούς ορισμούς έτσι ώστε να γίνει πιο κατανοητή η μετέπειτα επεξήγηση.

RSS Fingerprint: Πρόκειται για συγκεκριμένη μορφοποίηση πληροφορίας την οποία την παρέχει η κινητή συσκευή από τις πληροφορίες που λαμβάνει από τα Access Points. Κάθε fingerprint θεωρούμε ότι είναι στιγμιαία πληροφορία η οποία προέρχεται την δεδομένη στιγμή από όλα τα Access Points που βρίσκει.

Ένα RSS Fingerprint έχει συνήθως την συγκεκριμένη μορφή: timestamp, Longitude, Latitude (ή X, Y όταν πρόκειται για εσωτερικούς χώρους), MAC Address του Access Point, και το RSS (Received Signal Strength).

Το RSS Fingerprint είναι πληροφορία, η οποία είναι απαραίτητη για την τοπικοποίηση σε εσωτερικούς χώρους. Η συλλογή RSS Fingerprints είναι η προετοιμασία, κατά κάποιον τρόπο, η οποία γίνεται από τον χρήστη υποχρεωτικά προκειμένου να μπορέσει μετέπειτα το σύστημα να εντοπίζει τη τοποθεσία του στον εσωτερικό χώρο, τον οποίο βρίσκεται. Για την διεκπεραίωση αυτού του στόχου χρησιμοποιείται, στην περίπτωση μας, χρήση κινητής συσκευής είτε Android, είτε iOS με λογισμικό, το οποίο διεκπεραιώνει αυτή την λειτουργία.

```
# Timestamp, X, Y, MAC Address of AP, RSS
1324951068652 41.36557388305664 37.36746597290039 00:0b:fd:4a:71:a6 -70
1324951068652 41.36557388305664 37.36746597290039 00:0b:fd:4a:71:b5 -89
1324951068652 41.36557388305664 37.36746597290039 00:0e:38:3e:07:6b -89
1324951068652 41.36557388305664 37.36746597290039 00:0b:fd:4a:71:ad -92
1324951068652 41.36557388305664 37.36746597290039 00:0b:fd:4a:71:a9 -93
1324951068652 41.36557388305664 37.36746597290039 00:0b:fd:4a:71:d6 -95
```

Σχήμα 1.1: RSS Fingerprint

Κάθε RSS Fingerprint αποθηκεύεται σε αρχείο στη κινητή συσκευή, το οποίο χρησιμοποιείται για την μετατροπή του σε ένα άλλο αρχείο, το οποίο το ονομάζουμε radio-map. Για την δημιουργία του εν λόγω αρχείου, στην περίπτωση μας, γίνεται χρήση server με τον οποίο επικοινωνεί η κινητή μας συσκευή και το αποστέλει σ' αυτόν. Ακολούθως ο server βάσει του rss-log αρχείου δημιουργεί το radio-map, το οποίο έχει την ακόλουθη μορφοποίηση: #Longitude, Latitude, (ή #X, Y για εσωτερικούς χώρους) [Όλα τα mac-addresses όλων των Access Points τα οποία αναγνωρίστηκαν] και ακολούθως σε κάθε νέα γραμμή έχουμε και μια τοποθεσία η οποία ακολουθείται από όλα τα rss-values για κάθε ένα Access Point που αναγνώρισε η κινητή συσκευή.

Για παράδειγμα για το πιο πάνω RSS Fingerprint (Σχήμα 1.1) θα είχαμε το ακόλουθο Radio-map:

```
#X, Y, 00:0b:fd:4a:71:a6, 00:0b:fd:4a:71:b5, 00:0e:38:3e:07:6b, 00:0b:fd:4a:71:ad,
00:0b:fd:4a:71:a9, 00:0b:fd:4a:71:d6
41.36557388305664, 37.36746597290039, -70, -89, -89, -92, -93, -95
```

Σχήμα 1.2: radio-map

Το radio-map μας βοηθά στην τοπικοποίηση ακριβώς γιατί πλέον γνωρίζουμε, με την βοήθειά του, πόσο δυνατό σήμα έχουμε από κάθε Access Point για κάθε τοποθεσία μας. Περισσότερα θα αναλυθούν στη παρουσίαση της εφαρμογής Airplace, εφαρμογή την οποία και χρησιμοποιήσαμε. Επίσης, το ίδιο εφαρμόστηκε και στη πειραματική μέθοδο της παρούσας διπλωματικής εργασίας με δύο είδη data-set, τα οποία ήταν ουσιαστικά radio-maps.

1.2 **Υποκίνηση της εργασίας**

Παρόλη την ακρίβεια και την επιτυχία που παρουσιάζει η ανάπτυξη της πιο πάνω εφαρμογής - την οποία παρουσιάζω πολύ περιληπτικά στο υποκεφάλαιο 1.1- φαίνεται να παρουσιάζει πρόβλημα σχετικά με τον χρόνο τοπικοποίησης του χρήστη στο

εσωτερικό περιβάλλον στο οποίο βρίσκεται. Ειδικότερα, στην περίπτωση όπου ο χρήστης θέλει να συλλέξει στοιχεία από ένα μεγάλο εσωτερικό χώρο και οι τοποθεσίες στον χώρο που πρέπει να συλλέξει τα RSS Fingerprints είναι πολλές. Σε αυτή την περίπτωση δημιουργείται ένα μεγάλο radio-map, το οποίο η συσκευή στην προσπάθεια της να το διαπεράσει καθυστερεί αρκετά λόγω της καθυστέρησης του I/O που έχουν τα συστήματα των ηλεκτρονικών υπολογιστών. Σημειώνεται δε ότι κάθε φορά που ο χρήστης ζητά την τοποικοποίηση πρέπει είτε να διαπεράσει όλο το radio-map η εφαρμογή είτε πρέπει να διαπεραστεί η δομή, π.χ. δομή τύπου hashmap[5], στην οποία είναι ανεβασμένο το radio-map.

Για τον λόγο αυτό, αποτιμάται ένας νέος τρόπος τοπικοποίησης σε εσωτερικούς χώρους κατά τον οποίο η κινητή συσκευή έχει ένα πιο μικρό αρχείο για την τοπικοποίηση του χρήστη, τη στιγμή ακριβώς που θέλει να τοπικοποηθεί. Άρα, έχει την απαραίτητη πληροφορία που θέλει για το συγκεκριμένο σημείο που βρίσκεται την παρούσα στιγμή. Αυτό το αρχείο είναι αρκετό για να παρουσιάσει στον χρήστη την σωστή τοποθεσία του καθώς η διαπέραση όλου του αρχείου δεν έχει κανένα νόημα, αφού η χρήσιμη πληροφορία βρίσκεται μόνο σε ένα μέρος του αρχείου για μια συγκεκριμένη τοποθεσία στον χώρο. Κατ' επέκταση η προσπέλαση όλου του αρχείο κάθε φορά είναι αχρείαστη καθώς η πληροφορία που αναζητούμε βρίσκεται μόνο σε ένα μέρος του radio-map.

Τέλος, είναι πολύ σημαντικό, κατά την τοπικοποίηση του χρήστη, ο διακομιστής (server) να μην μπορεί να μάθει - βάσει των δεδομένων που ενδεχομένως στέλλει ο χρήστης - τη τοποθεσία του χρήστη. Αυτός είναι ένας σημαντικό παράγοντας που θα πρέπει να ληφθεί υπόψιν για την ανάπτυξη μιας νέας προσέγγισης τοπικοποίησης σε εσωτερικούς χώρους.

1.3 **Περίγραμμα της εργασίας**

Στο *πρώτο κεφάλαιο* παρουσιάστηκε η υποκίνηση, για την ανάπτυξη τρόπων κατά τους οποίους η τοπικοποίηση σε εσωτερικούς χώρους θα γίνεται πιο αποδοτικά αλλά και με ασφάλεια όσον αφορά την γνώση που μπορεί να έχει ο διακομιστής για την τοποθεσία

του χρήστη στον χώρο. Επίσης, παρουσιάστηκε το ενοιολογικό υπόβαθρο έτσι ώστε να μπορούμε να αναφερόμαστε πιο άνετα στη διαδικασία ανάπτυξης μεθόδων. Στο *δεύτερο κεφάλαιο* πρόκειται να παρουσιάσουμε το μοντέλο, το οποίο αναπτύχθηκε από μια ομάδα φοιτητών στο πανεπιστήμιο μας το οποίο εφαρμόζει στην πράξη την τοπικοποιήση σε εσωτερικούς χώρους. Ακόμη, θα παρουσιάσουμε μια επέκταση αυτής της πλατφόρμας που υλοποιήθηκε στα πλαίσια της παρούσας διπλωματικής εργασίας. Στο *τρίτο κεφάλαιο* θα γίνει εκτενής παρουσίαση του προβλήματος που δημιουργούν οι δύο προσεγγίσεις (οι δύο αλγόριθμοι τοπικοποίησης), οι οποίες υπάρχουν μέχρι τώρα, και θα αναφερθούν με λεπτομέρεια οι λόγοι για τους οποίους οδηγηθήκαμε στην ανάπτυξη μια νέας προσέγγισης τοπικοποίησης σε εσωτερικούς χώρους. Επίσης, θα γίνουν κατανοητα, τα θετικά και τα αρνητικά των προσεγγίσεων αυτών, αλλά θα γίνει και μια αρχική νύξη αναφορικά με τη νέα προσέγγιση που προτείνεται. Στο τέταρτο κεφάλαιο θα αναλυθεί εκτενώς η ανάπτυξη της νέας προσέγγισης τοπικοποίησης σε εσωτερικούς χώρους και πώς αυτή η προσέγγιση θα παρέχει στον χρήστη μεγάλη ασφάλεια δεδομένων και γρήγορη τοπικοποίηση όσο επίσης και πιο λίγη κατανάλωση ενέργειας στη κινητή συσκευή του. Στο *πέμπτο* και *εκτό κεφάλαιο* θα γίνει αναφορά στον προσομοιωτή που υλοποιήθηκε για τη σύγκριση των τριών προσεγγίσεων και παρουσίαση κάποιων πειραματικών αποτελεσμάτων βάσει συγκεκριμένων data-sets. Τέλος, στο έβδομο κεφάλαιο θα είμαστε σε θέση να παρουσιάσουμε τα τελικά συμπεράσματα και τις επιπλέον βελτιστοποιήσεις που μπορούν να γίνουν. Τέλος, στο Παράρτημα Α παρουσιάζουμε ένα δείγμα των αναλυτικών αποτελεσμάτων της πειραματικής μας μελέτης σε πίνακες.

Κεφάλαιο 2

Η πλατφόρμα γεωτοποθέτησης Airplace

Στο κεφάλαιο αυτό θα αναλύσουμε με λεπτομέρεια πώς γίνεται η χρήση του Airplace και το μοντέλο, το οποίο χρησιμοποιείται για τοπικοποίηση. Εν συνεχεία, θα παρουσιάσουμε ακριβώς τον τρόπο της τοπικοποίησης, ούτως ώστε να γίνει εντελώς κατανοητή η DRA (Decentralized Radiomap Algorithm) προσέγγιση στην οποία βασίζεται το Airplace. Επίσης, θα αναπτυχθούν λεπτομερώς οι δύο προσεγγίσεις που υπάρχουν στις μέρες μας αλλά επίσης θα γίνει και αναφορά στη νέα προσέγγιση, η οποία προτείνεται σε αυτή την διπλωματική εργασία. Ακόμη, για να είμαστε πιο κατανοητοί, θα γίνουν κατανοητοί οι λόγοι για τους οποίους οδηγούμαστε στην Bloom προσέγγιση, η οποία μπορεί να λύσει τα προβλήματα των δύο άλλων προσεγγίσεων. Τέλος, παρουσιάζουμε μια επέκταση της πλατφόρμας Airplace που έγινε στα πλαίσια της παρούσας διπλωματικής εργασίας.

2.1 Εισαγωγή

Το Airplace είναι μια εφαρμογή η οποία αναπτύχθηκε κάτω από την επίβλεψη του κου Δημήτρη Ζεϊναλιπούρ (επιβλέποντα καθηγητή μου) και τον κο Χρίστο Λαουδιά (διδακτορικό φοιτητή στο Πανεπιστήμιο μας). Η αρχική ιδέα της εφαρμογής ήταν η

τοπικοποιήση σε εξωτερικούς χώρους χωρίς τη χρήση GPS. Αυτό υλοποιήθηκε στα πλαίσια του project του μαθήματος "Προγραμματισμός Συστημάτων" από ομάδα φοιτητών. Η υλοποίηση στεύθηκε με επιτυχία και μετά από πρόταση του καθηγητή μας - κου Ζεϊναλιπούρ - συνεχίσαμε την υλοποίηση της εφαρμογής για τοπικοποίηση σε εσωτερικούς χώρους. Η εφαρμογή ακουλούθως παρουσιάστηκε σε Demo στο συνέδριο IPIN (Indoor Positioning and Indoor Navigation) [6] στην Πορτογαλία, όπου είχε επιτυχία. Η συγκεκριμένη εφαρμογή επεκτάθηκε στα πλαίσια της παρούσας διπλωματικής με την λειτουργία αναγνώρισης ορόφων βάσει της παρούσας θέσης του χρήστη και τη λήψη των RSS τιμών.

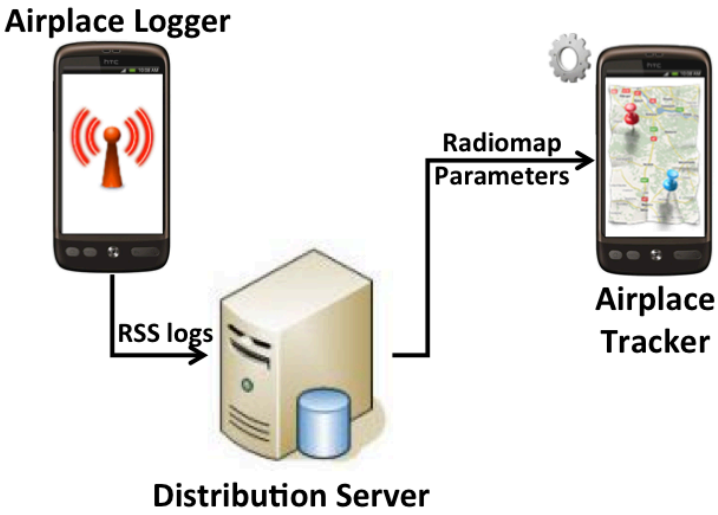
2.2 Μοντέλο του Airplace

Το σύστημα αυτό εφαρμόστηκε σε Android συσκευές, ακολουθώντας μια αρχιτεκτονική βασισμένη σε κινητές συσκευές και στη χρήση δικτύου. Αρχικά, το κόστος για επικοινωνία είναι χαμηλό, στη περίπτωση συνεχούς τοπικοποίησης από πολλούς χρήστες, διότι ουσιαστικά γίνεται η χρήση της μεθόδου όπου οι πληροφορίες για τοπικοποίηση στον συγκεκριμένο χώρο βρίσκονται αποκλειστικά στην ίδια την κινητή συσκευή. Επίσης, γι αυτόν ακριβώς τον λόγο δεν υπάρχει θέμα συμφόρισης στο δίκτυο καθώς ο χρήστης απλά κατεβάζει τις πληροφορίες από τον server μία και μόνο φορά, για κάθε χώρο, και πλοηγείται στον χώρο. Κατά δεύτερον δεν υπάρχει πρόβλημα διαρροής πληροφοριών του χρήστη προς τον server γιατί αρκιβώς ο υπολογισμός της τοπικοποίησης γίνεται πάνω στην κινητή συσκευή και όχι στον server.

Ένα τυπικό σενάριο τοπικοποίησης μπορεί να περιγραφεί σε τρία βήματα: (α) Ο χρήστης μπαίνει σε ένα εσωτερικό χώρο, όπως για παράδειγμα φοιτητικές εστίες ή σε ένα εμπορικό κέντρο ή αεροδρόμιο, ο οποίος είναι καλυμένος με διάφορα WLAN Access Points. (β) Η κινητή συσκευή του χρήστη λαμβάνει όλα τα αρχεία (RSS radio-map και parameters αρχείο) από τον τοπικό server με μια μόλις επικοινωνία και (γ) ο χρήστης τοπικοποιείται από μόνος του ανεξάρτητα χρησιμοποιώντας μόνο τη τοπική γνώση που έχει, δηλαδή, το κατεβασμένο radio-map που έχει στην συσκευή του και τη

συνεχή λήψη RSS fingerprints που λαμβάνει εκείνη την στιγμή ανάλογα με τον τόπο που είναι στον εσωτερικό χώρο.

Η αρχιτεκτονική του Airplace παρουσιάζεται στο Σχήμα 2.1 και αποτελείται από τρείς εφαρμογές. Οι 2 είναι εφαρμογές που τρέχουν σε κινητές συσκευές Android (Airplace Logger και Airplace Tracker) και η άλλη είναι η εφαρμογή RadioMap Server, η οποία τρέχει στον υπολογιστή και έχει τη λειτουργία ενός server. Οι πιο πάνω εφαρμογές θα παρουσιαστούν πιο κάτω αναλυτικότερα. Γενικότερα η εφαρμογή Airplace Logger παρέχει τη δυνατότητα συλλογής RSS δεδομένων, τα οποία αποθηκεύει τοπικά στην κινητή συσκευή. Επειτα, η εφαρμογή Airplace Tracker υλοποιεί τη τοπικοποίηση του χρήστη στον χώρο, ενώ ο RadioMap Server είναι υπεύθυνος για τη δημιουργία του radio-map χρησιμοποιώντας ακριβώς το αρχείο που συλλέκτηκε και δημιουργήθηκε από την εφαρμογή Airplace Logger. Επίσης, έχει την ευθύνη αποστολής του τελικού αρχείου radio-map σε πολλαπλούς αιτητές αλλά ακόμη και επιπλέον αρχεία, τα οποία είναι σημαντικά κατά την τοπικοποίηση, όπως είναι το parameters αρχείο.



Σχήμα 2.1: Αρχιτεκτονική συστήματος τοπικοποίησης του Airplace.

Εφαρμογή Airplace Logger: Η εφαρμογή Airplace υλοποιήθηκε γύρω από το Android RSS API για σάρωση και λήψη δειγμάτων των δεδομένων για συγκεκριμένες τοποθεσίες. Αυτά τα δείγματα περιλαμβάνουν τις MAC διευθύνσεις και τα επίπεδα των RSS (Received Signal Strength) για κάθε ένα από τα γειτονικά Access Points που βρίσκονται γύρω από τη κινητή συσκευή. Επίσης, αποθηκεύει τις συνεταγμένες για τη συγκεκριμένη τοποθεσία που βρίσκεται ο χρήστης. Η αρχική οθόνη της εφαρμογής παρέχει στον χρήστη τις βασικές λειτουργίες για να αρχίσει ή για να σταματήσει την αναζήτηση για Access Points. Επίσης υπάρχουν ρυθμίσεις, στις οποίες ο χρήστης έχει τη δυνατότητα να ορίσει το αρχιτεκτονικό σχέδιο του εσωτερικού χώρου στο οποίο θέλει να σαρώσει, να ορίσει το όνομα του αρχείου στο οποίο θέλει να αποθηκεύονται τα δεδομένα αλλά και να ορίσει πόσα δείγματα να παραλαμβάνει η κινητή συσκευή κάθε φορά που θέλει να κάνει λήψη πληροφοριών (Βλέπε σχήμα 2.2). Αρχικά ο χρήστης ορίζει τον αριθμό των δειγμάτων που θέλει να λαμβάνει για κάθε τοποθεσία του στον χώρο που θέλει να συλλέξει στοιχεία (Βλέπε σχήμα 2.2, στο μέσο) και μετά μετακινείται στον χώρο, ενώ για την συλλογή των συντεταγμένων ο χρήστης κάνει κλικ στο αρχιτεκτονικό σχέδιο τη τοποθεσία που βρίσκεται εκείνη τη συγκεκριμένη στιγμή. Τέλος, αφού ο χρήστης πιέσει στην οθόνη τη τοποθεσία που βρίσκεται και παρουσιαστούν οι συντεταγμένες (Βλέπε σχήμα 2.2 αριστέρα, δεξιά του κουμπιού Record info..), συλλέγει τα δεδομένα πιέζοντας το Record info... όπως φένεται ξεκάθαρα στο σχήμα 2.2.

Τα δεδομένα που συλλέχτηκαν αποθηκεύονται τοπικά σε ένα log αρχείο και οι χρήστες μπορούν ανά πάσα στιγμή να διανέμουν το αρχείο στο σύστημα για να ενημερώσουν έτσι το radio-map, το οποίο θα δώσει αργότερα τη δυνατότητα πλοήγησης του χρήστη στον χώρο. Συγκεκριμένα, οι χρήστες μπορούν να ορίσουν την IP διεύθυνση και τον αριθμό του port του Radiomap Server μέσω των Preferences, ούτως ώστε να κατορθώσουν με αυτόν τον τρόπο να ανεβάσουν το τελικό log αρχείο τους χρησιμοποιώντας πρωτόκολλο κειμένου.

Σχήμα 2.2: Στιγμιότυπα από την εφαρμογή Airplace Logger.

Εφαρμογή Airplace Tracker: Η εφαρμογή, η οποία είναι υπεύθυνη για τη τοπικοποίηση του χρήστη, τρέχει στα Android smartphones και επικοινωνεί με τον server για να κατεβάσει το radio-map και το αρχείο parameters, τα οποία είναι αναγκαία. Με την αποθήκευση αυτών των αρχείων ο χρήστης έχει τη δυνατότητα πλέον να αρχίσει τη τοπικοποίηση χωρίς να έχει καμιά εξάρτηση με τον Radiomap Server. Η διεπαφή του Airplace Tracker παρουσιάζεται στο Σχήμα 2.3 (αριστέρα), όπου ο χρήστης μπορεί να ορίσει τα δικά του Preferences (για παράδειγμα το αρχιτεκτονικό σχέδιο, τις ρυθμίσεις για κατέβασμα των αρχείων από τον server κ.ο.κ). Επίσης, μπορεί να επιλέξει οποιοδήποτε αλγόριθμο επιθυμεί να τρέχει για τη τοπικοποίηση. Η ομάδα, η οποία ευθύνεται για τη τελική υλοποίηση όλης της πλατφόρμας, υλοποίησε διάφορους αλγόριθμους βασισμένους στη χρήση RSS Fingerprint. Στη λίστα αυτών των αλγορίθμων περιλαμβάνονται και οι εξής: K-Nearest Neighbour (KNN), Weighted K-Nearest Neighbour (WKNN) [2][13], probabilistic Maximun A Posteriori (MAP), Minimum Mean Square Error (MMSE)[17],[15]. Επίσης, η πλατφόρμα υποστηρίζει δύο

state-of-the-art αλγόριθμους, οι οποίοι είναι αναπτυγμένοι σε εσωτερικό. Πρόκειται για τους Radial Basis Function (RBF) networks [12] και για τον Subtract on Negative Add on Positive (SNAP)[11] αλγόριθμο.

Η συγκεκριμένη εφαρμογή υποστηρίζει διπλή λειτουργία όσον αφορά το θέμα της τοπικοποίησης. Η πρώτη λειτουργία είναι η χρήση της τοπικοποίησης σε πραγματικό χρόνο (Online Mode) ενώ η δεύτερη λειτουργία είναι η εκτίμηση και η σύγκριση των διαφορετικών αλγορίθμων χρησιμοποιώντας προυπάρχοντα δεδομένα για δοκιμή (Offline Mode). Στην λειτουργία του Online Mode μετά από το κατέβασμα του radio-map από τον Radiomap Server και μετά από την επιλογή του αλγορίθμου που θα επιλέξει ο χρήστης μπορεί να τοπικοποιηθεί στιγμιαία πατώντας απλά το κουμπί Find Me και αμέσως, στο αρχιτεκτονικό σχέδιο που έχει επιλέξει από πριν, θα εμφανίσει την τοποθεσία του. Μία εναλακτική λειτουργία που έχει τη δυνατότητα να επιλέξει ο χρήστης, χρησιμοποιώντας την Online διαδικασία, είναι η χρήση του Track Me. Όταν ο χρήστης λοιπόν ενεργοποιήσει την λειρουργία του Track Me (πατώντας το αντίστοιχο κουμπί) τότε η εφαρμογή εντοπίζει τη τοποθεσία του χρήστη την ώρα ακριβώς που πλοηγείται στο κτήριο ή τον εσωτερικό χώρο που επέλεξε να πλοηγηθεί. Σε αυτή την περίπτωση, δηλαδή της λειτουργίας Track Me, ο υπολογισμός της τοποθεσίας του χρήστη (πράσινος κύκλος) ενημερώνεται κάθε ένα δευτερόλεπτο ενώ οι παλιές τοποθοσίες του χρήστη στο αρχιτεκτονικό σχέδιο δηλώνονται μέσω των κόκκινων τελειών. (βλέπε Σχήμα 2.3, στο μέσο).

Στην περίπτωση του Offline Mode, η εφαρμογή προσφέρει στον χρήστη την επιλογή της προσομοίωσης ενός μεγάλου αριθμού επιτυχημένων αιτήματων τοπικοποίησης (ειδικότερα, ένα αρχείο το οποίο αποτελείται από μερικά RSS Fingerprints τα οποία έχουν συλλεγεί από διαφορετικές τοποθεσίες και μπορούν να αποθηκευτούν στην sd-card όπως ακριβώς όταν συλλέγονται σε πραγματικό χρόνο). Αυτή η λειτουργία, η οποία παρουσιάζεται στο Σχήμα 2.3 (δεξιά), είναι μάλλον χρήσιμη για τη φόρτωση δεδομένων προς δοκιμή (για παράδειγμα ένα RSS log αρχείο το οποίο συλλέχτηκε με την εφαρμογή Airplace Logger). Αυτή η λειτουργία (Offline Mode) έχει σκοπό να δει την απόδοση του κάθε αλγορίθμου ανάλογμα με τα πιο κάτω κριτήρια:

Χρόνος εκτέλεσης: Μέτρηση του μέσου χρόνου που χρειάζεται στην πράξη για να εκτελεστεί η τοπικοποιήση στα smartphones.

Ακρίβεια τοπικοποίησης:

Παρουσιάζει το μέσο σφάλμα τοπικοποίησης που αφορούν τα test fingerprints.

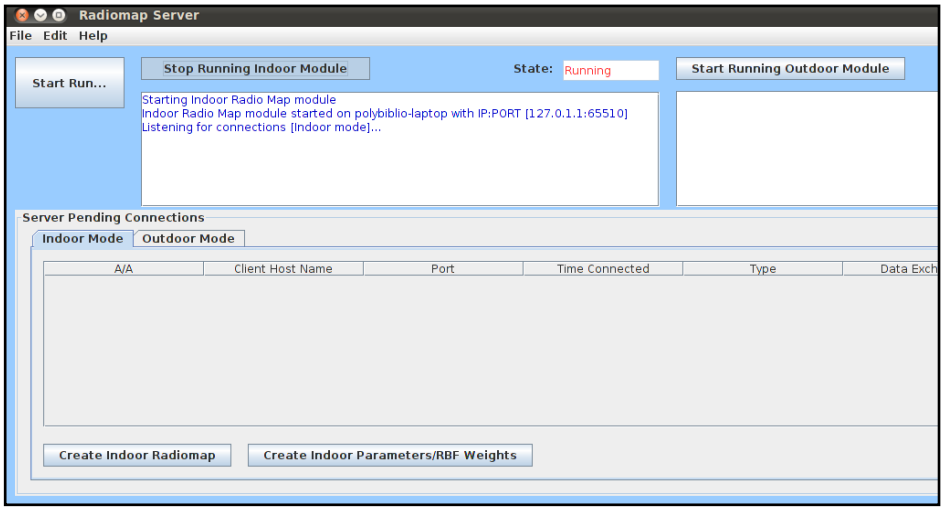
Σχήμα 2.3: Στιγμιότυπα από την εφαρμογή Airplace Tracker.

Κατανάλωση ενέργειας: Υπολογισμός της κατανάλωση ενέργειας που μπορεί να κοστίσει στην μπαταρία, με την χρήση του Power Tutor [14] κατά την διάρκεια τοπικοποίησης στις κινητές συσκευές.

Εφαρμογή του Radiomap Server: Ο συγκεκριμένος server είναι ο υπεύθυνος για δημιουργία του τελικού radio-map και όλων των παρεμφερών αρχείων, τα οποία βοηθούν την εφαρμογή Airplace Tracker να παρουσιάσει την τοποθεσία του χρήστη στον χώρο. Όταν ο server ενεργοποιείται τότε αναμένει αιτήσεις από διάφορους χρήστες, οι οποίοι είτε έχουν έτοιμα δεδομένα μαζεμένα για αποστολή σ' αυτόν είτε αναζητούν από τον server τα απαραίτητα αρχεία για να αρχίσουν τοπικοποίηση στον χώρο (Σχήμα 2.4). Για να δημιουργήσει ο server το radio-map αρχείο διαπερνά όλα τα

διαθέσιμα RSS Log αρχεία, τα οποία πιθανώς να δημιουργηθήκαν από διάφορους χρήστες, και υπλογίζει την μέση RSS τιμή ανά MAC διεύθυνση. Αυτό το κάνει με τον υπολογισμό του μέσου όρου από όλα τα δείγματα (samples) τα οποία μαζεύτηκαν από κάθε ξεχωριστή τοποθεσία. Τέλος, όλες οι τιμές συμπιέζονται και αποθηκεύονται σε ένα μοναδικό radio-map αρχείο έτσι ώστε κάθε γραμμή να αναφέρεται στον μέσο όρο των RSS τιμών σε συγκεκριμένη τοποθεσία (συντεταγμένες δηλαδή X,Y) για κάθε στήλη στην αντίστοιχη MAC διεύθυνση. Με αυτή την μέθοδο εξαλείφεται ο θορύβος στις RSS μετρήσεις ως επίσης και οι ακραίες τιμές. Επίσης, το τελικό radio-map είναι αποδοτικά συμπιεσμένο, δίνοντας στον χρήστη τη δυνατότητα τοπικοποίησης στον χώρο όσο πιο γρήγορα γίνεται καθώς και μια μοναδική τοπικοποίηση (η εφαρμογή χρειάζεται την προσπέλαση όλου του αρχείου). Ακόμη με την συμπίεση του αρχείου μπορεί η εφαρμογή Airplace Tracker να κατεβάσει σε πιο μικρό χρονικό διάστημα τα δεδομένα που χρειάζεται. Για παράδειγμα, στο Κέντρο Ερευνών ΚΟΙΟΣ του πανεπιστημίου Κύπρου με χώρο 560 τετραγωνικά μέτρα, με 9 WLAN Access Points και με 105 μοναδικές τοποθεσίες το μέγεθος του radio-map είναι περίπου 32KB και χρειάζεται ελάχιστα δευτερόλεπτα για να το κατεβάσει.

Ένα άλλο πολύ ενδιαφέρον χαρακτηριστικό του server είναι ότι μπορεί να προσαρμώσει τις παραμέτρους που χρειάζεται κάθε αλγόριθμος για ένα συγκεκριμένο σύνολο δεδομένων που έχει αποστείλει ο χρήστης για επεξεργασία. Για παράδειγμα η επιλογή μιας τιμής K για τον KNN αλγόριθμο δεν είναι ασήμαντη και μπορεί να επηρεάσει την επίδοση του συγκεκριμένου αλγορίθμου. Η υλοποίηση έχει γίνει με τέτοιο τρόπο έτσι ώστε να εφαρμοστεί μια αποδοτική προσέγγιση για την αντιμετώπιση αυτού του ζητήματος χρησιμοποιώντας την εφαρμογή Airplace Logger, μαζεύοντας επιπλέον RSS δείγματα για επικύρωση των δεδομένων. Έπειτα ο KNN αλγόριθμος τρέχει, στην μεριά του server, με διαφορετικές τιμές του K και η πιο κατάλληλη τιμή επιλέγεται. Μια παρόμοια προσέγγιση εφαρμόζεται και για τους άλλους αλγόριθμους. Με αυτόν ακριβώς τον τρόπο, ο server δημιουργεί το RSS radio-map και μετά επιλέγει τις κατάλληλες τιμές για το σύνολο το παραμέτρων που αποθηκεύει σε ένα ξεχωριστό αρχείο το οποίο μεταφέρεται με το αντίστοιχο radio-map στην εφαρμογή Airplace Tracker.



Σχήμα 2.4: Στιγμιότυπο από την εφαρμογή του Radiomap Server.

Σημειώνουμε ότι το Airplace είναι βασισμένο στην Decentralized προσέγγιση.

2.3 Επέκταση της πλατφόρμας Airplace

Στα πλαίσια της διπλωματικής εργασίας έγινε και επέκταση πάνω στην υφιστάμενη πλατφόρμα που παρουσιάστηκε στο IPIN ’11. Η επέκταση αφορούσε την περίπτωση όπου ο χρήστης θα ήθελε να πλοηγηθεί σε ένα ολόκληρο κτήριο, το οποίο θα είχε πολλαπλούς ορόφους. Σε αυτή την περίπτωση θα ήθελε, κατά την κίνηση του στον χώρο, η εφαρμογή να αναγνωρίζει την περίπτωση που ο χρήστης μετακινηθεί σε άλλο όροφο να γίνει αυτόματη εναλλαγή του αρχιτεκτονικού σχέδιου και τέλος να φορτωθούν τα κατάλληλα δεδομένα για συνέχιση της πλοήγησης στον άλλο όροφο.

Ο τρόπος για να μπορέσει η εφαρμογή να αναγνωρίσει τον διαφορετικό όροφο ήταν η χρήση των RSS τιμών σε συνδυασμό με κάποια βοηθητικά αρχεία .floor, τα οποία αποτελούνται από κάποια δεδομένα του αντίστοιχου ορόφου και βάσει αυτών κάθε φορά που η κινητή συσκευή λαμβάνει σε πραγματικό χρόνο RSS τιμές, γίνονται κάποιοι έλεγχοι και αποφασίζει σε πιο .floor αρχείο αντιστοιχεί.

Προεργασία: Κατά τη φάση της προεργασίας θα πρέπει να συλλεγούν κανονικά οι τιμές με τον Airplace Logger (βλέπε υποκεφάλαιο 2.2) και να αποσταλούν στον server για

δημιουργία του radio-map. Ο server δημιουργεί το radio-map. Έπειτα βάσει του radio-map βρίσκουμε τον μέσο όρο των RSS τιμών για κάθε MAC διεύθυνση και το καταγράφουμε στο .floor αρχείο του ορόφου που αντιστοιχούσε το radio-map.

```
00:1f:6c:a8:e6:a1, -103.3
00:1f:6c:a8:e6:a0, -107.72500000000001
00:0e:84:4b:0b:9f, -105.97142857142858
00:0e:84:4b:0b:c0, -98.71600000000001
74:ea:3a:e7:85:9a, -102.47999999999999
```

Σχήμα 2.5: Μέρος από ένα .floor αρχείο. Παρουσιάζει την MAC διεύθυνση και την μέση RSS τιμή του.

Αφού δημιουργηθούν τα .floor αρχεία για κάθε όροφο προστίθενται στην sd-card μαζί με τα άλλα αρχεία. Κατά την φάση της τοπικοποίησης, όπου ο χρήστης λαμβάνει τις RSS τιμές από τα γειτονικά Access Points, τοπικοποιείται στον χώρο αλλά ταυτόχρονα ελέγχει εάν οι RSS τιμές για την συγκεκριμένη MAC διεύθυνση του συγκεκριμένου Access Point, είναι πιο κοντά σε κάποιο από τα .floor αρχεία. Σε όποιο όροφο αντιστοιχεί, βάσει του .floor αρχείο, παίρνει +1 πιθανότητα να είναι στον συγκεκριμένο όροφο. Για παράδειγμα αν ο χρήστης έλαβε την MAC διεύθυνση: 00:1f:6c:a8:e6:a0 με RSS τιμή -100 και έχουμε cs1.floor (για τον πρώτο όροφο) 00:1f:6c:a8:e6:a0 με RSS τιμή -92,3 και cs2.floor (για τον δεύτερο όροφο) και 00:1f:6c:a8:e6:a0 με RSS τιμή -101,3 τότε η συγκεκριμένη MAC διεύθυνση θεωρούμε ότι ανήκει στον πρώτο όροφο. Αυτή η διαδικασία γίνεται για κάθε πληροφορία από κάθε Access Point που λαμβάνει ο χρήστης και σε όποιον όροφο αντιστοιχούν περισσότερες MAC διευθύνσεις τότε επιλέγεται ότι θα είναι αυτός που πρέπει να παρουσιάζεται στον χρήστη. Αν είναι ο ίδιος τότε παραμένει ως έχει. Αν είναι διαφορετικός από το παρόν αρχιτεκτονικό σχέδιο τότε τα δεδομένα αλλάζουν δυναμικά.

Η πιο πάνω μέθοδος μπορεί να εφαρμοστεί με τη προυπόθεση ότι ο χρήστης συνέλεξε περίπου ίσο αριθμό τοποθεσιών στους δύο ή περισσότερους ορόφους που έχει το κτήριο.

2.4 Εγκατάσταση του Airplace στο Τμήμα Πληροφορικής

Στα πλαίσια της παρούσας διπλωματικής εργασίας έγινε χαρτογράφηση όλου του κτηρίου του τμήματος της πληροφορικής. Βέβαια μιλόντας για χαρτογράφηση εννοούμε την συλλογή RSS Fingerprints σε όλο το κτήριο από σημείο σε σημείο των εσωτερικών χώρων.

Αυτό έχει γίνει για δύο λόγους. Από την μια να έχουμε ένα μεγάλο data-set, το οποίο μας έχει βοηθήσει για την πειραματική μας μεθοδολογία με την χρήση του προσομοιωτή και από την άλλη για να μπορεί όποιοσδήποτε χρήστης ο οποίος έχει Android κινητή συσκευή να πλοηγηθεί σ' αυτό ανά πάσα στιγμή.

Για να μπορεί λοιπόν ο χρήστης να πλοηγηθεί στο κτήριο μας δημιουργήσαμε μια ιστοσελίδα (βλέπε σχήμα 2.1) στην οποία ο χρήστης μπορεί να κατεβάσει τις εφαρμογές του Airplace, να βρεί τον τρόπο που χρησιμοποιείται η πλατφόρμα και τέλος έχει την δυνατότητα να ο ίδιος να κάνει τις την χαρτογράφιση (με την χρήση του Airplace Logger) για τον δικό του χώρο και να πλοηγηθεί ανά πάσα στιγμή.

Δημιουργήσαμε την πιο πάνω σελίδα αφενός για να δείξουμε και να παρουσιάσουμε σε όποινδήποτε πως λειτουργεί αυτή η εφαρμογή και τι κάνει, αφετέρου για να μπορεί όποιος πρώτη φορά εισέρχεται στο κτήριο της πληροφορικής να πλοηγηθεί στο κτήριο χωρίς κάποιο κόπο.

Σχήμα 2.1: Στιγμιότυπο από την ιστοσελίδα του Airplace

Κεφάλαιο 3

Αλγόριθμοι Τοπικοποίησης Με Πλήρες Radiomap

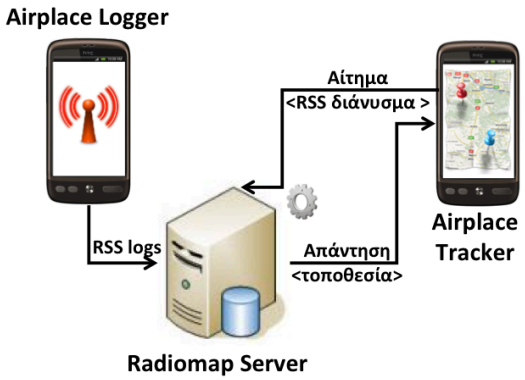
3.1 Εισαγωγικά

Πιο κάτω θα γίνουν κατανοητοί οι αλγορίθμοι για τοπικοποιήση, τους οποίους υλοποιήσαμε στον προσομοιωτή, αναπτύχθηκαν στα πλαίσια της παρούσας εργασίας και θα δούμε τους λόγους ακριβώς που οδηγούμαστε στην νέα Bloom προσέγγιση που ήταν και το βασικό ζήτημα σ' αυτήν την διπλωματική εργασία. Στα πλαίσια της περιγραφής των αλγορίθμων θα γίνουν επίσης κατανοητά τα διάφορα προβλήματα που παρουσιάζει κάθε αλγόριθμος για τοπικοποίηση και πώς αυτά επιλύονται με την εισαγωγή της νέας μας προσέγγισης.

3.2 CRA τεχνική (Centralized Radiomap Algorithm)

Σε αυτή την τεχνική ο χρήστης αφού βρίσκεται σε μια τοποθεσία, εσωτερικά του κτηρίου, αποστέλλει στον server τα δεδομένα, τα οποία λαμβάνει εκείνη την στιγμή από την συγκεκριμένη τοποθεσία που βρίσκεται. Στην περίπτωση μας ο client στέλλει τα RSS δεδομένα που λαμβάνει σε πραγματικό χρόνο, στον server. Ο server λαμβάνει τα συγκεκριμένα δεδομένα, έπειτα εντοπίζει την ακριβή τοποθεσία του χρήστη, δεδομένου ότι έχει το αντίστοιχο radio-map, και τέλος στέλλει πίσω την τοποθεσία του χρήστη. Η συγκεκριμένη προσέγγιση έχει τα ακόλουθα χαρακτηριστικά:

- *Κατανάλωση ενέργειας:* Η χρήση της CRA προσέγγισης συνεπάγεται τη χαμηλή κατανάλωση ενέργειας. Αυτό συμβαίνει επειδή όλοι οι υπολογισμοί γίνονται στην μεριά του server. Επομένως, η κινητή μας συσκευή δεν θα έχει κάποιο επιπλέον κόστος στην κατανάλωση ενέργειας.
- *Χρόνος υπολογισμού της τοποθεσίας:* Η υπολογίσιμη ισχύς στον server είναι πολύ μεγαλύτερη από του client, ως εκ τούτου ο υπολογισμός της τοποθεσίας του χρήστη γίνεται πολύ πιο γρήγορα παρά την ώρα που καταναλώνει ο client για υπολογισμό της τοποθεσίας του. Επίσης, η αποστολή της τοποθεσίας από τον server στον client, κοστίζει ελάχιστους διαδικτυακούς πόρους.
- *Προστασία προσωπικών δεδομένων:* Όσον αφορά τη προστασία των προσωπικών δεδομένων του χρήστη η CRA προσέγγιση δεν είναι καθόλου καλή καθώς ο server γνωρίζει ανά πάσα στιγμή που βρίσκεται ο χρήστης, αφού αναζητείται η τοποθεσία του από τον server.



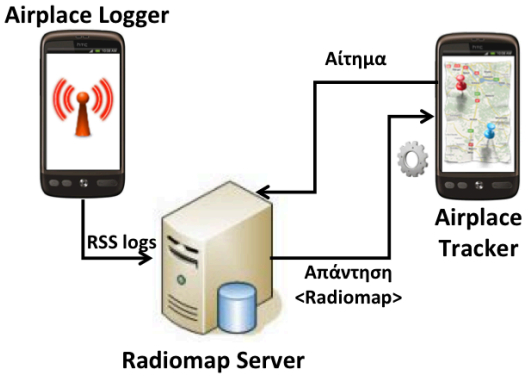
Σχήμα 3.1: Αρχιτεκτονική της CRA

3.3 DRA τεχνική (Distributed Radiomap Algorithm)

Προκειμένου να περιορίσουμε το πρόβλημα της προστασίας προσωπικών δεδομένων προτείνεται η DRA προσέγγιση (Distributed Radiomap Algorithm) (βλέπε Σχήμα 3.2). Σ' αυτή την προσέγγιση είναι βασισμένη και η πλατφόρμα του Airplace, την οποία εξηγήσαμε με λεπτομέρεια στο υποκεφάλαιο 2.3.

Η DRA τεχνική έχει τα πιο κάτω χαρακτηριστικά:

- *Κατανάλωση ενέργειας:* Το αρχείο, το οποίο αποστέλλει ο χρήστης στον server, όταν έχει όλα τα δεδομένα που χρειάζεται είναι πολύ μεγάλο. Το ίδιο και το αρχείο radio-map που ο server αποστέλλει στον χρήστη. Για αυτόν τον λόγο η κινητή συσκευή πρέπει να διαπεράσει αυτό το αρχείο κάθε φορά που πρέπει να κάνει τοπικοποίηση. Αυτό έχει ως αποτέλεσμα τη υψηλή κατανάλωση ενέργειας στην κινητή συσκευή, πράγμα ανεπιθύμητο ειδικά για τις κινητές συσκευές που η κατανάλωση της ενέργειας πρέπει να είναι περιορισμένη.
- *Χρόνος υπολογισμού της τοποθεσίας:* Ο client σε αυτή την προσέγγιση έχει υποχρέωση να κάνει τη τοπικοποίηση. Στη περίπτωση που το αρχείο, radio-map, είναι πολύ μεγάλο (λόγω του μεγάλου εσωτερικού χώρου) τότε ο χρόνος τοπικοποίησης κάθε φορά θα είναι πολύ αργός καθώς η προσπέλαση ενός μεγάλου αρχείου από μια συσκευή, με σχετικά μικρή υπολογιστική ισχύ, κοστίζει πολύ χρόνο.
- *Προστασία προσωπικών δεδομένων:* Αναφορικά με τη προστασία των προσωπικών δεδομένων του χρήστη η DRA προσέγγιση είναι πολύ καλή καθώς η τοπικοποίηση γίνεται ακριβώς μόνο στη πλευρά του client - δηλαδή του χρήστη - και έτσι κανείς άλλος δεν γνωρίζει την τοποθεσία του χρήστη.



Σχήμα 3.2: Αρχιτεκτονική της DRA.

3.4 Αξιολόγηση των τεχνικών

Όπως βλέπουμε και με τη περιγραφή των δύο προσεγγίσεων κατά τις οποίες επιτυχάνεται η τοπικοποίηση σε εσωτερικούς χώρους, οι δύο αλγόριθμοι υστερούν κατά πολύ σε βασικά ζητήματα, τα οποία ο χρήστης θα ήθελε να είχε επιλυμένα. Κάποια απ' αυτά είναι για παράδειγμα το ζήτημα προσωπικών δεδομένων, το οποίο δεν είναι καθόλου καλό στην προσέγγιση CRA. Σε αυτή την περίπτωση, όπως εξηγήθηκε και αναλυτικά στο υποκεφάλαιο 3.2, ο χρήστης δεν θα ήθελε σε καμιά περίπτωση ένας τρίτος, στην περίπτωση μας ο server, να ήξερε την ακριβή τοποθεσία του χρήστη. Παρόλο που ο αλγόριθμος CRA στα ζητήματα του χρόνου διαδικασίας της τοπικοποίησης και της κατανάλωσης ενέργειας είναι εξαιρετικός, στο ζήτημα των προσωπικών πληροφορίων υστερεί κατά πολύ. Αυτό έχει ως αποτέλεσμα την αναζήτηση για μια νέα προσέγγιση που θα υπερσκελίζει το ζήτημα της διαφύλαξης των προσωπικών δεδομένων.

Από την άλλη, ο DRA αλγόριθμος παρόλο που είναι εξαιρετικός στο ζήτημα της διαφύλαξης των προσωπικών δεδομένων του χρήστη υστερεί σε κάποια άλλα σοβαρά ζητήματα. Αυτά τα ζητήματα είναι ο χρόνος τοπικοποίησης αλλά και η κατανάλωση ενέργειας. Στο ζήτημα του χρόνου τοπικοποίησης ο χρήστης απαιτεί μια προσέγγιση κατά την οποία ακριβώς θα ήθελε να πλοηγείται στον χώρο εύκολα και γρήγορα. Ειδικά σε μεγάλους χώρους, όπως αεροδρόμιο κτλ, που το radio-map προφανώς θα είναι μεγάλο, ο DRA αλγόριθμος θα αδυνατεί να προσφέρει στον χρήστη γρήγορη τοπικοποίηση καθώς η προσπέλαση του (μεγάλου αρχείου) radio-map, κατά την τοπικοποίηση, απαιτεί πολύ χρόνο στην κινητή συσκευή η οποία έχει και χαμηλή υπολογίσιμη ισχύ. Επίσης, η προσπέλαση αυτού το μεγάλου αρχείου κάθε φορά σπαταλάει μεγάλη κατανάλωση ενέργειας. Αυτό όμως δεν το θέλουμε σε καμιά περίπτωση γνωρίζοντας ότι η εξοικονόμηση ενέργειας στην κινητή συσκευή είναι πολύ σημαντική.

Αυτά τα συμπεράσματα τα παρουσιάζουμε στην πράξη προσομοιώνοντας όλες τις προσεγγίσεις για να μπορέσουμε να δούμε τα θετικά και τα αρνητικά. Κατ' επέκταση

αξιολογούμε τους αλγόριθμους για να κατορθώσουμε στο τέλος να συγκρίνουμε τη νέα προσέγγιση που προτείνεται.

Κατά την διάρκεια υλοποίησης του νέου αλγόριθμου τοπικοποίησης ελήφθησαν υπόψιν τα χαρακτηριστικά που υστερεί ο κάθε αλγόριθμος τοπικοποίησης. Πιο συγκεκριμένα η εστίαση γίνεται στο πρόβλημα του CRA όσον αφορά την απόκρυψη των προσωπικών δεδομένων. Επίσης και στα δύο ζητήματα του DRA όσον αφορά το χρόνο τοπικοποίησης και τη μεγάλη κατανάλωση ενέργειας. Αυτά λήφθηκαν υπόψιν προς αποφυγήν, ενώ τα πλεονεκτήματα του κάθε αλγόριθμού επίσης ελήφθησαν υπόψιν προς εκμετάλευση.

Κεφάλαιο 4

Τεχνική γεωτοποθέτησης με φίλτρα Bloom

4.1 Εισαγωγικά

Σε αυτό το Κεφάλαιο θα παρουσιάσουμε το θεωρητικό μέρος του αλγορίθμου και το πώς ο συγκεκριμένος αλγόριθμος μπορεί επιτύχει όσον αφορά τους στόχους τους οποίους βάλαμε. Στόχος του BloomMap αλγόριθμου είναι από τη μια να πάρει όλα τα θετικά χαρακτηριστικά των δύο προσεγγίσεων που περιγράφησαν πριν (βλέπε υποκεφάλαια 3.1 και 3.2) αλλά, ταυτόχρονα, να αποφύγει τα αρνητικά των δύο προσεγγίσεων. Οι τομείς λοιπόν που λήφθηκαν σοβαρά υπόψιν είναι ο τομέας της απόκρυψης των προσωπικών δεδομένων, ο τομέας την κανάλωσης ενέργειας και τέλος ο τομέας του χρόνου της τοπικοποίησης.

Πριν όμως από την αναλυτική περιγραφή του BloomMap αλγορίθμου (BMA) θα παρουσιαστεί μια αρχική γνώση σχετικά με την απόκρυψη της τοποθεσίας του χρήστη και τα φίλτρα Bloom.

4.2 Φίλτρα Bloom

Τα φίλτρα Bloom προτάθηκαν από τον Burton Howard Bloom το 1970[3], με σκοπό την εξοικονόμηση του χώρου πάνω σε δομές δεδομένων, τα οποία χρησιμοποιούνται για να απαντήσουν αποδοτικά σε ένα σύνολο επερωτημάτων. Η ιδέα είναι η δέσμευση ενός διανύσματος από b bits, ορισμένα όλα εξ' αρχής με 0, και κάνοντας χρήστη h ανεξάρτητα hash functions κατακερματίζεται ένα στοιχείο σε h τοποθεσίες στο διάνυσμα με μια ομοιόμορφη τυχαία κατανομή.

Για να δημιουργήσει κάποιος ένα Bloom filter για ένα στοιχείο τροφοδοτείται το στοιχείο για κάθε ένα από τα h hash functions για να πάρουμε h θέσεις και να τις ορίσουμε με 1. Για να ελέγξουμε κατά πόσον ένα στοιχείο είναι μέλος του συνόλου πρέπει να συγκρίνουμε το διάνυσμα του επερωτήματος με το διάνυσμα του συνόλου. Αν μια μοναδική μη μηδενική θέση στο διάνυσμα επερωτήσεως δεν ταιριάζει με τη μη μηδενική θέση στο διάνυσμα του συνόλου, τότε το επερώτημα δεν υπάρχει στο σύνολο. Αν όλες οι μη-μηδενικές θέσεις ταιριάζουν τότε το στοιχείο πρέπει να είναι μέλος του συνόλου εφόσον το Bloom filter δεν εμποδίζει false positives αποτελέσματα.

Το πιο σημαντικό χαρακτηριστικό των Bloom filters είναι ότι υπάρχει ένα καθαρό "tradeoff" μεταξύ του b και της πιθανότητας για λανθασμένο θετικό αποτέλεσμα. Δίνοντας, h βέλτιστες hash functions, b bits του Bloom filter και M στοιχεία μπορούμε να υπολογίσουμε το ποσοστό των false positives, τα οποία παράγονται από το Bloom filter, όπως θα εξηγήσουμε στο επόμενο υποκεφάλαιο.

4.3 Αλγόριμος τοπικοποίησης BloomMap

Σε αυτό το υποκεφάλαιο θα παρουσιάσουμε την περιγραφή του BloomMap Αλγόριθμου (BMA), ο οποίος μειώνει τη κατανάλωση ενέργειας και το κόστος στο χρόνο τοπικοποίησης και εγγυάται την απόκρυψη της τοποθεσίας του χρήστη από τρίτους. Αντί την αποστολή του RSS διανύσματος από τον χρήστη στον server (όπως γίνεται με τις άλλες προσεγγίσεις τοπικοποίησης) ο χρήστης προωθεί ένα Bloom filter,

το οποίο δημιουργείται από ένα γειτονικό Access Point και από την αντίστοιχη RSS τιμή στον server. Ο server με τη σειρά του χρησιμοποιεί αυτό το Bloom filter για να βρει έτσι τον μικρό αριθμό (r << M) του πίνακα, ο οποίος θα βοηθήσει την συσκευή από πλευράς του χρήστη να εντοπίσει την τοποθεσία του.

Τα βασικά βήματα του BloomMap είναι τα πιο κάτω:

1. Ο χρήστης επιλέγει ένα μοναδικό Access Point - id από τον χώρο, τον οποίο βρίσκεται εκείνη την στιγμή και δημιοργείται ένα Bloom filter με ένα αριθμό επιλεγμένο από τον χρήστη k σύμφωνα με το "trade-off" μεταξύ της ισχύος (για παράδειγμα μέγεθος του συνόλου απάντησης) και τον βαθμό της απόκρυψης της προσωπικής πληροφορίας της τοποθεσίας από τρίτους, που επιθυμεί ο χρήστης.
2. Ο χρήστης προωθεί το Bloom filter στον server.
3. Ο server βρίσκει τα (k) AP-ids, τα οποία χρησιμοποιούνται για να δημιουργηθεί αυτό το Bloom filter και τις γραμμές του πίνακα από τις οποίες οι RSS τιμές δεν είναι μηδέν.
4. Ο server αποστέλλει το αποτέλεσμα στον client, το οποίο είναι κάποιες γραμμές από το radio-map για να μπορέσει έτσι ο χρήστης να προχωρήσει προς την τοπικοποίησης.

Υποθέτουμε ότι ο χρήστης γνωρίζει μία καλή προσέγγιση του αριθμού M από τα AP-ids στο σύστημα και τα h hash functions (αυτά μπορούν να ορισθούν από τον server την ώρα της αρχικοποίησης). Με αυτό τον τρόπο μπορούμε να είμαστε σίγουροι ότι υπάρχει ανωνυμία από την στιγμή που ο server δεν θα έχει την δυνατότητα να αναγνωρίσει ποια Access Points ο χρήστης πραγματικά αναγνωρίζει την προκειμένη φάση. Το false positive ration (*fpr*) δίνεται από την πιο κάτω εξίσωση:

$$fpr = (1-e^{-hM/b})^h,$$

που σημαίνει ότι το h, M και το k (ορισμένο από τον χρήστη) μπορούμε να αναγνωρίσουμε τον αριθμό των b bits για να χρησιμοποιηθούν στο Bloom filter.

Είναι σημαντικό να αναφέρουμε ότι μόνο ένα AP-id από τα Access Points στο εύρως που έχει την δυνατότητα να αναγνωρίσει η συσκευή του χρήστη, χρησιμοποιείται για

να δημιουργηθεί το Bloom filter. Αν πάνω από ένα Access Point χρησιμοποιηθούν τότε ο server θα είναι σε θέση να αναγνωρίσει τα false positives τα οποία δίνονται από το Bloom filter από την στιγμή που θα γνωρίζει τα τυχαία false positives που κατά πάσα πιθανότητα να μην υπερσκελιστούν. Έτσι θα είναι αδύνατο για τον χρήστη να ακούει από δύο Access Points.

Το σύνολο της απάντησης από τον server μπορεί να μειωθεί περεταίρω με ένα μικρό "trade-off" στο ποσοστό της απόκρυψης της τοποθεσίας του χρήστη. Μαζί με το Bloom filter ο χρήστης μπορεί να προωθήσει κάποιες RSS πληροφορίες στον server για να μπορέσει ο server να αποστείλει πίσω όσο πιο μικρό αριθμό γραμμών από το radio-map. Επίσης, η διαδικασία που συμβαίνει στο μέρος του server μπορεί να γίνει με πιο γρήγορο ρυθμό εφαρμόζοντας διάφορες τεχνικές όπως ένα index πάνω στα AP-ids στις γραμμές οι οποίες έχουν μη-μηδενικές τιμές RSS για το AP-id ή ακόμη και χαμηλό επίπεδο bit operators για να γίνει η ταύτιση ακόμη πιο γρήγορα.

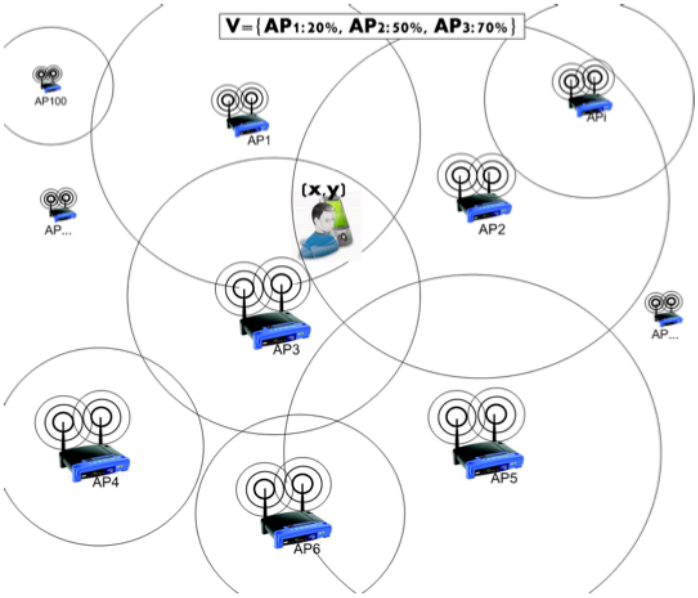
4.4 Παράδειγμα χρήσης του BMA

Όπως βλέπουμε στο Σχήμα 4.1, υποθέτουμε ότι ο χρήστης στην θέση (x,y) όπου τα σήματα εκπέμπουν από τρία διαφορετικά Access Points. Αυτά τα Access Points έχουν προσωπικούς κωδικούς AP1, AP2 και AP3 και τα σήματα τους είναι 20%, 50% και 70% αντίστοιχα. Ο συνόλικος αριθμός του M είναι 100 Access Points και έχουμε ορίσει από πριν ένα σύνολο από h=3 hash functions και το RSS διάνυσμα του χρήστη είναι V = {AP1 : 20%, AP2 : 50%, AP3 : 70%}, υποθέτουμε ότι ο χρήστης θέλει να χρησιμοποιήσει το επίπεδο ανωνυμίας για k = 3.

Ο χρήστης αρχικά υπολογίζει το μέγεθος που χρειάζεται το Bloom filter το οποίο διαβεβαιώνει ότι υπάρχουν k-1 false positives στην μεριά του server. Με k=3 και M=100, η false positive αναλογία του Bloom filter υπολογίζεται από τον τύπο fpr = k/M = 0.003. Κάνοντας χρήση την πιο κάτω positive ratio εξίσωση:

$$b = -hM \,/\, \ln\left(1-\sqrt[h]{fpr}\right)$$

ο χρήστης υπολογίζει το b ≈ 12.5 και ορίζει το b = 12 για να είναι σίγουρος για τουλάχιστον k-1 false positives. Έπειτα ο χρήστης επιλέγει ένα AP-id από RSS διάνυσμα (για παράδειγμα AP2) και το τροφοδοτεί στις, ορισμένες από πριν, hash-functions του Bloom filter για να δημιουργήσει τελικά το διάνυσμα με μέγεθος b=12 με κάποια bits να αλλάζουν σε 1, (για παράδειγμα {0,0,0,1,0,1,0,0,0,1,0,0}). Μετά αποστέλει το Bloom filter και την RSS τιμή του AP2 (RSS = 50) στον server.

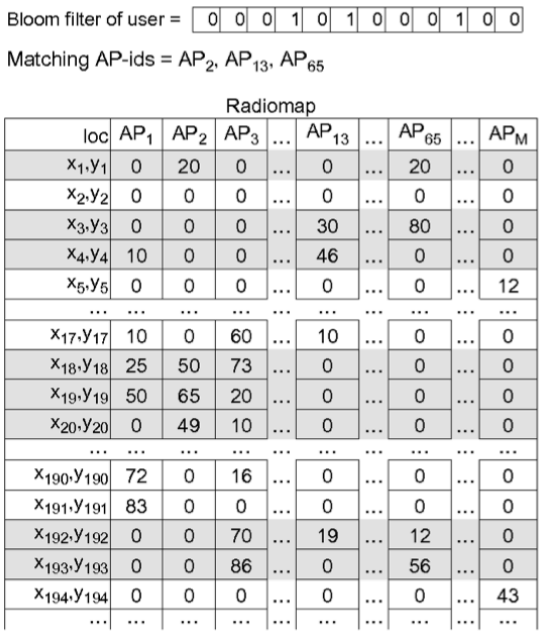


Σχήμα 4.1: Παράδειγμα ενός RSS διανύσματος ενός χρήστη

Στην μεριά του server (βλέπε σχήμα 4.2) το Bloom filter χρησιμοποιείται για να αναγνωρίσει τα AP-ids τα οποία θα του δώσουν την δυνατότητα να δημιουργηθεί το ίδιο Bloom filter και ανάλογα με το επιλεγμένο fpr να αντιστοιχηθεί με τουλάχιστο 3 AP-ids στον server. Ένα από αυτά τα AP-ids, είναι το πραγματικό AP-id το οποίο ο χρήστης έχει στο RSS διάνυσμα (βλέπε Σχήμα 4.1). Τα άλλα δύο Access Points, δηλαδή AP13 και AP65 είναι τα άκυρα στοιχεία τα οποία μας διασφαλίζουν την ανωνυμία και την απόκρυψη της τοποθεσίας του χρήστη αφήνοντας τον server να γνωρίζει μόνο την πιθανότητα του 1/k. Σημειώνουμε δε ότι η απόκρυψη της πληροφορίας της τοποθεσίας επιτυχάνεται καθώς ο server δεν έχει να αποφασίσει μεταξύ k=3 συγκεκριμένες τοποθεσίες αλλά μεταξύ k=3 διαφορετικά Access Points. Ακόμη και το γεγονός ότι ο

server κάνει χρήση την RSS τιμή η οποία δίνεται από τον χρήστη θα είναι μόνο σε θέση να περιορίσει αυτούς του χώρους αλλά όχι αλλά όχι να έχει εις γνώση του τις 3 συγκεκριμένες τοποθεσίες του χρήστη.

Ο server μετά αποστέλλει για κάθε Access Point που αντιστοίχησε τις γραμμές (του radio-map) που περιλαμβάνουν μη μηδενική RSS τιμή για το κάθε αντιστοιχησμένο Access Point. Οι γραμμές του radio-map επιστρέφονται τελικά στον χρήστη ο οποίος τώρα μπορεί να βρεί την θέση του στον χώρο με την χρήση των γνωστών αλγορίθμων που περιγράφησαν, σύντομα, και στο Κεφάλαιο 2, δηλαδή οι KNN, WKNN, MAP, RBF, SNAP κτλ. [7]



loc	AP_1	AP_2	AP_3	...	AP_{13}	...	AP_{65}	...	AP_M
x_1,y_1	0	20	0	...	0	...	20	...	0
x_2,y_2	0	0	0	...	0	...	0	...	0
x_3,y_3	0	0	0	...	30	...	80	...	0
x_4,y_4	10	0	0	...	46	...	0	...	0
x_5,y_5	0	0	0	...	0	...	0	...	12
...	...	...	...	...	...	...	...	...	...
x_{17},y_{17}	10	0	60	...	10	...	0	...	0
x_{18},y_{18}	25	50	73	...	0	...	0	...	0
x_{19},y_{19}	50	65	20	...	0	...	0	...	0
x_{20},y_{20}	0	49	10	...	0	...	0	...	0
...	...	...	...	...	...	...	...	...	...
x_{190},y_{190}	72	0	16	...	0	...	0	...	0
x_{191},y_{191}	83	0	0	...	0	...	0	...	0
x_{192},y_{192}	0	0	70	...	19	...	12	...	0
x_{193},y_{193}	0	0	86	...	0	...	56	...	0
x_{194},y_{194}	0	0	0	...	0	...	0	...	43
...	...	...	...	...	...	...	...	...	...

Σχήμα 4.2: Παράδειγμα αναγνώρισης των γραμμών του MATRIX με την χρήση του bloom filter στην μεριά του server.

4.5 Συζήτηση

Τελικώς, κάποια χαρακτηριστικά του νέου αλγορίθμου για τοπικοποίηση σε εσωτερικούς χώρους είναι τα ακόλουθα:

- *Κατανάλωση ενέργειας:* Πολύ καλός στο τομέα της κατανάλωσης ενέργειας. Αυτό συμβαίνει καθώς μετά την αποστολή του client του Bloom διανύσματος στον

server ο server αποστέλλει πίσω μόνο το αναγκαίο μέρος του radio-map που χρειάζεται ο χρήστης για να κάνει τοπικοποίηση. Έτσι πέρα από το γεγονός ότι στην πλευρά του client χρειάζεται προσπέλαση ενός μικρού αρχείο radio-map υπάρχει και πιο λίγος χρόνος λήψης του αρχείου που αυτό συνεπάγεται και πιο λίγη κατανάλωση ενέργειας από την συσκευή.

- *Χρόνος υπολογισμού της τοποθεσίας:* Όσον αφορά τον συνολικό χρόνο της διαδικασίας της τοπικοποίησης ο Bloom αλγόριθμος παρουσιάζεται να είναι σε ικανοποιητικά επίπεδα. Αυτό συμβαίνει γιατί κατά την διάρκεια της τοπικοποίησης έχει να προσπελάσει ένα σχετικά μικρό αρχείο αφού ο server τελικά αποστέλει ένα μέρος του radio-map. Επίσης, λόγο του ότι ο server έχει να αποστείλει ένα πιο μικρό σε μέγεθος radio-map αποστέλλεται σε αρκετά λίγο χρόνο.

- *Προστασία προσωπικών δεδομένων:* Όσον αφορά στην προστασία των προσωπικών δεδομένων ο Bloom αλγόριθμος είναι ο κατάλληλος. Αυτό συμβαίνει καθώς ο αλγόριθμος παρέχει μια απλή απόκρυψη και τελικά εξασφαλίζει ότι ο server μπορεί μόνο να αναγνωρίσει ένα μεγαλύτερο εύρως της τοποθεσίας του χρήστη με μια πιθανότητα λιγότερη από 1/k.

Όπως βλέπουμε πιο πάνω ο Bloom αλγόριθμος καταφέρνει να φέρει εις πέρας τους στόχους τους οποίους έχουμε βάλει. Δηλαδή την εκμετάλλευση των θετικών των προσεγγίσεων DRA και CRA και αποφυγή των αρνητικών τους.

Κεφάλαιο 5

Πειραματική Μεθοδολογία

Για την εκτέλεση της πειραματικής μεθοδολογίας υλοποιήσαμε έναν προσομοιωτή ο οποίος έχει την δυνατότητα να προσομειώνει τους τρείς αλγορίθμους και να μας παρουσιάσει εν τέλει τα αποτελέσματα για τον κάθε ένα απ' αυτούς. Μέσω του προσομοιωτή έχουμε την δυνατότητα επίσης να ορίσουμε όποιο data-set θέλουμε και αναλόγως να κάνουμε τις απαραίτητες συγκρίσεις για τα data-set. Στην παρούσα δουλειά κάναμε χρήση δύο data-set τα οποία ήταν αρκετά ικανοποιητικά για να προβάλουμε τις διαφορές των αλγορίθμων και να δείξουμε την υπεροχή του Bloom αλγόριθμου έναντι των υπολοίπων.

5.1 Πειραματική υποδομή

5.1.2 *Περιγραγή των data-sets*

Κάναμε χρήση των πιο κάτω αντιπροσωπευτικά datasets για την αξιολόγηση των αλγορίθμων τοπικοποίησης:

5.1.2.1 *UCY RadioMap*

Αυτό το radio-map σχεδιάστηκε από δεδομένα τα οποία συλλέχθησαν από ένα τυπικό κτήριο του τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου. Κάναμε χρήση τριών Android smartphones (HTC Hero, HTC Desire και Samsung Nexus S, βλέπε Σχέδιο 5.1) με τα οποία μαζέψαμε 30 RSS fingerprints (δηλαδή RSS τιμές από τα Access Points για κάθε συγκεκριμένη τοποθεσία) από 1500 μοναδικές τοποθεσίες και συνολικά 45000 fingerprints. Υπάρχουν δε, 120 WLAN Access Points εγκατεστημένα στους τέσσερις ορόφους αυτού του κτηρίου συμπεριλαμβανομένων και των Access Points τα οποία βρίσκονται σε γειτονικά κτήρια, τα οποία μπορούν να ανιχνευθούν από διαφορετικά μέρη του κτηρίου του τμήματος Πληροφορικής. Κατά μέσον όρο, 10.6 Access Points ανιχνεύθησαν ανά τοποθεσία στον κτήριο. Συλλέξαμε τα δεδομένα περπατώντας στο κτήριο το οποίο αποτελείτο από 2900 τοποθεσίες.

5.1.2.2 *ΚΙΟΣ Radiomap*

Μαζέψαμε τα RSS δεδομένα μέσα σε τυπικό περιβάλλον γραφείου στο ΚΙΟΣ το οποίο είναι κέντρο ερευνών του Πανεπιστημίου Κύπρου. Ο συνολικός χώρος είναι περίπου 560 τετραγωνικά μέτρα και ο όροφος αποτελείται από αρκετά ανοιχτά (στον όροφο) γραφεία και προσωπικά γραφεία, εργαστήρια, δωμάτιο συνεδριάσεων και διαδρόμους, Υπάρχουν εγκατεστημένα 9 Access Points τα οποία χρησιμοποιούν το IEEE 802.11b/g και παρέχουν πλήρης κάλυψη σε όλο τον όροφο. Επίσης, υπάρχουν μερικά γειτονικά Access Points τα οποία μπορούν να αναγνωριστούν από διαφορετικά τμήματα του ορόφου. Κάναμε χρήση τριών διαφορετικών συσκευών Android (HTC Desire, HTC Flyer και Samsung Nexus S, βλέπε Σχέδιο 5.1) για να συλλέξουμε RSS δεδομένα από τα διαθέσιμα Access Points για 105 μοναδικές τοποθεσίες στον όροφο. Αυτές οι τοποθεσίες, στον χώρο, ήταν χωριστές 2-3 μέτρα μεταξύ τους. Συλλέγησαν 2100 training fingerprints που αντιστοιχούν 20 fingerprints για κάθε τοποθεσία στον όροφο. Για δοκιμαστικούς λόγους συλλέγησαν επίσης 960 fingerprints για 96 τοποθεσίες (10 fingerprits ανά τοποθεσία) μέσα στον χώρο μας.

Σχήμα 5.1: HTC Flyer, HTC Desire, HTC Hero, Samsung Nexus S (από αριστερά προς τα δεξιά)

5. 2 Πειραματική διαδικασία

Σε αυτή την παράγραφο θα παρουσιάσουμε τις ακριβής λειτουργίες του προσομοιωτή καθώς και τα δεδομένα που χρησιμοποιήθηκαν για την εκτέλεση των πειραμάτων.

5.2.1 *Προσομοιωτής*

Ο προσομοιωτής υλοποιήθηκε στην γλώσσα προγραμματισμού Java. Επιλέξαμε την γλώσσα προγραμματισμού Java για την υλοποίηση του καθώς η συγκεκριμένη γλώσσα μας προσφέρει τεράστιες δυνατότητες όσον αφορά στους υπολογισμούς και τις έτοιμες κλάσεις τις οποίες προσφέρει. Επίσης, μας έχει βοηθήσει ιδιαίτερα όσον αφορά την υλοποίηση του Bloom αλγόριθμου για το οποίο, η Java, είχε την δυνατότητα να μας προσφέρει κάποιες έτοιμες hash functions και έτσι να μπορέσουμε να προχωρήσουμε με πιο πολλή ευκολία της υλοποίηση μας.

Ο προσομοιωτής υλοποιούσε ουσιαστικά προτόκολλο client - server. Θεωρήσαμε πως ο client είναι ο χρήστης ο οποίος προβένει στην τοπικοποίηση και ο server είναι ο εξυπηρετητής ο οποίος βοηθά έναν ή πολλούς χρήστες προς αυτή την κατεύθυνση. Από την πλευρά του ο client διαπερνούσε ένα αρχείο, το οποίο αποτελείτο από ένα μεγάλο εύρως δεδομένων RSS διανυσμάτων τα οποία τα συλλέξαμε πιο πριν, και έκανε τις λειτουργίες του ανάλογα με την προσέγγιση που θέλαμε να τρέξουμε στον προσομοιωτή. Για παράδειγμα αν θέλαμε να δοκιμάσουμε τον αλγόριθμο CRA, θα

έστελνε κάθε γραμμή του data-set στον server και ο server με τη σειρά του θα διαπερνούσε το radio-map (που θα έχει ήδη αποθηκευμένο στον server) και θα υπολόγιζε και να απέστελνε πίσω την τοποθεσία του. Σημειώνουμε δε ότι αυτή την διαδικασία θα την εκτελούσε για κάθε γραμμή του αρχείου που υπήρχε στην πλευρά του client θεωρόντας έτσι ότι ο χρήστης πλοηγείται στον εσωτερικό χώρο και κάθε φορά αναγνωρίζει διαφορετικά RSS δεδομένα.

Επίσης, ο προσομοιωτής, μετά την εκτέλεση του αντίστοιχου αλγόριθμου τοπικοποίησης, δημιουργεί ένα Excel αρχείο το οποίο παρουσιάζει για κάθε μια αίτηση του client για τοπικοποίηση τον χρόνο, τα μηνύματα συνδιαλλαγής κτλ (βλέπε Σχήμα 5.2). Αυτό το θέλουμε για να παρατηρήσουμε με πιο πολλή λεπτομέρια την συμπεριφορά του συστήματος μας αναλόγως του κάθε αιτήματος.

Τέλος, ο προσομοιωτής εμφανίζει στον χρήστη (του προσομοιωτή) τον χρόνο που θέλει να γίνει όλη η διαδικασία και την αποστολή και λήψη των συνολικών μηνυμάτων. Επίσης, η κατανάλωση ενέργειας για κάθε αλγόριθμο υπολογιζόταν εκτός του προσομοιωτή με συγκεκριμένη διαδικασία που θα την παρουσιάσουμε πιο κάτω αναλυτικότερα.

Bytes Client Send	Bytes Server	Client CPU time	Server CPU time	Msgs Sent	Msgs Rsv
1174.0	10.0	N/A	0.035	1	1
1174.0	10.0	N/A	0.0090	1	1
1174.0	10.0	N/A	0.0040	1	1
1174.0	10.0	N/A	0.0060	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0090	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0030	1	1

Σχήμα 5.2: Αναλυτικά αποτελέσματα του προσομοιωτή για κάθε αίτημα του client για τοπικοποίηση για την CRA προσέγγιση.

Μετά από το τρέξιμο του κάθε αλγόριθμου και την εξαγωγή του αντίστοιχου Excel αρχείου πέρναμε τα στοιχεία που μας παρουσίαζε: Bytes Client Send, Bytes Server Send, Client CPU Time, Server CPU Time, Msg Send και Msg Received. Μετά από αυτή την εξαγωγή, των πιο πάνω στοιχείων, τα εφαρμόζαμε σε ειδική φόρμουλα η οποία μας παρουσιάζε, εν τέλει, τον χρόνο και την κατανάλωση ενέργειας για μια απόπειρα του χρήστη να τοπικοποιηθεί.

5.2.2 *Δικτυακό και ενεργειακό μοντέλο*

Στις ρυθμίσεις του προσομοιωτή μας, δηλαδή στο αρχείο παραμέτρων, ορίσαμε ένα απλό CRA REQ μήνυμα στα 4 bytes, ένα DATA μήνυμα (δηλαδή μια γραμμή του radiomap) που αποτελείται από μια MAC διεύθυνση του αντίστοιχου Access Point που αποτελείτο από 17 bytes και 6-bytes ορίσαμε την RSS τιμή του Access Point. Επίσης, ορίσαμε μια μεταβλητή CRA response η όποια περιλάμβανε την τοποθεσία του χρήστη στα 10 bytes. Η επικοινωνία μεταξύ του smartphone και του server γίνεται κάτω από 802.11b δίκτυο το οποίο έχει TCP downlink στα 1022kbps και ένα TCP uplink στα 123 kbps, 237ms TCP χρόνος καθυστέρησης γι handshake και χρόνος handshake της εφαρμογής στα 493ms.

Κεφάλαιο 6

Πειραματικά Αποτελέσματα

Σε αυτό το κεφάλαιο θα γίνει ακριβώς η παρουσιάση των πειραματικών αποτελεσμάτων και βάσει αυτών, η αξιολόγηση της κάθε προσέγγισης βάσει της πειραματικής διαδικασίας που ορίστικέ στο προηγούμενο κεφάλαιο. Αρχικά θα δώσουμε μια γενική ιδέα των αποτεσμάτων ενώ μετά θα προχωρήσουμε σε μια πιο αναλυτική παρουσίαση των αποτελεσμάτων συγκρίνοντας τους τρείς αλγορίθμους σε τρία βασικά χαρακτηριστικά: Χρόνος διαδικασίας, κατανάλωση ενέργειας και τέλος αριθμός ανταλλαγής μηνυμάτων. Τέλος, θα γίνει προσπάθεια επικύρωσης της θεωρητικής ανάλυσης που προηγήθηκε.

6.1 Εισαγωγή

Η ανάλυση θα γίνει σε τρία επίπεδα. Το πρώτο αφορά τον χρόνο της ολικής διαδικασίας τοπικοποίησης του χρήστη (σ' αυτό συμπεριλαμβάνονται ο χρόνος τοπικοποίησης αλλά και ο χρόνος αποστολής και λήψης πληροφοριών από και προς τον server), κατανάλωση ενέργειας του client και τέλος ο αριθμός ανταλλαγής μηνυμάτων μεταξύ των δύο πλευρών. Βέβαια, θα αξιολογήσουμε και το γεγονός της απόκρυψης της τοποθεσίας του χρήστη από τον server. Αυτό εξάλλου ήταν και ένα μεγάλο ζήτημα στην

δημιουργία μιας νέας προσέγγισης για τοπικοποίησης σε εσωτερικούς χώρους. Η πιο πάνω ανάλυση έγινε για κάθε ένα αλγόριθμο που εκτελέστηκε στην προσομειώτη.

Γενικά πάντως, βάσει των πειραματικών αποτελεσμάτων που θα δούμε και πιο κάτω, παρατηρούμε ακριβώς ότι περιμέναμε. Δηλαδή, στο ζήτημα του χρόνου εκτέλεσης της διαδικασίας έχουμε βέλτιστο στην προσέγγιση του CRA, αφού την τοποθεσία του την υπολογίζει ο server, έπειτα ακολουθεί η δική μας προσέγγιση η BMA και τέλος με πολύ μεγαλύτερη διαφορά των δύο πρώτων ο DRA. Όσον αφορά το ζήτημα της κατανάλωσης ενέργειας η CRA προσέγγιση δεν έχει σχεδόν καμιά κατανάλωση ενέργειας, η προσέγγιση DRA αρκετή λόγο του μεγάλου αρχείου radio-map που έχει να διαπεράσει για τοπικοποίηση και ο BMA παρουσιάζεται να έχει αρκετή μειωμένη την κατανάλωση ενέργειας λόγο του πιο μικρού radio-map που έχει να διαχειρηστεί. Τέλος όσον αφορά την ανταλλαγή μηνυμάτων παρατηρούνται ακριβώς οι ίδιες αναλογίες.

6.2 Παρουσίαση και ανάλυση αποτελεσμάτων

Στο παρόν υποκεφάλαιο θα γίνει η παρουσίαση και ανάλυση των αποτελεσμάτων, όπως αυτά εξάγαμε από τον προσομείωτη, για κάθε ένα dataset ξεχωριστά που είχαμε ως είσοδο. Για κάθε ένα dataset θα δείξουμε τρείς γραφικές παραστάσεις οι οποίες θα παρουσιάζουν τα χαρακτηριστικά, που αναφέραμε στο υποκεφάλαιο 6.1, για κάθε έναν αλγόριθμο τοπικοποίσης που χρησιμοποιήθηκε. Οι τρείς γραφικές παραστάσεις που θα παρουσιάσουμε είναι οι ακόλουθες:

- ***Μέσος χρόνος εκτέλεσης της διαδικασίας:*** Η εν λόγω γραφική παράσταση παρουσιάζει τον μέσο χρόνο σε δευτερόλεπτα που μπορεί να εκτελέσει κάθε αλγόριθμος τοπικοποίησης για μια μοναδική τοπικοποίηση του χρήστη. Σημειώνουμε δε ότι εννοώντας τον μέσο χρόνο εκτέλεσης της διαδικασίας εννοούμε την διαδικασία πρώτης επικοινωνίας του client με τον server μέχρι την ώρα που ο client θα έχει τις απαραίτητες πληροφορίες από τον server για να

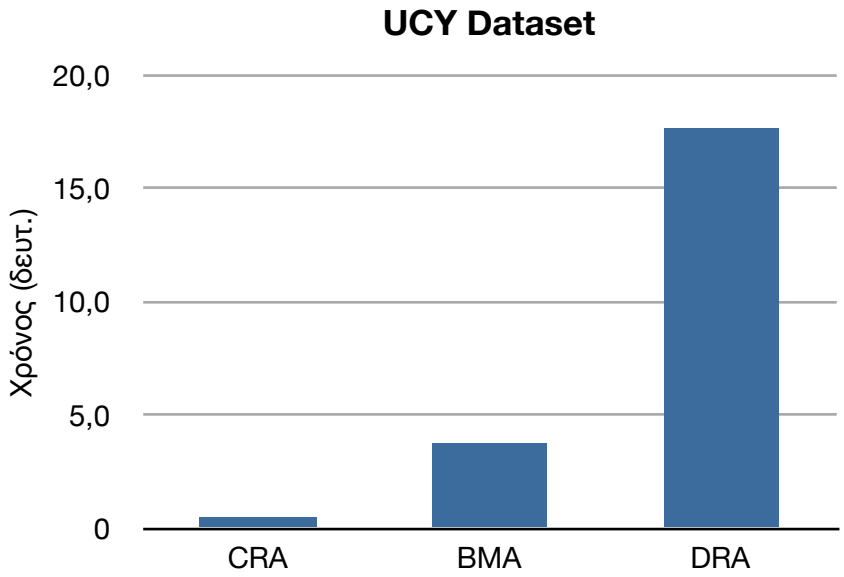
μπορέι να προχωρήσει προς την τοπικοποίηση.

- ***Μέση ενέργεια κατανάλωσης:*** Στην συγκεκριμένη γραφική παράσταση παρουσιάζουμε την μέση κατανάλωση ενέργειας για κάθε ένα αλγόριθμο τοπικοποίησης σε mW.

- ***Μέσος αριθμός ανταλλαγής μηνυμάτων:*** Σε αυτή την γραφική παράσταση παρουσιάζουμε τον μέσο αριθμό ανταλλαγής μηνυμάτων για κάθε ένα αλγόριθμο τοπικοποίησης.

Να αναφέρουμε σε αυτή την φάση ότι στο Παράρτημα Α βρίσκονται όλα τα αποτελέσματα για κάθε φορά που προσομοιώνεται το αίτημα του client στον server. Σε αυτό το Παράρτημα παρουσιάζουμε τα αποτελέσματα για τους τρεις αλγορίθμους τοπικοποίησης που συγκρίναμε. Η παρουσίαση γίνεται σε μορφή πίνακα, έτσι όπως ακριβώς μας τα πάραξε ο προσομοιωτής. Στις γραφικές παραστάσεις δε, παρουσιάζεται ο τροποποιημένες τιμές βάσει των μετρήσεων ενώ στο Παράρτημα Α παρουσιάζουμε δείγμα (λόγο του μεγάλου μήκους των μετρήσεων) για 100 απόπειρες τοπικοποίησης από τον χρήστη.

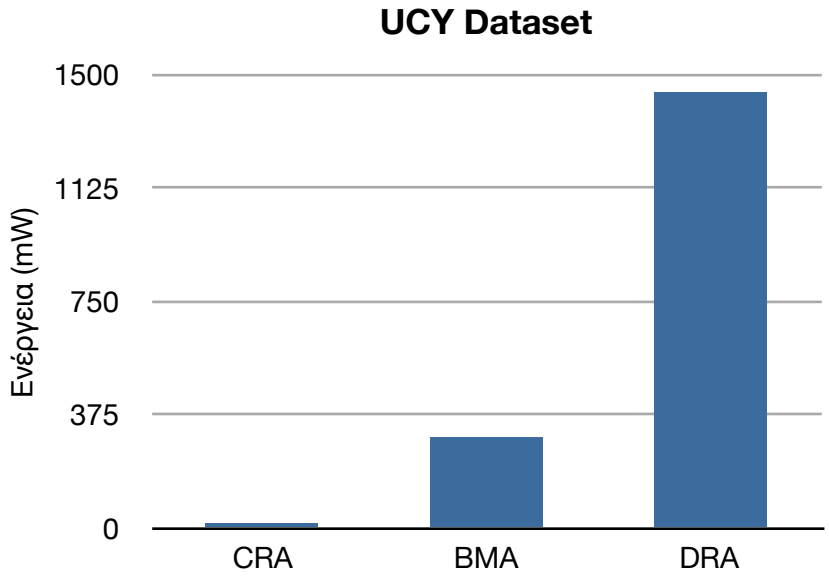
6.2.1 UCY Dataset



Σχήμα 6.1: Μέσος χρόνος για κάθε μια αλγοριθμική προσέγγιση τοπικοποίησης για το UCY Dataset

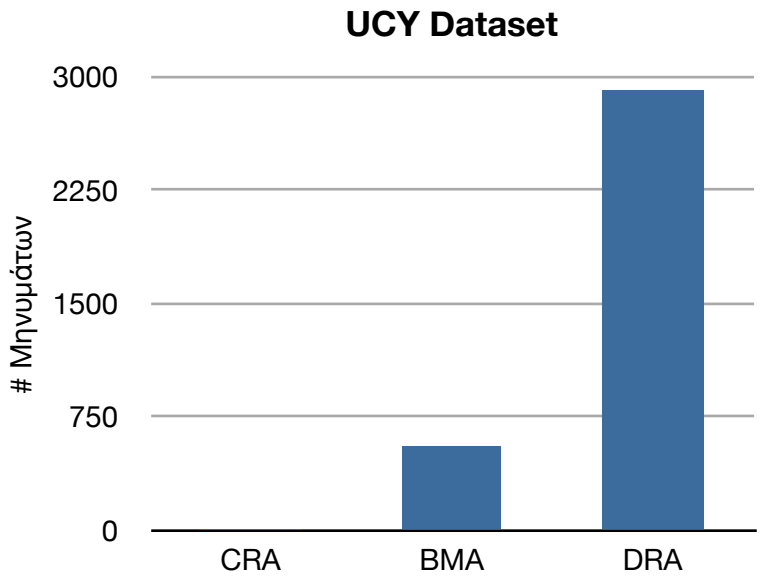
Σύμφωνα με την πιο πάνω γραφική παράσταση παρατηρούμε ξεκάθαρα ότι ο χρόνος για το CRA είναι αμελητέος. Για τον DRA έχουμε αρκετά ψηλό χρόνο και για τον BMA αρκετά μειωμένο χρόνο. Με την νέα προσέγγιση καταφέραμε να κρατήσουμε την απόκρυψη της τοποθεσία του χρήστη (όπως ακριβώς κάνει ο DRA) και μειώσαμε τον χρόνο κατά πολύ.

Πιο συγκεκριμένα για την προσέγγιση CRA έχουμε μέσο χρόνο 0,4646 δευτερόλεπτα, για τον BMA: 3,7262 δευτερόλεπτα και για τον DRA 17,6497 δευτερόλεπτα.



Σχήμα 6.2: Μέση κατανάλωση ενέργειας για κάθε μια αλγοριθμική προσέγγιση τοπικοποίησης για το UCY Dataset

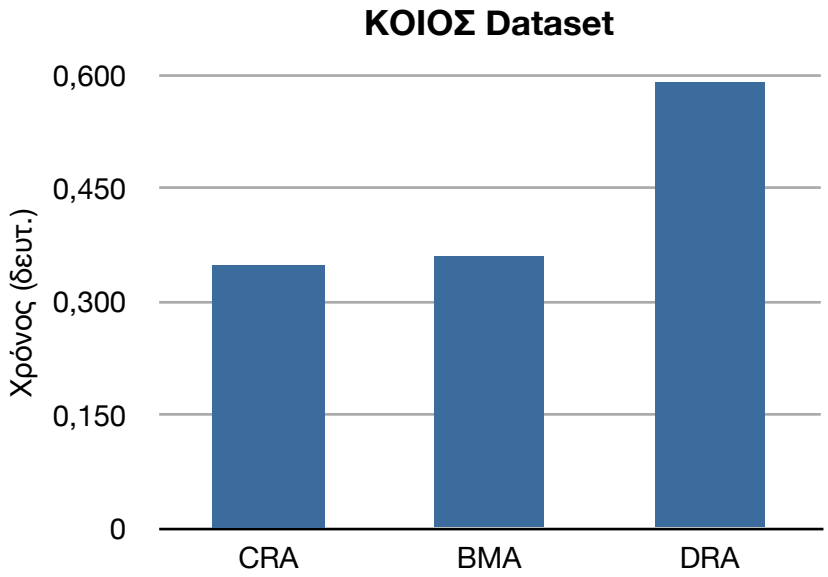
Στην γραφική παράσταση (Σχήμα 6.2) παρουσιάζουμε την μέση κατανάλωση ενέργειας για κάθε μια προσέγγιση για το UCY Dataset. Παρατηρούμε το ίδιο φαινόμενο, δηλαδή ότι ο CRA έχει αμελητέα κατανάλωση ενέργειας, ο BMA έχει 300,61 mW και τέλος ο DRA 1442,96 mW. Στην περίπτωση του νέου αλγόριθμου που προτείνεται, δηλαδή του BMA, παρατηρούμε ότι ενώ διατηρεί την απόκρυψη της τοποθεσίας του χρήστη από τον server εν τούτοις παρουσιάζει αρκετά μειωμένη κατανάλωση ενέργειας σε σύκριση με τον DRA του οποίου η κατανάλωση ενέργειας είναι υπερδιπλάσια.



Σχήμα 6.3: Μέσος αριθμός ανταλλαγής μηνυμάτων για κάθε μια αλγοριθμική προσέγγιση τοπικοποίησης για το UCY Dataset

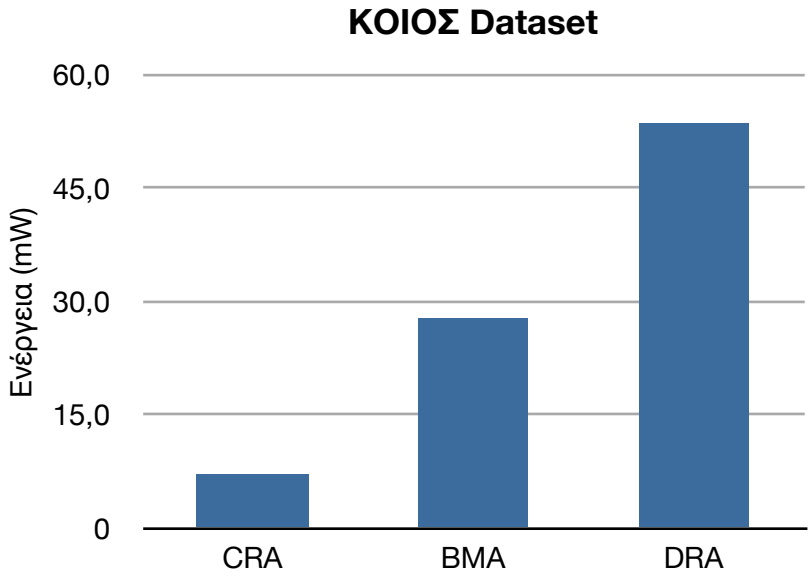
Στην γραφική παράσταση (Σχήμα 6.3) παρακολουθούμε αρκετά μεγάλη μείωση των αριθμό των μηνυμάτων του BMA σε σύγκριση με τον DRA. Αντιθέτως παρατηρούμε ότι ο CRA παρουσιάζει μόλις 2 μηνύματα ανταλλαγής μεταξύ client και server μέχρι την τελική τοπικοποίηση του χρήστη. Αυτά τα αποτελέσματα είναι απολύτως φυσιολογικά καθώς ο CRA δεν χρειάζεται να ανταλλάξει άλλα μηνύματα, λόγω του τρόπου τοπικοποίησης (τοπικοποίησης καθαρά από την μεριά του server), ενώ στον BMA και DRA το radio-map αποστέλλεται από τον server στον client μόνο που για τον πρώτο αποστέλλεται ένα μέρος του.

6.2.2 ΚΟΙΟΣ Dataset



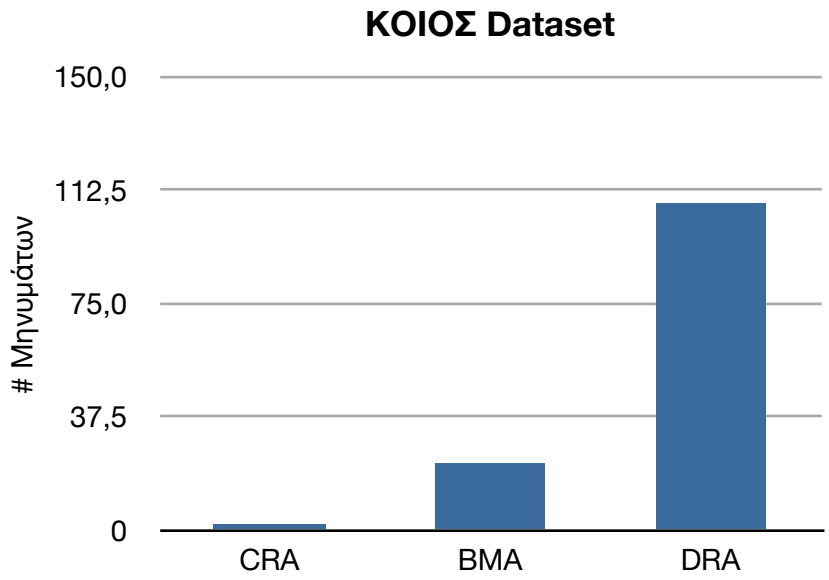
Σχήμα 6.4: Μέσος χρόνος για κάθε μια αλγοριθμική προσέγγιση τοπικοποίησης για το ΚΟΙΟΣ Dataset

Στη γραφική παράσταση (Σχήμα 6.4) παρουσιάζουμε τον συνολικό χρόνο που χρειάζεται ο χρήστης για να εξασφαλίσει την τοποθεσία του στον χώρο. Παρατηρούμε τις ίδιες αναλογίες, περίπου όπως φαίνονται ξεκάθαρα στο Σχήμα 6.1 τις οποίες και περιγράψα πιο πάνω. Η διαφορά σε αυτή την περίπτωση είναι ότι ο CRA παρουσιάζει πιο μεγάλο ποσοστο χρόνου για αυτό το dataset. Αυτό πολύ πιθανόν να συμβαίνει για τον λόγο ότι επειδή το dataset αυτό είναι φανερά πιο μικρής έκτασης και μειώνεται ο χρόνος τοπικοποίησης για τους αλγορίθμους BMA και DRA. Αντιθέτως για τον CRA ο χρόνος μένει ο ίδιος περίπου και έτσι αναλογικά αυξάνεται στην περίπτωση του ΚΟΙΟΣ dataset. Το ΚΟΙΟΣ dataset είναι ένα dataset το οποίο αποτελείται από ένα τυπικό εσωτερικό χώρο. Αυτό έχει σημασία στις παρατηρήσεις μας καθώς όπως βλέπουμε λόγο του ότι έχουμε να κάνουμε με σχετικά μικρό dataset (σε σύγκριση με το UCY) ο διαφορά του χρόνου της διαδικασίας τοπικοποίησης για τον CRA με τον BMA είναι μιδαμηνή.



Σχήμα 6.5: Μέση κατανάλωση ενέργειας για κάθε μια αλγοριθμική προσέγγιση τοπικοποίησης για το ΚΟΙΟΣ Dataset

Στην γραφική παράσταση (Σχήμα 6.5) φαίνεται ξεκάθαρα ότι φαίνεται ακριβώς και στο Σχήμα 6.2 του οποίου δώσαμε την περιγραφή πιο πάνω. Φαίνεται δηλαδή η υπεροχή που έχει ο CRA, λόγω της υλοποίησης του, όσον αφορά την κατανάλωση ενέργειας. Επίσης, ακολουθεί ο BMA του οποίου η κατανάλωση του φθάνει στα 22mW ενώ τελευταίος έρχεται ο DRA ο οποίος ανέρχεται στα 105mW. Για ακόμη μια φορά φαίνεται η υπεροχή του BMA ο οποίος, σε αντίθεση με τον CRA, εξασφαλίζει ασφάλεια της τοποθεσίας του χρήστη αλλά το εξασφαλίζει και με λίγο σχετικά κόστος.



Σχήμα 6.6: Μέσος αριθμός ανταλλαγής μηνυμάτων για κάθε μια αλγοριθμική προσέγγιση τοπικοποίησης για το ΚΟΙΟΣ Dataset

Στην γραφική παράσταση (Σχήμα 6.6) φαίνεται ότι ακριβώς φαίνεται και στο Σχήμα 6.3 δηλαδή ο BMA χρειάζεται αρκετά πιο λίγα μηνύματα, για τοπικοποίηση, από τον DRA ενώ αντιθέτως ο CRA να χρειάζεται μόνο μόλις 2 μηνύματα λόγω της υλοποίησής του. Δηλαδή, με λίγα λόγια η αναλογία του μέσου αριθμού μηνυμάτων τόσο για το ΚΟΙΟΣ dataset όσο και για το UCY dataset είναι στα ίδια επίπεδα δίνοντας μας έτσι τελικά τα αποτελέσματα που αναμέναμε.

Συνοψίζοντας, στις πιο πάνω γραφικές παραστάσεις (Σχήματα από 6.1 μέχρι 6.6) παρουσιάσαμε τα αποτελέσματα από τις τρεις προσεγγίσεις από την όψη της κατανάλωσης της ενέργειας, του χρόνου και του αριθμού των μηνυμάτων για τοπικοποίηση χρησιμοποιώντας δύο datasets, ΚΙΟΣ και UCY. Για το UCY dataset ο BloomMap αλγόριθμος βελτιώνει την επίδοση της DRA προσέγγισης (η οποία χρησιμοποιείται στο Airplace (βλέπε Κεφάλαιο 2)) κατά 80% στο θέμα του χρόνου, 83% στο θέμα της κατανάλωσης ενέργειας και κάνει πιο λίγη χρήση το WiFi του κατά 80%. Παρομοίως στην περίπτωση του ΚΟΙΟΣ dataset ο BloomMap αλγόριθμος παρέχει 60% πιο λίγη καθυστέρηση στο ζήτημα του χρόνου και 60% πιο λίγη κατανάλωση ενέργειας. Επίσης, χρησιμοποιεί λιγότερο το δίκτυο κατά 80%. Η CRA προσέγγιση υπερτερεί και στα δύο datasets έναντι του BloomMap αλγόριθμου σε όλους του τομείς.

Παρόλα ταύτα η BloomMap προσέγγιση εγγυάται την τοπικοποίηση χωρίς να δείχνει σε τρίτους την πραγματική θέση του χρήστη.

Κεφάλαιο 7

Συμπεράσματα και Μελλοντικές Επεκτάσεις

7.1 Συμπεράσματα

Σύμφωνα με την μελέτη λοιπόν που έγινε τόσο σε θεωρητικό βαθμό (βλέπε Κεφάλαιο . 4) όσο και σε πειραματικό επίπεδο (βλέπε Κεφάλαιο 6) ο νέος αλγόριθμος για τοπικοποίηση, δηλαδή ο BloomMap αλγόριθμος, που προτάθηκε αποδεικνύεται ότι μπορεί να είναι ο ιδανικότερος για αυτό τον τομέα. Αυτό καθώς ο συγκεκριμένος αλγόριθμος εκμεταλλεύεται τη φιλοσοφία του CRA αλγορίθμου, που είναι η χρήση του server για κάποιον υπολογισμό, και επίσης εκμεταλλεύεται την φιλοσοφία του DRA, που η τοπικοποίηση γίνεται πάνω στην ίδια τη συσκευή του χρήστη, αποκρύπτοντας έτσι την θέση του χρήστη από τον server.

Χρησιμοποιώντας λοιπόν την έξυπνη μεθοδολία του bloom-map καταφέραμε να "εξαναγκάσουμε" τον server να εντοπίζει κάθε φορά που αιτείται ο χρήστης τα Access Points, τα οποία ενδιαφέρεται και έτσι να του αποστέλλει πίσω το μέρος του radio-map που χρειάζεται για τοπικοποίηση. Έτσι δεν είναι αναγκαίο πλέον να εφαρμόζεται η αχρείαστη λογική προσπέλασης όλου του radio-map κάθε φορά που ο χρήστης θέλει να μάθει που βρίσκεται στο κτήριο αλλά εφαρμόζεται η λογική προσπέλασης του μέρους του αρχείου που χρειάζεται για την τοπικοποίηση.

Επίσης, τα πειραματικά αποτελέσματα, με τη χρήση προσομοιωτή έδειξαν θετικά και αναμενώμενα αποτελέσματα στη νέα προσέγγιση για τοπικοποίηση σε εσωτερικούς χώρους και ότι και στην πράξη η εφαρμογή του αλγόριθμου αυτού, δηλαδή στην πραγματική κινητή συσκευή, μπορεί να εξελίξει μια εφαρμογή αποδοτική και ευχάριστη για τον χρήστη, απαλλάσοντας τον από το θέμα της ασφάλειας (που δεν παρέχει ο CRA) και το θέμα του χρόνου τοπικοποίησης (πρόβλημα που έχει η προσέγγιση DRA).

Τελειώνοντας, σημαντικό είναι να αναφέρουμε ότι στο dataset (ΚΟΙΟΣ) - το οποίο αντικατοπτρίζει έναν τυπικό εσωτερικό χώρο και όχι ένα μεγάλο κρήριο όπως συλλέξαμε για το τμήμα Πληροφορικής (UCY dataset) - παρουσιάζεται ένας πολύ καλός χρόνο για τον BloomMap αλγόριθμο που φτάνει στα επίπεδα της CRA προσέγγισης. Αυτό μας δίνει το τελικό συμπέρασμα ότι η BloomMap προσέγγιση αγγίζει τα όρια της ιδάνικης προσέγγισης για το θέμα της εσωτερικής τοπικοποίησης.

7.2 Μελλοντικές Επεκτάσεις

Αν και η λογική του BloomMap αλγόριθμου φαίνεται να αποδίδει και να πετυχαίνει τους στόχους που έχουμε βάλει εξ αρχής, θεωρώ ότι χρειάζεται κι άλλη αξιολόγηση. Τα δύο dataset, τα οποία δοκιμάσαμε κατά την πειραματική διαδικασία αν και ήταν σημαντικά τα αποτελέσματα τους, δεν είναι αρκετά. Θα πρέπει η προσέγγιση μας να δοκιμαστεί και με άλλα dataset για να μπορέσουμε να είμαστε σίγουροι ότι πάντα αυτή η μεθοδολογία θα δουλεύει σωστά και θα είναι πάντα αποτελεσματική.

Επίσης, ο αλγόριθμος θα πρέπει να υλοποιηθεί κανονικά και σε κινητή συσκευή και να δοκιμαστεί στην πράξη, όπως ακριβώς είδαμε και με το Airplace (Κεφάλαιο 2) (όπου η DRA προσέγγιση), για να μπορέσουμε να το δούμε στην πράξη και έτσι να αξιολογηθεί πλέον εμπειρικά. Αυτό θα γίνει με μια μικρή μετατροπή του Airplace όπου αντί να χρησιμοποιείται η παρόμοια μέθοδος του DRA θα χρησιμοποιείται η μέθοδος του BMA όπως είναι ακριβώς υλοποιημένη στον προσομοιωτή.

Η υλοποίηση έχει γίνει προς το τέλος της συγκεκριμένης ΑΔΕ και έχουμε δουλέψει και εμπειρικά τον Bloom αλγόριθμο σε πραγματικό χρόνο. Παρόλα ταύτα επειδή η υλοποίηση βρίσκεται σε πειραματικό στάδιο θα πρέπει να τελειοποιηθεί και να λειτουργεί κανονικά με συγκεκριμένες προσθήκες τόσο στο server όσο και στην κινητή συσκευή. Το γεγονός πάντως ότι υλοποιήθηκε η BMA τεχνική είναι ακόμη μια απόδειξη ότι η τεχνική μπορεί να λειτουργήσεις κανονικά και να έχει τα θετικά τα οποία παρουσιάσαμε τόσο στην πειραματική διαδικασία όσο στα συμπεράσματα.

Βιβλιογραφία

[1] Airplace: http://www2.ucy.ac.cy/~laoudias/pages/platform.html

[2] P. Bahl and V. Padmanabhan, "RADAR: an in-building RF-based user location and tracking system," in IEEE INFOCOM, 2000, pp. 775-784.

[3]B. Bloom. Space/time trade-offs in hash coding with allowable errors. Communications of ACM, 13(7), 1970.

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

[5] Hashmap: http://docs.oracle.com/javase/1.4.2/docs/api/java/util/HashMap.html

[6] IPIN: http://ipin2011.dsi.uminho.pt/

[7] A. Konstantinidis, G. Chatzimilioudis, C. Laoudias, S. Nicolaou, D. Zeinalipour - Yazti, "Towards In-Situ Localization on Smartphones with a Partial Radiomap" ACM International Workshop on Hot Topics in Planet -Scale Measurement, 2012.

[8] C. Laoudias, G. Constantinou, M. Constantinides, S. Nicolaou, D. Zeinalipour-Yazti, C. G. Panayiotou, "A Platform for the Evaluation of Fingerprint Positioning Algorithms on Android Smartphones", International Conference on Indoor Positioning and Indoor Navigation (IPIN), 2011.

[9] C. Laoudias, G. Constantinou, M. Constantinides, S. Nicolaou, D. Zeinalipour-Yazti, C. G. Panayiotou, "The Airplace Indoor Positioning Platform for Android Smartphones", Demo at the 13th IEEE International Conference on Mobile Data Management (MDM'12)

[10] C. Laoudias, G. Constantinou, M. Constantinides, S. Nicolaou, D. Zeinalipour-Yazti and C. G. Panayiotou, "Demo: The Airplace Indoor Positioning Platform", Demo at the 10th ACM International Conference on Mobile Systems, Applications and Services (MobiSys'12)

[11] C. Laoudias, M. Michaelides, and C. Panayiotou, "Fault tolerant fingerprint-based positioning." in IEEE International Conference on Communications (ICC), 2011, pp. 1-5.

[12] C. Laoudias, M. Michaelides, and C. Panayiotou, "Lovalization using radial basis function networks and signal strength fingerprints in WLAN," in IEEE GLOBECOM, 2009, pp. 1-6

[13] B. Li, J Salter, A. Dempster, and C. Rizos, "Indoor positioning techniques based on wireless LAN," in 1st IEEE International Conference on Wireless Broadband and Ultra Wideband Communications, 2006, pp. 13-16

[14] PowerTutor: http://powertutor.org/

[15] T. Roos, P.Myllymaki, H. Tirri, P. Misikangas, and J. Sievanen, "A probabilistic approach to WLAN user location estimation," International Journal of Wireless Information Networks, vol. 9, no. 3, pp. 155-164, Jul. 2002.

[16] Source: Strategy Analytics http://www.strategyanalytics.com.

[17] M. Youssef and A. Agrawala, "The Horus WLAN location determination system," in ACM MobiSys, 2005, pp. 205-218

Παράρτημα Α

Το σύστημα Aiplace

Στο παράρτημα Α παρουσιάζουμε τον τρόπο δημιουργίας βοηθητικών αρχείων .floor τα οποία βοηθούν στην χρήση του Floor Mode κατά την οποία η εφαρμογή Airplace αναγνωρίζει τον όροφο στον οποίο βρίσκεται ο χρήστης και εναλάσσεις αναλόγως.

Επίσης, θα παρουσιάσουμε τους αντίστοιχους κώδικες τόσο του Airplace Tracker (με την λειτουργία των ορόφων περιλαμβανομένη) αλλά και τον κώδικα της δημιουργίας των .floor αρχείων.

Δημιουργία .floor αρχείων

Αφού δημιοργηθούν τα κατάλληλα αρχεία για τοπικοποίηση από τον Radiomap Server, ο χρήστης βάζει στην main κλάση του κώδικα που παρουσιάζουμε πιο κάτω το όνομα του radiomap-mean το οποίο το αντιγράφει στον φάκελο που βρίσκεται ο πηγαίος κώδικας.

Έπειτα, εκτελεί την εφαρμογή και δημιουργείται αμέσως το .floor αρχείο.

Κώδικας για δημιουργία .floor αρχείου

```java
import java.io.IOException;


public class MainClass
{

    /**
     * @param args
     * @throws IOException
     */
    public static void main(String[] args) throws IOException
    {
        String nameOfTheFile = "name-of-the-radiomap-mean.txt";
        ParseFile pf = new ParseFile(nameOfTheFile);
        RadioMapHelp rmh = new RadioMapHelp(pf);
        rmh.doTheProcedure();
        pf.closeTheFile();
    }

}
```

```
public class LogRecord
{

    private String bssid;
    private float rss;

    public LogRecord(String bssid, float rss)
    {
        super();
        this.bssid = bssid;
        this.rss = rss;
    }

    public String getBssid()
    {
        return bssid;
    }

    public float getRss()
    {
        return rss;
    }

    public String toString()
    {
        String str = new String();
        str = String.valueOf(bssid) + " " + String.valueOf(rss) + "\n";
        return str;
    }

}
```

```java
import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;


public class ParseFile
{
    DataInputStream in = null;
    BufferedReader br = null;

    private ArrayList<String> macAddresses = new ArrayList<String>();

    private String nameOfTheRadioMap=null;
    private int sizeOfFile = 0;
    public ParseFile(String nameOfTheFile)
    {
        nameOfTheRadioMap=nameOfTheFile;
        FileInputStream fstream = null;
        try
        {
            fstream = new FileInputStream(nameOfTheFile);
        } catch (FileNotFoundException e)
        {
            e.printStackTrace();
        }
        // Get the object of DataInputStream
        in = new DataInputStream(fstream);
        br = new BufferedReader(new InputStreamReader(in));

        //Parse the first mac address line
        String strLine = null;
        try
        {
            strLine = br.readLine();
        } catch (IOException e)
        {
            e.printStackTrace();
        }
        String[] macAddressesArray = strLine.split(", ");

        for(int i=2; i<macAddressesArray.length; ++i)
        {
            macAddresses.add(macAddressesArray[i]);
        }
    }

    public void closeTheFile() throws IOException
    {
        in.close();
    }

    public String getTheNextLine() throws IOException
    {
        ++sizeOfFile;
        return parseCentralizedLine();
```



```java
    }

    public ArrayList<String> getMacAddressList()
    {
        return macAddresses;
    }

    public int getTheSizeOfFile()
    {
        return sizeOfFile-1;
    }

    public String nameOfFile()
    {
        return this.nameOfTheRadioMap;
    }

    /**
     * Used for Centralized Approach
     * Convert the arrayList into a specific format for sending
     * @return the formated arrayList
     * @throws IOException
     */
    private String parseCentralizedLine() throws IOException
    {
        String currentLine = br.readLine();
        if(currentLine==null)
        {
            return null;
        }
        else
        {
            String[] currentRss = currentLine.split(", ");
            String str = "";
            for(int j=0,z=2; j<macAddresses.size(); ++j,++z)
            {
                String macStr=macAddresses.get(j);
                str+=macStr+", "+currentRss[z];

                if(!((macAddresses.size()-1) == j))
                {
                    str+=", ";
                }
            }
            return str;
        }
    }
}
```

```
import java.io.IOException;
import java.util.ArrayList;


public class RadioMapHelp
{
    private ArrayList<String> macAddresses = new ArrayList<String>();
    private ParseFile pf = null;
    private Double[] rss = null;
    private Integer[] noOfValidMacAddresses = null;


    WriteData wd = null;

    /**
     * Constructor
     *
     * @param pf
     * @throws IOException
     */
    RadioMapHelp(ParseFile pf) throws IOException
    {
        this.macAddresses=pf.getMacAddressList();
        this.pf=pf;
        rss = new Double[macAddresses.size()];
        noOfValidMacAddresses = new Integer[macAddresses.size()];

        //Initialize the arrays
        rss=initializeTheArray(rss);
        initializeTheArray();           //init the int array

        int indexOfFull= pf.nameOfFile().indexOf(".");
        String nameOfHelpFile = pf.nameOfFile().substring(0,indexOfFull);
        nameOfHelpFile = nameOfHelpFile+".floor";
        System.out.println(nameOfHelpFile+" Size: "+macAddresses.size());
        wd = new WriteData(nameOfHelpFile);

    }



    public void doTheProcedure() throws IOException
    {
        String incStr = this.pf.getTheNextLine();

        while(incStr!=null)
        {
            //Do the procedure
            /**
             * Format the line
             * Take the line and return an Array - Double with the rss
                 values
             */
            Double[] arrayCurrentRss = proccessLine(incStr);


            for(int i=0; i<rss.length; ++i)
            {
                //summing
```



```
                if(String.valueOf(arrayCurrentRss[i]).equals("-110.0"))
                {
                    //do nothing

                }
                else
                {
                    rss[i]+=arrayCurrentRss[i];
                    noOfValidMacAddresses[i]++;
                }

            }

            incStr = this.pf.getTheNextLine();
        }

        /**
         * Find the average and store to a new file
         */
        for(int i=0; i<rss.length; ++i)
        {
            wd.writeLineToFile(macAddresses.get(i)+", "+rss[i]/
                noOfValidMacAddresses[i]+", "+noOfValidMacAddresses[i]);
        }
        wd.closeTheFile();
        System.out.println("Size: "+pf.getTheSizeOfFile());
    }

    /**
     * Create the temporary array of double values to use for summing
         after
     *
     * @param line
     * @return array of Double values
     */

    private Double[] proccessLine(String line)
    {
        String[] temp = line.split(", ");
        Double[] dArray = new Double[macAddresses.size()];

        for(int i=0, j=0; i<temp.length-1; i+=2,++j)
        {
//          System.out.println(temp[i]+" "+temp[i+1]);

            Double tempD = Double.valueOf(temp[i+1]);
            dArray[j]=tempD;
        }

        return dArray;
    }

    /**
     * Initialize the array with 0.0
     *
     * @param  Array of doubles rss
     * @return Array of doubles
     */
    private Double[] initializeTheArray(Double[] rss)
```

```
        {
            for(int i=0; i<rss.length; ++i)
            {
                rss[i]=0.0;
            }

            return rss;
        }

        /**
         * Initialize the array with 0
         *
         */
        private void initializeTheArray()
        {
            for(int i=0; i<noOfValidMacAddresses.length; ++i)
            {
                noOfValidMacAddresses[i]=0;
            }

        }
    }
```

```java
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.Writer;
import java.util.ArrayList;


public class WriteData
{
    private Writer output = null;
    private File file = null;
    public WriteData(String fileName) throws IOException
    {
        file = new File(fileName);
        output= new BufferedWriter(new FileWriter(file));
    }

    /**
     *
     * @param line
     */
    public void writeLineToFile(String line)
    {
        try
        {
            output.write(line+"\n");
        } catch (IOException e)
        {
            e.printStackTrace();
        }
    }

    public void closeTheFile()
    {
        try
        {
            output.close();
        } catch (IOException e)
        {
            e.printStackTrace();
        }
    }

    /**
     *
     * @param incomingList
     * @throws IOException
     */
    public void writeArrayListToFile(ArrayList<String> incomingList)
        throws IOException
    {
        for(String str : incomingList)
        {
            str+="\n";
            output.write(str);
        }
```



```java
            output.close();
        }
    }
```

```
/*
* Copyright (c) 2011, KIOS Research Center and Data Management Systems Lab
    ,
* University of Cyprus. All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
    modification,
* are permitted provided that the following conditions are met:
*
*    * Redistributions of source code must retain the above copyright
    notice, this
*      list of conditions and the following disclaimer.
*    * Redistributions in binary form must reproduce the above copyright
    notice,
*      this list of conditions and the following disclaimer in the
    documentation
*      and/or other materials provided with the distribution.
*    * Neither the name of the Sony Ericsson Mobile Communication AB nor
    the names
*      of its contributors may be used to endorse or promote products
    derived from
*      this software without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
    IS" AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
    IMPLIED
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
    DISCLAIMED.
* IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY
    DIRECT,
* INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
    (INCLUDING,
* BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
    USE,
* DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
    THEORY OF
* LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
    NEGLIGENCE
* OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
    IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
*/

package cy.com.airplace;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;

import android.util.Log;

public class Algorithms {

    /**
     *
```



```
     * @param latestScanList
     *            the current scan list of APs
     * @param RM
     *            the constructed Radio Map
     *
     * @param algorithm_choice
     *            choice of several algorithms
     *
     * @return the location of user
     */
    public static String ProcessingAlgorithms(ArrayList<LogRecord>
        latestScanList, RadioMap RM, int algorithm_choice,
            ArrayList<WeightRecord> WeightsList) {

        int i, j;

        ArrayList<String> MacAdressList = RM.getMacAdressList();
        ArrayList<String> Observed_RSS_Values = new ArrayList<String>();
        LogRecord temp_LR;
        int notFoundCounter = 0;
        // Read parameter of algorithm
        String NaNValue = readParameter(RM.getRadiomapMean_File(), 0);

        // Check which mac addresses of radio map, we are currently
            listening.
        for (i = 0; i < MacAdressList.size(); ++i) {

            for (j = 0; j < latestScanList.size(); ++j) {

                temp_LR = latestScanList.get(j);

                // MAC Address Matched
                if (MacAdressList.get(i).compareTo(temp_LR.getBssid()) ==
                    0) {
                    Observed_RSS_Values.add(String.valueOf(temp_LR.getRss
                        ()));
                    break;
                }
            }
            // A MAC Address is missing so we place a small value, NaN
                value
            if (j == latestScanList.size()){
                Observed_RSS_Values.add(String.valueOf(NaNValue));
                ++notFoundCounter;
            }
        }

        if (notFoundCounter == MacAdressList.size())
            return null;

        // Read parameter of algorithm
        String parameter = readParameter(RM.getRadiomapMean_File(),
            algorithm_choice);

        if (parameter == null)
            return null;

        switch (algorithm_choice) {
```

```
            case 1:
                return KNN_WKNN_Algorithm(RM, Observed_RSS_Values, parameter,
                    false);
            case 2:
                return KNN_WKNN_Algorithm(RM, Observed_RSS_Values, parameter,
                    true);
            case 3:
                return MAP_MMSE_Algorithm(RM, Observed_RSS_Values, parameter,
                    false);
            case 4:
                return MAP_MMSE_Algorithm(RM, Observed_RSS_Values, parameter,
                    true);
            case 5:
                return RBF_Algorithm(RM, Observed_RSS_Values, parameter,
                    WeightsList);
            case 6:
                return SNAP_Algorithm(RM, Observed_RSS_Values, parameter);
            }
            return null;

        }

        /**
         * Calculates user location based on Weighted/Not Weighted K Nearest
         * Neighbor (KNN) Algorithm
         *
         * @param RM
         *            The radio map structure
         *
         * @param Observed_RSS_Values
         *            RSS values currently observed
         * @param parameter
         *
         * @param isWeighted
         *            To be weighted or not
         *
         * @param K
         *            The number of locations used
         *
         * @return The estimated user location
         */
        private static String KNN_WKNN_Algorithm(RadioMap RM, ArrayList<String
            > Observed_RSS_Values, String parameter, boolean isWeighted) {

            ArrayList<String> RSS_Values;
            float curResult = 0;
            ArrayList<LocDistance> LocDistance_Results_List = new ArrayList<
                LocDistance>();
            String myLocation = null;
            int K;

            try {
                K = Integer.parseInt(parameter);
            } catch (Exception e) {
                return null;
            }

            // Construct a list with locations-distances pairs for currently
            // observed RSS values
```



```
            for (String location : RM.getLocationRSS_HashMap().keySet()) {
                RSS_Values = RM.getLocationRSS_HashMap().get(location);
                curResult = calculateEuclideanDistance(RSS_Values,
                    Observed_RSS_Values);

                if (curResult == Float.NEGATIVE_INFINITY)
                    return null;

                LocDistance_Results_List.add(0, new LocDistance(curResult,
                    location));
            }

            // Sort locations-distances pairs based on minimum distances
            Collections.sort(LocDistance_Results_List, new Comparator<
                LocDistance>() {

                public int compare(LocDistance gd1, LocDistance gd2) {
                    return (gd1.getDistance() > gd2.getDistance() ? 1 : (gd1.
                        getDistance() == gd2.getDistance() ? 0 : -1));
                }
            });

            if (!isWeighted) {
                myLocation = calculateAverageKDistanceLocations
                    (LocDistance_Results_List, K);
            } else {
                myLocation = calculateWeightedAverageKDistanceLocations
                    (LocDistance_Results_List, K);
            }

            return myLocation;

        }

        /**
         * Calculates user location based on Probabilistic Maximum A
             Posteriori
         * (MAP) Algorithm or Probabilistic Minimum Mean Square Error (MMSE)
         * Algorithm
         *
         * @param RM
         *            The radio map structure
         *
         * @param Observed_RSS_Values
         *            RSS values currently observed
         * @param parameter
         *
         * @param isWeighted
         *            To be weighted or not
         *
         * @return The estimated user location
         */
        private static String MAP_MMSE_Algorithm(RadioMap RM, ArrayList<String
            > Observed_RSS_Values, String parameter, boolean isWeighted) {

            ArrayList<String> RSS_Values;
            double curResult = 0.0d;
            String myLocation = null;
            double highestProbability = Double.NEGATIVE_INFINITY;
```

```
        ArrayList<LocDistance> LocDistance_Results_List = new ArrayList<
            LocDistance>();
        float sGreek;

        try {
            sGreek = Float.parseFloat(parameter);
        } catch (Exception e) {
            return null;
        }

        // Find the location of user with the highest probability
        for (String location : RM.getLocationRSS_HashMap().keySet()) {

            RSS_Values = RM.getLocationRSS_HashMap().get(location);
            curResult = calculateProbability(RSS_Values,
                Observed_RSS_Values, sGreek);

            if (curResult == Double.NEGATIVE_INFINITY)
                return null;
            else if (curResult > highestProbability) {
                highestProbability = curResult;
                myLocation = location;
            }

            if (isWeighted)
                LocDistance_Results_List.add(0, new LocDistance(curResult,
                    location));
        }

        if (isWeighted)
            myLocation = calculateWeightedAverageProbabilityLocations
                (LocDistance_Results_List);

        return myLocation;
    }

    /**
     * Calculates user location based on Radial Basis Function (RBF)
         Algorithm
     *
     * @param RM
     *            The radio map structure
     * @param parameter
     *
     * @param Observed_RSS_Values
     *            RSS values currently observed
     *
     * @return The estimated user location
     */
    private static String RBF_Algorithm(RadioMap RM, ArrayList<String>
        observedRSSValues, String parameter, ArrayList<WeightRecord>
        WeightsList) {

        ArrayList<String> RSS_Values;
        float curResultX = 0.0f;
        float curResultY = 0.0f;
        String myLocation = null;
        float numerator = 0.0f;
        float denominator = 0.0f;
```



```
        float sGreekRBF;

        try {
            sGreekRBF = Float.parseFloat(parameter);
        } catch (Exception e) {
            return null;
        }

        int i = 0;

        if (WeightsList == null || WeightsList.size() != RM.getOrderList()
            .size())
            return null;

        for (String location : RM.getOrderList()) {
            RSS_Values = RM.getLocationRSS_HashMap().get(location);

            numerator = computeNumerator(observedRSSValues, RSS_Values,
                sGreekRBF);

            denominator += numerator;

            if (numerator == Float.NEGATIVE_INFINITY)
                return null;

            curResultX += numerator * WeightsList.get(i).getWeightX();
            curResultY += numerator * WeightsList.get(i).getWeightY();

            ++i;
        }
        curResultX /= denominator;
        curResultY /= denominator;
        myLocation = curResultX + " " + curResultY;
        return myLocation;
    }

    /**
     * Calculates user location based on Subtract on Negative Add on
         Positive
     * (SNAP) Algorithm
     *
     * @param RM
     *            The radio map structure
     * @param parameter
     *
     * @param Observed_RSS_Values
     *            RSS values currently observed
     *
     * @return The estimated user location
     */
    private static String SNAP_Algorithm(RadioMap RM, ArrayList<String>
        observedRSSValues, String parameter) {

        ArrayList<Zone> Zones = new ArrayList<Zone>();
        ArrayList<Boolean[][]> ZoneArrays = null;
        ArrayList<Integer> sumElementsL = null;
        String myLocation = null;
        int[][] LikelihoodMatrixL = null;
```

```
            int M;
            int MIN_RSS_VALUE;
            int MAX_RSS_VALUE;
            float A;

            String[] temp;
            parameter = parameter.replace(", ", " ");
            temp = parameter.split(" ");

            // Must have more than 2 fields
            if (temp.length < 3)
                return null;

            // Read the parameters
            try {
                M = Integer.parseInt(temp[0]);
                MIN_RSS_VALUE = Integer.parseInt(temp[1]);
                MAX_RSS_VALUE = Integer.parseInt(temp[2]);
            } catch (Exception e) {
                return null;
            }

            A = (MAX_RSS_VALUE - MIN_RSS_VALUE) / (float) M;

            // Calculate the Zones based on M parameter
            Zone z = null;
            for (int i = 1; i <= M; ++i) {
                z = calculateZone(MIN_RSS_VALUE, A, i);
                Zones.add(z);
            }

            Log.d("SNAP", "Size: " + Zones.size() + "min:" + z.getMin() + "
                max:" + z.getMax());

            // Calculate the Zone Arrays
            ZoneArrays = createZoneArrays(Zones, RM);

            if (ZoneArrays == null)
                return null;

            // Calculate the L (likelihood matrix L)
            LikelihoodMatrixL = createLikelihoodMatrixL(ZoneArrays, Zones,
                observedRSSValues, RM);

            if (LikelihoodMatrixL == null)
                return null;

            sumElementsL = sumElementsLikelihoodMatrixL(LikelihoodMatrixL);

            myLocation = calculateMaxScoreLocation(sumElementsL, RM);

            return myLocation;
        }

        /**
         * Calculates the location of max score locations
         *
         * @param sumElementsL
         *            the list with the score in location ordering
```



```
         *
         * @param RM
         *            The radio map structure
         *
         * @return The estimated user location
         */
        private static String calculateMaxScoreLocation(ArrayList<Integer>
            sumElementsL, RadioMap RM) {

            int MaxLocationSum = Integer.MIN_VALUE;
            float sumX = 0.0f;
            float sumY = 0.0f;
            float x, y;

            int countMaxLocationSum = 0;
            String[] LocationArray = new String[2];

            for (int i = 0; i < sumElementsL.size(); ++i) {
                if (sumElementsL.get(i) > MaxLocationSum)
                    MaxLocationSum = sumElementsL.get(i);
            }

            for (int i = 0; i < sumElementsL.size(); ++i) {
                if (sumElementsL.get(i) == MaxLocationSum) {
                    countMaxLocationSum++;
                    LocationArray = RM.getOrderList().get(i).split(" ");
                    try {
                        x = Float.valueOf(LocationArray[0].trim()).floatValue
                            ();
                        y = Float.valueOf(LocationArray[1].trim()).floatValue
                            ();
                    } catch (Exception e) {
                        return null;
                    }

                    sumX += x;
                    sumY += y;
                }

            }

            // Calculate the average
            sumX /= countMaxLocationSum;
            sumY /= countMaxLocationSum;

            return sumX + " " + sumY;
        }

        /**
         * Sums up the elements of L (row wise) and this gives the score for
             each
         * location
         *
         * @param LikelihoodMatrixL
         *            the L matrix
         *
         * @return a list with the sums, in order of locations
         */
        private static ArrayList<Integer> sumElementsLikelihoodMatrixL(int[][]
```

```
            LikelihoodMatrixL) {

            int LocationSum = 0;
            ArrayList<Integer> sumElementsL = new ArrayList<Integer>();

            for (int i = 0; i < LikelihoodMatrixL.length; ++i) {
                LocationSum = 0;
                for (int j = 0; j < LikelihoodMatrixL[0].length; ++j) {
                    LocationSum += LikelihoodMatrixL[i][j];
                }
                sumElementsL.add(LocationSum);
            }
            return sumElementsL;
        }

        /**
         * Creates a lxn matrix array (denoted as likelihood matrix L) and
             each
         * element (i, j) in matrix L contains the contribution of the j-th AP
             to
         * the location i
         *
         * @param ZoneArrays
         *            the Zone matrices
         *
         * @param RM
         *            The radio map structure
         *
         * @param Zones
         *            the zones min and max values
         *
         * @param Observed_RSS_Values
         *            RSS values currently observed
         *
         * @return matrix L
         */
        private static int[][] createLikelihoodMatrixL(ArrayList<Boolean[][]>
            ZoneArrays, ArrayList<Zone> Zones, ArrayList<String>
            observedRSSValues,
                RadioMap RM) {

            final int row = RM.getOrderList().size();
            final int column = RM.getMacAdressList().size();
            int[][] LikelihoodMatrixL = new int[row][column];
            Boolean[][] ZoneArray = null;
            Boolean[][] NeighborZoneArray = null;
            int zone_index;

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

                zone_index = getZoneIndex(observedRSSValues.get(i), Zones);
                int j;

                if (zone_index == -1) {
                    return null;
                }

                if (zone_index == Zones.size()) {
                    for (j = 0; j < row; ++j)
```



```
                        LikelihoodMatrixL[j][i]--;
                    continue;
                }

                ZoneArray = ZoneArrays.get(zone_index);

                for (j = 0; j < row; ++j) {
                    if (ZoneArray[j][i] == true) {
                        LikelihoodMatrixL[j][i]++;
                    } else {
                        int start = (zone_index - 1 >= 0) ? (zone_index - 1) :
                            zone_index;
                        int end = (zone_index + 1 <= ZoneArrays.size() - 1) ?
                            (zone_index + 1) : zone_index;
                        int z;

                        for (z = start; z <= end; ++z) {
                            if (z == zone_index)
                                continue;

                            NeighborZoneArray = ZoneArrays.get(z);

                            if (NeighborZoneArray[j][i] == true) {
                                break;
                            }
                        }

                        if (z != end + 1) {
                            continue;
                        }

                        for (z = 0; z < ZoneArrays.size(); ++z) {
                            if (Math.abs(zone_index - z) <= 1)
                                continue;

                            NeighborZoneArray = ZoneArrays.get(z);

                            if (NeighborZoneArray[j][i] == true) {
                                LikelihoodMatrixL[j][i]--;
                                break;
                            }
                        }
                    }
                }
            }

            return LikelihoodMatrixL;
        }

        private static int getZoneIndex(String RSS_Val, ArrayList<Zone> Zones)
            {

            Zone Z = null;
            float RSS_Value;

            try {
                RSS_Value = Float.parseFloat(RSS_Val);
            } catch (Exception e) {
                return -1;
```

```
        }

        int z;
        for (z = 0; z < Zones.size(); ++z) {
            Z = Zones.get(z);

            if (RSS_Value >= Z.getMin() && RSS_Value <= Z.getMax())
                break;
        }

        return z;
    }

    /**
     * Calculates Zi, Zm=[MIN_RSS_VALUE + (m-1)*A, min + m*A], where m=1,.
       ..,M
     * and A=(max-min)/2
     *
     * @param MIN_RSS_VALUE
     *            The minimum value of radiomap
     *
     * @param A
     *            A=(max-min)/2
     *
     * @param m
     *            the number of zone
     *
     * @return The new zone
     */
    private static Zone calculateZone(int MIN_RSS_VALUE, float A, int m) {

        float min = MIN_RSS_VALUE + (m - 1) * A;
        float max = MIN_RSS_VALUE + m * A;

        return new Zone(min, max);
    }

    /**
     * Calculates the lxn matrices for each zone of boolean values, where
       l is
     * the number of distinct locations and n is the total number of APs,
       by
     * checking whether the RSS value of an AP in the radiomap falls into
       the
     * respective zone
     *
     * @param Zones
     *            The constructed zones
     *
     * @param RM
     *            the radio map
     *
     *
     * @return the lxn matrices list
     */
    private static ArrayList<Boolean[][]> createZoneArrays(ArrayList<Zone>
        Zones, RadioMap RM) {

        ArrayList<Boolean[][]> ZoneArrays = new ArrayList<Boolean[][]>();
```



```
        Boolean[][] ZoneArray = null;
        ArrayList<String> RSS_Values = null;

        int row = RM.getOrderList().size();
        int column = RM.getMacAdressList().size();

        // Create lxn matrix for each zone
        for (int z = 0; z < Zones.size(); ++z) {
            ZoneArray = new Boolean[row][column];
            ZoneArrays.add(ZoneArray);
        }

        float RSS_Value = -110;
        Zone Z = null;
        ArrayList<String> OrderList = null;
        String location = null;

        OrderList = RM.getOrderList();

        // Fill the boolean values
        for (int i = 0; i < OrderList.size(); ++i) {
            location = OrderList.get(i);
            RSS_Values = RM.getLocationRSS_HashMap().get(location);

            for (int j = 0; j < RSS_Values.size(); ++j) {

                try {
                    RSS_Value = Float.parseFloat(RSS_Values.get(j));
                } catch (Exception e) {
                    return null;
                }

                for (int z = 0; z < Zones.size(); ++z) {
                    Z = Zones.get(z);
                    ZoneArray = ZoneArrays.get(z);

                    if (RSS_Value >= Z.getMin() && RSS_Value <= Z.getMax()
                        ) {
                        ZoneArray[i][j] = true;

                        for (z = z + 1; z < Zones.size(); ++z) {
                            ZoneArray = ZoneArrays.get(z);
                            ZoneArray[i][j] = false;
                        }

                        break;
                    } else
                        ZoneArray[i][j] = false;
                }
            }
        }
        return ZoneArrays;
    }

    public static ArrayList<WeightRecord> readWeights(File file) {

        String line;
        BufferedReader reader = null;
        FileReader fr = null;
```

```
            ArrayList<WeightRecord> returnWeightList = new ArrayList<
                WeightRecord>();

            try {
                fr = new FileReader(file.getAbsolutePath() + "-rbf-weights");
                reader = new BufferedReader(fr);

                while ((line = reader.readLine()) != null) {

                    /* Ignore the labels */
                    if (line.startsWith("#") || line.trim().equals("")) {
                        continue;
                    }

                    line = line.replace(", ", " ");
                    /* Split fields */
                    String[] temp = line.split(" ");

                    /* The file may be corrupted so ignore reading it */
                    if (temp.length != 2) {
                        return null;
                    }

                    returnWeightList.add(new WeightRecord(Float.parseFloat
                        (temp[0]), Float.parseFloat(temp[1])));

                }

                fr.close();
                reader.close();
            } catch (Exception e) {
                return null;
            }
            return returnWeightList;
        }

        /**
         * Computes exp(-1/2s^2 * || Si - Cj ||^2)
         *
         * @param Srow
         *            a single row of RSS Values in Radiomap file
         * @param Crow
         *            a single row of RSS Values in Radiomap mean file
         *
         * @return the Numerator of u(Si,Cj) function
         */
        private static float computeNumerator(ArrayList<String> Srow,
            ArrayList<String> Crow, float sGreekRBF) {

            float Si;
            float Cj;

            float firstParameter = (float) (-1 / (2 * sGreekRBF * sGreekRBF));
            float secondParameter = 0.0f;

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

                try {
                    Si = Float.parseFloat(Srow.get(i));
```



```
                    Cj = Float.parseFloat(Crow.get(i));
                } catch (Exception e) {
                    return Float.NEGATIVE_INFINITY;
                }

                secondParameter = secondParameter + (Si - Cj) * (Si - Cj);
            }

            return (float) Math.exp(firstParameter * secondParameter);
        }

        /**
         * Calculates the Euclidean distance between the currently observed
             RSS
         * values and the RSS values for a specific location.
         *
         * @param l1
         *            RSS values of a location in radiomap
         * @param l2
         *            RSS values currently observed
         *
         * @return The Euclidean distance, or MIN_VALUE for error
         */
        private static float calculateEuclideanDistance(ArrayList<String> l1,
            ArrayList<String> l2) {

            float finalResult = 0;
            float v1;
            float v2;
            float temp;
            String str;

            for (int i = 0; i < l1.size(); ++i) {

                try {
                    str = l1.get(i);
                    v1 = Float.valueOf(str.trim()).floatValue();
                    str = l2.get(i);
                    v2 = Float.valueOf(str.trim()).floatValue();
                } catch (Exception e) {
                    return Float.NEGATIVE_INFINITY;
                }

                // do the procedure
                temp = v1 - v2;
                temp *= temp;

                // do the procedure
                finalResult += temp;
            }
            return ((float) Math.sqrt(finalResult));
        }

        /**
         * Calculates the Probability of the user being in the currently
             observed
         * RSS values and the RSS values for a specific location.
         *
         * @param l1
```

```
     *            RSS values of a location in radiomap
     * @param l2
     *            RSS values currently observed
     *
     * @return The Probability for this location, or MIN_VALUE for error
     */
    public static double calculateProbability(ArrayList<String> l1,
        ArrayList<String> l2, float sGreek) {

        double finalResult = 1;
        float v1;
        float v2;
        double temp;
        String str;

        for (int i = 0; i < l1.size(); ++i) {

            try {
                str = l1.get(i);
                v1 = Float.valueOf(str.trim()).floatValue();
                str = l2.get(i);
                v2 = Float.valueOf(str.trim()).floatValue();
            } catch (Exception e) {
                return Double.NEGATIVE_INFINITY;
            }

            temp = v1 - v2;

            temp *= temp;

            temp = -temp;

            temp /= (double) (sGreek * sGreek);
            temp = (double) Math.exp(temp);

            finalResult *= temp;
        }
        return finalResult;
    }

    /**
     * Calculates the Average of the K locations that have the shortest
     * distances D
     *
     * @param LocDistance_Results_List
     *            Locations-Distances pairs sorted by distance
     * @param K
     *            The number of locations used
     * @return The estimated user location, or null for error
     */
    private static String calculateAverageKDistanceLocations(ArrayList<
        LocDistance> LocDistance_Results_List, int K) {

        float sumX = 0.0f;
        float sumY = 0.0f;

        String[] LocationArray = new String[2];
        float x, y;
```



```
        int K_Min = K < LocDistance_Results_List.size() ? K :
            LocDistance_Results_List.size();

        // Calculate the sum of X and Y
        for (int i = 0; i < K_Min; ++i) {
            LocationArray = LocDistance_Results_List.get(i).getLocation().
                split(" ");

            try {
                x = Float.valueOf(LocationArray[0].trim()).floatValue();
                y = Float.valueOf(LocationArray[1].trim()).floatValue();
            } catch (Exception e) {
                return null;
            }

            sumX += x;
            sumY += y;
        }

        // Calculate the average
        sumX /= K_Min;
        sumY /= K_Min;

        return sumX + " " + sumY;

    }

    /**
     * Calculates the Weighted Average of the K locations that have the
         shortest
     * distances D
     *
     * @param LocDistance_Results_List
     *            Locations-Distances pairs sorted by distance
     * @param K
     *            The number of locations used
     * @return The estimated user location, or null for error
     */
    public static String calculateWeightedAverageKDistanceLocations
        (ArrayList<LocDistance> LocDistance_Results_List, int K) {

        double LocationWeight = 0.0f;
        double sumWeights = 0.0f;
        double WeightedSumX = 0.0f;
        double WeightedSumY = 0.0f;

        String[] LocationArray = new String[2];
        float x, y;

        int K_Min = K < LocDistance_Results_List.size() ? K :
            LocDistance_Results_List.size();

        // Calculate the weighted sum of X and Y
        for (int i = 0; i < K_Min; ++i) {

            LocationWeight = 1 / LocDistance_Results_List.get(i).
                getDistance();
            LocationArray = LocDistance_Results_List.get(i).getLocation().
                split(" ");
```

```
            try {
                x = Float.valueOf(LocationArray[0].trim()).floatValue();
                y = Float.valueOf(LocationArray[1].trim()).floatValue();
            } catch (Exception e) {
                return null;
            }

            sumWeights += LocationWeight;
            WeightedSumX += LocationWeight * x;
            WeightedSumY += LocationWeight * y;

        }

        WeightedSumX /= sumWeights;
        WeightedSumY /= sumWeights;

        return WeightedSumX + " " + WeightedSumY;
    }

    /**
     * Calculates the Weighted Average over ALL locations where the
         weights are
     * the Normalized Probabilities
     *
     * @param LocDistance_Results_List
     *            Locations-Probability pairs
     *
     * @return The estimated user location, or null for error
     */
    public static String calculateWeightedAverageProbabilityLocations
        (ArrayList<LocDistance> LocDistance_Results_List) {

        double sumProbabilities = 0.0f;
        double WeightedSumX = 0.0f;
        double WeightedSumY = 0.0f;
        double NP;
        float x, y;
        String[] LocationArray = new String[2];

        // Calculate the sum of all probabilities
        for (int i = 0; i < LocDistance_Results_List.size(); ++i)
            sumProbabilities += LocDistance_Results_List.get(i).
                getDistance();

        // Calculate the weighted (Normalized Probabilities) sum of X and
            Y
        for (int i = 0; i < LocDistance_Results_List.size(); ++i) {
            LocationArray = LocDistance_Results_List.get(i).getLocation().
                split(" ");

            try {
                x = Float.valueOf(LocationArray[0].trim()).floatValue();
                y = Float.valueOf(LocationArray[1].trim()).floatValue();
            } catch (Exception e) {
                return null;
            }

            NP = LocDistance_Results_List.get(i).getDistance() /
```



```
                    sumProbabilities;

            WeightedSumX += (x * NP);
            WeightedSumY += (y * NP);

        }

        return WeightedSumX + " " + WeightedSumY;

    }

    /**
     * Reads the parameters from the file
     *
     * @param file
     *            the file of radiomap, to read parameters
     *
     * @param algorithm_choice
     *            choice of several algorithms
     *
     * @return The parameter for the algorithm
     *
     */
    private static String readParameter(File file, int algorithm_choice) {

        String line;
        BufferedReader reader = null;
        FileReader fr = null;

        String parameter = null;

        try {
            fr = new FileReader(file.getAbsolutePath() + "-parameters");
            reader = new BufferedReader(fr);

            while ((line = reader.readLine()) != null) {

                /* Ignore the labels */
                if (line.startsWith("#") || line.trim().equals("")) {
                    continue;
                }

                /* Split fields */
                String[] temp = line.split(":");

                /* The file may be corrupted so ignore reading it */
                if (temp.length != 2) {
                    return null;
                }

                if (algorithm_choice == 0 && temp[0].equals("NaN")) {
                    parameter = temp[1];
                    break;
                } else if (algorithm_choice == 1 && temp[0].equals("KNN"))
                    {
                    parameter = temp[1];
                    break;
                } else if (algorithm_choice == 2 && temp[0].equals("WKNN")
                    ) {
```

```
                        parameter = temp[1];
                        break;
                    } else if (algorithm_choice == 3 && temp[0].equals("MAP"))
                        {
                        parameter = temp[1];
                        break;
                    } else if (algorithm_choice == 4 && temp[0].equals("MMSE")
                        ) {
                        parameter = temp[1];
                        break;
                    } else if (algorithm_choice == 5 && temp[0].equals("RBF"))
                        {
                        parameter = temp[1];
                        break;
                    } else if (algorithm_choice == 6 && temp[0].equals("SNAP")
                        ) {
                        parameter = temp[1];
                        break;
                    }

                }
                fr.close();
                reader.close();
            } catch (Exception e) {
                return null;
            }

            return parameter;
        }

    }
```

Airplace Tracker code


```
/*
* Copyright (c) 2011, KIOS Research Center and Data Management Systems Lab
    ,
* University of Cyprus. All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
    modification,
* are permitted provided that the following conditions are met:
*
*    * Redistributions of source code must retain the above copyright
    notice, this
*      list of conditions and the following disclaimer.
*    * Redistributions in binary form must reproduce the above copyright
    notice,
*      this list of conditions and the following disclaimer in the
    documentation
*      and/or other materials provided with the distribution.
*    * Neither the name of the Sony Ericsson Mobile Communication AB nor
    the names
*      of its contributors may be used to endorse or promote products
    derived from
*      this software without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
    IS" AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
    IMPLIED
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
    DISCLAIMED.
* IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY
    DIRECT,
* INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
    (INCLUDING,
* BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
    USE,
* DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
    THEORY OF
* LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
    NEGLIGENCE
* OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
    IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
*/

package cy.com.airplace;

import java.util.Observable;

public class BooleanObservable extends Observable {

    private boolean isTracking;

    public boolean get() {
        return isTracking;
    }

    public void setBoolean(boolean track) {

        isTracking = track;
```



```
            setChanged();
        }
    }
```

Airplace Tracker code


```
/*
* Copyright (c) 2011, KIOS Research Center and Data Management Systems Lab
    ,
* University of Cyprus. All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
    modification,
* are permitted provided that the following conditions are met:
*
*    * Redistributions of source code must retain the above copyright
    notice, this
*      list of conditions and the following disclaimer.
*    * Redistributions in binary form must reproduce the above copyright
    notice,
*      this list of conditions and the following disclaimer in the
    documentation
*      and/or other materials provided with the distribution.
*    * Neither the name of the Sony Ericsson Mobile Communication AB nor
    the names
*      of its contributors may be used to endorse or promote products
    derived from
*      this software without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
    IS" AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
    IMPLIED
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
    DISCLAIMED.
* IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY
    DIRECT,
* INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
    (INCLUDING,
* BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
    USE,
* DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
    THEORY OF
* LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
    NEGLIGENCE
* OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
    IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
*/

package cy.com.airplace;

import cy.com.airplace.R;

import android.app.AlertDialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.preference.Preference;
import android.preference.Preference.OnPreferenceClickListener;
import android.preference.PreferenceActivity;
import android.view.View;

public class ChooseAnAlgorithm extends PreferenceActivity {

    final CharSequence[] items = { "KNN", "WKNN", "MAP", "MMSE", "RBF",
```



```
        "SNAP" };
    final CharSequence[] alg_items = { "K Nearest Neighbor (KNN)
        Algorithm", "Weighted K Nearest Neighbor (WKNN) Algorithm",
            "Probabilistic Maximum A Posteriori (MAP) Algorithm",
                "Probabilistic Minimum Mean Square Error (MMSE) Algorithm"
                ,
            "RBF-Based Positioning Algorithm", "SNAP" };

    private Preference pref;
    private AlertDialog alert;
    private View view;

    /**
     * Called when the activity is first created.
     * */
    @Override
    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        getPreferenceManager().setSharedPreferencesName(FindMe.
            SHARED_PREFS_INDOOR);

        addPreferencesFromResource(R.xml.choose_algorithm);

        pref = (Preference) findPreference("Short_Desc");
        view = pref.getView(getCurrentFocus(), getListView());

        // Customize the description of algorithms
        pref.setOnPreferenceClickListener(new OnPreferenceClickListener()
            {

            @Override
            public boolean onPreferenceClick(Preference preference) {

                // TODO Auto-generated method stub
                AlertDialog.Builder builder = new AlertDialog.Builder(view
                    .getContext());

                builder.setNeutralButton("Cancel", new DialogInterface.
                    OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which)
                        {

                        // Show something if does not exit the app
                        dialog.dismiss();
                    }
                });

                builder.setTitle("Algorithms Short Description");
                builder.setSingleChoiceItems(items, -1, new
                    DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int item)
                        {
                        switch (item) {
                        case 0:
                            popup_msg(
                                    "The KNN algorithm estimates the
```

Airplace Tracker code


```
                        unknown user location as the
                        average of K locations which have
                        the shortest Euclidean distances
                        between the currently observed RSS
                        fingerprint and the respective
                        mean value fingerprints in the
                        radiomap. For more information
                        see:\n\nP. Bahl and V. Padmanabhan
                        , ìRADAR: an in-building RF-based
                        user location and tracking system,
                        î in IEEE International Conference
                        on Computer Communications INFOCOM
                        , vol. 2, 2000, pp. 775ñ784.",
                "K-Nearest Neighbor (KNN)", 0);
            break;
        case 1:
            popup_msg(
                    "The WKNN algorithm is a variant of
                        KNN and estimates the unknown user
                        location as the weighted average
                        of K locations which have the
                        shortest Euclidean distances
                        between the currently observed RSS
                        fingerprint and the respective
                        mean value fingerprints in the
                        radiomap. The weight of each
                        location is set equal to the
                        inverse of the distance. For more
                        information see:\n\nB. Li, J.
                        Salter, A. Dempster, and C. Rizos,
                        ìIndoor positioning techniques
                        based on wireless LAN,î in 1st
                        IEEE International Conference on
                        Wireless Broadband and Ultra
                        Wideband Communications, 2006, pp.
                        13ñ16.",
                    "Weighted K-Nearest Neighbor (WKNN)",
                        0);
            break;
        case 2:
            popup_msg(
                    "The MAP algorithm is a probabilistic
                        approach that estimates the
                        unknown user location by
                        maximizing the conditional
                        probability of each location in
                        the radiomap given the currently
                        observed fingerprint (posterior).
                        For more information see:\n\nM.
                        Youssef and A. Agrawala, ìThe
                        Horus WLAN location determination
                        system,î in 3rd ACM International
                        Conference on Mobile systems,
                        applications, and services, 2005,
                        pp. 205ñ218.",
                    "Maximum A Posteriori (MAP)", 0);
            break;
        case 3:
            popup_msg(
```



```
                    "The MMSE algorithm is a probabilistic
                        approach that estimates the
                        unknown user location by
                        calculating the expected value of
                        the location, which is equivalent
                        to the weighted sum of the
                        loations in the radiomap while the
                        weights are set equal to the
                        conditional probability of each
                        location in the radiomap given the
                        currently observed fingerprint
                        (posterior). For more information
                        see:\n\nT. Roos, P. Myllymaki, H.
                        Tirri, P. Misikangas, and J.
                        Sievanen, ìA probabilistic
                        approach to WLAN user location
                        estimation,î International Journal
                        of Wireless Information Networks,
                        vol. 9, no. 3, pp. 155ñ164, Jul.
                        2002.",
                    "Minimum Mean Square Error (MMSE)", 0)
                        ;
            break;
        case 4:
            popup_msg(
                    "The RBF algorithm is a Neural
                        Networks-based approach that uses
                        the fingerprints in the radiomap
                        as training data to create a
                        mapping form the input RSS space
                        to locations in the area of
                        interest. The RBF network uses the
                        currently observed fingerprint to
                        estimate the unknown user location
                        according to the internal weights.
                        For more information see:\n\nC.
                        Laoudias, P. Kemppi, and C. G.
                        Panayiotou, ìLocalization using
                        radial basis function networks and
                        signal strength fingerprints in
                        WLAN,î in IEEE Global
                        Telecommunications Conference
                        (GLOBECOM), 2009, pp. 1ñ6.",
                    "Radial Basis Functions (RBF)", 0);
            break;
        case 5:
            popup_msg(
                    "The SNAP algorithm is a voting-based
                        approach that uses the notion of
                        signal zones to determine the
                        unknown user location based on the
                        currently observed fingerprint. It
                        provides a trade off between
                        positioning accuracy and
                        computational complexity. For more
                        information see:\n\nC. Laoudias, M
                        . P. Michaelides, C. G. Panayiotou
                        , ìFault Tolerant Fingerprint-
                        based Positioningì, IEEE
```

```
                                        International Conference on
                                        Communications (ICC), 2011.",
                                    "Subtract on Negative Add on Positive
                                        (SNAP)", 0);
                                break;
                            }

                        }
                    });

                    alert = builder.create();

                    alert.show();
                    return true;

                }
            });
        }

        private void popup_msg(String msg, String title, int imageID) {

            AlertDialog.Builder alert_box = new AlertDialog.Builder(this);
            alert_box.setTitle(title);
            alert_box.setMessage(msg);
            alert_box.setIcon(imageID);

            alert_box.setNeutralButton("Hide", new DialogInterface.
                OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    dialog.cancel();
                }
            });

            AlertDialog alert = alert_box.create();
            alert.show();
        }

    }
```

```
/*
* Copyright (c) 2011, KIOS Research Center and Data Management Systems Lab
    ,
* University of Cyprus. All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
    modification,
* are permitted provided that the following conditions are met:
*
*    * Redistributions of source code must retain the above copyright
    notice, this
*      list of conditions and the following disclaimer.
*    * Redistributions in binary form must reproduce the above copyright
    notice,
*      this list of conditions and the following disclaimer in the
    documentation
*      and/or other materials provided with the distribution.
*    * Neither the name of the Sony Ericsson Mobile Communication AB nor
    the names
*      of its contributors may be used to endorse or promote products
    derived from
*      this software without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
    IS" AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
    IMPLIED
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
    DISCLAIMED.
* IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY
    DIRECT,
* INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
    (INCLUDING,
* BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
    USE,
* DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
    THEORY OF
* LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
    NEGLIGENCE
* OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
    IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
*/

package cy.com.airplace;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;
import java.util.Observable;
import java.util.Observer;
import Wifi.SimpleWifiManager;
import Wifi.WifiReceiver;
import android.app.Activity;
import android.app.AlertDialog;
```



```
import android.app.ProgressDialog;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.SharedPreferences;
import android.content.SharedPreferences.OnSharedPreferenceChangeListener;
import android.net.wifi.ScanResult;
import android.net.wifi.WifiManager;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.preference.PreferenceManager;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.ToggleButton;
import cy.com.CalculationModes.OfflineMode;
import cy.com.Downloading.DownloadingSettings;
import cy.com.airplace.R;

public class FindMe extends Activity implements OnClickListener,
    OnSharedPreferenceChangeListener, Observer {

    // Text views to show results
    private TextView title;

    // TextView showing the current scan results
    private TextView scanResults;
    private TextView LocX;
    private TextView LocY;
    private TextView latitudeTextView;
    private TextView longitudeTextView;

    private DecimalFormat myFormatter = new DecimalFormat("###0.00");

    // Button to download radiomap
    private Button btnDownload;

    // Button for positioning
    private Button btnFindMe;

    // Button for position error
    private Button btnPosError;

    // Button for tracking
    private ToggleButton tglBtnTrackMe;

    // Flag to show if there is an ongoing progress
    private Boolean inProgress;

    // The radiomap read
    private RadioMap RM;
```

```
        // The latest scan list of APs
        static ArrayList<LogRecord> LatestScanList;

        // WiFi manager
        private SimpleWifiManager wifi;

        // WiFi Receiver
        private WifiReceiver receiverWifi;

        // Preferences name for indoor and outdoor
        public static final String SHARED_PREFS_INDOOR = "Indoor_Preferences";

        private SharedPreferences Preferences;

        // Path and filename to store radio-map file
        private String folder_path;

        private String imagePath;
        // Image width and height in meters
        private String building_width;
        private String building_height;

        // The filename of downloading radiomap
        private String filename_radiomap_download;

        // Filename of radiomap to use for positioning
        private String filename_radiomap;

        // Flag for offline or online mode
        private boolean isOffline;

        //Flag for Floor mode on or Floor mode off
        private boolean isFloorMode;

        private String algorithmSelection;

        private String test_data = "";

        private ProgressDialog progressDialog;
        private OfflineMode offmode;
        private DownloadingSettings downloadSet;

        private final BooleanObservable trackMe = new BooleanObservable();

        private BooleanObservable withPosErr;
        private FindMeOnBuild fmob;

        //Current floor
        protected static int currentFloor=0;

        //Name of the architecture plan
        private String nameOfArchFile;

        //Name of radio-map (without the roof number)
        private String  nameOfRM="";

        //Used for reading the file from sd-card only when is needed
        private boolean firstTime = true;
```



```
        int currentFloorForFindMe = -100;

        AccelerationServiceReceiver accelerationReceiver=null;

        /**
         * Called when the activity is first created.
         * */
        @Override
        public void onCreate(Bundle savedInstanceState) {

            super.onCreate(savedInstanceState);

            setContentView(R.layout.building_layout);
            this.fmob = new FindMeOnBuild(this);
            this.fmob.setTrackMe(this.trackMe);

            LatestScanList = new ArrayList<LogRecord>();

            title = (TextView) findViewById(R.id.title);
            title.setText("Starting the Application");

            scanResults = (TextView) findViewById(R.id.scanResults);
            scanResults.setText("AP detected: " + 0);
            latitudeTextView = (TextView) findViewById(R.id.latitude);
            longitudeTextView = (TextView) findViewById(R.id.longitude);

            LocX = (TextView) findViewById(R.id.LatTitle);
            LocY = (TextView) findViewById(R.id.LonTitle);

            inProgress = new Boolean(false);

            // Create the Radio map
            RM = new RadioMap();

            // Button to download indoor radio map
            btnDownload = (Button) findViewById(R.id.downloadRadioMap);
            btnDownload.setOnClickListener(this);


            // Button to find user on map
            btnFindMe = (Button) findViewById(R.id.find_me);
            btnFindMe.setOnClickListener(this);

            btnPosError = (Button) findViewById(R.id.pos_error);
            btnPosError.setOnClickListener(this);

            btnPosError.setVisibility(View.INVISIBLE);

            tglBtnTrackMe = (ToggleButton) findViewById(R.id.trackme);
            tglBtnTrackMe.setOnClickListener(this);

            // WiFi manager to manage scans
            wifi = new SimpleWifiManager(getApplicationContext());
            wifi.setScanResultsTextView(scanResults);

            // Create new receiver to get broadcasts
            receiverWifi = new SimpleWifiReceiver();

            // Configure preferences
```

```
            Preferences = PreferenceManager.getDefaultSharedPreferences(this);

            PreferenceManager.setDefaultValues(this, SHARED_PREFS_INDOOR,
                MODE_PRIVATE, R.xml.preferences, true);
            Preferences = FindMe.this.getSharedPreferences(SHARED_PREFS_INDOOR
                , MODE_PRIVATE);
            LocX.setText("X:  ");
            LocY.setText("Y: ");

            Preferences.registerOnSharedPreferenceChangeListener(this);
            onSharedPreferenceChanged(Preferences, "modeIn");
            onSharedPreferenceChanged(Preferences, "image_custom");

            //Preserve the floor mode
            this.onSharedPreferenceChanged(Preferences, "modeFloor");

        }

        /**
         * The WifiReceiver is responsible to Receive Access Points results
         * */
        public class SimpleWifiReceiver extends WifiReceiver {

            public void onReceive(Context c, Intent intent) {

                try {
                    if (intent == null || c == null || intent.getAction() ==
                        null)
                        return;

                    String action = intent.getAction();

                    if (!action.equals(WifiManager.
                        SCAN_RESULTS_AVAILABLE_ACTION))
                        return;

                    List<ScanResult> wifiList = wifi.getScanResults();
                    scanResults.setText("AP detected: " + wifiList.size());

                    // Set in progress (true)
                    synchronized (inProgress) {
                        if (inProgress == true)
                            return;
                        inProgress = true;
                    }

                    LatestScanList.clear();
                    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++) {
                            lr = new LogRecord(wifiList.get(i).BSSID, wifiList
                                .get(i).level);
                            LatestScanList.add(lr);
                        }
                    }

                    // Unset in progress (false)
```



```
                    synchronized (inProgress) {
                        inProgress = false;
                    }

                    if (trackMe.get()) {
                        if (!FindMe_Method()) {
                            tglBtnTrackMe.setChecked(false);
                            trackMe.setBoolean(false);
                            trackMe.notifyObservers();
                        }
                    }

                } catch (RuntimeException e) {
                    return;
                }

            }

        }

        /**
         * Draw the menu
         *
         * @param menu
         *            the menu to add items for Indoor RSS
         * */
        public boolean onCreateOptionsMenu(Menu menu) {
            MenuInflater inflater = getMenuInflater();
            inflater.inflate(R.menu.menu, menu);
            return true;
        }

        /**
         * Handles menu choices
         *
         * @param item
         *            the item clicked from menu
         * */
        public boolean onOptionsItemSelected(MenuItem item) {

            switch (item.getItemId()) {
            // Launch Preferences
            case R.id.Preferences:
                Intent prefs = new Intent(this, Preferences.class);
                startActivity(prefs);
                return true;
                // Launch Preferences to choose one of the algorithms
                    implemented
            case R.id.Choose_Algorithm:
                Intent algorithm_prefs = new Intent(this, ChooseAnAlgorithm.
                    class);
                startActivity(algorithm_prefs);
                return true;
                // Exit application
            case R.id.Exit_App:
                this.finish();
                return true;
            }
            return false;
```

```
        }

        @Override
        public void onSharedPreferenceChanged(SharedPreferences prefs, String
            key) {

            if (key == null)
                return;

            if (key.equals("modeIn"))
            {
                isOffline = Preferences.getBoolean("modeIn", false);

                if (isOffline)
                {
                    title.setText("Offline Mode");

                    latitudeTextView.setVisibility(View.INVISIBLE);
                    longitudeTextView.setVisibility(View.INVISIBLE);
                    LocX.setVisibility(View.INVISIBLE);
                    LocY.setVisibility(View.INVISIBLE);

                    tglBtnTrackMe.setVisibility(View.INVISIBLE);

                    tglBtnTrackMe.setChecked(false);
                    this.trackMe.setBoolean(false);
                    this.trackMe.notifyObservers();

                    // Disables the WiFi if is in Offline Mode
                    wifi.stopScan(receiverWifi);

                    btnFindMe.setText("Statistics\nIndoor");

                }
                else
                {

                    title.setText("Online Mode");

                    latitudeTextView.setVisibility(View.VISIBLE);
                    longitudeTextView.setVisibility(View.VISIBLE);
                    LocX.setVisibility(View.VISIBLE);
                    LocY.setVisibility(View.VISIBLE);

                    tglBtnTrackMe.setVisibility(View.VISIBLE);

                    // Enables the WiFi if is in Online Mode
                    wifi.startScan(receiverWifi, "2000");

                    btnFindMe.setText("Find Me\nIndoor");

                }
            }
            else if (key.equals("image_custom"))
            {


                imagePath = (String) Preferences.getString("image_custom", "")
                    .trim();
```



```
                //Take the file name
                filename_radiomap = (String) Preferences.getString
                    ("radiomap_file", "").trim();
                isolateTheNameOfRM(filename_radiomap);


                if (imagePath.equals("")) {
                    return;
                }

                String[] temp=imagePath.split("/");
                /**
                 * Maybe problem here with parsing the string.
                 * User must choose a file that in the end has the number of
                     the floor
                 */


                if(temp[3].contains("-"))
                {
                    nameOfArchFile=temp[3].substring(0, temp[3].length()-6);
                }
                else
                {
                    nameOfArchFile=temp[3].substring(0, temp[3].length()-5);
                }
                Integer tmpCurFloor = getTheNoOfFloor(temp[3], temp[3].
                    contains("-"));
                if(tmpCurFloor!=null)
                {
                    currentFloor=tmpCurFloor;
                }

                building_width = null;
                building_height = null;

                if (!ReadWidthHeigthFromFile(new File(imagePath))) {
                    popup_msg("Corrupted image configuration file.\nPlease set
                        a different floor plan or previous floor plan will be
                        used if available.",
                            "Error", R.drawable.error);
                    imagePath = null;
                    building_width = null;
                    building_height = null;
                    return;
                }

                if (!fmob.setFloorPlan(imagePath, building_width,
                    building_height)) {
                    imagePath = null;
                    building_width = null;
                    building_height = null;
                } else {
                    latitudeTextView.setText("");
                    longitudeTextView.setText("");
                }
            }
```

```
            else if (key.equals("change_floor_image_custom"))
            {
                imagePath="/mnt/sdcard/"+nameOfArchFile+currentFloor+".jpg";
                System.out.println("Image Path: "+imagePath);

                if (imagePath.equals("")) {
                    return;
                }

                building_width = null;
                building_height = null;

                if (!ReadWidthHeigthFromFile(new File(imagePath))) {
                    popup_msg("Corrupted image configuration file.\nPlease set
                        a different floor plan or previous floor plan will be
                        used if available.",
                            "Error", R.drawable.error);
                    imagePath = null;
                    building_width = null;
                    building_height = null;
                    return;
                }

                if (!fmob.setFloorPlan(imagePath, building_width,
                    building_height)) {
                    imagePath = null;
                    building_width = null;
                    building_height = null;
                } else {
                    latitudeTextView.setText("");
                    longitudeTextView.setText("");
                }
            }
            else if(key.equals("modeFloor"))
            {
                isFloorMode = Preferences.getBoolean("modeFloor", false);


                if(isFloorMode)
                {
                    popup_msg("Be sure that all the radio-maps, test-files and
                        floor plans have the right settings",
                            "Warning of floor mode!", R.drawable.info);

                    //Must take the currentFloor to inform the FloorService
                    //Start service if is not started

                    IntentFilter movementFilter;
                    movementFilter = new IntentFilter(FloorService.
                        MOVEMENT_UPDATE);
                    accelerationReceiver = new AccelerationServiceReceiver();
                    registerReceiver(accelerationReceiver, movementFilter);

                    Intent intent = new Intent("cy.com.airplace.FloorService")
                        ;
                    this.startService(intent);
                }
                else
                {
```



```
                    Intent intent = new Intent("cy.com.airplace.FloorService")
                        ;
                    this.stopService(intent);
                }
            }
        }

        /**
         * Find the name of the radio-map from the path
         *
         * @param nameOfRadioMap
         */
        private void isolateTheNameOfRM(String nameOfRadioMap)
        {
            nameOfRM="";
            String[] temp = nameOfRadioMap.split("/");

            //Check for (-) and take the last component that has minus
            String[] temp2 = temp[temp.length-1].split("-");
            String specificName = temp2[temp2.length-1];
            specificName=specificName.replace(".txt", "");

            for(int i=0; i<specificName.length(); ++i)
            {
                if(Character.isDigit(specificName.charAt(i)))
                {
                    if(specificName.contains("-"))
                    {
                        specificName="-"+specificName.replaceAll("[0-9]*$","")
                            ;
                    }
                    else
                    {
                        specificName=""+specificName.replaceAll("[0-9]*$","");
                    }
                }
            }

            //Back to right path
            for(int i=0; i<temp.length-1; ++i)
            {
                nameOfRM+=temp[i]+"/";
            }
            for(int i=0; i<temp2.length-1; ++i)
            {
                nameOfRM+=temp2[i]+"-";
            }
            nameOfRM+=specificName;
        }

        /**
         * Find the numerical value in the string of a file name
         * This method is used to find the current no of floor
         *
         * @param fileName
         * @param minus
         * @return the no of the floor
         */
        private Integer getTheNoOfFloor(String fileName, Boolean minus)
```

```
        {
            String tempFloor;
            for(int i=0; i<fileName.length(); ++i)
            {
                if(Character.isDigit(fileName.charAt(i)))
                {
                    if(minus)
                    {
                        tempFloor="-"+fileName.charAt(i);
                    }
                    else
                    {
                        tempFloor=""+fileName.charAt(i);
                    }
                    return Integer.valueOf(tempFloor);
                }
            }

            return null;
        }

        /**
         *
         * Class that takes the new floor if and only if the FloorService
             indicates a new floor
         *
         */

        public class AccelerationServiceReceiver extends BroadcastReceiver
        {
          @Override
            public void onReceive(Context context, Intent intent)//this method
                receives broadcast messages. Be sure to modify AndroidManifest
                .xml file in order to enable message receiving
            {
                    //Take the new floor that must change
                    double floor = intent.getDoubleExtra(FloorService.MSG_SEND
                        , 0);

                    //Assign the new floor
                    currentFloor=(int) floor;

                    //Change the architecture map
                    onSharedPreferenceChanged(null,"change_floor_image_custom"
                        );

                    //Change the radio-map, parameters file etc.
                    toastPrint("Floor changed to "+currentFloor, Toast.
                        LENGTH_LONG);
            }
        }

        /**
         * Control the clicks on buttons
         *
         * @param v
         *            the view clicked
         * */
        @Override
```



```
        public void onClick(View v) {
            switch (v.getId()) {
            // Download new radiomap
            case R.id.downloadRadioMap:
                Download();
                break;
            // Positioning
            case R.id.find_me:
                tglBtnTrackMe.setChecked(false);
                this.trackMe.setBoolean(false);
                this.trackMe.notifyObservers();
                FindMe_Method();
                break;
            // Tracking
            case R.id.trackme:
                if (tglBtnTrackMe.isChecked()) {
                    this.trackMe.setBoolean(true);
                    this.trackMe.notifyObservers();
                } else {
                    this.trackMe.setBoolean(false);
                    this.trackMe.notifyObservers();
                }
                break;
            }

        }

        /**
         * Downloads new radiomap
         * */
        private void Download() {

            String serverAddress = null;
            String portNumber = null;

            serverAddress = Preferences.getString("serverIP", "").trim();
            portNumber = Preferences.getString("serverPORT", "").trim();
            folder_path = (String) Preferences.getString("folder_browser", "")
                .trim();
            filename_radiomap_download = (String) Preferences.getString
                ("radiomap_file_download", "").trim();

            // Check IP/Port of Server
            if (serverAddress.equals("")) {
                popup_msg("Unable to start connection with the server\n"
                        + "Go to Menu::Preferences::Radiomap Settings::
                            RadioMap Download Settings::Connection Settings::
                            Server Address", "User Error",
                        R.drawable.error);
                return;
            } else if (portNumber.equals("")) {
                popup_msg("Unable to start connection with the server\n"
                        + "Go to Menu::Preferences::Radiomap Settings::
                            RadioMap Download Settings::Connection Settings::
                            Port number", "User Error",
                        R.drawable.error);
                return;
            }
```

```
            // Check folder path to store radio map
            if (folder_path.equals("")) {
                popup_msg("Folder path is not specified\n"
                        + "Go to Menu::Preferences::Radiomap Settings::
                            RadioMap Download Settings::Downloading Settings::
                            Radio Map Folder", "User Error",
                        R.drawable.error);
                return;
            } else if ((!(new File(folder_path).canWrite()))) {
                popup_msg("Folder path is not writable\n"
                        + "Go to Menu::Preferences::Radiomap Settings::
                            RadioMap Download Settings::Downloading Settings::
                            Radio Map Folder", "User Error",
                        R.drawable.error);
                return;
            }

            // Check radiomap filename
            if (filename_radiomap_download.equals("")) {
                popup_msg("Filename of radio map not specified\n"
                        + "Go to Menu::Preferences::Radiomap Settings::
                            RadioMap Download Settings::Downloading Settings::
                            Radio Map Filename",
                        "User Error", R.drawable.error);
                return;
            }

            toastPrint("Trying to connect to " + serverAddress + ":" +
                portNumber + "...", Toast.LENGTH_LONG);

            // Set in progress (true)
            synchronized (inProgress) {
                if (inProgress == true)
                    return;
                inProgress = true;
            }

            progressDialog = ProgressDialog.show(FindMe.this, "", "Downloading
                . Please wait...", true, false);

            downloadSet = new DownloadingSettings(serverAddress, portNumber,
                folder_path, filename_radiomap_download, handler);

            downloadSet.start();

            // Unset in progress (false)
            synchronized (inProgress) {
                inProgress = false;
            }

        }

        /**
         * Starts the appropriate positioning algorithm
         * */
        private boolean FindMe_Method() {

            /**
             * Change radio-map dynamically I have just to put the current
```



```
                 floor
             * So, concatenate filename_radiomap and test_data and put just
                 the new number in the end
             */

            algorithmSelection = (String) Preferences.getString("Algorithms",
                "1").trim();
            test_data = Preferences.getString("test_data_file", "").trim();

            if(isFloorMode)
            {
                filename_radiomap = (String) Preferences.getString
                    ("radiomap_file", "").trim();
                isolateTheNameOfRM(filename_radiomap);
                if(!nameOfRM.equals(""))
                {
                    filename_radiomap=nameOfRM+currentFloor+".txt";
                }
            }
            else
            {
                filename_radiomap = (String) Preferences.getString
                    ("radiomap_file", "").trim();
                System.out.println("Name of the radio map: "+filename_radiomap
                    );
                isOffline = Preferences.getBoolean("modeIn", false);
                System.out.println("Name of the test data: "+filename_radiomap
                    );
            }

            if (!isOffline) {

                if (!fmob.okBuildingSettings()) {
                    popup_msg("Building floor plan not specified\nGo to Menu::
                        Preferences::Building Settings::Floor Plan", "Error",
                        R.drawable.error);
                    return false;
                }
            }

            // Check that radiomap file is readable
            if (filename_radiomap.equals("")) {
                popup_msg("Radiomap file not specified\nGo to Menu::
                    Preferences::Radiomap Settings::Radiomap File", "User
                    Error", R.drawable.error);
                return false;

            } else if ((!(new File(filename_radiomap).canRead()))) {
                popup_msg("Radiomap file is not readable\nGo to Menu::
                    Preferences::Radiomap Settings::Radiomap File", "User
                    Error", R.drawable.error);
                return false;
            }

            // Check algorithm selection
            if (algorithmSelection.equals("") || Integer.parseInt
                (algorithmSelection) < 1 || Integer.parseInt
                (algorithmSelection) > 6) {
                popup_msg("Unable to find the location\nSpecify Algorithm",
```

```
                "User Error", R.drawable.error);
            return false;
        }

        if (isOffline && test_data.equals("")) {
            popup_msg("Test Data file not specified\nGo to Menu::
                Preferences::Mode::Test Data File", "User Error", R.
                drawable.error);
            return false;
        }

        // Set in progress (true)
        synchronized (inProgress) {
            if (inProgress == true)
                return false;
            inProgress = true;
        }


        if(firstTime || (currentFloor!=currentFloorForFindMe))
        {
            //New RM to save the new results inside
            RM = new RadioMap();

            // Error reading Radio Map
            if (!RM.ConstructRadioMap(new File(filename_radiomap))) {
                popup_msg("Error while reading radio map.\nDownload new
                    Radio Map and try again", "User Error", R.drawable.
                    error);

                // Unset in progress (false)
                synchronized (inProgress) {
                    inProgress = false;
                }

                return false;
            }
            currentFloorForFindMe=currentFloor;
            firstTime=false;
        }


        /**
         * If is true then go offline, otherwise continue online mode
         */
        if (isOffline) {
            goOffline(RM, new File(test_data), Integer.parseInt
                (algorithmSelection));
        } else {

            if (LatestScanList.isEmpty()) {
                popup_msg("No Access Point Received.\nWait for a scan
                    first and try again.", "Warning", R.drawable.warning);

                // Unset in progress (false)
                synchronized (inProgress) {
                    inProgress = false;
                }
```



```
                return false;
            }

            if (!calculatePosition(Integer.parseInt(algorithmSelection)))
                {
                popup_msg("Can't find location. Check that radio map file
                    refers to the same area.", "Error", R.drawable.error);

                // Unset in progress (false)
                synchronized (inProgress) {
                    inProgress = false;
                }

                return false;
            }

        }

        // Unset in progress (false)
        synchronized (inProgress) {
            inProgress = false;
        }

        return true;

    }


    /**
     *
     * @param imageFile
     *
     * @return true if all was ok, else return false
     */
    private boolean ReadWidthHeigthFromFile(File imageFile) {

        BufferedReader reader = null;
        FileReader fr = null;
        String line = null;
        int i = 0;

        try {
            String conf = imageFile.getAbsolutePath().replace(".jpg", ".
                config").replace(".JPG", ".config");
            fr = new FileReader(conf);
        } catch (Exception e) {
            return false;
        }

        reader = new BufferedReader(fr);

        try {

            while ((line = reader.readLine()) != null) {

                /* Ignore the labels */
                if (line.startsWith("#") || line.trim().equals("")) {
                    continue;
                }
```

```
                line = line.replace(": ", " ");
                /* Split fields */
                String[] temp = line.split(" ");

                if (temp.length != 2) {
                    return false;
                }

                if (i == 0)
                    building_width = temp[1];
                else
                    building_height = temp[1];

                ++i;

            }
        } catch (Exception e) {
            return false;
        }

        if (!checkBuildingDimensions(building_width, building_height)) {
            return false;
        }

        return true;
    }

    /**
     * Checks the building width and height
     *
     * @param building_width
     *            the width of the building to check
     *
     * @param building_height
     *            the height of the building to check
     *
     * @return true if the width and height are ok, otherwise false
     *
     * */
    private boolean checkBuildingDimensions(String building_width, String
        building_height) {

        if (building_height.equals("")) {
            popup_msg("Corrupted image configuration file", "User Error",
                R.drawable.error);
            return false;
        }

        try {
            Float.parseFloat(building_height);
        } catch (Exception e) {
            popup_msg("Error Building Height: " + e.getMessage(), "User
                Error", R.drawable.error);
            return false;
        }

        if (building_width.equals("")) {
            popup_msg("Corrupted image configuration file", "User Error",
```



```
                    R.drawable.error);
            return false;
        }

        try {
            Float.parseFloat(building_width);
        } catch (Exception e) {
            popup_msg("Error Building Width: " + e.getMessage(), "User
                Error", R.drawable.error);
            return false;
        }

        return true;
    }

    private boolean calculatePosition(int choice) {

        ArrayList<WeightRecord> WeightsList = Algorithms.readWeights(RM.
            getRadiomapMean_File());
        String calculatedLocation = Algorithms.ProcessingAlgorithms
            (LatestScanList, RM, choice, WeightsList);

        if (calculatedLocation == null) {
            return false;
        }

        String[] x_y = calculatedLocation.split(" ");
        if (x_y.length != 2) {
            return false;
        }

        try {
            if (x_y[0].equals("NaN") || x_y[1].equals("NaN"))
                return false;
            latitudeTextView.setText(myFormatter.format(Float.parseFloat
                (x_y[0])) + "m");
            longitudeTextView.setText(myFormatter.format(Float.parseFloat
                (x_y[1])) + "m");
        } catch (Exception e) {
            return false;
        }

        fmob.setLocationOnFloorPlan(calculatedLocation);

        return true;

    }

    private void goOffline(RadioMap RM, File inFile, int
        algorithm_selection) {
        progressDialog = ProgressDialog.show(FindMe.this, "", "Calculating
            . Please wait...\n0%", true, false);

        offmode = new OfflineMode(RM, inFile, algorithm_selection, handler
            );

        offmode.start();
    }
```

```
        final Handler handler = new Handler() {
            public void handleMessage(Message msg) {

                if (msg.what >= 0)
                    progressDialog.setMessage("Calculating. Please wait...\n"
                        + msg.what + "%");

                switch (msg.what) {
                case -1:
                    progressDialog.dismiss();
                    if (offmode.getErrMsg() != null)
                        popup_msg(offmode.getErrMsg(), "Error", R.drawable.
                            error);
                    else {
                        popup_statistics("Average Positioning Error: " +
                            myFormatter.format(offmode.getAverage_pos_err()) +
                            "m\nAverage Execution Time: "
                                + myFormatter.format(offmode.
                                    getAverage_exe_time()) + "ms", "Info", R.
                                    drawable.info);
                    }
                    break;

                case -2:
                    progressDialog.dismiss();
                    if (downloadSet.getErrMsg() != null)
                        popup_msg(downloadSet.getErrMsg(), "Error", R.drawable
                            .error);
                    else
                        popup_msg("Radio Map, RBF Weights and Parameters
                            Successfully Downloaded and Stored, on "
                                + Calendar.getInstance().getTime().toString(),
                                    "Info", R.drawable.info);
                    break;
                }
            }
        };

        public String showStatistics() {
            /**
             * Estimate the cpu consumption
             */
            Power power;
            String myfile = "sdcard/";
            String file = PowerTutor.getLastFilePowerTutor("sdcard/");

            if (file == null) {
                return "No Power Tutor log file found";
            }

            power = PowerTutor.getPower(myfile + file);

            if (power != null)
                return "Total Power (CPU): " + myFormatter.format(power.CPU) +
                    "mW";
            else
                return "Could not calculate Total Power (CPU)";
        }
```



```
        private void popup_msg(String msg, String title, int imageID) {

            AlertDialog.Builder alert_box = new AlertDialog.Builder(this);
            alert_box.setTitle(title);
            alert_box.setMessage(msg);
            alert_box.setIcon(imageID);

            alert_box.setNeutralButton("Hide", new DialogInterface.
                OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    dialog.cancel();
                }
            });

            AlertDialog alert = alert_box.create();
            alert.show();
        }

        private void popup_statistics(final String msg, final String title,
            final int imageID) {

            final AlertDialog.Builder alert_box = new AlertDialog.Builder(this
                );
            alert_box.setTitle(title);
            alert_box.setMessage(msg);
            alert_box.setIcon(imageID);

            alert_box.setNeutralButton("Hide", new DialogInterface.
                OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    dialog.cancel();
                }
            }).setPositiveButton("Show Power", new DialogInterface.
                OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int id) {
                    popup_msg(msg + "\n" + showStatistics(), title, imageID);
                }
            });

            AlertDialog alert = alert_box.create();
            alert.show();
        }

        /**
         * Method used to print pop up message to user
         * */
        private void toastPrint(String textMSG, int duration) {
            Toast.makeText(this, textMSG, duration).show();
        }

        /**
         * Back button pressed to exit program
         */
        @Override
        public void onBackPressed() {
```

```
            AlertDialog.Builder builder = new AlertDialog.Builder(this);

            builder.setMessage("Are you sure you want to exit?").setCancelable
                (false).setPositiveButton("Yes", new DialogInterface.
                OnClickListener() {
                public void onClick(DialogInterface dialog, int id) {
                    FindMe.this.finish();
                }
            }).setNegativeButton("No", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int id) {
                    dialog.cancel();
                }
            });
            AlertDialog alert = builder.create();
            alert.show();

        }

        @Override
        public void onResume() {
            super.onResume();
        }

        @Override
        public void onPause() {
            super.onPause();
        }

        @Override
        protected void onDestroy() {

            wifi.stopScan(receiverWifi);
            Intent intent = new Intent("cy.com.airplace.FloorService");
            stopService(intent);
            super.onDestroy();
        }

        @Override
        public void update(Observable observable, Object data) {
            if (this.withPosErr != null && this.withPosErr.get())
                btnPosError.setVisibility(View.VISIBLE);
            else
                btnPosError.setVisibility(View.INVISIBLE);
        }
    }
```

Airplace Tracker code


```
/*
* Copyright (c) 2011, KIOS Research Center and Data Management Systems Lab
    ,
* University of Cyprus. All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
    modification,
* are permitted provided that the following conditions are met:
*
*    * Redistributions of source code must retain the above copyright
    notice, this
*       list of conditions and the following disclaimer.
*    * Redistributions in binary form must reproduce the above copyright
    notice,
*       this list of conditions and the following disclaimer in the
    documentation
*       and/or other materials provided with the distribution.
*    * Neither the name of the Sony Ericsson Mobile Communication AB nor
    the names
*       of its contributors may be used to endorse or promote products
    derived from
*       this software without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
    IS" AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
    IMPLIED
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
    DISCLAIMED.
* IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY
    DIRECT,
* INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
    (INCLUDING,
* BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
    USE,
* DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
    THEORY OF
* LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
    NEGLIGENCE
* OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
    IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
*/

package cy.com.airplace;

import java.util.Observable;
import java.util.Observer;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import cy.com.airplace.R;
import cy.com.zoom.ClickPoint;
import cy.com.zoom.DynamicZoomControl;
import cy.com.zoom.ImageZoomView;
import cy.com.zoom.LongPressZoomListener;

public class FindMeOnBuild implements Observer {

    private final ClickPoint curClick = new ClickPoint(-1, -1);
```



```
    private String imagePath;
    private String building_width;
    private String building_height;

    /** Image zoom view */
    private ImageZoomView mZoomView;

    /** Zoom control */
    private DynamicZoomControl mZoomControl;

    /** Decoded bitmap image */
    private Bitmap mBitmap;

    /** On touch listener for zoom view */
    private LongPressZoomListener mZoomListener;

    private BooleanObservable trackMe;

    public FindMeOnBuild(FindMe fm) {

        // Set Zooming and Panning Settings
        mZoomControl = new DynamicZoomControl();
        mZoomListener = new LongPressZoomListener(fm.getApplicationContext
            ());
        mZoomListener.setZoomControl(mZoomControl);

        mZoomView = (ImageZoomView) fm.findViewById(R.id.zoomview);
        mZoomView.setZoomState(mZoomControl.getZoomState());
        mZoomView.setOnTouchListener(mZoomListener);
        mZoomView.setCurClick(curClick);

        mZoomControl.setAspectQuotient(mZoomView.getAspectQuotient());

    }

    public void setTrackMe(BooleanObservable trackMe) {
        mZoomView.setTrackMe(trackMe);

        if (this.trackMe != null) {
            this.trackMe.deleteObserver(this);
        }
        this.trackMe = trackMe;
        this.trackMe.addObserver(this);
    }

    public boolean setFloorPlan(String imagePath, String building_width,
        String building_height) {

        if (imagePath == null || imagePath.equals(""))
            return false;

        if (this.imagePath != null && this.imagePath.equals(imagePath))
            return true;

        Bitmap tempBitmap = BitmapFactory.decodeFile(imagePath);

        if (tempBitmap != null) {

            if (mBitmap != null)
```

```
                    mBitmap.recycle();

                System.gc();

                this.imagePath = imagePath;
                this.mBitmap = tempBitmap;
                this.building_width = building_width;
                this.building_height = building_height;
                mZoomView.setImage(this.mBitmap);
                resetZoomState();
                resetLocation();
            } else
                return false;

            return true;
        }

        public boolean setLocationOnFloorPlan(String Geolocation) {

            if (Geolocation == null || mBitmap == null || building_width ==
                null || building_height == null)
                return false;

            String coordinates[] = Geolocation.replace(", ", " ").split(" ");
            float x, y;
            float bitmapWidth;
            float bitmapHeight;

            try {
                x = Float.parseFloat(coordinates[0]);
                y = Float.parseFloat(coordinates[1]);
                bitmapWidth = Float.parseFloat(building_width);
                bitmapHeight = Float.parseFloat(building_height);
            } catch (Exception e) {
                return false;
            }

            // Clear all overlays if it is not tracking
            if (!trackMe.get())
                mZoomView.clearPoints();

            int x_pixels = (int) ((x * mBitmap.getWidth()) / bitmapWidth);
            int y_pixels = (int) ((y * mBitmap.getHeight()) / bitmapHeight);
            curClick.setClickPoint(x_pixels, y_pixels);
            curClick.notifyObservers();

            return true;
        }

        protected void onDestroy() {

            if (mBitmap != null)
                mBitmap.recycle();
            if (mZoomView != null)
                mZoomView.setOnTouchListener(null);
            if (mZoomControl != null)
                mZoomControl.getZoomState().deleteObservers();
        }
```



```
        /**
         * Reset zoom state and notify observers
         */
        private void resetZoomState() {
            mZoomControl.getZoomState().setPanX(0.5f);
            mZoomControl.getZoomState().setPanY(0.5f);
            mZoomControl.getZoomState().setZoom(1f);
            mZoomControl.getZoomState().notifyObservers();
            mZoomView.clearPoints();
        }

        private void resetLocation() {
            curClick.setClickPoint(-1, -1);
            curClick.notifyObservers();
        }

        @Override
        public void update(Observable observable, Object data) {
            // Clear all overlays if it is not tracking
            if (!trackMe.get()) {
                mZoomView.clearPoints();
            }

        }

        public boolean okBuildingSettings() {
            return this.mBitmap != null && this.building_height != null &&
                this.building_width != null;
        }

    }
```

Airplace Tracker code


```
package cy.com.airplace;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;

import cy.com.FloorMode.FloorProcedure;
import cy.com.FloorMode.ParseRMHelp;
import cy.com.FloorMode.RMHelp;
import android.app.Service;
import android.content.Intent;
import android.os.Environment;
import android.os.Handler;
import android.os.IBinder;

public class FloorService extends Service
{
    private final int PERIODIC_EVENT_TIMEOUT = 5000;
    public static final String MOVEMENT_UPDATE = "cy.com.airplace.
        FloorService.action.MOVEMENT_UPDATE";
    public static final String MSG_SEND = "cy.com.airplace.FloorService.
        MSG_SEND";
    private Handler periodicEvent;
    private int floor = 0;

    @Override
    public IBinder onBind(Intent intent)
    {
        return null;
    }

    @Override
    public void onCreate()
    {

        periodicEvent = new Handler();
        periodicEvent.postDelayed(doPeriodicTask, PERIODIC_EVENT_TIMEOUT);
        this.floor=FindMe.currentFloor;
    }


    private Runnable doPeriodicTask = new Runnable()
    {
        public void run()
        {
            /**
             * Find first where the user Is
             */
            //Array max contains 1st the no of floor 2nd the score of
                floor
            int[] max = { FindMe.currentFloor, -800 };

            //Stores the current floors that every time compares

            ArrayList<Integer> tempFloors = new ArrayList<Integer>();

            String resultOfFloors = "";

            // Take the names of all .floor files and take the content of
```



```
                all
            // the .floor files (sort)
            ArrayList<ArrayList<RMHelp>> arrayRMHelp = readTheFloorFiles
                (tempFloors);
            int[][] floorsScore = new int[tempFloors.size()][2];
            for(int i=0; i<tempFloors.size(); ++i)
            {
                floorsScore[i][0]=tempFloors.get(i);
            }

            ArrayList<ArrayList<RMHelp>> arrayRMHelpTemp = new ArrayList<
                ArrayList<RMHelp>>();

            for (int i = 0; i < arrayRMHelp.size() - 1; ++i)
            {
                arrayRMHelpTemp.add(arrayRMHelp.get(i));
                arrayRMHelpTemp.add(arrayRMHelp.get(i + 1));
                FloorProcedure findFloor = new FloorProcedure
                    (arrayRMHelpTemp);
                if (FindMe.LatestScanList.isEmpty())
                {
                    return;
                }

                // Take the score of the two floors that are in
                    competition and
                // the max

                resultOfFloors = findFloor.findTheMAC(FindMe.
                    LatestScanList);
                // Parse the result
                String[] tempStr = resultOfFloors.split(" ");


                //Assign the score to each floor
                String[] temp = tempStr[0].split(",");
                floorsScore[i][1]=Integer.valueOf(temp[1]);

                temp = tempStr[1].split(",");
                floorsScore[i+1][1]=Integer.valueOf(temp[1]);

                arrayRMHelpTemp.clear();


            }


            /**
             * Find the highest score
             */
            for(int i=0; i<tempFloors.size(); ++i)
            {
                if(floorsScore[i][1]>max[1])
                {

                    max[0]=floorsScore[i][0];
                    max[1]=floorsScore[i][1];
                }
            }
```

Airplace Tracker code


```
            //Inform the main thread if and ONLY if the view must change!
            if(max[0]!=floor && !FindMe.LatestScanList.isEmpty())
            {
                informFindMeForChanges((double)max[0]);
                floor=FindMe.currentFloor=max[0];
            }
            periodicEvent.postDelayed(doPeriodicTask,
                PERIODIC_EVENT_TIMEOUT);
        }
    };

    /**
     * Inform the FindMe class if the floor is changed
     *
     * @param newValue
     *                  new floor
     */

    private void informFindMeForChanges(double newValue)
    {
        Intent intent = new Intent(MOVEMENT_UPDATE);
        intent.putExtra(MSG_SEND,newValue);
        sendBroadcast(intent);
    }

    /**
     * Method that reads all the .floor files.
     * Files that will help the device to decide in which floor is the
         user
     *
     * @param tempFloors
     *                      the no of all the floors
     * @return floorFiles
     *                      a list of the name of the files
     */
    private ArrayList<ArrayList<RMHelp>> readTheFloorFiles(ArrayList<
        Integer> tempFloors)
    {
        ArrayList<ArrayList<RMHelp>> rmh = new ArrayList<ArrayList<RMHelp>
            >();
        File sdCardDir = Environment.getExternalStorageDirectory();
        ArrayList<String> floorFiles = new ArrayList<String>();
        String files;
        String floor;

        File[] listOfFiles = sdCardDir.listFiles();
        for (int i = 0; i < listOfFiles.length; i++)
        {

            if (listOfFiles[i].isFile())
            {
                files = listOfFiles[i].getName();
                if (files.endsWith(".floor"))
                {
                    floor=files.substring(files.length() - 8, files.length
                        () - 6);

                    if(floor.contains("-"))
```



```
                    {
                        tempFloors.add(Integer.valueOf("-"+floor.substring
                            (1)));
                    }
                    else
                    {
                        tempFloors.add(Integer.valueOf(floor.substring(1))
                            );
                    }
                    try
                    {
                        ParseRMHelp newRMFloor = new ParseRMHelp(sdCardDir
                            .getAbsolutePath()+"/"+files);
                        rmh.add(newRMFloor.getArrayRMH());
                    } catch (IOException e)
                    {
                        e.printStackTrace();
                    }
                    floorFiles.add(files);
                }
            }
        }
        return rmh;
    }


    @Override
    public void onDestroy()
    {
        periodicEvent.removeCallbacks(doPeriodicTask);
        super.onDestroy();
    }

    @Override
    public void onStart(Intent intent, int startid)
    {
    }

}
```

Airplace Tracker code


```
/*
* Copyright (c) 2011, KIOS Research Center and Data Management Systems Lab
    ,
* University of Cyprus. All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
    modification,
* are permitted provided that the following conditions are met:
*
*    * Redistributions of source code must retain the above copyright
    notice, this
*      list of conditions and the following disclaimer.
*    * Redistributions in binary form must reproduce the above copyright
    notice,
*      this list of conditions and the following disclaimer in the
    documentation
*      and/or other materials provided with the distribution.
*    * Neither the name of the Sony Ericsson Mobile Communication AB nor
    the names
*      of its contributors may be used to endorse or promote products
    derived from
*      this software without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
    IS" AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
    IMPLIED
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
    DISCLAIMED.
* IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY
    DIRECT,
* INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
    (INCLUDING,
* BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
    USE,
* DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
    THEORY OF
* LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
    NEGLIGENCE
* OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
    IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
*/

package cy.com.airplace;

public class LocDistance {
    private double distance;
    private String location;

    public LocDistance(double distance, String location) {
        this.distance = distance;
        this.location = location;
    }

    public double getDistance() {
        return distance;
    }
```



```
    public String getLocation() {
        return location;
    }
}
```

Airplace Tracker code


```
/*
* Copyright (c) 2011, KIOS Research Center and Data Management Systems Lab
    ,
* University of Cyprus. All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
    modification,
* are permitted provided that the following conditions are met:
*
*    * Redistributions of source code must retain the above copyright
    notice, this
*      list of conditions and the following disclaimer.
*    * Redistributions in binary form must reproduce the above copyright
    notice,
*      this list of conditions and the following disclaimer in the
    documentation
*      and/or other materials provided with the distribution.
*    * Neither the name of the Sony Ericsson Mobile Communication AB nor
    the names
*      of its contributors may be used to endorse or promote products
    derived from
*      this software without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
    IS" AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
    IMPLIED
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
    DISCLAIMED.
* IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY
    DIRECT,
* INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
    (INCLUDING,
* BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
    USE,
* DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
    THEORY OF
* LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
    NEGLIGENCE
* OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
    IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
*/

package cy.com.airplace;

public class LogRecord {

    private String bssid;
    private int rss;

    public LogRecord(String bssid, int rss) {
        super();
        this.bssid = bssid;
        this.rss = rss;
    }

    public String getBssid() {
        return bssid;
```



```
    }

    public int getRss() {
        return rss;
    }

    public String toString() {
        String str = new String();
        str = String.valueOf(bssid) + " " + String.valueOf(rss) + "\n";
        return str;
    }

}
```

Airplace Tracker code


```
/*
* Copyright (c) 2011, KIOS Research Center and Data Management Systems Lab
    ,
* University of Cyprus. All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
    modification,
* are permitted provided that the following conditions are met:
*
*    * Redistributions of source code must retain the above copyright
    notice, this
*      list of conditions and the following disclaimer.
*    * Redistributions in binary form must reproduce the above copyright
    notice,
*      this list of conditions and the following disclaimer in the
    documentation
*      and/or other materials provided with the distribution.
*    * Neither the name of the Sony Ericsson Mobile Communication AB nor
    the names
*      of its contributors may be used to endorse or promote products
    derived from
*      this software without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
    IS" AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
    IMPLIED
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
    DISCLAIMED.
* IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY
    DIRECT,
* INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
    (INCLUDING,
* BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
    USE,
* DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
    THEORY OF
* LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
    NEGLIGENCE
* OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
    IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
*/

package cy.com.airplace;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.DataInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;

public class PowerTutor {

    public static String getLastFilePowerTutor(String path) {
```



```
        File dir = new File(path);

        Long l = -1l;
        Long curLogTime;

        String[] children = dir.list();
        if (children == null) {
            return null;
        } else {
            for (int i = 0; i < children.length; i++) {
                // Get filename of file or directory
                if (children[i].contains("PowerTrace")) {
                    curLogTime = Long.parseLong((String) children[i].
                        subSequence(10, 23));
                    if (curLogTime > l)
                        l = curLogTime;
                }
            }
        }
        if (l == -1)
            return null;
        else
            return "PowerTrace" + l + ".log";

    }

    public static Power getPower(String file) {

        String strLine;
        String pID = null;
        int wifi = 0, cpu = 0;

        try {

            FileInputStream fstream = new FileInputStream(file);
            DataInputStream in = new DataInputStream(fstream);
            BufferedReader br = new BufferedReader(new InputStreamReader
                (in));

            while ((strLine = br.readLine()) != null) {

                if (pID == null && strLine.contains("cy.com.findme")) {
                    pID = getProcessId(strLine.split(" "));
                    continue;
                }

                if (pID != null) {
                    String[] str = strLine.split(" ");
                    if (str.length == 2) {
                        if (str[0].contains(pID) && str[0].contains("CPU-"
                            + pID)) {
                            cpu = cpu + Integer.parseInt(str[1]);
                        }
                        if (str[0].contains(String.valueOf(pID)) && str[0]
                            .contains("Wifi-" + pID)) {
                            wifi = wifi + Integer.parseInt(str[1]);
                        }
                    }
                }
```

```
                }

                in.close();

            } catch (Exception e) {
                return null;

            }

            return new Power(cpu, wifi);

        }

        public static String getProcessId(String[] str) {

            if (str.length == 3) // org.com.clientpositioning
                return str[1];

            return null;
        }

        public static void writePowerInFile(ArrayList<String[]> str, String id
            , String Path) throws IOException {

            FileWriter fstream = new FileWriter(Path);

            BufferedWriter out = new BufferedWriter(fstream);

            out.write("CPU: \n");
            for (int i = 0; i < str.size(); ++i) {
                if (str.get(i).length == 2) {
                    // System.out.println(str.get(i)[0]);
                    if (str.get(i)[0].contains(id) && str.get(i)[0].contains
                        ("CPU-" + id)) {
                        out.write(str.get(i)[1] + "\n");

                    }
                }
            }

            out.write("\nWIFI: \n");
            for (int i = 0; i < str.size(); ++i) {
                if (str.get(i).length == 2) {
                    // System.out.println(str.get(i)[0]);
                    if (str.get(i)[0].contains(id) && str.get(i)[0].contains
                        ("Wifi-" + id)) {
                        out.write(str.get(i)[1] + "\n");

                    }
                }
            }
            out.close();
        }

    }

    class Power {

        int CPU;
```



```
        int WIFI;

        Power(int cpu, int wifi) {
            CPU = cpu;
            WIFI = wifi;
        }
    }
```

```
/*
* Copyright (c) 2011, KIOS Research Center and Data Management Systems Lab
    ,
* University of Cyprus. All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
    modification,
* are permitted provided that the following conditions are met:
*
*    * Redistributions of source code must retain the above copyright
    notice, this
*      list of conditions and the following disclaimer.
*    * Redistributions in binary form must reproduce the above copyright
    notice,
*      this list of conditions and the following disclaimer in the
    documentation
*      and/or other materials provided with the distribution.
*    * Neither the name of the Sony Ericsson Mobile Communication AB nor
    the names
*      of its contributors may be used to endorse or promote products
    derived from
*      this software without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
    IS" AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
    IMPLIED
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
    DISCLAIMED.
* IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY
    DIRECT,
* INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
    (INCLUDING,
* BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
    USE,
* DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
    THEORY OF
* LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
    NEGLIGENCE
* OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
    IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
*/


package cy.com.airplace;

import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.preference.Preference;
import android.preference.PreferenceActivity;
import android.preference.Preference.OnPreferenceClickListener;
import android.provider.MediaStore.MediaColumns;
import cy.com.FileBrowser.AndroidFileBrowser;
import cy.com.airplace.R;
```



```
public class Preferences extends PreferenceActivity implements
    SharedPreferences.OnSharedPreferenceChangeListener {

    private static final int SELECT_IMAGE = 7;
    private static final int SELECT_PATH = 8;
    private static final int SELECT_FILE = 9;
    private static final int SELECT_TEST_FILE = 10;

    /**
     * Called when the activity is first created.
     * */
    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        // Load the appropriate preferences
        getPreferenceManager().setSharedPreferencesName(FindMe.
            SHARED_PREFS_INDOOR);

        addPreferencesFromResource(R.xml.preferences);

        getPreferenceManager().getSharedPreferences().
            registerOnSharedPreferenceChangeListener(this);

        // Building choice is only visible in indoor mode

        // Custom button to choose image from path
        getPreferenceManager().findPreference("image_custom").
            setOnPreferenceClickListener(new OnPreferenceClickListener() {
            @Override
            public boolean onPreferenceClick(Preference preference) {
                Intent i = new Intent(Intent.ACTION_PICK, android.provider
                    .MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
                i.setType("image/*");
                startActivityForResult(i, SELECT_IMAGE);
                return true;
            }
        });

        // Custom button to choose folder
        getPreferenceManager().findPreference("folder_browser").
            setOnPreferenceClickListener(new OnPreferenceClickListener() {

            @Override
            public boolean onPreferenceClick(Preference preference) {

                Intent i = new Intent(getBaseContext(), AndroidFileBrowser
                    .class);

                Bundle extras = new Bundle();

                // Send flag to browse for folder true
                extras.putInt("to_Browse", 1);

                i.putExtras(extras);

                startActivityForResult(i, SELECT_PATH);
```

```
                return true;
            }
        });

        // Custom button to choose radio map file to use for positioning
        getPreferenceManager().findPreference("radiomap_file").
            setOnPreferenceClickListener(new OnPreferenceClickListener() {

            @Override
            public boolean onPreferenceClick(Preference preference) {

                Intent i = new Intent(getBaseContext(), AndroidFileBrowser
                    .class);

                Bundle extras = new Bundle();
                // Send flag to browse for folder false, it is a file
                    selection.
                extras.putInt("to_Browse", 2);

                i.putExtras(extras);

                startActivityForResult(i, SELECT_FILE);
                return true;
            }
        });


        // Custom button to choose radio map file to use for positioning
        getPreferenceManager().findPreference("test_data_file").
            setOnPreferenceClickListener(new OnPreferenceClickListener() {

            @Override
            public boolean onPreferenceClick(Preference preference) {

                Intent i = new Intent(getBaseContext(), AndroidFileBrowser
                    .class);

                Bundle extras = new Bundle();
                // Send flag to browse for folder false, it is a file
                    selection.
                extras.putInt("to_Browse", 3);

                i.putExtras(extras);

                startActivityForResult(i, SELECT_TEST_FILE);
                return true;
            }
        });
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent
        data) {
        super.onActivityResult(requestCode, resultCode, data);

        SharedPreferences customSharedPreference;

        customSharedPreference = getSharedPreferences(FindMe.
            SHARED_PREFS_INDOOR, MODE_PRIVATE);
```



```
        switch (requestCode) {

        case SELECT_IMAGE:
            if (resultCode == Activity.RESULT_OK) {
                Uri selectedImage = data.getData();
                String RealPath;
                SharedPreferences.Editor editor = customSharedPreference.
                    edit();
                RealPath = getRealPathFromURI(selectedImage);
                editor.putString("image_custom", RealPath);
                editor.commit();
            }
            break;
        case SELECT_PATH:
            if (resultCode == Activity.RESULT_OK) {
                Uri selectedFolder = data.getData();
                String path = selectedFolder.toString();
                SharedPreferences.Editor editor = customSharedPreference.
                    edit();
                editor.putString("folder_browser", path);
                editor.commit();
            }
            break;
        case SELECT_FILE:
            if (resultCode == Activity.RESULT_OK) {
                Uri selectedFile = data.getData();
                String file = selectedFile.toString();
                SharedPreferences.Editor editor = customSharedPreference.
                    edit();
                editor.putString("radiomap_file", file);
                editor.commit();
            }
            break;
        case SELECT_TEST_FILE:
            if (resultCode == Activity.RESULT_OK) {
                Uri selectedFile = data.getData();
                String file = selectedFile.toString();
                SharedPreferences.Editor editor = customSharedPreference.
                    edit();
                editor.putString("test_data_file", file);
                editor.commit();
            }
            break;
        }

    }

    public String getRealPathFromURI(Uri contentUri) {
        String[] proj = { MediaColumns.DATA };
        Cursor cursor = managedQuery(contentUri, proj, null, null, null);
        int column_index = cursor.getColumnIndexOrThrow(MediaColumns.DATA)
            ;
        cursor.moveToFirst();
        return cursor.getString(column_index);
    }

    @Override
    protected void onResume() {
```

```
            super.onResume();
            // Set up a listener whenever a key changes
            getPreferenceScreen().getSharedPreferences().
                registerOnSharedPreferenceChangeListener(this);
        }

        @Override
        protected void onDestroy() {
            super.onDestroy();
            // Unregister the listener whenever a key changes
            getPreferenceManager().getSharedPreferences().
                unregisterOnSharedPreferenceChangeListener(this);
        }

        @Override
        protected void onPause() {
            super.onPause();
            // Unregister the listener whenever a key changes
            getPreferenceScreen().getSharedPreferences().
                unregisterOnSharedPreferenceChangeListener(this);
        }

        @Override
        public void onSharedPreferenceChanged(SharedPreferences arg0, String
            arg1) {
        }

    }
```

```
/*
* Copyright (c) 2011, KIOS Research Center and Data Management Systems Lab
    ,
* University of Cyprus. All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
    modification,
* are permitted provided that the following conditions are met:
*
*    * Redistributions of source code must retain the above copyright
    notice, this
*      list of conditions and the following disclaimer.
*    * Redistributions in binary form must reproduce the above copyright
    notice,
*      this list of conditions and the following disclaimer in the
    documentation
*      and/or other materials provided with the distribution.
*    * Neither the name of the Sony Ericsson Mobile Communication AB nor
    the names
*      of its contributors may be used to endorse or promote products
    derived from
*      this software without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
    IS" AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
    IMPLIED
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
    DISCLAIMED.
* IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY
    DIRECT,
* INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
    (INCLUDING,
* BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
    USE,
* DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
    THEORY OF
* LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
    NEGLIGENCE
* OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
    IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
*/

package cy.com.airplace;/**
 *

 */
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.util.ArrayList;
import java.util.HashMap;

public class RadioMap {

    private File RadiomapMean_File = null;
    private ArrayList<String> MacAdressList = null;
    private HashMap<String, ArrayList<String>> LocationRSS_HashMap = null;
```



```
    private ArrayList<String> OrderList = null;

    public RadioMap() {
        super();
        MacAdressList = new ArrayList<String>();
        LocationRSS_HashMap = new HashMap<String, ArrayList<String>>();
        OrderList = new ArrayList<String>();
    }

    /**
     * Getter of MAC Address list in file order
     *
     * @return
     *            the list of MAC Addresses
     * */
    public ArrayList<String> getMacAdressList() {
        return MacAdressList;
    }

    /**
     * Getter of HashMap Location-RSS Values list in no particular order
     *
     * @return
     *            the HashMap Location-RSS Values
     * */
    public HashMap<String, ArrayList<String>> getLocationRSS_HashMap() {
        return LocationRSS_HashMap;
    }

    /**
     * Getter of Location list in file order
     *
     * @return
     *            the Location list
     * */
    public ArrayList<String> getOrderList() {
        return OrderList;
    }

    /**
     * Getter of radio map mean filename
     *
     * @return
     *            the filename of radiomap mean used
     * */
    public File getRadiomapMean_File() {
        return this.RadiomapMean_File;
    }

    /**
     * Construct a radio map
     *
     * @param inFile
     *            the radio map file to read
     *
     * @return
     *            true if radio map constructed successfully, otherwise
         false
     * */
```

```
        public boolean ConstructRadioMap(File inFile) {

            if (!inFile.exists() || !inFile.canRead()) {
                return false;
            }

            this.RadiomapMean_File = inFile;

            this.OrderList.clear();
            this.MacAdressList.clear();
            this.LocationRSS_HashMap.clear();

            ArrayList<String> RSS_Values = null;
            BufferedReader reader = null;
            String line = null;
            String[] temp = null;
            String key = null;

            try {

                reader = new BufferedReader(new FileReader(inFile));

                // Read the first line
                line = reader.readLine();

                // Must exists
                if (line == null)
                    return false;

                line = line.replace(", ", " ");
                temp = line.split(" ");

                // Must have more than 4 fields
                if (temp.length < 4)
                    return false;

                // Store all Mac Addresses
                for (int i = 3; i < temp.length; ++i)
                    this.MacAdressList.add(temp[i]);

                while ((line = reader.readLine()) != null) {

                    if (line.trim().equals(""))
                        continue;

                    line = line.replace(", ", " ");
                    temp = line.split(" ");

                    if (temp.length < 3)
                        return false;

                    key = temp[0] + " " + temp[1];

                    RSS_Values = new ArrayList<String>();

                    for (int i = 2; i < temp.length; ++i)
                        RSS_Values.add(temp[i]);

                    // Equal number of MAC address and RSS Values
```



```
                    if (this.MacAdressList.size() != RSS_Values.size())
                        return false;

                    this.LocationRSS_HashMap.put(key, RSS_Values);

                    this.OrderList.add(key);
                }
                reader.close();
            } catch (Exception ex) {
                return false;
            }
            return true;
        }

        public String toString() {
            String str = "MAC Adresses: ";
            ArrayList<String> temp;
            for (int i = 0; i < MacAdressList.size(); ++i)
                str += MacAdressList.get(i) + " ";

            str += "\nLocations\n";
            for (String location : LocationRSS_HashMap.keySet()) {
                str += location + " ";
                temp = LocationRSS_HashMap.get(location);
                for (int i = 0; i < temp.size(); ++i)
                    str += temp.get(i) + " ";
                str += "\n";
            }

            return str;
        }
    }
```

```
/*
* Copyright (c) 2011, KIOS Research Center and Data Management Systems Lab
    ,
* University of Cyprus. All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
    modification,
* are permitted provided that the following conditions are met:
*
*    * Redistributions of source code must retain the above copyright
    notice, this
*      list of conditions and the following disclaimer.
*    * Redistributions in binary form must reproduce the above copyright
    notice,
*      this list of conditions and the following disclaimer in the
    documentation
*      and/or other materials provided with the distribution.
*    * Neither the name of the Sony Ericsson Mobile Communication AB nor
    the names
*      of its contributors may be used to endorse or promote products
    derived from
*      this software without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
    IS" AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
    IMPLIED
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
    DISCLAIMED.
* IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY
    DIRECT,
* INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
    (INCLUDING,
* BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
    USE,
* DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
    THEORY OF
* LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
    NEGLIGENCE
* OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
    IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
*/

package cy.com.airplace;

public class WeightRecord {

    private float wx;
    private float wy;

    public WeightRecord(float weightX, float weightY) {
        this.wx = weightX;
        this.wy = weightY;
    }

    public float getWeightX() {
        return wx;
    }
```



```
    public float getWeightY() {
        return wy;
    }
}
```

Airplace Tracker code


```
/*
* Copyright (c) 2011, KIOS Research Center and Data Management Systems Lab
    ,
* University of Cyprus. All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
    modification,
* are permitted provided that the following conditions are met:
*
*    * Redistributions of source code must retain the above copyright
    notice, this
*      list of conditions and the following disclaimer.
*    * Redistributions in binary form must reproduce the above copyright
    notice,
*      this list of conditions and the following disclaimer in the
    documentation
*      and/or other materials provided with the distribution.
*    * Neither the name of the Sony Ericsson Mobile Communication AB nor
    the names
*      of its contributors may be used to endorse or promote products
    derived from
*      this software without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
    IS" AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
    IMPLIED
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
    DISCLAIMED.
* IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY
    DIRECT,
* INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
    (INCLUDING,
* BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
    USE,
* DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
    THEORY OF
* LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
    NEGLIGENCE
* OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
    IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
*/

package cy.com.airplace;

public class Zone {

    private final float min;
    private final float max;

    Zone(float min, float max){
        this.min=min;
        this.max=max;
    }

    public float getMin() {
        return min;
    }
```



```
    public float getMax() {
        return max;
    }
}
```

Airplace Tracker code


```
/*
* Copyright (c) 2011, KIOS Research Center and Data Management Systems Lab
    ,
* University of Cyprus. All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
    modification,
* are permitted provided that the following conditions are met:
*
*    * Redistributions of source code must retain the above copyright
    notice, this
*      list of conditions and the following disclaimer.
*    * Redistributions in binary form must reproduce the above copyright
    notice,
*      this list of conditions and the following disclaimer in the
    documentation
*      and/or other materials provided with the distribution.
*    * Neither the name of the Sony Ericsson Mobile Communication AB nor
    the names
*      of its contributors may be used to endorse or promote products
    derived from
*      this software without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
    IS" AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
    IMPLIED
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
    DISCLAIMED.
* IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY
    DIRECT,
* INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
    (INCLUDING,
* BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
    USE,
* DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
    THEORY OF
* LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
    NEGLIGENCE
* OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
    IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
*/

package cy.com.CalculationModes;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.util.ArrayList;
import android.os.Handler;
import cy.com.airplace.Algorithms;
import cy.com.airplace.LogRecord;
import cy.com.airplace.RadioMap;
import cy.com.airplace.WeightRecord;

public class OfflineMode extends Thread {

    private String errMsg = null;
```



```
    private Handler handler;
    private final RadioMap RM;
    private final File test_data_file;
    private final int algorithm_selection;

    private double average_pos_err, average_exe_time;

    // The scan list to use for offline
    private ArrayList<LogRecord> OfflineScanList;

    public OfflineMode(RadioMap RM, File test_data_file, int
        algorithm_selection, Handler handler) {
        this.RM = RM;
        this.test_data_file = test_data_file;
        this.handler = handler;
        this.algorithm_selection = algorithm_selection;
        this.OfflineScanList = new ArrayList<LogRecord>();
    }

    public void run() {

        if (!test_data_file.isFile() || !test_data_file.exists() || !
            test_data_file.canRead()) {
            errMsg = test_data_file + " does not exist or is not readable"
                ;
            handler.sendEmptyMessage(-1);
            return;
        }

        OfflineScanList.clear();

        BufferedReader reader = null;
        String line;
        String[] temp;

        String test_geo;
        int count_test_pos = 0;
        double pos_error;
        double sum_pos_error = 0;

        long bytesRead = 0;
        long bytesTotal = test_data_file.length();
        int perc = 0;

        long start = 0;
        long finish = 0;
        long total = 0;

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

        try {

            reader = new BufferedReader(new FileReader(test_data_file));

            /* Read the first line */
            line = reader.readLine();

            // Must exists
            if (line == null) {
```

```
                    errMsg = test_data_file + " file is corrupted";
                    handler.sendEmptyMessage(-1);
                    return;
                }

                bytesRead += line.length() + 1;

                if (perc < (int) (((float) bytesRead / (float) bytesTotal) *
                    100)) {
                    perc = (int) (((float) bytesRead / (float) bytesTotal) *
                        100);
                    handler.sendEmptyMessage(perc);
                }

                /* Store the Mac Addresses */
                if (line.startsWith("#")) {
                    line = line.replace(", ", " ");
                    temp = line.split(" ");

                    // Must have more than 4 fields
                    if (temp.length < 4) {
                        errMsg = test_data_file + " file is corrupted";
                        handler.sendEmptyMessage(-1);
                        return;
                    }

                    // Store all Mac Addresses
                    for (int i = 3; i < temp.length; ++i)
                        MacAdressList.add(temp[i]);
                } else {
                    errMsg = test_data_file + " file is corrupted";
                    handler.sendEmptyMessage(-1);
                    return;
                }

                ArrayList<WeightRecord> WeightsList = Algorithms.readWeights
                    (RM.getRadiomapMean_File());

                count_test_pos = 0;

                while ((line = reader.readLine()) != null) {

                    bytesRead += line.length() + 1;

                    line = line.trim().replace(", ", " ");
                    temp = line.split(" ");

                    if (temp.length < 3) {
                        errMsg = test_data_file + " file is corrupted";
                        handler.sendEmptyMessage(-1);
                        return;
                    }

                    if (MacAdressList.size() != temp.length - 2) {
                        errMsg = test_data_file + " file is corrupted";
                        handler.sendEmptyMessage(-1);
                        return;
                    }
```



```
                    for (int i = 2; i < temp.length; ++i) {
                        LogRecord lr = new LogRecord(MacAdressList.get(i - 2),
                            Integer.parseInt(temp[i]));
                        OfflineScanList.add(lr);
                    }

                    if (perc < (int) (((float) bytesRead / (float) bytesTotal)
                        * 100)) {
                        perc = (int) (((float) bytesRead / (float) bytesTotal)
                            * 100);
                        handler.sendEmptyMessage(perc);
                    }

                    start = System.currentTimeMillis();

                    test_geo = Algorithms.ProcessingAlgorithms(OfflineScanList
                        , RM, algorithm_selection, WeightsList);

                    if (test_geo == null) {
                        errMsg = "Can't calculate a location. Check that test
                            data and radio map files refer to the same area.";
                        handler.sendEmptyMessage(-1);
                        return;
                    }

                    finish = System.currentTimeMillis();

                    total += (finish - start);

                    OfflineScanList.clear();

                    pos_error = calculateEuclideanDistance(temp[0] + " " +
                        temp[1], test_geo);

                    if (pos_error != -1) {
                        sum_pos_error += pos_error;
                        count_test_pos++;
                    }
                }

                reader.close();

                handler.sendEmptyMessage(100);

                average_pos_err = sum_pos_error / (double) count_test_pos;
                average_exe_time = total / (double) count_test_pos;

                handler.sendEmptyMessage(-1);
                errMsg = null;

            } catch (Exception ex) {
                errMsg = "Can't calculate a location.\nError: " + ex.
                    getMessage() + "." + "\nCheck that test data and radio map
                    files are not corrupted.";
                handler.sendEmptyMessage(-1);
            }
        }

        private double calculateEuclideanDistance(String real, String estimate
```

```
            ) {

            double pos_error;
            String[] temp_real;
            String[] temp_estimate;
            double x1, x2;

            temp_real = real.split(" ");
            temp_estimate = estimate.split(" ");

            try {
                x1 = Math.pow((Double.parseDouble(temp_real[0]) - Double.
                    parseDouble(temp_estimate[0])), 2);
                x2 = Math.pow((Double.parseDouble(temp_real[1]) - Double.
                    parseDouble(temp_estimate[1])), 2);
            } catch (Exception e) {
                return -1;
            }

            pos_error = Math.sqrt((x1 + x2));

            return pos_error;
        }

        public String getErrMsg() {
            return errMsg;
        }

        public double getAverage_pos_err() {
            return average_pos_err;
        }

        public double getAverage_exe_time() {
            return average_exe_time;
        }

    }
```

Airplace Tracker code


```
/*
* Copyright (c) 2011, KIOS Research Center and Data Management Systems Lab
    ,
* University of Cyprus. All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
    modification,
* are permitted provided that the following conditions are met:
*
*    * Redistributions of source code must retain the above copyright
    notice, this
*      list of conditions and the following disclaimer.
*    * Redistributions in binary form must reproduce the above copyright
    notice,
*      this list of conditions and the following disclaimer in the
    documentation
*      and/or other materials provided with the distribution.
*    * Neither the name of the Sony Ericsson Mobile Communication AB nor
    the names
*      of its contributors may be used to endorse or promote products
    derived from
*      this software without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
    IS" AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
    IMPLIED
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
    DISCLAIMED.
* IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY
    DIRECT,
* INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
    (INCLUDING,
* BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
    USE,
* DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
    THEORY OF
* LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
    NEGLIGENCE
* OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
    IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
*/

package cy.com.Downloading;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import android.os.Handler;

public class DownloadingSettings extends Thread {

    private final String IP;
    private final String PORT;
    private final String filename_radiomap_download;
```



```
    private final String folder_path;

    private String errMsg;
    private final Handler handler;

    /**
     * @param IP
     *            the IP address of radiomap distribution server
     *
     * @param PORT
     *            the port that radiomap distribution server is listening
     * */
    public DownloadingSettings(String IP, String PORT, String folder_path,
        String filename, Handler handler) {
        this.filename_radiomap_download = filename;
        this.PORT = PORT;
        this.IP = IP;
        this.folder_path = folder_path;
        this.handler = handler;
    }

    public String getErrMsg() {
        return errMsg;
    }

    /**
     * Establishes a connection on IP/PORT and download radiomap
     *
     * */
    public void run() {

        Socket connection = null;
        FileOutputStream fos = null;
        File root = new File(folder_path);

        String radiomap_mean = filename_radiomap_download;
        String rbf_weights = radiomap_mean + "-rbf-weights";
        String parameters = radiomap_mean + "-parameters";

        try {

            // Create new socket
            connection = new Socket(IP, Integer.parseInt(PORT));

            // Check that path is writable
            if (root.canWrite()) {
                fos = new FileOutputStream(new File(root, radiomap_mean),
                    false);
            } else {
                errMsg = "Directory: " + root.getAbsolutePath() + " is not
                    writable.\nYou may need an external memory card";
                handler.sendEmptyMessage(-2);
                return;
            }

            PrintWriter out = new PrintWriter(connection.getOutputStream()
                , true);
            BufferedReader in = new BufferedReader(new InputStreamReader
                (connection.getInputStream()));
```

```
                String inputLine, outputLine;

                inputLine = in.readLine();

                if (!inputLine.equalsIgnoreCase("+OK READY")) {
                    fos.close();
                    out.close();
                    in.close();
                    connection.close();
                    errMsg = "Server not ready.\nTry again later.";
                    handler.sendEmptyMessage(-2);
                    return;
                }

                outputLine = "GET radiomap";
                out.println(outputLine);
                inputLine = in.readLine();

                if (!inputLine.startsWith("RADIOMAP")) {
                    fos.close();
                    out.close();
                    in.close();
                    connection.close();
                    errMsg = inputLine + "";
                    handler.sendEmptyMessage(-2);
                    return;
                }

                inputLine = inputLine.replaceFirst("RADIOMAP ", "");
                fos.write((inputLine + "\n").getBytes());

                // Get files: radiomap, parameters and rbf weights
                while ((inputLine = in.readLine()) != null) {
                    if (inputLine.compareTo("null") == 0 || inputLine.
                        startsWith("CORRUPTED"))
                        break;

                    if (inputLine.equalsIgnoreCase("PARAMETERS")) {
                        fos = new FileOutputStream(new File(root, parameters),
                            false);
                    } else if (inputLine.equalsIgnoreCase("RBF_WEIGHTS")) {
                        fos = new FileOutputStream(new File(root, rbf_weights)
                            , false);
                    } else {
                        fos.write((inputLine + "\n").getBytes());
                    }
                }

                fos.close();
                out.close();
                in.close();
                connection.close();

                errMsg = null;
                handler.sendEmptyMessage(-2);

            } catch (Exception e) {
                errMsg = "Error: " + e.getMessage();
                handler.sendEmptyMessage(-2);
```



```
            }

        }

    }
```

```
/*
* Copyright (c) 2011, KIOS Research Center and Data Management Systems Lab
    ,
* University of Cyprus. All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
    modification,
* are permitted provided that the following conditions are met:
*
*    * Redistributions of source code must retain the above copyright
    notice, this
*      list of conditions and the following disclaimer.
*    * Redistributions in binary form must reproduce the above copyright
    notice,
*      this list of conditions and the following disclaimer in the
    documentation
*      and/or other materials provided with the distribution.
*    * Neither the name of the Sony Ericsson Mobile Communication AB nor
    the names
*      of its contributors may be used to endorse or promote products
    derived from
*      this software without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
    IS" AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
    IMPLIED
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
    DISCLAIMED.
* IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY
    DIRECT,
* INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
    (INCLUDING,
* BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
    USE,
* DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
    THEORY OF
* LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
    NEGLIGENCE
* OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
    IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
*/

package cy.com.FileBrowser;

import java.io.File;
import java.util.ArrayList;
import java.util.List;

import cy.com.airplace.FindMe;
import cy.com.airplace.R;
import android.app.AlertDialog;
import android.app.ListActivity;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.net.Uri;
```



```
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.View.OnClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.TextView;

public class AndroidFileBrowser extends ListActivity implements
    OnClickListener {

    // Enum for the Display Mode
    private enum DISPLAYMODE {
        ABSOLUTE, RELATIVE;
    }

    private final DISPLAYMODE displayMode = DISPLAYMODE.ABSOLUTE;
    private List<String> directoryEntries = new ArrayList<String>();
    private File currentDirectory = null;
    private final File homeDirectory = new File("/");
    private File file = null;
    private TextView pwd = null;
    private Button select_file_folder = null;
    private int selectFolder = 1;
    private String folder_path = null;
    private String file_path = null;
    private SharedPreferences sharedPreferences = null;

    @Override
    public void onCreate(Bundle icicle) {

        super.onCreate(icicle);
        setContentView(R.layout.main_choose_file_or_directory);

        pwd = (TextView) findViewById(R.id.pwd);
        select_file_folder = (Button) findViewById(R.id.select_file_folder
            );
        select_file_folder.setOnClickListener(this);

        // Get flags for browsing and indoor or outdoor
        Bundle extras = getIntent().getExtras();
        if (extras != null) {
            selectFolder = extras.getInt("to_Browse");
        }

        sharedPreferences = getSharedPreferences(FindMe.
            SHARED_PREFS_INDOOR, MODE_PRIVATE);

        // Get folder path selected before
        folder_path = sharedPreferences.getString("folder_browser", "").
            trim();

        // Get file of radiomap selected before

        switch (selectFolder) {
        case 1:
```

```
                select_file_folder.setText("Save in this folder");
                break;

            case 2:

                file_path = sharedPreferences.getString("radiomap_file", "").
                    trim();
                select_file_folder.setText("Use radiomap");

                if (file_path != null && !file_path.equals("") && new File
                    (file_path).canRead())
                    pwd.setText(file_path);

                break;

            case 3:

                file_path = sharedPreferences.getString("test_data_file", "").
                    trim();
                select_file_folder.setText("Use test data");

                if (file_path != null && !file_path.equals("") && new File
                    (file_path).canRead())
                    pwd.setText(file_path);

                break;
            }

            if (folder_path == null || folder_path.equals("")) {
                browseToHome();
            } else {
                currentDirectory = new File(makePath(folder_path));
                browseTo(currentDirectory);
            }
        }

        /**
         * Sets home directory (/) as the current directory
         *
         * */
        private void browseToHome() {
            currentDirectory = new File("/");
            browseTo(homeDirectory);
        }

        /**
         * Make a path if exists, can read and is a directory
         *
         * @param path
         *            the given path to check
         *
         *
         * @return the constructed path
         * */
        private String makePath(String path) {

            if (path.length() == 0) {
                return "/";
            }
```



```
            File check = new File(path);
            if (!check.exists() || !check.canRead() || !check.isDirectory()) {
                return "/";
            }

            if (check.isDirectory())
                return path;

            return "/";
        }

        /**
         * Browses to aDirectory path in mobile device file system
         *
         * @param aDirectory
         *            the given directory to enter
         *
         * */
        private void browseTo(final File aDirectory) {

            if (aDirectory.isDirectory()) {
                this.currentDirectory = aDirectory;
                fill(aDirectory.listFiles());

                if (selectFolder == 1)
                    pwd.setText(currentDirectory.getAbsolutePath());
            }
        }

        /**
         * Fills directoryEntries with the files in current path
         *
         * @param files
         *            in the current directory
         *
         * */
        private void fill(File[] files) {

            // Clear list
            this.directoryEntries.clear();

            // Add the "~" for home directory
            this.directoryEntries.add(getString(R.string.homeDir));

            // And the ".." for parent directory
            if (this.currentDirectory.getParent() != null)
                this.directoryEntries.add(getString(R.string.parentDir));

            switch (this.displayMode) {

            case ABSOLUTE:
                for (File file : files) {
                    this.directoryEntries.add(file.getAbsolutePath());
                }
                break;
            case RELATIVE:
                int currentPathStringLenght = this.currentDirectory.
                    getAbsolutePath().length();
```

```
                for (File file : files) {
                    this.directoryEntries.add(file.getAbsolutePath().substring
                        (currentPathStringLenght));
                }
                break;
            }

            MyCustomAdapter directoryList = new MyCustomAdapter(this, R.xml.
                file_row, this.directoryEntries);

            this.setListAdapter(directoryList);
        }

        @Override
        protected void onListItemClick(ListView l, View v, int position, long
            id) {

            file = new File(directoryEntries.get(position));

            if (file.isDirectory()) {
                if (file.canRead()) {
                    if (file.getName().equals(getString(R.string.parentDir)))
                        {
                        browseTo(currentDirectory.getParentFile());
                    } else if (file.getName().equals(getString(R.string.
                        homeDir))) {
                        browseTo(homeDirectory);
                    } else
                        browseTo(file);
                } else {
                    showAlert("Read Permission Denied", "Warning", this);
                }
            } else if (!file.isDirectory() && selectFolder == 1) {
                showAlert("Not a directory", "Warning", this);
            } else {
                pwd.setText(file.getAbsolutePath());
            }
            super.onListItemClick(l, v, position, id);
        }

        /**
         * Creates an Alert box
         *
         * @param message
         *            the message to show to user
         *
         * @param title
         *            the title of alert box
         *
         * @param ctx
         *            the context
         * */
        private void showAlert(String message, String title, Context ctx) {
            // Create a builder
            AlertDialog.Builder builder = new AlertDialog.Builder(ctx);
            builder.setTitle(title);

            // Add buttons and listener
            builder.setMessage(message).setCancelable(false).setPositiveButton
```



```
                ("OK", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int id) {
                    dialog.cancel();
                }
            });

            // Create the dialog
            AlertDialog ad = builder.create();

            // Show
            ad.show();
        }

        @Override
        public void onClick(View v) {

            switch (v.getId()) {

            case R.id.select_file_folder:

                // Select a path mode
                if (selectFolder == 1) {

                    if (currentDirectory.canWrite()) {
                        Intent data = new Intent();
                        data.setData(Uri.parse(currentDirectory.
                            getAbsolutePath()));
                        setResult(RESULT_OK, data);
                        AndroidFileBrowser.this.finish();
                    } else
                        showAlert("Write Permission Denied", "Warning", this);

                }
                // Select a file mode
                else {
                    file = new File(pwd.getText().toString());
                    if (file.exists()) {
                        if (file.canRead()) {
                            Intent data = new Intent();
                            data.setData(Uri.parse(pwd.getText().toString()));
                            setResult(RESULT_OK, data);
                            AndroidFileBrowser.this.finish();
                        } else
                            showAlert("Read Permission Denied", "Warning",
                                this);
                    } else
                        showAlert("File not available", "Warning", this);
                }
                break;
            }
        }

        /**
         * Control when back button is pressed
         * */
        @Override
        public void onBackPressed() {
            finish();
        }
```

```
    /**
     * Public Inner Class which help to put images for each file that is
     * different type
     *
     * */
    public class MyCustomAdapter extends ArrayAdapter<String> {

        List<String> myList;

        /**
         * Constructor
         */
        public MyCustomAdapter(Context context, int textViewResourceId,
            List<String> objects) {

            super(context, textViewResourceId, objects);
            myList = objects;
        }

        @Override
        public View getView(int position, View convertView, ViewGroup
            parent) {

            View row = convertView;
            // Check If Row Is Null
            if (row == null) {
                // Make New Layoutinflater
                LayoutInflater vi = (LayoutInflater) getSystemService
                    (Context.LAYOUT_INFLATER_SERVICE);
                row = vi.inflate(R.xml.file_row, parent, false);
            }
            TextView label = (TextView) row.findViewById(R.id.file);
            String stringFile = myList.get(position).toString();

            // Change The Symbol Of The Home Directory
            if (stringFile.equals(getString(R.string.homeDir)))
                label.setText("~");
            else
                label.setText(stringFile);

            File file = new File(stringFile);
            ImageView icon = (ImageView) row.findViewById(R.id.icon);

            if (file.isDirectory())
                icon.setImageResource(R.drawable.directory);
            else
                icon.setImageResource(FindDrawable(stringFile));

            return row;
        }

        /**
         * Find the right icon for each file type
         *
         * @param file
         *            the file to get image source
         *
         * @return the id of image source
```



```
         * */
        //
        private int FindDrawable(String file) {
            if (file.endsWith(".txt")) {
                return R.drawable.txt;
            } else if (file.endsWith(".pdf")) {
                return R.drawable.pdf;
            } else if (file.endsWith(".exe")) {
                return R.drawable.exe;
            } else if (file.endsWith(".apk")) {
                return R.drawable.apk;
            } else if (file.endsWith(".png")) {
                return R.drawable.png;
            } else if (file.endsWith(".gif")) {
                return R.drawable.gif;
            } else if (file.endsWith(".jpg")) {
                return R.drawable.jpg;
            } else if (file.endsWith(".rar")) {
                return R.drawable.rar;
            } else if (file.endsWith(".zip")) {
                return R.drawable.zip;
            } else if (file.endsWith(".gz")) {
                return R.drawable.gz;
            } else if (file.endsWith(".mp3") || file.endsWith(".wav") ||
                file.endsWith(".amp")) {
                return R.drawable.sound;
            } else if (file.endsWith(".mp4") || file.endsWith(".avi") ||
                file.endsWith(".flv")) {
                return R.drawable.video;
            } else
                return R.drawable.txt;
        }
    }
}
```

Airplace Tracker code


```
package cy.com.FloorMode;

import java.util.ArrayList;
import cy.com.airplace.LogRecord;


public class FloorProcedure
{
    private ArrayList<ArrayList<RMHelp>> arrayRMHelp = null;


    /**
     * Take an arrayList of arrayLists.
     * Each arraList is the content of the help files, of the two floors
        that
     * I want to find me!
     *
     * @param arrayList of arrayLists of RMHelp lists
     */
    public FloorProcedure(ArrayList<ArrayList<RMHelp>> array)
    {
        this.arrayRMHelp=array;
    }


    /**
     * Find the nearest mac Address of the current live received AP's.
     * After that decide in which floor is this AP and change to this
        floor.
     *
     * @param the current results of the detected AP's
     * @return the result of the two check in .floor files
     */
    public String findTheMAC(ArrayList<LogRecord> lr)
    {
        ArrayList<LogRecordFloor> lrf = new ArrayList<LogRecordFloor>();
        Double maxRSS = 0.0;
        Double minRSS = 0.0;
        String strRe = "";
        boolean flagEq = true;

        /**
         * Take the lr type LogRecord and convert it to LogRecordFloor
         */

        for(LogRecord current : lr)
        {
            lrf.add(new LogRecordFloor(current.getBssid(), current.getRss
                (), 2));
        }

        //Find the strongest Access Point
        boolean flag = true;
        LogRecordFloor maxMac = null;
        for(int i=0; i<lrf.size(); ++i)
        {
            if(flag)
            {
                maxMac = lrf.get(i);
```



```
                flag=false;
            }
            if(maxMac.getRss()<lrf.get(i).getRss())
            {
                maxMac = lrf.get(i);
            }
        }

        /*
         * Find where it exists the maxMac
         */
        for(int i=0; i< arrayRMHelp.size(); ++i)
            {
                for(int j=0; j< arrayRMHelp.get(i).size(); ++j)
                {
                    ArrayList<RMHelp> temp = arrayRMHelp.get(i);

                    if(arrayRMHelp.get(i).get(j).getMacAddress().equals
                        (maxMac.getBssid()))
                    {
                        temp.get(j).setVisible(true);
                    }
                }
            }


        /* Check the case of existing the maxMac in both .floor files */
        for(int j=0; j< arrayRMHelp.get(0).size(); ++j)
        {
            RMHelp temp0 = arrayRMHelp.get(0).get(j);

            for(int z=0; z<arrayRMHelp.get(1).size(); ++z)
            {
                RMHelp temp1 = arrayRMHelp.get(1).get(z);

                /**
                 * Check if one macAddress belongs to the two floors
                 */
                if(temp0.getMacAddress().equals(temp1.getMacAddress()))
                {
                    /**
                     * Find the macAddress from the current results if
                        exist
                     */
                    if(maxMac.getBssid().equals(temp0.getMacAddress()))
                    {
                        /**
                         * Decide in which floor you will label the
                            macAddress
                         */
                        if(temp0.getRss()>temp1.getRss())
                        {
                            //lrf.get(i).setFloor(false, 1);
                            flagEq=false;
                            maxRSS = temp0.getRss();
                            minRSS = temp1.getRss();
                            strRe="1,"+maxRSS.intValue()+" 2,"+minRSS.
                                intValue();
                        }
```

Airplace Tracker code


```
                            else
                            {
                                flagEq=false;
                                maxRSS = temp1.getRss();
                                minRSS = temp0.getRss();
                                strRe="1,"+minRSS.intValue()+" 2,"+maxRSS.
                                    intValue();
                            }
                        }
                    }
                }
            }

            if(flagEq)
            {
                for(int i=0; i< arrayRMHelp.size(); ++i)
                {
                    for(int j=0; j< arrayRMHelp.get(i).size(); ++j)
                    {
                        ArrayList<RMHelp> temp = arrayRMHelp.get(i);
                        for(RMHelp current : temp)
                        {
                            if(temp.get(j).isVisible())
                            {
                                if(i==0)
                                {
                                    return "1,"+temp.get(j).getRss().intValue
                                        ()+" 2,"+(-800);
                                }
                                else if(i==1)
                                {
                                    return "1,"+(-800)+" 2,"+temp.get(j).
                                        getRss().intValue();
                                }

                            }
                        }
                    }
                }
            }

            if(strRe.equals(""))
            {
                return "1,"+(-800)+" 2,"+(-800);
            }

            return strRe;

        }
    }
```

```
package cy.com.FloorMode;

import cy.com.airplace.LogRecord;


public class LogRecordFloor extends LogRecord
{
    private boolean[] floor;

    public LogRecordFloor(String bssid, int rss, int noOfFloors)
    {
        super(bssid, rss);
        this.floor = new boolean[noOfFloors];

        for(int i=0; i<this.floor.length; ++i)
        {
            this.floor[i]=true;
        }
    }

    /**
     * Set in which floor the macAddress belongs at final
     *
     * @param validFloor
     * @param floor
     */
    public void setFloor(boolean validFloor, int floor)
    {
        this.floor[floor] = validFloor;
    }

    /**
     * Get the result of the specific number of the floor
     *
     * @param noOfFloor
     * @return
     */
    public boolean getFloor(int noOfFloor)
    {
        return floor[noOfFloor];
    }

    public String toString()
    {
        String str = new String();
        str = this.getBssid() + " " + String.valueOf(this.getRss()) + "\n"
            +" "+this.floor[0]+" "+this.floor[1];
        return str;
    }
}
```

Airplace Tracker code


```
package cy.com.FloorMode;

import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;


public class ParseRMHelp
{
    private DataInputStream in = null;
    private BufferedReader br = null;
    private String nameOfFile=null;
    private ArrayList<RMHelp> arrayRMH = new ArrayList<RMHelp>();

    private ArrayList<String> inArray = new ArrayList<String>();

    public ParseRMHelp(String strName) throws IOException
    {
        this.nameOfFile=strName;

        FileInputStream fstream = null;
        try
        {
            fstream = new FileInputStream(this.nameOfFile);
        } catch (FileNotFoundException e)
        {
            e.printStackTrace();
        }

        // Get the object of DataInputStream
        in = new DataInputStream(fstream);
        br = new BufferedReader(new InputStreamReader(in));
        String line = null;
        while((line=br.readLine())!=null)
        {
            inArray.add(line);
        }

        br.close();
        in.close();
        this.createRMHelp();
    }

    private void createRMHelp()
    {

        for(String str : inArray)
        {
            String[] temp = str.split(", ");
            arrayRMH.add(new RMHelp(temp[0], Double.valueOf(temp[1]),
                Integer.valueOf(temp[2])));
        }
    }

    public ArrayList<RMHelp> getArrayRMH()
```



```
        {
            return arrayRMH;
        }
    }
```

Airplace Tracker code


```
package cy.com.FloorMode;


public class RMHelp
{
    private String macAddress;
    private Double rss;
    private int times;
    private boolean visible;

    /**
     * Constructor of RMHelp class to upload the .floor file
     *
     * @param macAddress
     * @param rss
     * @param times
     */
    public RMHelp(String macAddress, Double rss, int times)
    {
        this.setMacAddress(macAddress);
        this.setRss(rss);
        this.setTimes(times);
        this.setVisible(false);
    }

    /**
     * Constructor that takes only two parameters
     *
     * @param macAddress
     * @param rss
     */
    public RMHelp(String macAddress, Double rss)
    {
        this.setMacAddress(macAddress);
        this.setRss(rss);
    }

    public void setMacAddress(String macAddress)
    {
        this.macAddress = macAddress;
    }

    public String getMacAddress()
    {
        return macAddress;
    }

    public void setRss(Double rss)
    {
        this.rss = rss;
    }

    public Double getRss()
    {
        return rss;
    }

    public void setTimes(int times)
    {
```



```
        this.times = times;
    }

    public int getTimes()
    {
        return times;
    }

    public String toString()
    {
        return macAddress+" "+rss+" "+times+" "+visible;
    }

    public void setVisible(boolean visible)
    {
        this.visible = visible;
    }

    public boolean isVisible()
    {
        return visible;
    }

}
```

```
/*
* Copyright (c) 2011, KIOS Research Center and Data Management Systems Lab
    ,
* University of Cyprus. All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
    modification,
* are permitted provided that the following conditions are met:
*
*    * Redistributions of source code must retain the above copyright
    notice, this
*      list of conditions and the following disclaimer.
*    * Redistributions in binary form must reproduce the above copyright
    notice,
*      this list of conditions and the following disclaimer in the
    documentation
*      and/or other materials provided with the distribution.
*    * Neither the name of the Sony Ericsson Mobile Communication AB nor
    the names
*      of its contributors may be used to endorse or promote products
    derived from
*      this software without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
    IS" AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
    IMPLIED
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
    DISCLAIMED.
* IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY
    DIRECT,
* INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
    (INCLUDING,
* BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
    USE,
* DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
    THEORY OF
* LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
    NEGLIGENCE
* OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
    IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
*/

/*
 * Copyright (c) 2010, Sony Ericsson Mobile Communication AB. All rights
     reserved.
 *
 * Redistribution and use in source and binary forms, with or without
     modification,
 * are permitted provided that the following conditions are met:
 *
 *    * Redistributions of source code must retain the above copyright
     notice, this
 *      list of conditions and the following disclaimer.
 *    * Redistributions in binary form must reproduce the above copyright
     notice,
 *      this list of conditions and the following disclaimer in the
     documentation
```



```
 *      and/or other materials provided with the distribution.
 *    * Neither the name of the Sony Ericsson Mobile Communication AB nor
     the names
 *      of its contributors may be used to endorse or promote products
     derived from
 *      this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
     IS" AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
     IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
     DISCLAIMED.
 * IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
     ANY DIRECT,
 * INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
     (INCLUDING,
 * BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS
     OF USE,
 * DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
     THEORY OF
 * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
     NEGLIGENCE
 * OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
     IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 */

package cy.com.util;

/**
 * Utility class used to handle flinging within a specified limit.
 */
public abstract class Dynamics {
    /**
     * The maximum delta time, in milliseconds, between two updates
     */
    private static final int MAX_TIMESTEP = 50;

    /** The current position */
    protected float mPosition;

    /** The current velocity */
    protected float mVelocity;

    /** The current maximum position */
    protected float mMaxPosition = Float.MAX_VALUE;

    /** The current minimum position */
    protected float mMinPosition = -Float.MAX_VALUE;

    /** The time of the last update */
    protected long mLastTime = 0;

    /**
     * Sets the state of the dynamics object. Should be called before
         starting
     * to call update.
     *
```

```
     * @param position The current position.
     * @param velocity The current velocity in pixels per second.
     * @param now The current time
     */
    public void setState(final float position, final float velocity, final
        long now) {
        mVelocity = velocity;
        mPosition = position;
        mLastTime = now;
    }

    /**
     * Returns the current position. Normally used after a call to update
         () in
     * order to get the updated position.
     *
     * @return The current position
     */
    public float getPosition() {
        return mPosition;
    }

    /**
     * Gets the velocity. Unit is in pixels per second.
     *
     * @return The velocity in pixels per second
     */
    public float getVelocity() {
        return mVelocity;
    }

    /**
     * Used to find out if the list is at rest, that is, has no velocity
         and is
     * inside the the limits. Normally used to know if more calls to
         update are
     * needed.
     *
     * @param velocityTolerance Velocity is regarded as 0 if less than
     *            velocityTolerance
     * @param positionTolerance Position is regarded as inside the limits
         even
     *            if positionTolerance above or below
     *
     * @return true if list is at rest, false otherwise
     */
    public boolean isAtRest(final float velocityTolerance, final float
        positionTolerance) {
        final boolean standingStill = Math.abs(mVelocity) <
            velocityTolerance;
        final boolean withinLimits = mPosition - positionTolerance <
            mMaxPosition
                && mPosition + positionTolerance > mMinPosition;
        return standingStill && withinLimits;
    }

    /**
     * Sets the maximum position.
     *
```



```
     * @param maxPosition The maximum value of the position
     */
    public void setMaxPosition(final float maxPosition) {
        mMaxPosition = maxPosition;
    }

    /**
     * Sets the minimum position.
     *
     * @param minPosition The minimum value of the position
     */
    public void setMinPosition(final float minPosition) {
        mMinPosition = minPosition;
    }

    /**
     * Updates the position and velocity.
     *
     * @param now The current time
     */
    public void update(final long now) {
        int dt = (int)(now - mLastTime);
        if (dt > MAX_TIMESTEP) {
            dt = MAX_TIMESTEP;
        }

        onUpdate(dt);

        mLastTime = now;
    }

    /**
     * Gets the distance to the closest limit (max and min position).
     *
     * @return If position is more than max position: distance to max
         position. If
     *         position is less than min position: distance to min
         position. If
     *         within limits: 0
     */
    protected float getDistanceToLimit() {
        float distanceToLimit = 0;

        if (mPosition > mMaxPosition) {
            distanceToLimit = mMaxPosition - mPosition;
        } else if (mPosition < mMinPosition) {
            distanceToLimit = mMinPosition - mPosition;
        }

        return distanceToLimit;
    }

    /**
     * Updates the position and velocity.
     *
     * @param dt The delta time since last time
     */
    abstract protected void onUpdate(int dt);
}
```

Airplace Tracker code


```
/*
* Copyright (c) 2011, KIOS Research Center and Data Management Systems Lab
    ,
* University of Cyprus. All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
    modification,
* are permitted provided that the following conditions are met:
*
*    * Redistributions of source code must retain the above copyright
    notice, this
*      list of conditions and the following disclaimer.
*    * Redistributions in binary form must reproduce the above copyright
    notice,
*      this list of conditions and the following disclaimer in the
    documentation
*      and/or other materials provided with the distribution.
*    * Neither the name of the Sony Ericsson Mobile Communication AB nor
    the names
*      of its contributors may be used to endorse or promote products
    derived from
*      this software without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
    IS" AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
    IMPLIED
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
    DISCLAIMED.
* IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY
    DIRECT,
* INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
    (INCLUDING,
* BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
    USE,
* DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
    THEORY OF
* LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
    NEGLIGENCE
* OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
    IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
*/

/*
 * Copyright (c) 2010, Sony Ericsson Mobile Communication AB. All rights
     reserved.
 *
 * Redistribution and use in source and binary forms, with or without
     modification,
 * are permitted provided that the following conditions are met:
 *
 *    * Redistributions of source code must retain the above copyright
     notice, this
 *      list of conditions and the following disclaimer.
 *    * Redistributions in binary form must reproduce the above copyright
     notice,
 *      this list of conditions and the following disclaimer in the
     documentation
```



```
 *      and/or other materials provided with the distribution.
 *    * Neither the name of the Sony Ericsson Mobile Communication AB nor
     the names
 *      of its contributors may be used to endorse or promote products
     derived from
 *      this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
     IS" AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
     IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
     DISCLAIMED.
 * IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
     ANY DIRECT,
 * INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
     (INCLUDING,
 * BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS
     OF USE,
 * DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
     THEORY OF
 * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
     NEGLIGENCE
 * OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
     IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 */

package cy.com.util;

/**
 * SpringDynamics is a Dynamics object that uses friction and spring
     physics to
 * snap to boundaries and give a natural and organic dynamic.
 */
public class SpringDynamics extends Dynamics {

    /** Friction factor */
    private float mFriction;

    /** Spring stiffness factor */
    private float mStiffness;

    /** Spring damping */
    private float mDamping;

    /**
     * Set friction parameter, friction physics are applied when inside of
         snap
     * bounds.
     *
     * @param friction Friction factor
     */
    public void setFriction(float friction) {
        mFriction = friction;
    }

    /**
     * Set spring parameters, spring physics are applied when outside of
```

```
             snap
         * bounds.
         *
         * @param stiffness Spring stiffness
         * @param dampingRatio Damping ratio, < 1 underdamped, > 1 overdamped
         */
        public void setSpring(float stiffness, float dampingRatio) {
            mStiffness = stiffness;
            mDamping = dampingRatio * 2 * (float)Math.sqrt(stiffness);
        }

        /**
         * Calculate acceleration at the current state
         *
         * @return Current acceleration
         */
        private float calculateAcceleration() {
            float acceleration;

            final float distanceFromLimit = getDistanceToLimit();
            if (distanceFromLimit != 0) {
                acceleration = distanceFromLimit * mStiffness - mDamping *
                    mVelocity;
            } else {
                acceleration = -mFriction * mVelocity;
            }

            return acceleration;
        }

        @Override
        protected void onUpdate(int dt) {
            // Calculate dt in seconds as float
            final float fdt = dt / 1000f;

            // Calculate current acceleration
            final float a = calculateAcceleration();

            // Calculate next position based on current velocity and
                acceleration
            mPosition += mVelocity * fdt + .5f * a * fdt * fdt;

            // Update velocity
            mVelocity += a * fdt;
        }

    }
```

```
/*
* Copyright (c) 2011, KIOS Research Center and Data Management Systems Lab
    ,
* University of Cyprus. All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
    modification,
* are permitted provided that the following conditions are met:
*
*    * Redistributions of source code must retain the above copyright
    notice, this
*      list of conditions and the following disclaimer.
*    * Redistributions in binary form must reproduce the above copyright
    notice,
*      this list of conditions and the following disclaimer in the
    documentation
*      and/or other materials provided with the distribution.
*    * Neither the name of the Sony Ericsson Mobile Communication AB nor
    the names
*      of its contributors may be used to endorse or promote products
    derived from
*      this software without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
    IS" AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
    IMPLIED
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
    DISCLAIMED.
* IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY
    DIRECT,
* INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
    (INCLUDING,
* BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
    USE,
* DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
    THEORY OF
* LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
    NEGLIGENCE
* OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
    IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
*/

/*
 * Copyright (c) 2010, Sony Ericsson Mobile Communication AB. All rights
     reserved.
 *
 * Redistribution and use in source and binary forms, with or without
     modification,
 * are permitted provided that the following conditions are met:
 *
 *    * Redistributions of source code must retain the above copyright
     notice, this
 *      list of conditions and the following disclaimer.
 *    * Redistributions in binary form must reproduce the above copyright
     notice,
 *      this list of conditions and the following disclaimer in the
     documentation
```



```
 *      and/or other materials provided with the distribution.
 *    * Neither the name of the Sony Ericsson Mobile Communication AB nor
     the names
 *      of its contributors may be used to endorse or promote products
     derived from
 *      this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
     IS" AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
     IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
     DISCLAIMED.
 * IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
     ANY DIRECT,
 * INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
     (INCLUDING,
 * BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS
     OF USE,
 * DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
     THEORY OF
 * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
     NEGLIGENCE
 * OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
     IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 */

package cy.com.zoom;

import java.util.Observable;

/**
 * Class that holds the aspect quotient, defined as content aspect ratio
     divided
 * by view aspect ratio.
 */
public class AspectQuotient extends Observable {

    /**
     * Aspect quotient
     */
    private float mAspectQuotient;

    // Public methods

    /**
     * Gets aspect quotient
     *
     * @return The aspect quotient
     */
    public float get() {
        return mAspectQuotient;
    }

    /**
     * Updates and recalculates aspect quotient based on supplied view and
     * content dimensions.
     *
```

```
         * @param viewWidth
         *            Width of view
         * @param viewHeight
         *            Height of view
         * @param contentWidth
         *            Width of content
         * @param contentHeight
         *            Height of content
         */
        public void updateAspectQuotient(float viewWidth, float viewHeight,
            float contentWidth, float contentHeight) {
            final float aspectQuotient = (contentWidth / contentHeight) /
                (viewWidth / viewHeight);

            if (aspectQuotient != mAspectQuotient) {
                mAspectQuotient = aspectQuotient;
                setChanged();
            }
        }

    }
```

Airplace Tracker code


```
/*
* Copyright (c) 2011, KIOS Research Center and Data Management Systems Lab
    ,
* University of Cyprus. All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
    modification,
* are permitted provided that the following conditions are met:
*
*    * Redistributions of source code must retain the above copyright
    notice, this
*      list of conditions and the following disclaimer.
*    * Redistributions in binary form must reproduce the above copyright
    notice,
*      this list of conditions and the following disclaimer in the
    documentation
*      and/or other materials provided with the distribution.
*    * Neither the name of the Sony Ericsson Mobile Communication AB nor
    the names
*      of its contributors may be used to endorse or promote products
    derived from
*      this software without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
    IS" AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
    IMPLIED
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
    DISCLAIMED.
* IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY
    DIRECT,
* INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
    (INCLUDING,
* BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
    USE,
* DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
    THEORY OF
* LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
    NEGLIGENCE
* OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
    IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
*/

package cy.com.zoom;

import java.util.Observable;

import android.graphics.PointF;

public class ClickPoint extends Observable {

    private PointF clickPoint;

    public ClickPoint(float x, float y) {
        clickPoint = new PointF(x, y);
    }

    public PointF get() {
```



```
        return clickPoint;
    }

    public void setClickPoint(float x, float y) {

        if (x != clickPoint.x || y != clickPoint.y) {
            clickPoint.x = x;
            clickPoint.y = y;
            setChanged();
        }
    }
}
```

```
/*
* Copyright (c) 2011, KIOS Research Center and Data Management Systems Lab
    ,
* University of Cyprus. All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
    modification,
* are permitted provided that the following conditions are met:
*
*    * Redistributions of source code must retain the above copyright
    notice, this
*      list of conditions and the following disclaimer.
*    * Redistributions in binary form must reproduce the above copyright
    notice,
*      this list of conditions and the following disclaimer in the
    documentation
*      and/or other materials provided with the distribution.
*    * Neither the name of the Sony Ericsson Mobile Communication AB nor
    the names
*      of its contributors may be used to endorse or promote products
    derived from
*      this software without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
    IS" AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
    IMPLIED
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
    DISCLAIMED.
* IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY
    DIRECT,
* INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
    (INCLUDING,
* BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
    USE,
* DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
    THEORY OF
* LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
    NEGLIGENCE
* OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
    IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
*/

/*
 * Copyright (c) 2010, Sony Ericsson Mobile Communication AB. All rights
     reserved.
 *
 * Redistribution and use in source and binary forms, with or without
     modification,
 * are permitted provided that the following conditions are met:
 *
 *    * Redistributions of source code must retain the above copyright
     notice, this
 *      list of conditions and the following disclaimer.
 *    * Redistributions in binary form must reproduce the above copyright
     notice,
 *      this list of conditions and the following disclaimer in the
     documentation
```



```
 *      and/or other materials provided with the distribution.
 *    * Neither the name of the Sony Ericsson Mobile Communication AB nor
     the names
 *      of its contributors may be used to endorse or promote products
     derived from
 *      this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
     IS" AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
     IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
     DISCLAIMED.
 * IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
     ANY DIRECT,
 * INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
     (INCLUDING,
 * BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS
     OF USE,
 * DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
     THEORY OF
 * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
     NEGLIGENCE
 * OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
     IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 */

package cy.com.zoom;

import cy.com.util.SpringDynamics;

import android.os.Handler;
import android.os.SystemClock;

import java.util.Observable;
import java.util.Observer;

/**
 * The DynamicZoomControl is responsible for controlling a ZoomState. It
     makes
 * sure that pan movement follows the finger, that limits are satisfied
     and that
 * we can zoom into specific positions.
 *
 * In order to implement these control mechanisms access to certain
     content and
 * view state data is required which is made possible through the
 * ZoomContentViewState.
 */
public class DynamicZoomControl implements Observer {

    /** Minimum zoom level limit */
    private static final float MIN_ZOOM = 1;

    /** Maximum zoom level limit */
    private static final float MAX_ZOOM = 16;

    /** Velocity tolerance for calculating if dynamic state is resting */
```

```
    private static final float REST_VELOCITY_TOLERANCE = 0.004f;

    /** Position tolerance for calculating if dynamic state is resting */
    private static final float REST_POSITION_TOLERANCE = 0.01f;

    /** Target FPS when animating behavior such as fling and snap to */
    private static final int FPS = 50;

    /** Factor applied to pan motion outside of pan snap limits. */
    private static final float PAN_OUTSIDE_SNAP_FACTOR = .4f;

    /** Zoom state under control */
    private final ZoomState mState = new ZoomState();

    /** Object holding aspect quotient of view and content */
    private AspectQuotient mAspectQuotient;

    /**
     * Dynamics object for creating dynamic fling and snap to behavior for
         pan
     * in x-dimension.
     */
    private final SpringDynamics mPanDynamicsX = new SpringDynamics();

    /**
     * Dynamics object for creating dynamic fling and snap to behavior for
         pan
     * in y-dimension.
     */
    private final SpringDynamics mPanDynamicsY = new SpringDynamics();

    /** Minimum snap to position for pan in x-dimension */
    private float mPanMinX;

    /** Maximum snap to position for pan in x-dimension */
    private float mPanMaxX;

    /** Minimum snap to position for pan in y-dimension */
    private float mPanMinY;

    /** Maximum snap to position for pan in y-dimension */
    private float mPanMaxY;

    /** Handler for posting runnables */
    private final Handler mHandler = new Handler();

    /** Creates new zoom control */
    public DynamicZoomControl() {
        mPanDynamicsX.setFriction(2f);
        mPanDynamicsY.setFriction(2f);
        mPanDynamicsX.setSpring(50f, 1f);
        mPanDynamicsY.setSpring(50f, 1f);
    }

    /**
     * Set reference object holding aspect quotient
     *
     * @param aspectQuotient
     *            Object holding aspect quotient
```



```
     */
    public void setAspectQuotient(AspectQuotient aspectQuotient) {
        if (mAspectQuotient != null) {
            mAspectQuotient.deleteObserver(this);
        }

        mAspectQuotient = aspectQuotient;
        mAspectQuotient.addObserver(this);
    }

    /**
     * Get zoom state being controlled
     *
     * @return The zoom state
     */
    public ZoomState getZoomState() {
        return mState;
    }

    /**
     * Zoom
     *
     * @param f
     *            Factor of zoom to apply
     * @param x
     *            X-coordinate of invariant position
     * @param y
     *            Y-coordinate of invariant position
     */
    public void zoom(float f, float x, float y) {
        final float aspectQuotient = mAspectQuotient.get();

        final float prevZoomX = mState.getZoomX(aspectQuotient);
        final float prevZoomY = mState.getZoomY(aspectQuotient);

        mState.setZoom(mState.getZoom() * f);
        limitZoom();

        final float newZoomX = mState.getZoomX(aspectQuotient);
        final float newZoomY = mState.getZoomY(aspectQuotient);

        // Pan to keep x and y coordinate invariant
        mState.setPanX(mState.getPanX() + (x - .5f) * (1f / prevZoomX - 1f
            / newZoomX));
        mState.setPanY(mState.getPanY() + (y - .5f) * (1f / prevZoomY - 1f
            / newZoomY));

        updatePanLimits();

        mState.notifyObservers();
    }

    /**
     * Pan
     *
     * @param dx
     *            Amount to pan in x-dimension
     * @param dy
     *            Amount to pan in y-dimension
```

```
     */
    public void pan(float dx, float dy) {
        final float aspectQuotient = mAspectQuotient.get();

        dx /= mState.getZoomX(aspectQuotient);
        dy /= mState.getZoomY(aspectQuotient);

        if (mState.getPanX() > mPanMaxX && dx > 0 || mState.getPanX() <
            mPanMinX && dx < 0) {
            dx *= PAN_OUTSIDE_SNAP_FACTOR;
        }
        if (mState.getPanY() > mPanMaxY && dy > 0 || mState.getPanY() <
            mPanMinY && dy < 0) {
            dy *= PAN_OUTSIDE_SNAP_FACTOR;
        }

        final float newPanX = mState.getPanX() + dx;
        final float newPanY = mState.getPanY() + dy;

        mState.setPanX(newPanX);
        mState.setPanY(newPanY);

        mState.notifyObservers();
    }

    /**
     * Runnable that updates dynamics state
     */
    private final Runnable mUpdateRunnable = new Runnable() {
        public void run() {
            final long startTime = SystemClock.uptimeMillis();
            mPanDynamicsX.update(startTime);
            mPanDynamicsY.update(startTime);
            final boolean isAtRest = mPanDynamicsX.isAtRest
                (REST_VELOCITY_TOLERANCE, REST_POSITION_TOLERANCE)
                    && mPanDynamicsY.isAtRest(REST_VELOCITY_TOLERANCE,
                        REST_POSITION_TOLERANCE);
            mState.setPanX(mPanDynamicsX.getPosition());
            mState.setPanY(mPanDynamicsY.getPosition());

            if (!isAtRest) {
                final long stopTime = SystemClock.uptimeMillis();
                mHandler.postDelayed(mUpdateRunnable, 1000 / FPS -
                    (stopTime - startTime));
            }

            mState.notifyObservers();
        }
    };

    /**
     * Release control and start pan fling animation
     *
     * @param vx
     *            Velocity in x-dimension
     * @param vy
     *            Velocity in y-dimension
     */
    public void startFling(float vx, float vy) {
```



```
        final float aspectQuotient = mAspectQuotient.get();
        final long now = SystemClock.uptimeMillis();

        mPanDynamicsX.setState(mState.getPanX(), vx / mState.getZoomX
            (aspectQuotient), now);
        mPanDynamicsY.setState(mState.getPanY(), vy / mState.getZoomY
            (aspectQuotient), now);

        mPanDynamicsX.setMinPosition(mPanMinX);
        mPanDynamicsX.setMaxPosition(mPanMaxX);
        mPanDynamicsY.setMinPosition(mPanMinY);
        mPanDynamicsY.setMaxPosition(mPanMaxY);

        mHandler.post(mUpdateRunnable);
    }

    /**
     * Stop fling animation
     */
    public void stopFling() {
        mHandler.removeCallbacks(mUpdateRunnable);
    }

    /**
     * Help function to figure out max delta of pan from center position.
     *
     * @param zoom
     *            Zoom value
     * @return Max delta of pan
     */
    private float getMaxPanDelta(float zoom) {
        return Math.max(0f, .5f * ((zoom - 1) / zoom));
    }

    /**
     * Force zoom to stay within limits
     */
    private void limitZoom() {
        if (mState.getZoom() < MIN_ZOOM) {
            mState.setZoom(MIN_ZOOM);
        } else if (mState.getZoom() > MAX_ZOOM) {
            mState.setZoom(MAX_ZOOM);
        }
    }

    /**
     * Update limit values for pan
     */
    private void updatePanLimits() {
        final float aspectQuotient = mAspectQuotient.get();

        final float zoomX = mState.getZoomX(aspectQuotient);
        final float zoomY = mState.getZoomY(aspectQuotient);

        mPanMinX = .5f - getMaxPanDelta(zoomX);
        mPanMaxX = .5f + getMaxPanDelta(zoomX);
        mPanMinY = .5f - getMaxPanDelta(zoomY);
        mPanMaxY = .5f + getMaxPanDelta(zoomY);
    }
```

```
        // Observable interface implementation

        public void update(Observable observable, Object data) {
            limitZoom();
            updatePanLimits();
        }

    }
```

```
/*
* Copyright (c) 2011, KIOS Research Center and Data Management Systems Lab
    ,
* University of Cyprus. All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
    modification,
* are permitted provided that the following conditions are met:
*
*    * Redistributions of source code must retain the above copyright
    notice, this
*      list of conditions and the following disclaimer.
*    * Redistributions in binary form must reproduce the above copyright
    notice,
*      this list of conditions and the following disclaimer in the
    documentation
*      and/or other materials provided with the distribution.
*    * Neither the name of the Sony Ericsson Mobile Communication AB nor
    the names
*      of its contributors may be used to endorse or promote products
    derived from
*      this software without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
    IS" AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
    IMPLIED
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
    DISCLAIMED.
* IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY
    DIRECT,
* INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
    (INCLUDING,
* BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
    USE,
* DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
    THEORY OF
* LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
    NEGLIGENCE
* OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
    IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
*/

/*
 * Copyright (c) 2010, Sony Ericsson Mobile Communication AB. All rights
     reserved.
 *
 * Redistribution and use in source and binary forms, with or without
     modification,
 * are permitted provided that the following conditions are met:
 *
 *    * Redistributions of source code must retain the above copyright
     notice, this
 *      list of conditions and the following disclaimer.
 *    * Redistributions in binary form must reproduce the above copyright
     notice,
 *      this list of conditions and the following disclaimer in the
     documentation
```



```
 *      and/or other materials provided with the distribution.
 *    * Neither the name of the Sony Ericsson Mobile Communication AB nor
     the names
 *      of its contributors may be used to endorse or promote products
     derived from
 *      this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
     IS" AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
     IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
     DISCLAIMED.
 * IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
     ANY DIRECT,
 * INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
     (INCLUDING,
 * BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS
     OF USE,
 * DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
     THEORY OF
 * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
     NEGLIGENCE
 * OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
     IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 */

package cy.com.zoom;

import java.util.ArrayList;
import java.util.Observable;
import java.util.Observer;

import cy.com.airplace.BooleanObservable;
import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.PointF;
import android.graphics.Rect;
import android.graphics.Paint.Style;
import android.util.AttributeSet;
import android.view.View;

/**
 * View capable of drawing an image at different zoom state levels
 */
public class ImageZoomView extends View implements Observer {

    /** Paint object used when drawing bitmap. */
    private final Paint mPaint = new Paint(Paint.FILTER_BITMAP_FLAG);

    /** Rectangle used (and re-used) for cropping source image. */
    private final Rect mRectSrc = new Rect();

    /** Rectangle used (and re-used) for specifying drawing area on canvas
        . */
```

```
        private final Rect mRectDst = new Rect();

        /** Object holding aspect quotient */
        private final AspectQuotient mAspectQuotient = new AspectQuotient();

        /** The bitmap that we're zooming in, and drawing on the screen. */
        private Bitmap mBitmap;

        /** State of the zoom. */
        private ZoomState mState;

        // Public methods
        private ClickPoint curClick;
        private ArrayList<PointF> points = new ArrayList<PointF>();
        private final Paint p = new Paint();
        private BooleanObservable trackMe;

        /**
         * Constructor
         */
        public ImageZoomView(Context context, AttributeSet attrs) {
            super(context, attrs);
        }

        public void setCurClick(ClickPoint click) {

            if (curClick != null) {
                curClick.deleteObserver(this);
            }
            curClick = click;
            curClick.addObserver(this);

            invalidate();
        }

        /**
         * Set image bitmap
         *
         * @param bitmap
         *            The bitmap to view and zoom into
         */
        public void setImage(Bitmap bitmap) {
            mBitmap = bitmap;

            mAspectQuotient.updateAspectQuotient(getWidth(), getHeight(),
                mBitmap.getWidth(), mBitmap.getHeight());
            mAspectQuotient.notifyObservers();
            points.clear();
            invalidate();
        }

        /**
         * Set object holding the zoom state that should be used
         *
         * @param state
         *            The zoom state
         */
        public void setZoomState(ZoomState state) {
            if (mState != null) {
```



```
                mState.deleteObserver(this);
            }

            mState = state;
            mState.addObserver(this);

            invalidate();
        }

        /**
         * Gets reference to object holding aspect quotient
         *
         * @return Object holding aspect quotient
         */
        public AspectQuotient getAspectQuotient() {
            return mAspectQuotient;
        }

        // Superclass overrides

        @Override
        protected void onDraw(Canvas canvas) {
            if (mBitmap != null && mState != null) {
                final float aspectQuotient = mAspectQuotient.get();

                final int viewWidth = getWidth();
                final int viewHeight = getHeight();
                final int bitmapWidth = mBitmap.getWidth();
                final int bitmapHeight = mBitmap.getHeight();

                final float panX = mState.getPanX();
                final float panY = mState.getPanY();
                final float zoomX = mState.getZoomX(aspectQuotient) *
                    viewWidth / bitmapWidth;
                final float zoomY = mState.getZoomY(aspectQuotient) *
                    viewHeight / bitmapHeight;

                // Setup source and destination rectangles
                mRectSrc.left = (int) (panX * bitmapWidth - viewWidth / (zoomX
                    * 2));
                mRectSrc.top = (int) (panY * bitmapHeight - viewHeight /
                    (zoomY * 2));
                mRectSrc.right = (int) (mRectSrc.left + viewWidth / zoomX);
                mRectSrc.bottom = (int) (mRectSrc.top + viewHeight / zoomY);

                mRectDst.left = getLeft();
                mRectDst.top = getTop();
                mRectDst.right = getRight();
                mRectDst.bottom = getBottom();

                // Adjust source rectangle so that it fits within the source
                    image.
                if (mRectSrc.left < 0) {
                    mRectDst.left += -mRectSrc.left * zoomX;
                    mRectSrc.left = 0;
                }
                if (mRectSrc.right > bitmapWidth) {
                    mRectDst.right -= (mRectSrc.right - bitmapWidth) * zoomX;
                    mRectSrc.right = bitmapWidth;
```

```
            }
            if (mRectSrc.top < 0) {
                mRectDst.top += -mRectSrc.top * zoomY;
                mRectSrc.top = 0;
            }
            if (mRectSrc.bottom > bitmapHeight) {
                mRectDst.bottom -= (mRectSrc.bottom - bitmapHeight) *
                    zoomY;
                mRectSrc.bottom = bitmapHeight;
            }

            float Xana = (float) (mRectSrc.right - mRectSrc.left) / (float
                ) (mRectDst.right - mRectDst.left);
            float Yana = (float) (mRectSrc.bottom - mRectSrc.top) / (float
                ) (mRectDst.bottom - mRectDst.top);

            canvas.drawBitmap(mBitmap, mRectSrc, mRectDst, mPaint);

            if (curClick.get().x >= 0 && curClick.get().x <= bitmapWidth &
                & curClick.get().y >= 0 && curClick.get().y <=
                bitmapHeight) {

                boolean isAlreadyMarked = false;
                PointF point = null;
                // Check if this point is already marked
                for (int i = 0; i < points.size(); ++i) {
                    point = (PointF) points.get(i);

                    if (curClick.get().equals(point.x, point.y)) {
                        isAlreadyMarked = true;

                        PointF pointLast = (PointF) points.get(points.size
                            () - 1);

                        if (pointLast != point) {
                            point = (PointF) points.set(i, pointLast);
                            points.set(points.size() - 1, point);
                        }
                        break;
                    }
                }

                if (!isAlreadyMarked)
                    points.add(new PointF(curClick.get().x, curClick.get()
                        .y));
            }

            p.setColor(Color.RED);
            int radius = 4;

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

                if (i == points.size() - 1) {
                    if (trackMe.get())
                        p.setColor(Color.GREEN);
                    else
                        p.setColor(0xff47e3ff);
                    radius = 7;
                }
```



```
                PointF temp = points.get(i);

                if (temp.x != -1 && temp.y != -1 && temp.x >= mRectSrc.
                    left - 5 && temp.x <= mRectSrc.right + 5 && temp.y >=
                    mRectSrc.top - 5
                        && temp.y <= mRectSrc.bottom + 5) {

                    canvas.drawCircle((temp.x - mRectSrc.left) / Xana +
                        mRectDst.left, (temp.y - mRectSrc.top) / Yana +
                        mRectDst.top, radius, this.p);

                    if (i == points.size() - 1) {
                        p.setColor(Color.RED);
                        canvas.drawCircle((temp.x - mRectSrc.left) / Xana
                            + mRectDst.left, (temp.y - mRectSrc.top) /
                            Yana + mRectDst.top, 2, this.p);
                        p.setColor(Color.BLUE);
                        p.setStyle(Style.STROKE);
                        canvas.drawCircle((temp.x - mRectSrc.left) / Xana
                            + mRectDst.left, (temp.y - mRectSrc.top) /
                            Yana + mRectDst.top, radius,
                                this.p);
                        p.setStyle(Style.FILL_AND_STROKE);
                    }
                }
            }
        }
    }

    @Override
    protected void onLayout(boolean changed, int left, int top, int right,
        int bottom) {
        super.onLayout(changed, left, top, right, bottom);

        if (mBitmap != null) {
            mAspectQuotient.updateAspectQuotient(right - left, bottom -
                top, mBitmap.getWidth(), mBitmap.getHeight());
            mAspectQuotient.notifyObservers();
        }
    }

    // implements Observer
    public void update(Observable observable, Object data) {
        invalidate();
    }

    public void clearPoints() {
        points.clear();
        invalidate();
    }

    public void setTrackMe(BooleanObservable trackMe) {
        if (this.trackMe != null) {
            this.trackMe.deleteObserver(this);
        }
        this.trackMe = trackMe;
        this.trackMe.addObserver(this);
    }
```

```
    }
```

```
/*
* Copyright (c) 2011, KIOS Research Center and Data Management Systems Lab
    ,
* University of Cyprus. All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
    modification,
* are permitted provided that the following conditions are met:
*
*    * Redistributions of source code must retain the above copyright
    notice, this
*      list of conditions and the following disclaimer.
*    * Redistributions in binary form must reproduce the above copyright
    notice,
*      this list of conditions and the following disclaimer in the
    documentation
*      and/or other materials provided with the distribution.
*    * Neither the name of the Sony Ericsson Mobile Communication AB nor
    the names
*      of its contributors may be used to endorse or promote products
    derived from
*      this software without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
    IS" AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
    IMPLIED
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
    DISCLAIMED.
* IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY
    DIRECT,
* INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
    (INCLUDING,
* BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
    USE,
* DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
    THEORY OF
* LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
    NEGLIGENCE
* OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
    IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
*/

/*
 * Copyright (c) 2010, Sony Ericsson Mobile Communication AB. All rights
     reserved.
 *
 * Redistribution and use in source and binary forms, with or without
     modification,
 * are permitted provided that the following conditions are met:
 *
 *    * Redistributions of source code must retain the above copyright
     notice, this
 *      list of conditions and the following disclaimer.
 *    * Redistributions in binary form must reproduce the above copyright
     notice,
 *      this list of conditions and the following disclaimer in the
     documentation
```



```
 *      and/or other materials provided with the distribution.
 *    * Neither the name of the Sony Ericsson Mobile Communication AB nor
     the names
 *      of its contributors may be used to endorse or promote products
     derived from
 *      this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
     IS" AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
     IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
     DISCLAIMED.
 * IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
     ANY DIRECT,
 * INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
     (INCLUDING,
 * BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS
     OF USE,
 * DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
     THEORY OF
 * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
     NEGLIGENCE
 * OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
     IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 */

package cy.com.zoom;

import android.content.Context;
import android.os.Vibrator;
import android.view.MotionEvent;
import android.view.VelocityTracker;
import android.view.View;
import android.view.ViewConfiguration;

/**
 * Listener for controlling zoom state through touch events
 */
public class LongPressZoomListener implements View.OnTouchListener {

    /**
     * Enum defining listener modes. Before the view is touched the
         listener is
     * in the UNDEFINED mode. Once touch starts it can enter either one of
         the
     * other two modes: If the user scrolls over the view the listener
         will
     * enter PAN mode, if the user lets his finger rest and makes a
         longpress
     * the listener will enter ZOOM mode.
     */
    private enum Mode {
        UNDEFINED, PAN, ZOOM
    }

    /** Time of tactile feedback vibration when entering zoom mode */
    private static final long VIBRATE_TIME = 50;
```

```
    /** Current listener mode */
    private Mode mMode = Mode.UNDEFINED;

    /** Zoom control to manipulate */
    private DynamicZoomControl mZoomControl;

    /** X-coordinate of previously handled touch event */
    private float mX;

    /** Y-coordinate of previously handled touch event */
    private float mY;

    /** X-coordinate of latest down event */
    private float mDownX;

    /** Y-coordinate of latest down event */
    private float mDownY;

    /** Velocity tracker for touch events */
    private VelocityTracker mVelocityTracker;

    /** Distance touch can wander before we think it's scrolling */
    private final int mScaledTouchSlop;

    /** Duration in ms before a press turns into a long press */
    private final int mLongPressTimeout;

    /** Vibrator for tactile feedback */
    private final Vibrator mVibrator;

    /** Maximum velocity for fling */
    private final int mScaledMaximumFlingVelocity;

    /**
     * Creates a new instance
     *
     * @param context
     *            Application context
     */
    public LongPressZoomListener(Context context) {
        mLongPressTimeout = ViewConfiguration.getLongPressTimeout();
        mScaledTouchSlop = ViewConfiguration.get(context).
            getScaledTouchSlop();
        mScaledMaximumFlingVelocity = ViewConfiguration.get(context).
            getScaledMaximumFlingVelocity();
        mVibrator = (Vibrator) context.getSystemService("vibrator");
    }

    /**
     * Sets the zoom control to manipulate
     *
     * @param control
     *            Zoom control
     */
    public void setZoomControl(DynamicZoomControl control) {
        mZoomControl = control;
    }
```



```
    /**
     * Runnable that enters zoom mode
     */
    private final Runnable mLongPressRunnable = new Runnable() {
        public void run() {
            mMode = Mode.ZOOM;
            mVibrator.vibrate(VIBRATE_TIME);
        }
    };

    // implements View.OnTouchListener
    public boolean onTouch(View v, MotionEvent event) {
        final int action = event.getAction();
        final float x = event.getX();
        final float y = event.getY();

        if (mVelocityTracker == null) {
            mVelocityTracker = VelocityTracker.obtain();
        }
        mVelocityTracker.addMovement(event);

        switch (action) {
        case MotionEvent.ACTION_DOWN:
            mZoomControl.stopFling();
            v.postDelayed(mLongPressRunnable, mLongPressTimeout);
            mDownX = x;
            mDownY = y;
            mX = x;
            mY = y;
            break;

        case MotionEvent.ACTION_MOVE: {
            final float dx = (x - mX) / v.getWidth();
            final float dy = (y - mY) / v.getHeight();

            if (mMode == Mode.ZOOM) {
                mZoomControl.zoom((float) Math.pow(20, -dy), mDownX / v.
                    getWidth(), mDownY / v.getHeight());
            } else if (mMode == Mode.PAN) {
                mZoomControl.pan(-dx, -dy);
            } else {
                final float scrollX = mDownX - x;
                final float scrollY = mDownY - y;

                final float dist = (float) Math.sqrt(scrollX * scrollX +
                    scrollY * scrollY);

                if (dist >= mScaledTouchSlop) {
                    v.removeCallbacks(mLongPressRunnable);
                    mMode = Mode.PAN;
                }
            }

            mX = x;
            mY = y;
            break;
        }

        case MotionEvent.ACTION_UP:
```

```
                if (mMode == Mode.PAN) {
                    mVelocityTracker.computeCurrentVelocity(1000,
                        mScaledMaximumFlingVelocity);
                    mZoomControl.startFling(-mVelocityTracker.getXVelocity() /
                        v.getWidth(), -mVelocityTracker.getYVelocity() / v.
                        getHeight());
                } else {
                    mZoomControl.startFling(0, 0);
                }
                mVelocityTracker.recycle();
                mVelocityTracker = null;
                v.removeCallbacks(mLongPressRunnable);
                mMode = Mode.UNDEFINED;

                break;


            case MotionEvent.ACTION_POINTER_DOWN:
                mMode = Mode.ZOOM;
                break;
            case MotionEvent.ACTION_POINTER_UP:
                mMode = Mode.UNDEFINED;
                break;

            default:
                mVelocityTracker.recycle();
                mVelocityTracker = null;
                v.removeCallbacks(mLongPressRunnable);
                mMode = Mode.UNDEFINED;
                break;

            }

            return true;
        }


    }
```

```
/*
* Copyright (c) 2011, KIOS Research Center and Data Management Systems Lab
    ,
* University of Cyprus. All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
    modification,
* are permitted provided that the following conditions are met:
*
*    * Redistributions of source code must retain the above copyright
    notice, this
*      list of conditions and the following disclaimer.
*    * Redistributions in binary form must reproduce the above copyright
    notice,
*      this list of conditions and the following disclaimer in the
    documentation
*      and/or other materials provided with the distribution.
*    * Neither the name of the Sony Ericsson Mobile Communication AB nor
    the names
*      of its contributors may be used to endorse or promote products
    derived from
*      this software without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
    IS" AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
    IMPLIED
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
    DISCLAIMED.
* IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY
    DIRECT,
* INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
    (INCLUDING,
* BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
    USE,
* DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
    THEORY OF
* LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
    NEGLIGENCE
* OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
    IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
*/

/*
 * Copyright (c) 2010, Sony Ericsson Mobile Communication AB. All rights
     reserved.
 *
 * Redistribution and use in source and binary forms, with or without
     modification,
 * are permitted provided that the following conditions are met:
 *
 *    * Redistributions of source code must retain the above copyright
     notice, this
 *      list of conditions and the following disclaimer.
 *    * Redistributions in binary form must reproduce the above copyright
     notice,
 *      this list of conditions and the following disclaimer in the
     documentation
```



```
 *      and/or other materials provided with the distribution.
 *    * Neither the name of the Sony Ericsson Mobile Communication AB nor
     the names
 *      of its contributors may be used to endorse or promote products
     derived from
 *      this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
     IS" AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
     IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
     DISCLAIMED.
 * IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
     ANY DIRECT,
 * INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
     (INCLUDING,
 * BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS
     OF USE,
 * DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
     THEORY OF
 * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
     NEGLIGENCE
 * OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
     IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 */

package cy.com.zoom;

import java.util.Observable;

/**
 * A ZoomState holds zoom and pan values and allows the user to read and
     listen
 * to changes. Clients that modify ZoomState should call notifyObservers()
 */
public class ZoomState extends Observable {
    /**
     * Zoom level A value of 1.0 means the content fits the view.
     */
    private float mZoom;

    /**
     * Pan position x-coordinate X-coordinate of zoom window center
         position,
     * relative to the width of the content.
     */
    private float mPanX;

    /**
     * Pan position y-coordinate Y-coordinate of zoom window center
         position,
     * relative to the height of the content.
     */
    private float mPanY;

    // Public methods
```

```
        /**
         * Get current x-pan
         *
         * @return current x-pan
         */
        public float getPanX() {
            return mPanX;
        }

        /**
         * Get current y-pan
         *
         * @return Current y-pan
         */
        public float getPanY() {
            return mPanY;
        }

        /**
         * Get current zoom value
         *
         * @return Current zoom value
         */
        public float getZoom() {
            return mZoom;
        }

        /**
         * Help function for calculating current zoom value in x-dimension
         *
         * @param aspectQuotient (Aspect ratio content) / (Aspect ratio view)
         * @return Current zoom value in x-dimension
         */
        public float getZoomX(float aspectQuotient) {
            return Math.min(mZoom, mZoom * aspectQuotient);
        }

        /**
         * Help function for calculating current zoom value in y-dimension
         *
         * @param aspectQuotient (Aspect ratio content) / (Aspect ratio view)
         * @return Current zoom value in y-dimension
         */
        public float getZoomY(float aspectQuotient) {
            return Math.min(mZoom, mZoom / aspectQuotient);
        }

        /**
         * Set pan-x
         *
         * @param panX Pan-x value to set
         */
        public void setPanX(float panX) {
            if (panX != mPanX) {
                mPanX = panX;
                setChanged();
            }
        }
```



```
        /**
         * Set pan-y
         *
         * @param panY Pan-y value to set
         */
        public void setPanY(float panY) {
            if (panY != mPanY) {
                mPanY = panY;
                setChanged();
            }
        }

        /**
         * Set zoom
         *
         * @param zoom Zoom value to set
         */
        public void setZoom(float zoom) {
            if (zoom != mZoom) {
                mZoom = zoom;
                setChanged();
            }
        }

    }
```

Παράρτημα Β

Αναλυτική Παρουσίαση Πειραματικών Αποτελεσμάτων

Σε αυτό το παράρτημα θα παρουσιάσουμε αναλυτικά τα αποτελέσματα που εξάγαμε από τον προσωμειωτή. Τα αποτελέσματα θα τα παρουσιάσουμε για κάθε προσέγγιση ξεχωριστά για κάθε dataset. Επειδή το πείραμα έγινε για εκατοντάδες τοπικοποιήσεις, για κάθε φορά (στο UCY dataset για παράδειγμα 5000), θα παρουσιάσουμε ένα δείγμα 100 τοπικοποιήσεων για κάθε προσέγγιση σε κάθε ένα dataset. Οι γραφικές παραστάσεις που παρουσιάσαμε στο Κεφάλαιο 6 είναι βάσει των στοιχείων που εξάγαμε από τον προσομοιωτή και παρουσιάζουμε σε αυτό το παράρτημα.

UCY dataset

CRA προσέγγιση

Bytes Client Send	Bytes Server Send	Client CPU time	Server CPU time	Msgs Sent	Msgs Rsv
2853.0	10.0	N/A	42	1	1
2853.0	10.0	N/A	0.0060	1	1
2853.0	10.0	N/A	0.0050	1	1
2853.0	10.0	N/A	0.0060	1	1
2853.0	10.0	N/A	0.0020	1	1
2853.0	10.0	N/A	0.0040	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0040	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0040	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0040	1	1
2853.0	10.0	N/A	0.0040	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0040	1	1

2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0020	1	1
2853.0	10.0	N/A	0.0020	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0040	1	1
2853.0	10.0	N/A	0.0040	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0040	1	1
2853.0	10.0	N/A	0.0020	1	1
2853.0	10.0	N/A	0.0040	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0040	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0050	1	1
2853.0	10.0	N/A	0.0040	1	1
2853.0	10.0	N/A	0.0040	1	1
2853.0	10.0	N/A	0.0050	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0050	1	1
2853.0	10.0	N/A	0.0040	1	1
2853.0	10.0	N/A	0.0040	1	1
2853.0	10.0	N/A	0.0060	1	1
2853.0	10.0	N/A	0.0040	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0040	1	1
2853.0	10.0	N/A	0.0050	1	1
2853.0	10.0	N/A	0.0040	1	1
2853.0	10.0	N/A	0.0060	1	1
2853.0	10.0	N/A	0.0020	1	1

2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0020	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0040	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0020	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0020	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0040	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0040	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0040	1	1
2853.0	10.0	N/A	0.0020	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0020	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0060	1	1
2853.0	10.0	N/A	0.0020	1	1
2853.0	10.0	N/A	0.0020	1	1
2853.0	10.0	N/A	0.0040	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0020	1	1

2853.0	10.0	N/A	0.0020	1	1
2853.0	10.0	N/A	0.0030	1	1
2853.0	10.0	N/A	0.0020	1	1
2853.0	10.0	N/A	0.0020	1	1
2853.0	10.0	N/A	0.0020	1	1
2853.0	10.0	N/A	0.0020	1	1
2853.0	10.0	N/A	0.0050	1	1
2853.0	10.0	N/A	0.0020	1	1
2853.0	10.0	N/A	0.0040	1	1
2853.0	10.0	N/A	0.0020	1	1
2853.0	10.0	N/A	0.0020	1	1

BMA προσέγγιση

Bytes Client Send	Bytes Server Send	Client CPU time	Server CPU time	Msgs Sent	Msgs Rsv
79.0	409203.0	0.1490000039339 0656	132	1	552
79.0	409203.0	0.2090000063180 9235	101	1	552
79.0	409203.0	0.0700000002980 2322	85	1	552
79.0	409203.0	0.0610000006854 53415	78	1	552
79.0	409203.0	0.0900000035762 7869	0.1	1	552
79.0	409203.0	0.0869999974966 0492	93	1	552
79.0	409203.0	0.0909999981522 5601	94	1	552
79.0	409203.0	0.0769999995827 6749	85	1	552
79.0	409203.0	0.1080000028014 183	81	1	552
79.0	409203.0	0.0810000002384 1858	81	1	552
79.0	409203.0	0.0700000002980 2322	76	1	552
79.0	409203.0	0.0850000008940 6967	81	1	552
79.0	409203.0	0.0680000036954 8798	99	1	552
79.0	409203.0	0.0679999962449 0738	83	1	552

79.0	409203.0	0.0649999976158 1421	76	1	552
79.0	409203.0	0.0589999966323 3757	72	1	552
79.0	409203.0	0.0719999969005 5847	0.07	1	552
79.0	409203.0	0.0560000017285 347	81	1	552
79.0	409203.0	0.0630000010132 7896	75	1	552
79.0	409203.0	0.1270000040531 1584	78	1	552
79.0	409203.0	0.0589999966323 3757	98	1	552
79.0	409203.0	0.0580000020563 60245	0.07	1	552
79.0	409203.0	0.0599999986588 9549	76	1	552
79.0	409203.0	0.0570000000298 0232	67	1	552
79.0	409203.0	0.0460000000894 0697	68	1	552
79.0	409203.0	0.0729999989271 164	67	1	552
79.0	409203.0	0.0810000002384 1858	67	1	552
79.0	409203.0	0.0850000008940 6967	86	1	552
79.0	409203.0	0.0610000006854 53415	102	1	552
79.0	409203.0	0.0780000016093 2541	79	1	552
79.0	409203.0	0.0579999983310 69946	85	1	552
79.0	409203.0	0.0500000007450 5806	67	1	552
79.0	409203.0	0.0509999990463 25684	66	1	552
79.0	409203.0	0.0619999989867 2104	63	1	552
79.0	409203.0	0.0579999983310 69946	72	1	552
79.0	409203.0	0.0509999990463 25684	69	1	552
79.0	409203.0	0.0729999989271 164	65	1	552
79.0	409203.0	0.0509999990463 25684	77	1	552
79.0	409203.0	0.0580000020563 60245	68	1	552

79.0	409203.0	0.0599999986588 9549	71	1	552
79.0	409203.0	0.0599999986588 9549	69	1	552
79.0	409203.0	0.0559999980032 444	72	1	552
79.0	409203.0	0.0570000000298 0232	65	1	552
79.0	409203.0	0.0829999968409 5383	78	1	552
79.0	409203.0	0.0539999976754 18854	83	1	552
79.0	409203.0	0.0539999976754 18854	72	1	552
79.0	409203.0	0.0549999997019 76776	72	1	552
79.0	409203.0	0.0580000020563 60245	66	1	552
79.0	409203.0	0.0589999966323 3757	69	1	552
79.0	409203.0	0.0680000036954 8798	0.07	1	552
79.0	409203.0	0.0599999986588 9549	96	1	552
79.0	409203.0	0.0509999990463 25684	68	1	552
79.0	409203.0	0.0560000017285 347	66	1	552
79.0	409203.0	0.0509999990463 25684	68	1	552
79.0	409203.0	0.0820000022649 765	74	1	552
79.0	409203.0	0.0570000000298 0232	69	1	552
79.0	409203.0	0.0630000010132 7896	78	1	552
79.0	409203.0	0.0579999983310 69946	66	1	552
79.0	409203.0	0.0670000016689 3005	92	1	552
79.0	409203.0	0.0649999976158 1421	102	1	552
79.0	409203.0	0.0659999996423 7213	79	1	552
79.0	409203.0	0.0689999982714 653	98	1	552
79.0	409203.0	0.0690000057220 459	76	1	552
79.0	409203.0	0.0890000015497 2076	91	1	552

79.0	409203.0	0.0810000002384 1858	115	1	552
79.0	409203.0	0.0480000004172 3251	84	1	552
79.0	409203.0	0.0570000000298 0232	65	1	552
79.0	409203.0	0.0610000006854 53415	73	1	552
79.0	409203.0	0.0590000003576 2787	71	1	552
79.0	409203.0	0.0570000000298 0232	69	1	552
79.0	409203.0	0.0579999983310 69946	69	1	552
79.0	409203.0	0.0630000010132 7896	72	1	552
79.0	409203.0	0.0580000020563 60245	83	1	552
79.0	409203.0	0.0879999995231 6284	103	1	552
79.0	409203.0	0.0870000049471 8552	101	1	552
79.0	409203.0	0.0960000008344 6503	89	1	552
79.0	409203.0	0.0899999961256 9809	113	1	552
79.0	409203.0	0.0670000016689 3005	85	1	552
79.0	409203.0	0.0679999962449 0738	105	1	552
79.0	409203.0	0.0599999986588 9549	105	1	552
79.0	409203.0	0.0820000022649 765	91	1	552
79.0	409203.0	0.0760000050067 9016	91	1	552
79.0	409203.0	0.0749999955296 5164	102	1	552
79.0	409203.0	0.0750000029802 3224	84	1	552
79.0	409203.0	0.0779999941587 4481	0.08	1	552
79.0	409203.0	0.0679999962449 0738	91	1	552
79.0	409203.0	0.0630000010132 7896	92	1	552
79.0	409203.0	0.0770000070333 4808	111	1	552
79.0	409203.0	0.0689999982714 653	115	1	552

79.0	409203.0	0.0700000002980 2322	83	1	552
79.0	409203.0	0.0879999995231 6284	0.08	1	552
79.0	409203.0	0.0670000016689 3005	116	1	552
79.0	409203.0	0.0619999989867 2104	88	1	552
79.0	409203.0	0.0549999997019 76776	91	1	552
79.0	409203.0	0.0770000070333 4808	82	1	552
79.0	409203.0	0.0740000009536 7432	81	1	552
79.0	409203.0	0.0700000002980 2322	78	1	552
79.0	409203.0	0.0599999986588 9549	86	1	552
79.0	409203.0	0.0670000016689 3005	0.08	1	552
79.0	409203.0	0.0769999995827 6749	79	1	552

DRA προσέγγιση

Bytes Client Send	Bytes Server Send	Client CPU time	Server CPU time	Msgs Sent	Msgs Rsv
4.0	2164916.0	0.4659999907016 754	N/A	1	2908
4.0	2164916.0	0.3619999885559 082	N/A	1	2908
4.0	2164916.0	0.2549999952316 284	N/A	1	2908
4.0	2164916.0	0.2509999871253 9673	N/A	1	2908
4.0	2164916.0	0.2660000026226 0437	N/A	1	2908
4.0	2164916.0	0.2649999856948 8525	N/A	1	2908
4.0	2164916.0	0.2409999966621 399	N/A	1	2908
4.0	2164916.0	0.2590000033378 601	N/A	1	2908
4.0	2164916.0	0.2329999953508 377	N/A	1	2908
4.0	2164916.0	0.2319999933242 7979	N/A	1	2908
4.0	2164916.0	0.2319999933242 7979	N/A	1	2908

4.0	2164916.0	0.2230000048875 8087	N/A	1	2908
4.0	2164916.0	0.2639999985694 885	N/A	1	2908
4.0	2164916.0	0.2430000007152 5574	N/A	1	2908
4.0	2164916.0	0.2230000048875 8087	N/A	1	2908
4.0	2164916.0	0.2249999940395 3552	N/A	1	2908
4.0	2164916.0	0.2450000047683 7158	N/A	1	2908
4.0	2164916.0	0.2259999960660 9344	N/A	1	2908
4.0	2164916.0	0.2169999927282 3334	N/A	1	2908
4.0	2164916.0	0.2150000035762 787	N/A	1	2908
4.0	2164916.0	0.2060000002384 1858	N/A	1	2908
4.0	2164916.0	0.2199999988079 071	N/A	1	2908
4.0	2164916.0	0.25	N/A	1	2908
4.0	2164916.0	0.2479999959468 8416	N/A	1	2908
4.0	2164916.0	0.2249999940395 3552	N/A	1	2908
4.0	2164916.0	0.2210000008344 6503	N/A	1	2908
4.0	2164916.0	0.2189999967813 4918	N/A	1	2908
4.0	2164916.0	0.2329999953508 377	N/A	1	2908
4.0	2164916.0	0.2189999967813 4918	N/A	1	2908
4.0	2164916.0	0.2339999973773 9563	N/A	1	2908
4.0	2164916.0	0.2370000034570 694	N/A	1	2908
4.0	2164916.0	0.2399999946355 8197	N/A	1	2908
4.0	2164916.0	0.2540000081062 317	N/A	1	2908
4.0	2164916.0	0.2419999986886 9781	N/A	1	2908
4.0	2164916.0	0.2380000054836 2732	N/A	1	2908
4.0	2164916.0	0.2560000121593 4753	N/A	1	2908

4.0	2164916.0	0.2469999939203 2623	N/A	1	2908
4.0	2164916.0	0.2290000021457 672	N/A	1	2908
4.0	2164916.0	0.2380000054836 2732	N/A	1	2908
4.0	2164916.0	0.2750000059604 645	N/A	1	2908
4.0	2164916.0	0.2430000007152 5574	N/A	1	2908
4.0	2164916.0	0.2440000027418 1366	N/A	1	2908
4.0	2164916.0	0.2300000041723 2513	N/A	1	2908
4.0	2164916.0	0.2460000067949 295	N/A	1	2908
4.0	2164916.0	0.2290000021457 672	N/A	1	2908
4.0	2164916.0	0.2210000008344 6503	N/A	1	2908
4.0	2164916.0	0.2269999980926 5137	N/A	1	2908
4.0	2164916.0	0.2300000041723 2513	N/A	1	2908
4.0	2164916.0	0.2370000034570 694	N/A	1	2908
4.0	2164916.0	0.2249999940395 3552	N/A	1	2908
4.0	2164916.0	0.2469999939203 2623	N/A	1	2908
4.0	2164916.0	0.2399999946355 8197	N/A	1	2908
4.0	2164916.0	0.2409999966621 399	N/A	1	2908
4.0	2164916.0	0.2509999871253 9673	N/A	1	2908
4.0	2164916.0	0.2460000067949 295	N/A	1	2908
4.0	2164916.0	0.2339999973773 9563	N/A	1	2908
4.0	2164916.0	0.2569999992847 4426	N/A	1	2908
4.0	2164916.0	0.3009999990463 257	N/A	1	2908
4.0	2164916.0	0.2809999883174 896	N/A	1	2908
4.0	2164916.0	0.2680000066757 202	N/A	1	2908
4.0	2164916.0	0.2370000034570 694	N/A	1	2908

4.0	2164916.0	0.2300000041723 2513	N/A	1	2908
4.0	2164916.0	0.2360000014305 1147	N/A	1	2908
4.0	2164916.0	0.2759999930858 612	N/A	1	2908
4.0	2164916.0	0.2349999994039 5355	N/A	1	2908
4.0	2164916.0	0.2269999980926 5137	N/A	1	2908
4.0	2164916.0	0.2339999973773 9563	N/A	1	2908
4.0	2164916.0	0.2610000073909 7595	N/A	1	2908
4.0	2164916.0	0.2360000014305 1147	N/A	1	2908
4.0	2164916.0	0.2300000041723 2513	N/A	1	2908
4.0	2164916.0	0.2360000014305 1147	N/A	1	2908
4.0	2164916.0	0.2360000014305 1147	N/A	1	2908
4.0	2164916.0	0.2290000021457 672	N/A	1	2908
4.0	2164916.0	0.2709999978542 328	N/A	1	2908
4.0	2164916.0	0.2569999992847 4426	N/A	1	2908
4.0	2164916.0	0.2529999911785 126	N/A	1	2908
4.0	2164916.0	0.2489999979734 4208	N/A	1	2908
4.0	2164916.0	0.2300000041723 2513	N/A	1	2908
4.0	2164916.0	0.2310000061988 8306	N/A	1	2908
4.0	2164916.0	0.2290000021457 672	N/A	1	2908
4.0	2164916.0	0.2319999933242 7979	N/A	1	2908
4.0	2164916.0	0.2389999926090 2405	N/A	1	2908
4.0	2164916.0	0.2720000147819 519	N/A	1	2908
4.0	2164916.0	0.2750000059604 645	N/A	1	2908
4.0	2164916.0	0.2630000114440 918	N/A	1	2908
4.0	2164916.0	0.2339999973773 9563	N/A	1	2908

4.0	2164916.0	0.2269999980926 5137	N/A	1	2908
4.0	2164916.0	0.2329999953508 377	N/A	1	2908
4.0	2164916.0	0.2360000014305 1147	N/A	1	2908
4.0	2164916.0	0.2300000041723 2513	N/A	1	2908
4.0	2164916.0	0.2349999994039 5355	N/A	1	2908
4.0	2164916.0	0.2380000054836 2732	N/A	1	2908
4.0	2164916.0	0.2520000040531 1584	N/A	1	2908
4.0	2164916.0	0.2319999933242 7979	N/A	1	2908
4.0	2164916.0	0.2709999978542 328	N/A	1	2908
4.0	2164916.0	0.2540000081062 317	N/A	1	2908
4.0	2164916.0	0.2249999940395 3552	N/A	1	2908
4.0	2164916.0	0.2310000061988 8306	N/A	1	2908
4.0	2164916.0	0.2310000061988 8306	N/A	1	2908
4.0	2164916.0	0.2349999994039 5355	N/A	1	2908

ΚΟΙΟΣ dataset

CRA προσέγγιση

Bytes Client Send	Bytes Server Send	Client CPU time	Server CPU time	Msgs Sent	Msgs Rsv
1174.0	10.0	N/A	35	1	1
1174.0	10.0	N/A	0.0090	1	1
1174.0	10.0	N/A	0.0040	1	1
1174.0	10.0	N/A	0.0060	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0090	1	1
1174.0	10.0	N/A	0.0030	1	1

1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0050	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0010	1	1
1174.0	10.0	N/A	0.0040	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0040	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0040	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0040	1	1
1174.0	10.0	N/A	0.0060	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0040	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0020	1	1

1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0040	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0010	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0010	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0010	1	1
1174.0	10.0	N/A	0.0020	1	1

1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0010	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0010	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0040	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0030	1	1
1174.0	10.0	N/A	0.0020	1	1
1174.0	10.0	N/A	0.0030	1	1

BMA προσέγγιση

Bytes Client Send	Bytes Server Send	Client CPU time	Server CPU time	Msgs Sent	Msgs Rsv
79.0	5816.0	0.0639999955892 5629	0.0080	1	21
79.0	5816.0	0.0160000007599 5922	0.01	1	21
79.0	5816.0	0.0080000003799 7961	0.0070	1	21
79.0	5816.0	0.0160000007599 5922	18	1	21
79.0	5816.0	0.0040000001899 89805	0.0060	1	21
79.0	5816.0	0.0030000000260 77032	0.0040	1	21
79.0	5816.0	0.0030000000260 77032	0.0060	1	21
79.0	5816.0	0.0030000000260 77032	0.0040	1	21
79.0	5816.0	0.0030000000260 77032	0.0030	1	21
79.0	5816.0	0.0010000000474 974513	0.0020	1	21

79.0	5816.0	0.0020000000949949026	0.0020	1	21
79.0	5816.0	0.0020000000949949026	0.0020	1	21
79.0	5816.0	0.0030000000026077032	0.0020	1	21
79.0	5816.0	0.0020000000949949026	0.0030	1	21
79.0	5816.0	0.0020000000949949026	0.0030	1	21
79.0	5816.0	0.0020000000949949026	0.0030	1	21
79.0	5816.0	0.0020000000949949026	0.0080	1	21
79.0	5816.0	0.0030000000026077032	0.0020	1	21
79.0	5816.0	0.0020000000949949026	0.0030	1	21
79.0	5816.0	0.0030000000026077032	0.0010	1	21
79.0	5816.0	0.0030000000026077032	0.0030	1	21
79.0	5816.0	0.0030000000026077032	0.0030	1	21
79.0	5816.0	0.0010000000474974513	0.0030	1	21
79.0	5816.0	0.0010000000474974513	0.0020	1	21
79.0	5816.0	0.0030000000026077032	0.0020	1	21
79.0	5816.0	0.0020000000949949026	0.0020	1	21
79.0	5816.0	0.0010000000474974513	0.0020	1	21
79.0	5816.0	0.0010000000474974513	0.0010	1	21
79.0	5816.0	0.0020000000949949026	0.0010	1	21
79.0	5816.0	0.0010000000474974513	0.0020	1	21
79.0	5816.0	0.0020000000949949026	0.0030	1	21
79.0	5816.0	0.0020000000949949026	0.0030	1	21
79.0	5816.0	0.0020000000949949026	0.0020	1	21
79.0	5816.0	0.0030000000026077032	0.0030	1	21
79.0	5816.0	0.0020000000949949026	0.0020	1	21

79.0	5816.0	0.0020000000949949026	0.0020	1	21
79.0	5816.0	0.0030000000026077032	0.0030	1	21
79.0	5816.0	0.0020000000949949026	0.0030	1	21
79.0	5816.0	0.0020000000949949026	0.0010	1	21
79.0	5816.0	0.0010000000474974513	0.0020	1	21
79.0	5816.0	0.0020000000949949026	0.0020	1	21
79.0	5816.0	0.0030000000026077032	0.0020	1	21
79.0	5816.0	0.0020000000949949026	0.0030	1	21
79.0	5816.0	0.0030000000026077032	0.0020	1	21
79.0	5816.0	0.0010000000474974513	0.0020	1	21
79.0	5816.0	0.0020000000949949026	0.0010	1	21
79.0	5816.0	0.0020000000949949026	0.0020	1	21
79.0	5816.0	0.0010000000474974513	0.0020	1	21
79.0	5816.0	0.0010000000474974513	0.0020	1	21
79.0	5816.0	0.0020000000949949026	0.0010	1	21
79.0	5816.0	0.0010000000474974513	0.0010	1	21
79.0	5816.0	0.0010000000474974513	0.0010	1	21
79.0	5816.0	0.0010000000474974513	0.0010	1	21
79.0	5816.0	0.0020000000949949026	0.0010	1	21
79.0	5816.0	0.0	0.0010	1	21
79.0	5816.0	0.0030000000026077032	0.0010	1	21
79.0	5816.0	0.0010000000474974513	0.0010	1	21
79.0	5816.0	0.0010000000474974513	0.0010	1	21
79.0	5816.0	0.0	0.0	1	21
79.0	5816.0	0.0020000000949949026	0.0010	1	21
79.0	5816.0	0.0	0.0010	1	21

79.0	5816.0	0.0020000000949949026	0.0010	1	21
79.0	5816.0	0.0	0.0020	1	21
79.0	5816.0	0.0020000000949949026	0.0020	1	21
79.0	5816.0	0.0030000000026077032	0.0010	1	21
79.0	5816.0	0.0010000000474974513	0.0010	1	21
79.0	5816.0	0.0010000000474974513	0.0040	1	21
79.0	5816.0	0.0020000000949949026	0.0010	1	21
79.0	5816.0	0.0020000000949949026	0.0010	1	21
79.0	5816.0	0.0030000000026077032	0.0010	1	21
79.0	5816.0	0.0010000000474974513	0.0020	1	21
79.0	5816.0	0.0010000000474974513	0.0010	1	21
79.0	5816.0	0.0020000000949949026	0.0020	1	21
79.0	5816.0	0.0080000003799 7961	0.0020	1	21
79.0	5816.0	0.0010000000474974513	0.0010	1	21
79.0	5816.0	0.0020000000949949026	0.0010	1	21
79.0	5816.0	0.0010000000474974513	0.0020	1	21
79.0	5816.0	0.0020000000949949026	0.0010	1	21
79.0	5816.0	0.0020000000949949026	0.0010	1	21
79.0	5816.0	0.0020000000949949026	0.0010	1	21
79.0	5816.0	0.0020000000949949026	0.0010	1	21
79.0	5816.0	0.0010000000474974513	0.0010	1	21
79.0	5816.0	0.0	0.0010	1	21
79.0	5816.0	0.0020000000949949026	0.0010	1	21
79.0	5816.0	0.0010000000474974513	0.0020	1	21
79.0	5816.0	0.0020000000949949026	0.0010	1	21

79.0	5816.0	0.0020000000949949026	0.0010	1	21
79.0	5816.0	0.0030000000026077032	0.0020	1	21
79.0	5816.0	0.0020000000949949026	0.0020	1	21
79.0	5816.0	0.0030000000026077032	0.0020	1	21
79.0	5816.0	0.0020000000949949026	0.0010	1	21
79.0	5816.0	0.0010000000474974513	0.0010	1	21
79.0	5816.0	0.0010000000474974513	0.0030	1	21
79.0	5816.0	0.0030000000026077032	0.0020	1	21
79.0	5816.0	0.0010000000474974513	0.0010	1	21
79.0	5816.0	0.0020000000949949026	0.0020	1	21
79.0	5816.0	0.0020000000949949026	0.0020	1	21
79.0	5816.0	0.0010000000474974513	0.0010	1	21
79.0	5816.0	0.0020000000949949026	0.0010	1	21
79.0	5816.0	0.0020000000949949026	0.0010	1	21

DRA προσέγγιση

Bytes Client Send	Bytes Server Send	Client CPU time	Server CPU time	Msgs Sent	Msgs Rsv
4.0	33303.0	0.0920000001788 1393	N/A	1	107
4.0	33303.0	0.0230000000447 03484	N/A	1	107
4.0	33303.0	0.0309999994933 6052	N/A	1	107
4.0	33303.0	0.0270000007003 54576	N/A	1	107
4.0	33303.0	0.0219999998807 9071	N/A	1	107
4.0	33303.0	0.0120000001043 08128	N/A	1	107
4.0	33303.0	0.0120000001043 08128	N/A	1	107
4.0	33303.0	0.0149999996647 23873	N/A	1	107

4.0	33303.0	0.0109999999403 95355	N/A	1	107
4.0	33303.0	0.0089999996125 69809	N/A	1	107
4.0	33303.0	0.0099999997764 82582	N/A	1	107
4.0	33303.0	0.0089999996125 69809	N/A	1	107
4.0	33303.0	0.0080000003799 7961	N/A	1	107
4.0	33303.0	0.0070000002160 66837	N/A	1	107
4.0	33303.0	0.0080000003799 7961	N/A	1	107
4.0	33303.0	0.0080000003799 7961	N/A	1	107
4.0	33303.0	0.0080000003799 7961	N/A	1	107
4.0	33303.0	0.0070000002160 66837	N/A	1	107
4.0	33303.0	0.0080000003799 7961	N/A	1	107
4.0	33303.0	0.0089999996125 69809	N/A	1	107
4.0	33303.0	0.0080000003799 7961	N/A	1	107
4.0	33303.0	0.0080000003799 7961	N/A	1	107
4.0	33303.0	0.0120000001043 08128	N/A	1	107
4.0	33303.0	0.0080000003799 7961	N/A	1	107
4.0	33303.0	0.0080000003799 7961	N/A	1	107
4.0	33303.0	0.0099999997764 82582	N/A	1	107
4.0	33303.0	0.0060000000521 54064	N/A	1	107
4.0	33303.0	0.0080000003799 7961	N/A	1	107
4.0	33303.0	0.0130000002682 20901	N/A	1	107
4.0	33303.0	0.0060000000521 54064	N/A	1	107
4.0	33303.0	0.0060000000521 54064	N/A	1	107
4.0	33303.0	0.0070000002160 66837	N/A	1	107
4.0	33303.0	0.0070000002160 66837	N/A	1	107

4.0	33303.0	0.0040000001899 89805	N/A	1	107
4.0	33303.0	0.0049999998882 41291	N/A	1	107
4.0	33303.0	0.0049999998882 41291	N/A	1	107
4.0	33303.0	0.0060000000521 54064	N/A	1	107
4.0	33303.0	0.0060000000521 54064	N/A	1	107
4.0	33303.0	0.0049999998882 41291	N/A	1	107
4.0	33303.0	0.0070000002160 66837	N/A	1	107
4.0	33303.0	0.0040000001899 89805	N/A	1	107
4.0	33303.0	0.0060000000521 54064	N/A	1	107
4.0	33303.0	0.0049999998882 41291	N/A	1	107
4.0	33303.0	0.0060000000521 54064	N/A	1	107
4.0	33303.0	0.0080000003799 7961	N/A	1	107
4.0	33303.0	0.0080000003799 7961	N/A	1	107
4.0	33303.0	0.0080000003799 7961	N/A	1	107
4.0	33303.0	0.0049999998882 41291	N/A	1	107
4.0	33303.0	0.0060000000521 54064	N/A	1	107
4.0	33303.0	0.0040000001899 89805	N/A	1	107
4.0	33303.0	0.0070000002160 66837	N/A	1	107
4.0	33303.0	0.0049999998882 41291	N/A	1	107
4.0	33303.0	0.0040000001899 89805	N/A	1	107
4.0	33303.0	0.0040000001899 89805	N/A	1	107
4.0	33303.0	0.0049999998882 41291	N/A	1	107
4.0	33303.0	0.0080000003799 7961	N/A	1	107
4.0	33303.0	0.0070000002160 66837	N/A	1	107
4.0	33303.0	0.0070000002160 66837	N/A	1	107

4.0	33303.0	0.0060000000521 54064	N/A	1	107
4.0	33303.0	0.0049999998882 41291	N/A	1	107
4.0	33303.0	0.0049999998882 41291	N/A	1	107
4.0	33303.0	0.0080000003799 7961	N/A	1	107
4.0	33303.0	0.0070000002160 66837	N/A	1	107
4.0	33303.0	0.0070000002160 66837	N/A	1	107
4.0	33303.0	0.0080000003799 7961	N/A	1	107
4.0	33303.0	0.0060000000521 54064	N/A	1	107
4.0	33303.0	0.0060000000521 54064	N/A	1	107
4.0	33303.0	0.0049999998882 41291	N/A	1	107
4.0	33303.0	0.0060000000521 54064	N/A	1	107
4.0	33303.0	0.0060000000521 54064	N/A	1	107
4.0	33303.0	0.0049999998882 41291	N/A	1	107
4.0	33303.0	0.0070000002160 66837	N/A	1	107
4.0	33303.0	0.0049999998882 41291	N/A	1	107
4.0	33303.0	0.0049999998882 41291	N/A	1	107
4.0	33303.0	0.0060000000521 54064	N/A	1	107
4.0	33303.0	0.0060000000521 54064	N/A	1	107
4.0	33303.0	0.0070000002160 66837	N/A	1	107
4.0	33303.0	0.0040000001899 89805	N/A	1	107
4.0	33303.0	0.0060000000521 54064	N/A	1	107
4.0	33303.0	0.0060000000521 54064	N/A	1	107
4.0	33303.0	0.0049999998882 41291	N/A	1	107
4.0	33303.0	0.0070000002160 66837	N/A	1	107
4.0	33303.0	0.0060000000521 54064	N/A	1	107

4.0	33303.0	0.0070000002160 66837	N/A	1	107
4.0	33303.0	0.0049999998882 41291	N/A	1	107
4.0	33303.0	0.0049999998882 41291	N/A	1	107
4.0	33303.0	0.0040000001899 89805	N/A	1	107
4.0	33303.0	0.0049999998882 41291	N/A	1	107
4.0	33303.0	0.0070000002160 66837	N/A	1	107
4.0	33303.0	0.0099999997764 82582	N/A	1	107
4.0	33303.0	0.0160000007599 5922	N/A	1	107
4.0	33303.0	0.0080000003799 7961	N/A	1	107
4.0	33303.0	0.0049999998882 41291	N/A	1	107
4.0	33303.0	0.0049999998882 41291	N/A	1	107
4.0	33303.0	0.0070000002160 66837	N/A	1	107
4.0	33303.0	0.0080000003799 7961	N/A	1	107
4.0	33303.0	0.0109999999403 95355	N/A	1	107
4.0	33303.0	0.0060000000521 54064	N/A	1	107
4.0	33303.0	0.0040000001899 89805	N/A	1	107
4.0	33303.0	0.0040000001899 89805	N/A	1	107

Παράρτημα Γ

Κώδικας προσομοιωτή

Πιο κάτω παρουσιάζουμε τον κώδικα του προσομοιωτή που υλοποιήθηκε στα πλαίσια της παρούσας διπλωματικής εργασίας. Πρόκειται για δύο εφαρμογές οι οποίες προσομοιάζουν τους τρείς αλγόριθμους τοπικοποίησης: DRA, CRA και BMA. Οι δύο εφαρμογές είναι ο client και ο server. Πρώτα θα παρουσιάζουμε τον Client και ακολούθως τον Server.

```
package Other;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;

public class Algorithms
{
    /**
     * @param latestScanList
     *            the current scan list of APs
     * @param RM
     *            the constructed Radio Map
     *
     * @param algorithm_choice
     *            choice of several algorithms
     *
     * @param WeightsList
     *             the RBF weights
     *
     * @return the location of user
     */
    public static String ProcessingAlgorithms(ArrayList<LogRecord>
        latestScanList, RadioMapMean RM, int algorithm_choice, String
        parameter)
    {

        int i = 0, j = 0;

        ArrayList<String> MacAdressList = RM.getMacAdressList();
        ArrayList<String> Observed_RSS_Values = new ArrayList<String>();
        LogRecord temp_LR = null;

        // Check which mac addresses of radio map, we are currently
            listening.
        for (i = 0; i < MacAdressList.size(); ++i)
        {
            for (j = 0; j < latestScanList.size(); ++j)
            {
                temp_LR = latestScanList.get(j);

                // MAC Address Matched
                if (MacAdressList.get(i).compareTo(temp_LR.getBssid()) ==
                    0)
                {
                    Observed_RSS_Values.add(String.valueOf(temp_LR.getRss
                        ()));
                    break;
                }
            }
            // A MAC Address is missing so we place a small NaN value
            if (j == latestScanList.size())
            {
                Observed_RSS_Values.add(String.valueOf("-110"));
            }
        }

        switch (algorithm_choice)
        {
```



```
            case 1:
                return KNN_WKNN_Algorithm(RM, Observed_RSS_Values,
                    parameter, false);
        }
        return null;

    }

    /**
     * Calculates user location based on Weighted/Not Weighted K Nearest
     * Neighbor (KNN) Algorithm
     *
     * @param RM
     *            The radio map structure
     *
     * @param Observed_RSS_Values
     *            RSS values currently observed
     * @param parameter
     *
     * @param isWeighted
     *            To be weighted or not
     *
     * @param K
     *            The number of locations used
     *
     * @return The estimated user location
     */
    private static String KNN_WKNN_Algorithm(RadioMapMean RM, ArrayList<
        String> Observed_RSS_Values, String parameter, boolean isWeighted)
    {
        ArrayList<String> RSS_Values;
        float curResult = 0;
        ArrayList<LocDistance> LocDistance_Results_List = new ArrayList<
            LocDistance>();
        String myLocation = null;
        int K;

        try
        {
            K = Integer.parseInt(parameter);
        }
        catch (Exception e)
        {
            System.err.println("Error parsing K in KNN-WKNN: " + e.
                getMessage());
            return null;
        }

        // Construct a list with locations-distances pairs for currently
        // observed RSS values
        for (String location : RM.getLocationRSS_HashMap().keySet())
        {
            RSS_Values = RM.getLocationRSS_HashMap().get(location);
            curResult = calculateEuclideanDistance(RSS_Values,
                Observed_RSS_Values);

            if (curResult == Float.NEGATIVE_INFINITY)
            {
                return null;
```

```
                }

                LocDistance_Results_List.add(0, new LocDistance(curResult,
                    location));
            }

            // Sort locations-distances pairs based on minimum distances
            Collections.sort(LocDistance_Results_List, new Comparator<
                LocDistance>()
            {

                public int compare(LocDistance gd1, LocDistance gd2)
                {
                    return (gd1.getDistance() > gd2.getDistance() ? 1 : (gd1.
                        getDistance() == gd2.getDistance() ? 0 : -1));
                }
            });

            if (!isWeighted)
            {
                myLocation = calculateAverageKDistanceLocations
                    (LocDistance_Results_List, K);
            }
            else
            {
                myLocation = calculateWeightedAverageKDistanceLocations
                    (LocDistance_Results_List, K);
            }

            return myLocation;

        }

        /**
         * Calculates the Euclidean distance between the currently observed
             RSS
         * values and the RSS values for a specific location.
         *
         * @param l1
         *            RSS values of a location in radiomap
         * @param l2
         *            RSS values currently observed
         *
         * @return The Euclidean distance, or MIN_VALUE for error
         */
        private static float calculateEuclideanDistance(ArrayList<String> l1,
            ArrayList<String> l2)
        {

            float finalResult = 0;
            float v1;
            float v2;
            float temp;
            String str;

            for (int i = 0; i < l1.size(); ++i)
            {

                try
```



```
                {
                    str = l1.get(i);
                    v1 = Float.valueOf(str.trim()).floatValue();
                    str = l2.get(i);
                    v2 = Float.valueOf(str.trim()).floatValue();
                }
                catch (Exception e)
                {
                    System.err.println("Error while computing Euclidean
                        Distance: " + e.getMessage());
                    return Float.NEGATIVE_INFINITY;
                }

                // do the procedure
                temp = v1 - v2;
                temp *= temp;

                // do the procedure
                finalResult += temp;
            }
            return ((float) Math.sqrt(finalResult));
        }

        /**
         * Calculates the Probability of the user being in the currently
             observed
         * RSS values and the RSS values for a specific location.
         *
         * @param l1
         *            RSS values of a location in radiomap
         * @param l2
         *            RSS values currently observed
         *
         * @return The Probability for this location, or MIN_VALUE for error
         */
        public static double calculateProbability(ArrayList<String> l1,
            ArrayList<String> l2, float sGreek)
        {

            double finalResult = 1;
            float v1;
            float v2;
            double temp;
            String str;

            for (int i = 0; i < l1.size(); ++i)
            {

                try
                {
                    str = l1.get(i);
                    v1 = Float.valueOf(str.trim()).floatValue();
                    str = l2.get(i);
                    v2 = Float.valueOf(str.trim()).floatValue();
                }
                catch (Exception e)
                {
                    System.err.println("Error while computing Euclidean
                        Distance: " + e.getMessage());
```

```
                return Double.NEGATIVE_INFINITY;
            }

            temp = v1 - v2;

            temp *= temp;

            temp = -temp;

            temp /= (double) (sGreek * sGreek);
            temp = (double) Math.exp(temp);

            finalResult *= temp;
        }
        return finalResult;
    }

    /**
     * Calculates the Average of the K locations that have the shortest
     * distances D
     *
     * @param LocDistance_Results_List
     *            Locations-Distances pairs sorted by distance
     * @param K
     *            The number of locations used
     * @return The estimated user location, or null for error
     */
    private static String calculateAverageKDistanceLocations(ArrayList<
        LocDistance> LocDistance_Results_List, int K)
    {

        float sumX = 0.0f;
        float sumY = 0.0f;

        String[] LocationArray = new String[2];
        float x, y;

        int K_Min = K < LocDistance_Results_List.size() ? K :
            LocDistance_Results_List.size();

        // Calculate the sum of X and Y
        for (int i = 0; i < K_Min; ++i)
        {
            LocationArray = LocDistance_Results_List.get(i).getLocation().
                split(" ");

            try
            {
                x = Float.valueOf(LocationArray[0].trim()).floatValue();
                y = Float.valueOf(LocationArray[1].trim()).floatValue();
            }
            catch (Exception e)
            {
                System.err.println("Error while calculating Average K
                    Distance Locations: " + e.getMessage());
                return null;
            }

            sumX += x;
```



```
            sumY += y;
        }

        // Calculate the average
        sumX /= K_Min;
        sumY /= K_Min;

        return sumX + " " + sumY;

    }

    /**
     * Calculates the Weighted Average of the K locations that have the
         shortest
     * distances D
     *
     * @param LocDistance_Results_List
     *            Locations-Distances pairs sorted by distance
     * @param K
     *            The number of locations used
     * @return The estimated user location, or null for error
     */
    public static String calculateWeightedAverageKDistanceLocations
        (ArrayList<LocDistance> LocDistance_Results_List, int K)
    {

        double LocationWeight = 0.0f;
        double sumWeights = 0.0f;
        double WeightedSumX = 0.0f;
        double WeightedSumY = 0.0f;

        String[] LocationArray = new String[2];
        float x, y;

        int K_Min = K < LocDistance_Results_List.size() ? K :
            LocDistance_Results_List.size();

        // Calculate the weighted sum of X and Y
        for (int i = 0; i < K_Min; ++i)
        {

            LocationWeight = 1 / LocDistance_Results_List.get(i).
                getDistance();
            LocationArray = LocDistance_Results_List.get(i).getLocation().
                split(" ");

            try
            {
                x = Float.valueOf(LocationArray[0].trim()).floatValue();
                y = Float.valueOf(LocationArray[1].trim()).floatValue();
            }
            catch (Exception e)
            {
                System.err.println("Error while calculating Weighted
                    Average K Distance Locations: " + e.getMessage());
                return null;
            }

            sumWeights += LocationWeight;
```

```
                WeightedSumX += LocationWeight * x;
                WeightedSumY += LocationWeight * y;

            }

            WeightedSumX /= sumWeights;
            WeightedSumY /= sumWeights;

            return WeightedSumX + " " + WeightedSumY;
        }

        /**
         * Calculates the Weighted Average over ALL locations where the
             weights are
         * the Normalized Probabilities
         *
         * @param LocDistance_Results_List
         *            Locations-Probability pairs
         *
         * @return The estimated user location, or null for error
         */
        public static String calculateWeightedAverageProbabilityLocations
            (ArrayList<LocDistance> LocDistance_Results_List)
        {

            double sumProbabilities = 0.0f;
            double WeightedSumX = 0.0f;
            double WeightedSumY = 0.0f;
            double NP;
            float x, y;
            String[] LocationArray = new String[2];

            // Calculate the sum of all probabilities
            for (int i = 0; i < LocDistance_Results_List.size(); ++i)
            {
                sumProbabilities += LocDistance_Results_List.get(i).
                    getDistance();
            }

            // Calculate the weighted (Normalized Probabilities) sum of X and
                Y
            for (int i = 0; i < LocDistance_Results_List.size(); ++i)
            {
                LocationArray = LocDistance_Results_List.get(i).getLocation().
                    split(" ");

                try
                {
                    x = Float.valueOf(LocationArray[0].trim()).floatValue();
                    y = Float.valueOf(LocationArray[1].trim()).floatValue();
                }
                catch (Exception e)
                {
                    System.err.println("Error while calculating Weighted
                        Average Probability Locations: " + e.getMessage());
                    return null;
                }

                NP = LocDistance_Results_List.get(i).getDistance() /
```



```
                        sumProbabilities;

                WeightedSumX += (x * NP);
                WeightedSumY += (y * NP);

            }

            return WeightedSumX + " " + WeightedSumY;

        }
    }
```

```java
package approaches;

/**
 * General class that is superclass for all approaches
 * @author Silouanos
 *
 */
public class Approach implements Runnable
{
    protected String ip;
    protected Integer port;
    protected String nameOfDataFile;
    protected float elapsedTimeSec;

    /**
     * Initialize with the ip and the port
     * @param ip
     * @param port
     */
    public Approach(String ip, Integer port,String nameOfDataFile)
    {
        this.ip=ip;
        this.port=port;
        this.nameOfDataFile=nameOfDataFile;
    }

    @Override
    public void run()
    {
    }
}
```

Simulator Client code


```
package approaches;

import java.io.IOException;
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;

import Other.ParseFile;
import Other.RadioMapMean;

public class Bloom
{
    final private int K;
    final private int noOfHashFunctions;
    private Boolean[] bloomFilter = null;
    final private int sizeOfFinalVector;
    private ParseFile testFile = null;
    private RadioMapMean rmm = new RadioMapMean();
    private ArrayList<String> macAddresses = null;

    public Bloom(int kFromClient, ParseFile testFile, int noOfHashFunc)
        throws IOException
    {
        this.K=kFromClient;
        this.testFile=testFile;
        this.noOfHashFunctions=noOfHashFunc;

        //Create a radio-map to separate the MacAddresses and RSS values

        try
        {
            rmm.createRadioMap(testFile);
        } catch (IOException e)
        {
            e.printStackTrace();
        }


        //After the use in rmm I have to initialize again the testFile
        try
        {
            testFile.closeTheFile();
        } catch (IOException e)
        {
            e.printStackTrace();
        }
        testFile.closeTheFile();

        try
        {
            this.testFile = new ParseFile(testFile.getTheNameOfTheFile());
        } catch (IOException e)
        {
            e.printStackTrace();
        }

        //Get the macAddresses
        macAddresses = rmm.getMacAdressList();
```



```
        //Find the size of bloom Vector
        sizeOfFinalVector= computeTheSizeOfBloomVector();
    }

    /**
     * Take a random macAddress from the i (=iteration) line [that the RSS
         is NOT -110]
     * @param iteration
     * @return
     * @throws IOException
     */
    public String getTheNextBloomVector(int iteration) throws IOException
    {
        /**
         *  Initialize the Bloom vector
         *  Every time that user has new position
         */
        initializeTheBloomFilter(sizeOfFinalVector);

        String currentLine = null;
        String macAddress = null;

        //Take the current line
        currentLine=testFile.getTheNextLine();
        if(currentLine==null)
        {
            return null;
        }

        //Find the macAddress of any VALID mac Address (NOT -110)
        macAddress = findARandomMacAddress(currentLine);

        //Do the whole procedure of the bloom algorithm
        try
        {
            doTheBloomProcedure(macAddress);
        } catch (NoSuchAlgorithmException e)
        {
            e.printStackTrace();
        }
        return convertBloomToString();
    }

    /**
     * Convert the Vector to String
     *
     * @return the content of the bloom vector in string type
     */
    private String convertBloomToString()
    {
        String reStr="";
        for(int i=0; i<bloomFilter.length; ++i)
        {
            if(bloomFilter[i]==false)
            {
                reStr+=0;
            }
            else
```

```
                {
                    reStr+=1;
                }
            }

            return reStr;
        }

        /**
         * Doing the whole bloom procedure the 4 hash functions
         *
         * @param macAddress
         * @throws NoSuchAlgorithmException
         */
        private void doTheBloomProcedure(String macAddress) throws
            NoSuchAlgorithmException
        {
            String reversedMacAddress = new StringBuffer(macAddress).reverse()
                .toString();

            /**
             * Attention here!!! I put (-) if is negative
             */

            if(this.noOfHashFunctions>=1)
            {
                int resOfH1 = macAddress.hashCode();
                if(resOfH1<0)
                {
                    resOfH1*=(-1);
                }
                //H1
                bloomFilter[resOfH1%sizeOfFinalVector]=true;
            }

            if(this.noOfHashFunctions>=2)
            {
                int resOfH2 = MD5HashFunction(macAddress);
                if(resOfH2<0)
                {
                    resOfH2*=(-1);
                }
                //H2
                bloomFilter[resOfH2%sizeOfFinalVector]=true;
            }

            if(this.noOfHashFunctions>=3)
            {
                int resOfH3 = reversedMacAddress.hashCode();
                if(resOfH3<0)
                {
                    resOfH3*=(-1);
                }
                //H3
                bloomFilter[resOfH3%sizeOfFinalVector]=true;
            }

            if(this.noOfHashFunctions>=4)
            {
```



```
                int resOfH4 = MD5HashFunction(reversedMacAddress);

                if(resOfH4<0)
                {
                    resOfH4*=(-1);
                }
                //H4
                bloomFilter[resOfH4%sizeOfFinalVector]=true;
            }
        }

        /**
         * Use the hash function of MD5 and then use the result with hashCode
         *
         * @param macAddress
         * @return a value
         * @throws NoSuchAlgorithmException
         */
        private int MD5HashFunction(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();
        }

        /**
         * Check if a string is an integer
         *
         * @param i the string
         * @return true if is number or false if is not
         */
        public boolean isParsableToInt(String i)
        {
            try
            {
                Integer.parseInt(i);
                return true;
            }
            catch(NumberFormatException nfe)
            {
                return false;
            }
        }


        /**
         * Initialize the bloom vector with false and create it
         * @param size
         */
        private void initializeTheBloomFilter(int size)
        {
            bloomFilter=new Boolean[size];

            for(int i=0; i<size; ++i)
            {
                bloomFilter[i]=new Boolean(false);
            }
```

```
        }

        /**
         * Finds a random MacAddress from the test-data file but mac address
             that client listens
         *
         * @return the macAddress otherwise null
         */
        private String findARandomMacAddress(String currentLine)
        {
            int count=0;
            ArrayList<String> currentRss = getRssCurrentLine(currentLine);

            for(String curMacAddress : macAddresses)
            {
                if(!currentRss.get(count).equals("-110"))
                {
                    return curMacAddress;
                }
                ++count;
            }

            return null;
        }

        /**
         *
         * @param line that client receives
         * @return an arrayList with only RSS values
         */
        private ArrayList<String> getRssCurrentLine(String line)
        {
            ArrayList<String> rssValues = new ArrayList<String>();
            String[] temp = line.split(", ");

            for(int i=2; i<temp.length; ++i)
            {
                rssValues.add(temp[i]);
            }

            return rssValues;
        }

        /**
         * Compute the size of the bloom vector
         *
         * @return the computed size of the bloom vector
         */
        private int computeTheSizeOfBloomVector()
        {
            float fpr = (float)K/macAddresses.size();

            System.out.println("Size: "+macAddresses.size()+" fpr: "+fpr +" K:
                "+K);

            int upperNo = -(noOfHashFunctions*macAddresses.size());

            double downNo = Math.log((double)1-Math.pow((double)fpr, ((double)
                1/noOfHashFunctions)))/(Math.log(Math.E));
```



```
            return (int) ((int)upperNo/downNo);

        }
    }
```

Simulator Client code


```
package approaches;

import java.io.IOException;


import Other.Client;
import Other.CountTime;
import Other.DataInformation;
import Other.Information;
import Other.Parameters;
import Other.ParseFile;

public class BloomFilter extends Approach
{
    private String testData = null;
    private DataInformation dataInfo=null;

    public BloomFilter(String ip, Integer port, String testData, String
        dataFile)
    {
        super(ip,port,dataFile);
        this.testData=testData;
    }

    public void run()
    {
        //Upload all the test-data file to an arrayList
        ParseFile testDataFile = null;
        //ParseFile formattedTestDataFile = null;
        try
        {
            testDataFile = new ParseFile(testData);
        } catch (IOException e)
        {
            e.printStackTrace();
        }

        CountTime wholeProcedure = new CountTime();
        System.out.println("IN11");
        //Send to server
        Client clt = new Client(ip, port,nameOfDataFile);
        try
        {
            dataInfo=clt.sendReceiveBloomFilterDataToServer(testDataFile);
            System.out.println("Time to position: "+" size of the file: "+
                testDataFile.getSizeOfTheFile());

            int aveSizeOfSmallRM = (int) (Information.
                sumOfSizeOfRadioMapSmall/testDataFile.getSizeOfTheFile());
            Parameters pr = new Parameters();

            /* (X Lines of test-radio-map x ConstantOfPositioning x X
                Lines of radio-map) */
            Information.timeToPosition=testDataFile.getSizeOfTheFile()*
                aveSizeOfSmallRM*pr.mobileAlgExecution;

        } catch (IOException e)
        {
            e.printStackTrace();
```



```
        }
        try
        {
            clt.closeConnections();
        } catch (IOException e5)
        {
            e5.printStackTrace();
        }
        wholeProcedure.endCount();
        Information.timeWholeProcedure=Information.timeOfConnection+
            Information.timeToPosition+wholeProcedure.getSeconds();
        Information.showData();
    }
}
```

Simulator Client code


```
package approaches;

import java.io.IOException;

import Other.Client;
import Other.CountTime;
import Other.DataInformation;
import Other.Information;
import Other.ParseFile;

public class Centralized extends Approach
{
    private String testData;
    private String deviceID;
    private DataInformation dataInfo;
    /**
     * @param ip
     * @param port
     * @param testData
     *                  name of the test radio-map
     * @param deviceID
     *                  the ID of the "device"
     */

    public Centralized(String ip, Integer port,String testData, String
        deviceID, String nameOfDataFile)
    {
        super(ip, port,nameOfDataFile);
        this.testData=testData;
        this.deviceID=deviceID;
        dataInfo=null;
    }

    @Override
    public void run()
    {
        //Send data to server//

        //Upload all the test-data file to an arrayList
        ParseFile testDataFile = null;
        try
        {
            testDataFile = new ParseFile(testData);
        } catch (IOException e)
        {
            e.printStackTrace();
        }

        CountTime wholeProcedure = new CountTime();

        //Send to server
        Client clt = new Client(ip, port,nameOfDataFile);
        try
        {
            dataInfo = clt.sendDataToServer(testDataFile,deviceID);
        } catch (IOException e6)
        {
            e6.printStackTrace();
        }
```



```
        try
        {
            clt.closeConnections();
        } catch (IOException e5)
        {
            e5.printStackTrace();
        }
        wholeProcedure.endCount();
        Information.timeWholeProcedure=wholeProcedure.getSeconds();
        Information.showData();
    }
}
```

Simulator Client code


```
package Other;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.ArrayList;


import approaches.Bloom;


public class Client
{
    private Socket kkSocket = null;
    private PrintWriter out = null;
    private BufferedReader in = null;
    private BufferedReader stdIn = null;
    private ArrayList<String> RadioMap = null;
    private String nameOfDataFile;
    DataInformation dataInfo = null;
    WriteData dataFile = null;

    private double currentBytesSend = 0;
    private double currentBytesReceive = 0;
    private double currentServerCpuTime = 0;
    private double currentClientCpuTime = 0;


    private double bytesSend = 0;
    private double bytesReceive = 0;
    private double clientCpuTime = 0;
    private double serverCpuTime = 0;

    private int messagesReceived = 0;
    private int messagesSent = 0;
    private int currentMessagesRsv = 0;
    private int currentMessagesSent = 0;


    /**
     * Constructor, that start new connection with the server to send and
         receive data
     *
     * @param ipAddress
     * @param port
     */
    public Client(String ipAddress, int port, String nameOfDataFile)
    {
        try
        {
            kkSocket = new Socket(ipAddress, port);
            out = new PrintWriter(kkSocket.getOutputStream(), true);
            in = new BufferedReader(new InputStreamReader(kkSocket.
                getInputStream()));
            this.nameOfDataFile=nameOfDataFile;
            dataFile = new WriteData(nameOfDataFile);
            dataInfo = new DataInformation();
```



```
        } catch (UnknownHostException e)
        {
            System.err.println("Don't know about host: taranis.");
            System.exit(1);
        } catch (IOException e)
        {
            System.err.println("Couldn't get I/O for the connection to:
                the network.");
            System.exit(1);
        }

        stdIn = new BufferedReader(new InputStreamReader(System.in));
    }


    /**
     * This method is used by BloomFilter approach only
     *
     * @param testDataFile
     * @throws IOException
     */

     @SuppressWarnings("static-access")
    public DataInformation sendReceiveBloomFilterDataToServer(ParseFile
        testDataFile) throws IOException
     {
         dataFile.writeLineToFile("-Log file for BloomFilter Approach-");
         dataFile.writeLineToFile("Bytes Client Send"+"\t"+"Bytes Server
             Send"+"\t"+"Client CPU time"+"\t"+"Server CPU time"+"\t"+
             "Msgs Sent"+"\t"+"Msgs Rsv");

         Parameters pr = new Parameters();
         int count = 0;

         Bloom bl = new Bloom(10,testDataFile,4);

         ArrayList<String> smallRadioMap = new ArrayList<String>();
         String outputLine = null;
         Algorithms alg = null;
         RadioMapMean rmm = new RadioMapMean();
         int msgs = 1;              //init with 1 for the message END
         float bytes = 0;


         String fromServer = null;
         CountTime countTheConnection = null;
         while ((fromServer = in.readLine()) != null)
         {
             //1 message for fromServer and 1 message for the sizeInt
                 transmitting
             msgs+=2;
             ++currentMessagesRsv;
             ++messagesReceived;

             if(count==2)
             {
                 countTheConnection = new CountTime();
             }
```

Simulator Client code


```
                /**
                 * Count the time of bloom processing
                 */
                CountTime countTheBloomProc = new CountTime();
                String toSendStr = bl.getTheNextBloomVector(count);
                countTheBloomProc.endCount();

                if(toSendStr==null)
                {
                   out.println("END");
                   break;
                }

                ++currentMessagesSent;
                ++messagesSent;
                out.println(toSendStr);
                bytes+=pr.bloomVectorLen;
                currentBytesSend=pr.bloomVectorLen;
                bytesSend+=pr.bloomVectorLen;


                //Get the size of the radio-map that will receive //Problem
                    to convert it to parameter
                ++currentMessagesRsv;
                ++messagesReceived;
                String size = in.readLine();
                bytes+=size.length();
                currentBytesReceive=size.length();
                bytesReceive+=currentBytesReceive;

                int sizeInt = Integer.valueOf(size);

               //Receive the whole radio-map
                for(int i=0; i<sizeInt; ++i)
                {
                    String rec = in.readLine();
                    smallRadioMap.add(rec);

                    bytes+=pr.rssValueByteLen*testDataFile.
                        getTheSizeOfMacAddresses();
                    currentBytesReceive+=pr.rssValueByteLen*testDataFile.
                        getTheSizeOfMacAddresses();
                    bytesReceive+=pr.rssValueByteLen*testDataFile.
                        getTheSizeOfMacAddresses();
                    ++msgs;
                    ++currentMessagesRsv;
                    ++messagesReceived;
                }

                if(count%500==0)
                {
                   System.out.print(". "+smallRadioMap.size()+" ");
                }


                /**
                 * Getting the current CPU time of the processing in server.
```



```
                 */
                currentServerCpuTime+=Double.valueOf(in.readLine());
                serverCpuTime+=currentServerCpuTime;

                if(count==2)
                {
                    countTheConnection.endCount();
                    Information.timeOfConnection=countTheConnection.
                        getSeconds();
                }
                Information.sumOfSizeOfRadioMapSmall+=smallRadioMap.size();

                CountTime countPositioning = new CountTime();
                rmm.createRadioMap(smallRadioMap);
                smallRadioMap.clear();
                ArrayList<LogRecord>lr = parseLineOfTestData(toSendStr);
                alg = new Algorithms();
                outputLine = alg.ProcessingAlgorithms(lr, rmm, 1, "5");
                lr=null;
                //End the counting of the positioning
                countPositioning.endCount();

                /**
                 * Store to variables the time to position and the bloom
                     processing
                 */
                currentClientCpuTime+=countPositioning.getSeconds()+
                    countTheBloomProc.getSeconds();
                clientCpuTime+=currentClientCpuTime;
                ++count;

                /**
                  * Write the data of the current results in the Data file
                 */

               dataFile.writeLineToFile(currentBytesSend+"\t"+
                   currentBytesReceive+"\t"+currentClientCpuTime+"\t"+
                   currentServerCpuTime+"\t"+currentMessagesSent+"\t"+
                   currentMessagesRsv);
               //Init
                currentServerCpuTime=0;
                currentBytesReceive=0;
                currentClientCpuTime=0;
                currentMessagesRsv=0;
                currentMessagesSent=0;


            }
            Information.messages=msgs;
            Information.bytesSendReceive=bytes;
            dataInfo.setBytesClientSend(bytesSend);
            dataInfo.setBytesServerSend(bytesReceive);
            dataInfo.setServerCpuTime(serverCpuTime);
            dataInfo.setClientCpuTime(clientCpuTime);
            dataInfo.setMessagesRsv(messagesReceived);
            dataInfo.setMessagesSent(messagesSent);
            dataFile.closeTheFile();

            return dataInfo;
```

```
        }


    /**
     * This method is used from Centralized approach only
     *
     * @param sendArray
     * @throws IOException
     */
    public DataInformation sendDataToServer(ParseFile testDataFile,String
        deviceID) throws IOException
    {

        dataFile.writeLineToFile("-Log file for Centralized Approach-");
        dataFile.writeLineToFile("Bytes Client Send"+"\t"+"Bytes Server
            Send"+"\t"+"Client CPU time"+"\t"+"Server CPU time"+"\t"+"Msgs
            Sent"+"\t"+"Msgs Rsv");

        int count = 0;
        String fromServer = null;
        int msgs = 0;
        double bytes = 3;       //END

        Information.bytesSendReceive=0.0;
        Information.messages=0;
        boolean flag = true;

        //ACK length Constant +OK READY
        //Byte length of X and Y. Constant.
        CountTime countTheConnection = new CountTime();
        while ((fromServer = in.readLine()) != null)
        {
            msgs+=1;
            ++messagesReceived;
            ++currentMessagesRsv;

            if(count%500==0)
            {
                System.out.print(". ");
            }


            if (fromServer.equals("END"))
                break;

            /**
             * Get the next line directly from the file.
             */
            String strToSend = testDataFile.getTheNextLine();

            if(strToSend==null)
            {
                out.println("END");
                ++messagesSent;
                break;
            }
            ++messagesSent;
            ++currentMessagesSent;
```



```
            out.println(deviceID+", "+strToSend); //Make Constant and the
                deviceID //RSS Value byteLen(something)
            msgs+=1;
            bytes+=Parameters.deviceIDByteLen+Parameters.rssValueByteLen*
                testDataFile.getTheSizeOfMacAddresses()+Parameters.
                macAddressesByteLen*testDataFile.getTheSizeOfMacAddresses
                ();
            bytesSend+=Parameters.deviceIDByteLen+Parameters.
                rssValueByteLen*testDataFile.getTheSizeOfMacAddresses()+
                Parameters.macAddressesByteLen*testDataFile.
                getTheSizeOfMacAddresses();
            currentBytesSend=Parameters.deviceIDByteLen+Parameters.
                rssValueByteLen*testDataFile.getTheSizeOfMacAddresses()+
                Parameters.macAddressesByteLen*testDataFile.
                getTheSizeOfMacAddresses();
            ++count;

            if(flag)
            {
                countTheConnection.endCount();
                Information.timeOfConnection=countTheConnection.getSeconds
                    ();
                flag=false;
            }

            /**
             * Getting the time of the positioning in server.
             */
            String curStrCpuTime = in.readLine();
            serverCpuTime+= Double.valueOf(curStrCpuTime);
            currentServerCpuTime=Double.valueOf(curStrCpuTime);

            bytesReceive+=Parameters.byteLenXY;
            currentBytesReceive=Parameters.byteLenXY;


            /**
             * Write the data of the current results in the Data file
             */
            dataFile.writeLineToFile(currentBytesSend+"\t"+
                currentBytesReceive+"\t"+"N/A"+"\t"+currentServerCpuTime+
                "\t"+currentMessagesSent+"\t"+currentMessagesRsv);

            //init
            currentMessagesSent=0;
            currentMessagesRsv=0;
        }
        Information.bytesSendReceive=bytes;
        Information.messages=msgs;
        dataInfo.setBytesClientSend(bytesSend);
        dataInfo.setBytesServerSend(bytesReceive+3);        //+3 END
        dataInfo.setServerCpuTime(serverCpuTime);
        dataInfo.setMessagesSent(messagesSent);
        dataInfo.setMessagesRsv(messagesReceived);

        dataFile.closeTheFile();

        return dataInfo;
    }
```

```
/**
 * This method is used from Decentralized approach
 *
 * @throws IOException
 */

public DataInformation incomingDataFromServer(ParseFile testDataFile,
    int noOfMacAddresses) throws IOException
{
    RadioMapMean rmm = new RadioMapMean();
    Algorithms algorithms = new Algorithms();

    dataFile.writeLineToFile("-Log file for Decentralized Approach-");
    dataFile.writeLineToFile("Bytes Client Send"+"\t"+"Bytes Server
        Send"+"\t"+"Client CPU time"+"\t"+"Server CPU time"+"\t"+"Msgs
        Sent"+"\t"+"Msgs Rsv");

    Information.bytesSendReceive=0.0;
    int msgs=2;     //+2 the START ACKS
    double bytes=0.0;
    int index = 0;

    //Add the bytes of the Start ACK messages
    Information.bytesSendReceive+=Parameters.decStartAckBytes+
        Parameters.decStartRequestBytes;

    RadioMap = new ArrayList<String>();
    String fromServer = null;
    fromServer = in.readLine();
    ++messagesReceived;
    ++messagesSent;
    //System.out.println("From server: "+fromServer);
    if(fromServer.equals("INCOMING RADIO-MAP"))     //Constant and
        here
    {
        out.println("+OK READY TO RECEIVE RADIO-MAP"); //Constant and
            here ACK message
    }
    else
    {
        System.err.println("Problem receiving");
        System.exit(-1);
    }

    while(true)
    {
        RadioMap.clear();


        bytes+=Parameters.decAckBytes;
        bytesSend+=Parameters.decAckBytes;
        currentBytesSend=Parameters.decAckBytes;

        //+macAddresses length //in radio-map
        bytes+=Parameters.macAddressesByteLen*noOfMacAddresses;
        bytesReceive+=Parameters.macAddressesByteLen*noOfMacAddresses;
        currentBytesReceive+=Parameters.macAddressesByteLen*
            noOfMacAddresses;
```



```
        //END of RM message
        ++msgs;
        ++messagesReceived;
        ++currentMessagesRsv;

        while(!(fromServer = in.readLine()).equals("END OF RM"))
        {
            ++msgs;
            ++messagesReceived;
            ++currentMessagesRsv;
            RadioMap.add(fromServer);
            bytes+=Parameters.rssValueByteLen*noOfMacAddresses;
            currentBytesReceive+=Parameters.rssValueByteLen*
                noOfMacAddresses;
        }

        if(index%500==0)
        {
            System.out.print(". ");
        }

        String testStr = testDataFile.getTheNextLine();

        ++msgs;
        ++messagesSent;
        ++currentMessagesSent;
        if(testStr == null)
        {
            out.println("end");
            break;
        }
        else
        {
            //Positioning
            //Start the counting of the positioning
            CountTime countPositioning = new CountTime();
            rmm.createRadioMap(RadioMap);
            ArrayList<LogRecord>lr = parseLineOfTestData(testStr);
            algorithms.ProcessingAlgorithms(lr, rmm, 1, "5");
            //End the counting of the positioning
            countPositioning.endCount();

            currentClientCpuTime=countPositioning.getSeconds();
            clientCpuTime+=currentClientCpuTime;

            out.println("continue");    //Constant HERE ACK message
                continue
        }

        ++index;

        dataFile.writeLineToFile(currentBytesSend+"\t"+
            currentBytesReceive+"\t"+currentClientCpuTime+"\t"+"N/A"+
            "\t"+currentMessagesSent+"\t"+currentMessagesRsv);

        //Init the current variables
        currentBytesReceive=0;
```

```
                currentMessagesRsv=0;
                currentMessagesSent=0;
            }
            dataInfo.setBytesClientSend(bytesSend);
            dataInfo.setBytesServerSend(bytesReceive);
            dataInfo.setClientCpuTime(clientCpuTime);
            dataInfo.setMessagesSent(messagesSent);
            dataInfo.setMessagesRsv(messagesReceived);

            dataFile.closeTheFile();

            Information.messages=msgs;
            Information.bytesSendReceive=bytes;

            return dataInfo;
        }

        /**
         * Formats the line and puts in an ArrayList to use then for
             positioning
         * @param line
         * @return ArrayList<LogRecord>
         */
        private ArrayList<LogRecord> parseLineOfTestData(String line)
        {
            if(line.equals("END"))
            {
                return null;
            }
            ArrayList<LogRecord> lr = new ArrayList<LogRecord>();
            String[] lineArray = line.split(", ");
            for(int i=0; i<lineArray.length-1; i+=2)
            {
                String tempFloat = lineArray[i+1];
                tempFloat = tempFloat.replace(",", ".");
                LogRecord temp= new LogRecord(lineArray[i], Float.parseFloat
                    (tempFloat));
                lr.add(temp);
            }

            return lr;
        }

        /**
         *
         * @return the Radio Map
         */
        public ArrayList<String> getRadioMap()
        {
            return RadioMap;
        }

        /**
         * Close the connections like socket
         * @throws IOException
         */
        public void closeConnections() throws IOException
        {
            out.close();
```



```
            in.close();
            stdIn.close();
            kkSocket.close();
        }

    }
```

```
package Other;

public class CountTime
{
    private long start;
    private long elapsedTimeMillis;
    public CountTime()
    {
        start = System.currentTimeMillis();
    }

    public void endCount()
    {
        elapsedTimeMillis = System.currentTimeMillis()-start;
    }

    public float getSeconds()
    {
        return elapsedTimeMillis/1000F;
    }
}
```

Simulator Client code


```
package Other;

public class DataInformation
{
    private double bytesClientSend=0;
    private double bytesServerSend=0;
    private double clientCpuTime=0;
    private double serverCpuTime=0;
    private int messagesSent=0;
    private int messagesRsv=0;

    public void setBytesClientSend(double bytesClientSend)
    {
        this.bytesClientSend = bytesClientSend;
    }
    public double getBytesClientSend()
    {
        return bytesClientSend;
    }
    public void setBytesServerSend(double bytesServerSend)
    {
        this.bytesServerSend = bytesServerSend;
    }
    public double getBytesServerSend()
    {
        return bytesServerSend;
    }
    public void setClientCpuTime(double clientCpuTime)
    {
        this.clientCpuTime = clientCpuTime;
    }
    public double getClientCpuTime()
    {
        return clientCpuTime;
    }
    public void setServerCpuTime(double serverCpuTime)
    {
        this.serverCpuTime = serverCpuTime;
    }
    public double getServerCpuTime()
    {
        return serverCpuTime;
    }

    public void setMessagesSent(int messagesSent)
    {
        this.messagesSent = messagesSent;
    }
    public int getMessagesSent()
    {
        return messagesSent;
    }
    public void setMessagesRsv(int messagesRsv)
    {
        this.messagesRsv = messagesRsv;
    }
    public int getMessagesRsv()
    {
        return messagesRsv;
```



```
    }

    public String toString()
    {
        return "Bytes Client Send: "+bytesClientSend+" Bytes Server Send:
            "+bytesServerSend+" clientCpuTime: "+clientCpuTime+"
            serverCpuTime: "+serverCpuTime+" msgsSend: "+messagesSent+"
            msgsRsv: "+messagesRsv;
    }
}
```

```
package approaches;

import java.io.IOException;

import Other.*;

/**
 * Class for the whole procedure
 * @author Silouanos
 *
 */
public class Decentralized extends Approach
{
    private String radioMap;
    private String testRadioMap;

    /**
     * @param ip
     * @param port
     * @param radio_map
     *                  name of the radio-map
     */
    public Decentralized(String ip, Integer port, String radio_map, String
        testRadioMap, String dataFile)
    {
        super(ip, port,dataFile);
        this.radioMap = radio_map;
        this.testRadioMap=testRadioMap;
    }

    @Override
    public void run()
    {
        //Just take the first line from test-data-radio-map-mean for
            positioning
        ParseFile pf = null;
        DataInformation dataInfo=null;
        try
        {
            pf = new ParseFile(testRadioMap);
        } catch (IOException e)
        {
            e.printStackTrace();
        }

        CountTime ct = new CountTime();

        //Take the radio-map from server
        Client clt = new Client(ip, port,nameOfDataFile);
        try
        {
            synchronized (clt)
            {
                dataInfo = clt.incomingDataFromServer(pf,pf.
                    getTheNoOfMacAddresses());
            }


        } catch (IOException e4)
```



```
        {
            e4.printStackTrace();
        }
        try
        {
            clt.closeConnections();
        } catch (IOException e3)
        {
            e3.printStackTrace();
        }
        ct.endCount();

        // Write radio-map to a file
        WriteData wrm = null;
        try
        {
            wrm = new WriteData(radioMap);
        } catch (IOException e2)
        {
            e2.printStackTrace();
        }
        try
        {
            wrm.writeArrayListToFile(clt.getRadioMap());
        } catch (IOException e1)
        {
            e1.printStackTrace();
        }


        // Positioning with KNN algorithm
        //Just take the first line from test-data-radio-map-mean for
            positioning
        pf = null;
        try
        {
            pf = new ParseFile(testRadioMap);
        } catch (IOException e)
        {
            e.printStackTrace();
        }

        RadioMapMean rmm = new RadioMapMean();
        try
        {
            rmm.createRadioMap(pf);
        } catch (IOException e)
        {
            e.printStackTrace();
        }



        System.out.println("size of radio-map: "+pf.getSizeOfTheFile() +"
            radio-map22: "+clt.getRadioMap().size() );
        //How many lines I have in radio-map!!! (X Lines of test-radio-map
            x ConstantOfPositioning x X Lines of radio-map)
        Information.timeToPosition=pf.getSizeOfTheFile()*Parameters.
            mobileAlgExecution*clt.getRadioMap().size();
```

```
            /**
             * Here I have to do bytes + bytes that was sent each time of
                 positioning
             * For each positioning on mobile, server is sending a new radio-
                 map if the parameter is >0%
             */

            Information.timeOfConnection=ct.getSeconds();
            /**
             * The same as bytes because we assume that it takes some time to
                 send again the "new" radio map
             * time+=time*(parameter)%
             */

            Information.timeWholeProcedure=ct.getSeconds()+Information.
                timeToPosition+Information.timeOfConnection;

        }

    }
```

```
package Other;

public class FileException extends Exception
{
    String mistake;

    public FileException()
    {
        super();
        mistake="unknown";
    }

    public FileException(String error)
    {
        mistake=error;
    }

    public String getError()
    {
        return mistake;
    }

}
```

```
package Other;

public class Information
{
    public static int messages;
    public static float timeToPosition;
    public static float timeWholeProcedure;
    public static double bytesSendReceive;
    public static float timeOfConnection;

    /* Is used only by bloom approach */
    public static float sumOfSizeOfRadioMapSmall;

    /**
     * Method to show the current information
     */
    public static void showData()
    {
        System.out.println("Messages: "+messages+" Whole time: "+
            timeWholeProcedure+"s Bytes: "+bytesSendReceive+" Time to
            position: "+timeToPosition);
    }
}
```

```
package Other;

public class LocDistance
{
    private double distance;
    private String location;

    public LocDistance(double distance, String location)
    {
        this.distance = distance;
        this.location = location;
    }

    public double getDistance()
    {
        return distance;
    }

    public String getLocation()
    {
        return location;
    }
}
```

```
package Other;
public class LogRecord
{

    private String bssid;
    private float rss;

    public LogRecord(String bssid, float rss)
    {
        super();
        this.bssid = bssid;
        this.rss = rss;
    }

    public String getBssid()
    {
        return bssid;
    }

    public float getRss()
    {
        return rss;
    }

    public String toString()
    {
        String str = new String();
        str = String.valueOf(bssid) + " " + String.valueOf(rss) + "\n";
        return str;
    }

}
```

Simulator Client code


```
package Other;
import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;


public class Parameters
{
    public static String ipAddress;
    public static String port;
    public static String choice;
    public static String clients;
    public static String nameOfRadioMap;
    public static String testRadioMap;
    public static int decStartRequestBytes;
    public static int decStartAckBytes;
    public static int decAckBytes;
    public static float mobileAlgExecution;
    public static float serverAlgExecution;
    public static int bloomVectorLen;
    public static String nameOfDataFile;
    //Centralized
    public static int okReadyACK;
    public static int byteLenXY;
    public static int deviceIDByteLen;
    public static int rssValueByteLen;
    public static int macAddressesByteLen;


    public Parameters() throws IOException
    {
        ArrayList<String> arrayList = new ArrayList<String>();
        FileInputStream fstream = null;
        try
        {
            fstream = new FileInputStream("parameters.txt");
        } catch (FileNotFoundException e)
        {
            e.printStackTrace();
        }
        // Get the object of DataInputStream
        DataInputStream in = new DataInputStream(fstream);
        BufferedReader br = new BufferedReader(new InputStreamReader(in));
        String strLine;
        // Read File Line By Line
        while ((strLine = br.readLine()) != null)
        {
            // Print the content on the console
            String[] tempArray = strLine.split(" ");
            try
            {
                validateFile(tempArray[0]);
            }
            catch(FileException fe)
            {
```



```
                System.err.println("Error in file: "+fe.getError());
                System.exit(-1);
            }

            arrayList.add(tempArray[1]);
        }
        in.close();
        ipAddress=arrayList.get(0);
        port=arrayList.get(1);
        choice=arrayList.get(2);
        clients=arrayList.get(3);
        nameOfRadioMap=arrayList.get(4);
        testRadioMap=arrayList.get(5);
        decStartRequestBytes=Integer.valueOf(arrayList.get(6));
        decStartAckBytes = Integer.valueOf(arrayList.get(7));
        decAckBytes = Integer.valueOf(arrayList.get(8));
        mobileAlgExecution = Float.valueOf(arrayList.get(9));
        serverAlgExecution = Float.valueOf(arrayList.get(10));
        okReadyACK=Integer.valueOf(arrayList.get(11));
        byteLenXY=Integer.valueOf(arrayList.get(12));
        deviceIDByteLen = Integer.valueOf(arrayList.get(13));
        rssValueByteLen = Integer.valueOf(arrayList.get(14));
        macAddressesByteLen=Integer.valueOf(arrayList.get(15));
        bloomVectorLen=Integer.valueOf(arrayList.get(16).replace("%", ""))
            ;
        nameOfDataFile=arrayList.get(17)+choice+".xls";
    }

    private static void validateFile(String str) throws FileException
    {

        if(str.equals("IP") || str.equals("PORT") || str.equals("CHOICE")
            || str.equals("NO_OF_CLIENTS") || str.equals(
            "NAME_OF_RADIOMAP") || str.contains("DEC") || str.contains
            ("ALG") || str.equals("OK_READY_ACK") || str.contains("LEN") |
            | str.equals("BLOOM_VECTOR_LEN") || str.equals
            ("TEST_RADIOMAP_FOR_POS") || str.equals("NAME_OF_DATA_FILE"))
        {
            return;
        }
        else
        {
            throw new FileException("Invalidate parameter file!");
        }
    }
}
```

Simulator Client code


```
package Other;
import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;

public class ParseFile
{
    private ArrayList<String> dataArray = null;
    private ArrayList<String> formattedArrayList = null;
    public static int noOfMacAddresses;
    //Store the mac addresses
    private ArrayList<String> macAddresses = new ArrayList<String>();
    DataInputStream in = null;
    BufferedReader br = null;
    private static String nameOfTheRadioMap=null;
    private int sizeOfFile = 0;

    public ParseFile(String nameOfTheFile) throws IOException
    {

        //dataArray=new ArrayList<String>();
        this.nameOfTheRadioMap=nameOfTheFile;
        FileInputStream fstream = null;
        try
        {
            fstream = new FileInputStream(nameOfTheFile);
        } catch (FileNotFoundException e)
        {
            e.printStackTrace();
        }
        // Get the object of DataInputStream
        in = new DataInputStream(fstream);
        br = new BufferedReader(new InputStreamReader(in));

        //Parse the first mac address line
        String strLine = br.readLine();
        String[] macAddressesArray = strLine.split(", ");
        noOfMacAddresses=macAddressesArray.length;

        for(int i=2; i<macAddressesArray.length; ++i)
        {
            macAddresses.add(macAddressesArray[i]);
        }

        // Read File Line By Line
//      while ((strLine = br.readLine()) != null)
//      {
//          dataArray.add(strLine);
//      }
        //in.close();
    }

    public void closeTheFile() throws IOException
    {
        in.close();
```



```
    }

    public String getTheNextLine() throws IOException
    {
        ++sizeOfFile;
        return parseCentralizedLine();
    }

    public ArrayList<String> getTheAllTheMacAddresses()
    {
        return macAddresses;
    }

    public int getSizeOfTheFile()
    {
        return sizeOfFile;
    }

    /**
     * Used for Centralized Approach
     * Convert the arrayList into a specific format for sending
     * @return the formated arrayList
     * @throws IOException
     */
    private String parseCentralizedLine() throws IOException
    {
        String currentLine = br.readLine();
        if(currentLine==null)
        {
            return null;
        }
        else
        {
            String[] currentRss = currentLine.split(", ");
            String str = "";
            for(int j=0,z=2; j<macAddresses.size(); ++j,++z)
            {
                String macStr=macAddresses.get(j);
                str+=macStr+", "+currentRss[z];

                if(!((macAddresses.size()-1) == j))
                {
                    str+=", ";
                }
            }
            return str;
        }
    }

    public int getTheNoOfMacAddresses()
    {
        return noOfMacAddresses;
    }

    public ParseFile(ArrayList<String> incomingArrayList)
    {
        dataArray=incomingArrayList;
        formattedArrayList=new ArrayList<String>();
        parseCentralized();
```

```
        }

        public String getTheNextLine(int no)
        {
            return formattedArrayList.get(no);
        }

        public ArrayList<String> getFormattedArrayList()
        {
            return formattedArrayList;
        }


        /**
         * Used for Centralized Approach
         * Convert the arrayList into a specific format for sending
         * @return the formated arrayList
         */
        private ArrayList<String> parseCentralized()
        {
            //Store the mac addresses
            ArrayList<String> macAddresses = new ArrayList<String>();

            String[] macAddressesArray = dataArray.get(0).split(", ");
            noOfMacAddresses=macAddressesArray.length;

            for(int i=2; i<macAddressesArray.length; ++i)
            {
                macAddresses.add(macAddressesArray[i]);
            }

            for(int i=1; i<dataArray.size(); ++i)
            {
                String[] currentRss = dataArray.get(i).split(", ");
                String str = "";
                for(int j=0,z=2; j<macAddresses.size(); ++j,++z)
                {
                    String macStr=macAddresses.get(j);
                    str+=macStr+", "+currentRss[z];

                    if(!((macAddresses.size()-1) == j))
                    {
                        str+=", ";
                    }

                }
                formattedArrayList.add(str);
            }
            return formattedArrayList;
        }

        /**
         * Get the first line to take the mac Addresses
         *
         * @return an array list of LogRecord
         */
        public ArrayList<LogRecord> getFirstLineFromTestFile()
        {
            ArrayList<LogRecord> lr = new ArrayList<LogRecord>();
```



```
            String macAddressesStr= dataArray.get(0);
            macAddressesStr=macAddressesStr.replace("# ", "");
            String firstLine = dataArray.get(1);
            String[] macAddressesArray = macAddressesStr.split(", ");
            String[] firstLineArray = firstLine.split(", ");
            for(int i=2; i<macAddressesArray.length; ++i)
            {
                LogRecord l = new LogRecord(macAddressesArray[i], Float.
                    valueOf(firstLineArray[i].replace(",", ".")));
                lr.add(l);
            }

            return lr;
        }

        public int getTheSizeOfMacAddresses()
        {
            return noOfMacAddresses;
        }

        public ArrayList<String> getDataArray()
        {
            return dataArray;
        }

        public String getTheNameOfTheFile()
        {
            return nameOfTheRadioMap;
        }
    }
```

Simulator Client code


```
package Other;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;


public class RadioMapMean
{
    private ArrayList<String> MacAdressList = null; //Whole macAddresses
    private HashMap<String, ArrayList<String>> LocationRSS_HashMap = null;

    public RadioMapMean()
    {
        MacAdressList = new ArrayList<String>();
        LocationRSS_HashMap = new HashMap<String, ArrayList<String>>();
    }

    /**
     * Create the radio map from an incoming arraylist
     *
     * @param incomingArrayList
     * @throws IOException
     */

    public void createRadioMap(ParseFile incomingArrayList) throws
        IOException
    {
        String[] temp = null;
        MacAdressList=incomingArrayList.getTheAllTheMacAddresses();


        while(true)
        {
            //Take the rest data
            String str = incomingArrayList.getTheNextLine();
            if(str==null)
            {
                break;
            }
            str = str.replace(",", "");
            temp = str.split(" ");
            String key = temp[0]+" "+temp[1];
            ArrayList<String> rssValues = new ArrayList<String>();
            for(int j=2; j<temp.length; ++j)
            {
                rssValues.add(temp[j]);
            }

            LocationRSS_HashMap.put(key, rssValues);
        }
    }

    /**
     * Create the radio map
     *
     * @param incomingArrayList
     */
    public void createRadioMap(ArrayList<String> incomingArrayList)
    {
```



```
        //Take the first line and put it in MacAddressList
        String macAddresses = new String(incomingArrayList.get(0));
        macAddresses = macAddresses.replace(",", "");
        String [] temp = macAddresses.split(" ");

        for(int i=3; i<temp.length; ++i)
        {
            MacAdressList.add(temp[i]);
        }

        //Take the rest data
        for(int i=1; i<incomingArrayList.size(); ++i)
        {
            String str = incomingArrayList.get(i);
            str = str.replace(",", "");
            temp = str.split(" ");
            String key = temp[0]+" "+temp[1];
            ArrayList<String> rssValues = new ArrayList<String>();
            for(int j=2; j<temp.length; ++j)
            {
                rssValues.add(temp[j]);
            }
            LocationRSS_HashMap.put(key, rssValues);
        }
    }


    public HashMap<String, ArrayList<String>> getLocationRSS_HashMap()
    {
        return LocationRSS_HashMap;
    }

    public ArrayList<String> getMacAdressList()
    {
        return MacAdressList;
    }
}
```

Simulator Client code


```
package Other;
import approaches.*;
import java.io.IOException;

public class StartClient
{
    /**
     * @param args
     * @throws IOException
     * @throws InterruptedException
     */
    public static void main(String[] args) throws IOException,
        InterruptedException
    {
        // Get the parameters from the parameters file
        Parameters pr = new Parameters();
        Thread[] threadArray = new Thread[Integer.valueOf(pr.clients)];

        if (pr.choice.equals("DEC"))
        {
            for(int i=0; i<threadArray.length; ++i)
            {
                Decentralized dec = new Decentralized(pr.ipAddress,
                    Integer.valueOf(pr.port),pr.nameOfRadioMap,pr.
                    testRadioMap,pr.nameOfDataFile);
                threadArray[i] = new Thread(dec);
            }

            for(int i=0; i<threadArray.length; ++i)
            {
                threadArray[i].start();
            }

            for(int i=0; i<threadArray.length; ++i)
            {
                threadArray[i].join();
                Information.showData();
            }


        }
        else if(pr.choice.equals("CEN"))
        {
            Centralized cen = new Centralized(pr.ipAddress, Integer.
                valueOf(pr.port), pr.testRadioMap,"1",pr.nameOfDataFile);
            new Thread(cen).start();
        }
        else if(pr.choice.equals("BLOOM"))
        {
            System.out.println("Bloom is starting from client!");
            BloomFilter bf = new BloomFilter(pr.ipAddress, Integer.valueOf
                (pr.port), pr.testRadioMap,pr.nameOfDataFile);
            new Thread(bf).start();
        }
        else
        {
            //Wrong choice
            System.exit(-1);
        }
```



```
        }
    }
```

Simulator Client code


```
package Other;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.Writer;
import java.util.ArrayList;


public class WriteData
{
    private Writer output = null;
    private File file = null;

    /**
     * Initialize the streams
     *
     * @param fileName
     * @throws IOException
     */
    public WriteData(String fileName) throws IOException
    {
        file = new File(fileName);
        output= new BufferedWriter(new FileWriter(file));
    }

    /**
     *
     * @param line
     */
    public void writeLineToFile(String line)
    {
        try
        {
            output.write(line+"\n");
        } catch (IOException e)
        {
            e.printStackTrace();
        }
    }


    public void closeTheFile()
    {
        try
        {
            output.close();
        } catch (IOException e)
        {
            e.printStackTrace();
        }
    }

    /**
     *
     * @param incomingList
     * @throws IOException
     */
    public void writeArrayListToFile(ArrayList<String> incomingList)
```



```
            throws IOException
    {
        for(String str : incomingList)
        {
            str+="\n";
            output.write(str);
        }

        output.close();
    }
}
```

Simulator Server code


```
package Positioning;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;


public class Algorithms
{
    /**
     * @param latestScanList
     *            the current scan list of APs
     * @param RM
     *            the constructed Radio Map
     *
     * @param algorithm_choice
     *            choice of several algorithms
     *
     * @param WeightsList
     *             the RBF weights
     *
     * @return the location of user
     */
    public static String ProcessingAlgorithms(ArrayList<LogRecord>
        latestScanList, RadioMapMean RM, int algorithm_choice, String
        parameter)
    {

        int i = 0, j = 0;

        ArrayList<String> MacAdressList = RM.getMacAdressList();
        ArrayList<String> Observed_RSS_Values = new ArrayList<String>();
        LogRecord temp_LR = null;

        // Check which mac addresses of radio map, we are currently
            listening.
        for (i = 0; i < MacAdressList.size(); ++i)
        {
            for (j = 0; j < latestScanList.size(); ++j)
            {
                temp_LR = latestScanList.get(j);

                // MAC Address Matched
                if (MacAdressList.get(i).compareTo(temp_LR.getBssid()) ==
                    0)
                {
                    Observed_RSS_Values.add(String.valueOf(temp_LR.getRss
                        ()));
                    break;
                }
            }
            // A MAC Address is missing so we place a small NaN value
            if (j == latestScanList.size())
            {
                Observed_RSS_Values.add(String.valueOf("-110"));
            }
        }

        switch (algorithm_choice)
        {
```



```
                case 1:
                    return KNN_WKNN_Algorithm(RM, Observed_RSS_Values,
                        parameter, false);
//                  case 2:
//                      return KNN_WKNN_Algorithm(RM, Observed_RSS_Values,
    parameter, true);
//                  case 3:
//                      return MAP_MMSE_Algorithm(RM, Observed_RSS_Values,
    parameter, false);
//                  case 4:
//                      return MAP_MMSE_Algorithm(RM, Observed_RSS_Values,
    parameter, true);
                case 6:
//                      return SNAP_Algorithm(RM, Observed_RSS_Values, parameter
    );
            }
            return null;

        }

        /**
         * Calculates user location based on Weighted/Not Weighted K Nearest
         * Neighbor (KNN) Algorithm
         *
         * @param RM
         *            The radio map structure
         *
         * @param Observed_RSS_Values
         *            RSS values currently observed
         * @param parameter
         *
         * @param isWeighted
         *            To be weighted or not
         *
         * @param K
         *            The number of locations used
         *
         * @return The estimated user location
         */
        private static String KNN_WKNN_Algorithm(RadioMapMean RM, ArrayList<
            String> Observed_RSS_Values, String parameter, boolean isWeighted)
        {
//          System.out.println("RadioMapMean: "+RM.getLocationRSS_HashMap()+"\
    n"+RM.getMacAdressList()+"\n"+Observed_RSS_Values);

            ArrayList<String> RSS_Values;
            float curResult = 0;
            ArrayList<LocDistance> LocDistance_Results_List = new ArrayList<
                LocDistance>();
            String myLocation = null;
            int K;

            try
            {
                K = Integer.parseInt(parameter);
            }
            catch (Exception e)
            {
```

```
                System.err.println("Error parsing K in KNN-WKNN: " + e.
                    getMessage());
                return null;
            }

            // Construct a list with locations-distances pairs for currently
            // observed RSS values
            for (String location : RM.getLocationRSS_HashMap().keySet())
            {
                RSS_Values = RM.getLocationRSS_HashMap().get(location);
                curResult = calculateEuclideanDistance(RSS_Values,
                    Observed_RSS_Values);

                if (curResult == Float.NEGATIVE_INFINITY)
                {
                    return null;
                }

                LocDistance_Results_List.add(0, new LocDistance(curResult,
                    location));
            }

            // Sort locations-distances pairs based on minimum distances
            Collections.sort(LocDistance_Results_List, new Comparator<
                LocDistance>()
            {

                public int compare(LocDistance gd1, LocDistance gd2)
                {
                    return (gd1.getDistance() > gd2.getDistance() ? 1 : (gd1.
                        getDistance() == gd2.getDistance() ? 0 : -1));
                }
            });

            if (!isWeighted)
            {
                myLocation = calculateAverageKDistanceLocations
                    (LocDistance_Results_List, K);
            }
            else
            {
                myLocation = calculateWeightedAverageKDistanceLocations
                    (LocDistance_Results_List, K);
            }

            return myLocation;

        }

        /**
         * Calculates user location based on Probabilistic Maximum A
             Posteriori
         * (MAP) Algorithm or Probabilistic Minimum Mean Square Error (MMSE)
         * Algorithm
         *
         * @param RM
         *            The radio map structure
         *
         * @param Observed_RSS_Values
```



```
         *            RSS values currently observed
         * @param parameter
         *
         * @param isWeighted
         *            To be weighted or not
         *
         * @return The estimated user location
         */
        private static String MAP_MMSE_Algorithm(RadioMapMean RM, ArrayList<
            String> Observed_RSS_Values, String parameter, boolean isWeighted)
        {

            ArrayList<String> RSS_Values;
            double curResult = 0.0d;
            String myLocation = null;
            double highestProbability = Double.NEGATIVE_INFINITY;
            ArrayList<LocDistance> LocDistance_Results_List = new ArrayList<
                LocDistance>();
            float sGreek;

            try
            {
                sGreek = Float.parseFloat(parameter);
            }
            catch (Exception e)
            {
                System.err.println("Error parsing parameter of MAP-MMSE: " + e
                    .getMessage());
                return null;
            }

            // Find the location of user with the highest probability
            for (String location : RM.getLocationRSS_HashMap().keySet())
            {

                RSS_Values = RM.getLocationRSS_HashMap().get(location);
                curResult = calculateProbability(RSS_Values,
                    Observed_RSS_Values, sGreek);

                if (curResult == Double.NEGATIVE_INFINITY)
                {
                    return null;
                }
                else if (curResult > highestProbability)
                {
                    highestProbability = curResult;
                    myLocation = location;
                }

                if (isWeighted)
                {
                    LocDistance_Results_List.add(0, new LocDistance(curResult,
                        location));
                }
            }

            if (isWeighted)
            {
                myLocation = calculateWeightedAverageProbabilityLocations
```

```
                (LocDistance_Results_List);
        }

        return myLocation;
    }

    /**
     * Calculates the Euclidean distance between the currently observed
         RSS
     * values and the RSS values for a specific location.
     *
     * @param l1
     *            RSS values of a location in radiomap
     * @param l2
     *            RSS values currently observed
     *
     * @return The Euclidean distance, or MIN_VALUE for error
     */
    private static float calculateEuclideanDistance(ArrayList<String> l1,
        ArrayList<String> l2)
    {

        float finalResult = 0;
        float v1;
        float v2;
        float temp;
        String str;

        for (int i = 0; i < l1.size(); ++i)
        {

            try
            {
                str = l1.get(i);
                v1 = Float.valueOf(str.trim()).floatValue();
                str = l2.get(i);
                v2 = Float.valueOf(str.trim()).floatValue();
            }
            catch (Exception e)
            {
                System.err.println("Error while computing Euclidean
                    Distance: " + e.getMessage());
                return Float.NEGATIVE_INFINITY;
            }

            // do the procedure
            temp = v1 - v2;
            temp *= temp;

            // do the procedure
            finalResult += temp;
        }
        return ((float) Math.sqrt(finalResult));
    }

    /**
     * Calculates the Probability of the user being in the currently
         observed
     * RSS values and the RSS values for a specific location.
```



```
     *
     * @param l1
     *            RSS values of a location in radiomap
     * @param l2
     *            RSS values currently observed
     *
     * @return The Probability for this location, or MIN_VALUE for error
     */
    public static double calculateProbability(ArrayList<String> l1,
        ArrayList<String> l2, float sGreek)
    {

        double finalResult = 1;
        float v1;
        float v2;
        double temp;
        String str;

        for (int i = 0; i < l1.size(); ++i)
        {

            try
            {
                str = l1.get(i);
                v1 = Float.valueOf(str.trim()).floatValue();
                str = l2.get(i);
                v2 = Float.valueOf(str.trim()).floatValue();
            }
            catch (Exception e)
            {
                System.err.println("Error while computing Euclidean
                    Distance: " + e.getMessage());
                return Double.NEGATIVE_INFINITY;
            }

            temp = v1 - v2;

            temp *= temp;

            temp = -temp;

            temp /= (double) (sGreek * sGreek);
            temp = (double) Math.exp(temp);

            finalResult *= temp;
        }
        return finalResult;
    }

    /**
     * Calculates the Average of the K locations that have the shortest
     * distances D
     *
     * @param LocDistance_Results_List
     *            Locations-Distances pairs sorted by distance
     * @param K
     *            The number of locations used
     * @return The estimated user location, or null for error
     */
```

```
    private static String calculateAverageKDistanceLocations(ArrayList<
        LocDistance> LocDistance_Results_List, int K)
    {

        float sumX = 0.0f;
        float sumY = 0.0f;

        String[] LocationArray = new String[2];
        float x, y;

        int K_Min = K < LocDistance_Results_List.size() ? K :
            LocDistance_Results_List.size();

        // Calculate the sum of X and Y
        for (int i = 0; i < K_Min; ++i)
        {
            LocationArray = LocDistance_Results_List.get(i).getLocation().
                split(" ");

            try
            {
                x = Float.valueOf(LocationArray[0].trim()).floatValue();
                y = Float.valueOf(LocationArray[1].trim()).floatValue();
            }
            catch (Exception e)
            {
                System.err.println("Error while calculating Average K
                    Distance Locations: " + e.getMessage());
                return null;
            }

            sumX += x;
            sumY += y;
        }

        // Calculate the average
        sumX /= K_Min;
        sumY /= K_Min;

        return sumX + " " + sumY;

    }

    /**
     * Calculates the Weighted Average of the K locations that have the
         shortest
     * distances D
     *
     * @param LocDistance_Results_List
     *            Locations-Distances pairs sorted by distance
     * @param K
     *            The number of locations used
     * @return The estimated user location, or null for error
     */
    public static String calculateWeightedAverageKDistanceLocations
        (ArrayList<LocDistance> LocDistance_Results_List, int K)
    {

        double LocationWeight = 0.0f;
```



```
        double sumWeights = 0.0f;
        double WeightedSumX = 0.0f;
        double WeightedSumY = 0.0f;

        String[] LocationArray = new String[2];
        float x, y;

        int K_Min = K < LocDistance_Results_List.size() ? K :
            LocDistance_Results_List.size();

        // Calculate the weighted sum of X and Y
        for (int i = 0; i < K_Min; ++i)
        {

            LocationWeight = 1 / LocDistance_Results_List.get(i).
                getDistance();
            LocationArray = LocDistance_Results_List.get(i).getLocation().
                split(" ");

            try
            {
                x = Float.valueOf(LocationArray[0].trim()).floatValue();
                y = Float.valueOf(LocationArray[1].trim()).floatValue();
            }
            catch (Exception e)
            {
                System.err.println("Error while calculating Weighted
                    Average K Distance Locations: " + e.getMessage());
                return null;
            }

            sumWeights += LocationWeight;
            WeightedSumX += LocationWeight * x;
            WeightedSumY += LocationWeight * y;

        }

        WeightedSumX /= sumWeights;
        WeightedSumY /= sumWeights;

        return WeightedSumX + " " + WeightedSumY;
    }

    /**
     * Calculates the Weighted Average over ALL locations where the
         weights are
     * the Normalized Probabilities
     *
     * @param LocDistance_Results_List
     *            Locations-Probability pairs
     *
     * @return The estimated user location, or null for error
     */
    public static String calculateWeightedAverageProbabilityLocations
        (ArrayList<LocDistance> LocDistance_Results_List)
    {

        double sumProbabilities = 0.0f;
        double WeightedSumX = 0.0f;
```

```
            double WeightedSumY = 0.0f;
            double NP;
            float x, y;
            String[] LocationArray = new String[2];

            // Calculate the sum of all probabilities
            for (int i = 0; i < LocDistance_Results_List.size(); ++i)
            {
                sumProbabilities += LocDistance_Results_List.get(i).
                    getDistance();
            }

            // Calculate the weighted (Normalized Probabilities) sum of X and
                Y
            for (int i = 0; i < LocDistance_Results_List.size(); ++i)
            {
                LocationArray = LocDistance_Results_List.get(i).getLocation().
                    split(" ");

                try
                {
                    x = Float.valueOf(LocationArray[0].trim()).floatValue();
                    y = Float.valueOf(LocationArray[1].trim()).floatValue();
                }
                catch (Exception e)
                {
                    System.err.println("Error while calculating Weighted
                        Average Probability Locations: " + e.getMessage());
                    return null;
                }

                NP = LocDistance_Results_List.get(i).getDistance() /
                    sumProbabilities;

                WeightedSumX += (x * NP);
                WeightedSumY += (y * NP);

            }

            return WeightedSumX + " " + WeightedSumY;

        }
    }
```

```java
package approaches;

import java.net.Socket;

public class Approach implements Runnable
{
    protected final int port;
    protected Socket clientSocket = null;
    protected static int taskCount = 0;
    protected final int id = taskCount++;

    public Approach(int port, Socket clientSocket)
    {
        this.port = port;
        this.clientSocket=clientSocket;
    }

    @Override
    public void run()
    {
    }
}
```

Simulator Server code


```
package Other;

import Positioning.*;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;

import approaches.Bloom;


public class ApproachesThread
{
    private Socket clientSocket = null;
    private PrintWriter out = null;
    private BufferedReader in = null;
    private ArrayList<String> sendArrayList = null;

    public ApproachesThread(Socket clientSocketIncoming) throws
        IOException
    {
         this.clientSocket=clientSocketIncoming;
         out = new PrintWriter(clientSocket.getOutputStream(), true);
         in = new BufferedReader(new InputStreamReader(clientSocket.
             getInputStream()));
    }

    /**
     * If server wants to send a message to pressure client to wait
         because of
     * processing of the server
     *
     * @param message
     */
    public void sendToClientAMessage(String message)
    {
        out.println(message);
    }

    /**
     * Incoming bloom filter coding from client.
     * Do procedure and send a portion of a radio map.
     *
     * @throws IOException
     */
    public void selectAndSendRadioMap(ParseFile radioMap, Bloom bl) throws
        IOException
    {
        String inputLine = new String(), outputLine;
        ArrayList<String> sendArrayList = null;
        outputLine = "+OK READY";
        out.println(outputLine);
        boolean flag = true;
        CountTime ctOfServerProcedure = null;
        while ((inputLine = in.readLine()) != null)
```



```
        {
            if(!inputLine.equals("END"))
            {
                if(flag)
                {
                    ctOfServerProcedure = new CountTime();
                }

                CountTime timeToMatchAddresses = new CountTime();

                sendArrayList = bl.matchMacAddresses(inputLine);
                timeToMatchAddresses.endCount();

                if(flag)
                {
                    ctOfServerProcedure.endCount();
                    Information.timeWholeProcedure=ctOfServerProcedure.
                        getSeconds();
                    flag = false;
                }
                //Send the size of the radio-map
                out.println(sendArrayList.size());
                //Send the radio-map
                for(int i=0; i<sendArrayList.size(); ++i)
                {
                    out.println(sendArrayList.get(i));
                }

                //Send the time of processing in server
                out.println(timeToMatchAddresses.getSeconds());

                out.println("continue");
            }
        }
    }

    /**
     * Incoming data from client. Here Server receives the test-data-radio
         -map
     * Centralized Approach
     *
     * @throws IOException
     */
    @SuppressWarnings("static-access")
    public void incomingDataFromClient(RadioMapMean rmm) throws
        IOException
    {
        String inputLine = new String(), outputLine, timeToPosition;
        outputLine = "+OK READY";
        out.println(outputLine);
        Algorithms algorithms = null;
        while ((inputLine = in.readLine()) != null)
        {
            if(!inputLine.equals("END"))
            {
                //Positioning and send back the position to client
                String[] dIDLine = takeTheDeviceID(inputLine);

                //Start the counting of the positioning
```

```
                    CountTime countPositioning = new CountTime();
                    ArrayList<LogRecord>lr = parseLineOfTestData(dIDLine[1]);
                    algorithms = new Algorithms();
                    outputLine = algorithms.ProcessingAlgorithms(lr, rmm, 1,
                        "5");
                    //End the counting of the positioning
                    countPositioning.endCount();
                    timeToPosition = String.valueOf(countPositioning.
                        getSeconds());
                    out.println(timeToPosition);
                }
                else
                {
                    outputLine="END";
                    break;
                }
                out.println(outputLine);
                if (inputLine.equals("END"))
                {
                    System.out.println("END!!!");
                    break;
                }
            }
        }

        /**
         * Sending the line and isolates the device ID
         *
         * @param line
         * @return the device ID
         */
        private String[] takeTheDeviceID(String line)
        {
            String[] temp = line.split(", ");
            line="";
            String[] arrayToSend = new String[2];

            for(int i=1; i<temp.length; ++i)
            {
                if(i!=temp.length-1)
                {
                    line+=temp[i]+", ";
                }
                else
                {
                    line+=temp[i];
                }
            }

            arrayToSend[0]=temp[0];
            arrayToSend[1]=line;
            return arrayToSend;
        }


        /**
         * Formats the line and puts in an ArrayList to use then for
             positioning
         * @param line
```



```
         * @return ArrayList<LogRecord>
         */
        private ArrayList<LogRecord> parseLineOfTestData(String line)
        {
            if(line.equals("END"))
            {
                return null;
            }
            ArrayList<LogRecord> lr = new ArrayList<LogRecord>();
            String[] lineArray = line.split(", ");
            for(int i=0; i<lineArray.length-1; i+=2)
            {
                String tempFloat = lineArray[i+1];
                tempFloat = tempFloat.replace(",", ".");
                LogRecord temp= new LogRecord(lineArray[i], Float.parseFloat
                    (tempFloat));
                lr.add(temp);
            }

            return lr;
        }

        /**
         * Server is sending the final radio-map.txt to client
         * Decentralized Approach
         *
         * @throws IOException
         */
        public void outgoingData() throws IOException
        {
            System.out.println("size of the array: " + sendArrayList.size());
            String inputLine;
            int index = 0;
            out.println("INCOMING RADIO-MAP");


                while ((inputLine = in.readLine())!=null)
                {
                    if (!inputLine.equals("end"))
                    {
                        for(int i=0; i<sendArrayList.size(); ++i)
                        {
                            out.println(sendArrayList.get(i));
                        }
                        out.println("END OF RM");
                    }
                    else
                    {
                        break;
                    }
                    ++index;
                }
        }

        /**
         * Just process the radio-map to send it to the client
         *
         * @param LocationRSS_HashMap
         *            Contains the geolocation and the respectively rss values
```

```
         *
         * @param macAddresses
         *            All the mac addresses that took from the radio-map
         *
         * @throws IOException
         */

        public void sendRadioMapToClient(HashMap<String, ArrayList<String>>
            LocationRSS_HashMap,HashSet<String> macAddresses) throws
            IOException
        {
            System.out.println("LocationRSSHashMap: "+LocationRSS_HashMap);
            sendArrayList = new ArrayList<String>();
            String MacAddressesStr = "# Longitude, Latitude";
            for (String str : macAddresses)
            {
                MacAddressesStr += ", " + str;
            }
            sendArrayList.add(MacAddressesStr);
            MacAddressesStr = null;

            for (String key : LocationRSS_HashMap.keySet())
            {
                String tempStr = null;
                String[] temp = key.split(" ");
                tempStr = temp[0] + ", " + temp[1];

                ArrayList<String> rssValues = LocationRSS_HashMap.get(key);

                for (String str : rssValues)
                {
                    tempStr += ", " + str;
                }

                sendArrayList.add(tempStr);
            }
            // Method that sends the radio-map to client
            outgoingData();
        }

        /**
         * Method that helps us to set the array direct to the sendArrayList
             to send it without processing
         * @param incoming
         */
        public void setSendArrayList(ArrayList<String> incoming)
        {
            sendArrayList=incoming;
        }


        /**
         * Close all connections.
         *
         * @throws IOException
         */
        public void closeConnections() throws IOException
        {
            out.close();
```



```
            in.close();
            clientSocket.close();
        }
    }
```

Simulator Server code


```
package approaches;

import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;

import Positioning.RadioMapMean;


public class Bloom
{
    final private int K;
    final private int noOfHashFunctions;
    private int sizeOfFinalVector;
    private RadioMapMean rmm = new RadioMapMean();
    private ArrayList<String> macAddresses = new ArrayList<String>();
    private ArrayList<String> rssLocation = new ArrayList<String>();
    private ArrayList<String> radioMap = null;
    private HashMap<String, String> macAddressesBFilter = new HashMap<
        String, String>();

    /**
     *
     * @param radioMap
     * @param k
     * @param noOfHash
     */
    public Bloom(ArrayList<String> radioMap, int k, int noOfHash)
    {
        this.radioMap=radioMap;
        this.K=k;
        this.noOfHashFunctions=noOfHash;

        //Create a radio-map to separate the MacAddresses and RSS values
        rmm.createRadioMap(radioMap);

        //Get the macAddresses
        macAddresses = rmm.getMacAdressList();
        proccessTestFile();

        //Find the size of the Bloom vector
        sizeOfFinalVector=computeTheSizeOfBloomVector();

        //Full the HashMap
        for(int i=0; i<macAddresses.size(); ++i)
        {
            try
            {
                macAddressesBFilter.put(macAddresses.get(i),
                    doTheBloomProcedure(macAddresses.get(i)));
            } catch (NoSuchAlgorithmException e)
            {
                e.printStackTrace();
            }
        }
    }
```



```
    /**
     * Match the incoming macAddress with all the lines of radio-map that
         relates
     *
     * @param clientBloomStr
     * @return the new small radio-map
     */
    public ArrayList<String> matchMacAddresses(String clientBloomStr)
    {
        ArrayList<String> macAddressesMatched = new ArrayList<String>();

        for(String macAddress : macAddressesBFilter.keySet())
        {
            String strBloomServer = macAddressesBFilter.get(macAddress);
            if(strBloomServer.equals(clientBloomStr))
            {
                macAddressesMatched.add(macAddress);
            }
        }

        return removeDupFromArrayList(createTheNewRadioMap
            (macAddressesMatched));
    }

    /**
     * Creating the new small radio-map that will be send to client for
         positioning
     *
     * @param macAddressesMatched
     * @return the small radio-map
     */
    private ArrayList<String> createTheNewRadioMap(ArrayList<String>
        macAddressesMatched)
    {
        ArrayList<String> newRadioMap = new ArrayList<String>();
        for(int i=0; i<macAddressesMatched.size(); ++i)
        {
            int placeOfMacA = findTheNoOfTheMacAddress(macAddressesMatched
                .get(i));

            for(String line : rssLocation)
            {
                String[] temp = line.split(", ");
                if(!temp[placeOfMacA+2].equals("-110"))
                {
                    newRadioMap.add(line);
                }

            }
        }

        return newRadioMap;
    }

    /**
     * Compute the K
     *
     * @param sizeOfVector
```

```
     * @return k
     */

    private int computeTheK(int sizeOfVector)
    {
        int M = macAddresses.size();
        int b = sizeOfVector;

        int k = (int) (M*Math.pow((1-Math.exp((-noOfHashFunctions*M)/b)),
            noOfHashFunctions));

        return k;
    }

    /**
     * Return the number in the arrayList of macAddresses
     * @param macAddress
     * @return
     */

    private int findTheNoOfTheMacAddress(String macAddress)
    {
        for(int i=0; i<macAddresses.size(); ++i)
        {
            if(macAddress.equals(macAddresses.get(i)))
            {
                return i;
            }
        }
        return -1;
    }

    /**
     * Compares two Booleans arrays
     *
     * @param array1
     * @param array2
     * @return true or false
     */
    private Boolean compareBooleanArrays(Boolean[] array1, Boolean[]
        array2)
    {
        for(int i=0; i<array1.length; ++i)
        {
            if(array1[i]!=array2[i])
            {
                return false;
            }
        }
        return true;
    }
    /**
     * Convert a string bloom vector to boolean array
     *
     * @param clientBloomStr
     * @return the bloom vector in boolean type
     *
     */
    private Boolean[] convertStrToBooleanArray(String clientBloomStr)
```



```
    {
        Boolean[] arrayToReturn = new Boolean[sizeOfFinalVector];
        //System.out.println("Size of final vector: "+sizeOfFinalVector+"
            strLen: "+clientBloomStr.length()+" macAddressSize: "+
            macAddresses.size());

        for(int i=0; i<clientBloomStr.length()-1; ++i)
        {
            if(clientBloomStr.substring(i, i+1).equals("0"))
            {
                arrayToReturn[i]=false;
            }
            else
            {
                arrayToReturn[i]=true;
            }
        }

        return arrayToReturn;
    }

    /**
     * A little processing to radioMap arrayList
     */
    private void proccessTestFile()
    {
        for(int i=1; i<radioMap.size(); ++i)
        {
            rssLocation.add(radioMap.get(i));
        }
    }

    /**
     * Do the Bloom procedure for a specific macAddress
     *
     * @param macAddress
     * @return bloom vector in string type
     * @throws NoSuchAlgorithmException
     */
    private String doTheBloomProcedure(String macAddress) throws
        NoSuchAlgorithmException
    {
        Boolean[] bloomFilter = new Boolean[sizeOfFinalVector];
        bloomFilter = initializeTheBloomFilter(sizeOfFinalVector,
            bloomFilter);
        String reversedMacAddress = new StringBuffer(macAddress).reverse()
            .toString();

        if(this.noOfHashFunctions>=1)
        {
            int resOfH1 = macAddress.hashCode();
            if(resOfH1<0)
            {
                resOfH1*=(-1);
            }
            //H1
            bloomFilter[resOfH1%sizeOfFinalVector]=true;
        }
```

```
        if(this.noOfHashFunctions>=2)
        {
            int resOfH2 = MD5HashFunction(macAddress);
            if(resOfH2<0)
            {
                resOfH2*=(-1);
            }
            //H2
            bloomFilter[resOfH2%sizeOfFinalVector]=true;
        }

        /**
         * Attention here!!! I put (-) because is negative
         */

        if(this.noOfHashFunctions>=3)
        {
            int resOfH3 = reversedMacAddress.hashCode();
            if(resOfH3<0)
            {
                resOfH3*=(-1);
            }
            //H3
            bloomFilter[resOfH3%sizeOfFinalVector]=true;
        }

        if(this.noOfHashFunctions>=4)
        {
            int resOfH4 = MD5HashFunction(reversedMacAddress);

            if(resOfH4<0)
            {
                resOfH4*=(-1);
            }
            //H4
            bloomFilter[resOfH4%sizeOfFinalVector]=true;
        }

        return convertBloomToString(bloomFilter);
    }

    /**
     * Use the hash function of MD5 and then use the result with hashCode
     *
     * @param macAddress
     * @return a value
     * @throws NoSuchAlgorithmException
     */
    private int MD5HashFunction(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();
    }

    /**
     * Compute the size of the bloom vector
```



```
     * @return the computed size of the bloom vector
     */
    private int computeTheSizeOfBloomVector()
    {

        float fpr = (float)K/macAddresses.size();           //K=1 & size:
            29


        int upperNo = -(noOfHashFunctions*macAddresses.size());

        double downNo = Math.log((double)1-Math.pow((double)fpr, ((double)
            1/noOfHashFunctions)))/(Math.log(Math.E));

        return (int) ((int)upperNo/downNo);
    }

    /**
     * Initialize the bloom vector with false and create it
     * @param size
     */
    private Boolean[] initializeTheBloomFilter(int size, Boolean[]
        bloomFilter)
    {

        for(int i=0; i<size; ++i)
        {
            bloomFilter[i]=new Boolean(false);
        }

        return bloomFilter;
    }


    /**
     * Convert the Vector to String
     *
     * @return the content of the bloom vector in string type
     */
    private String convertBloomToString(Boolean[] bloom)
    {
        String reStr="";
        for(int i=0; i<bloom.length; ++i)
        {
            if(bloom[i]==false)
            {
                reStr+=0;
            }
            else
            {
                reStr+=1;
            }
        }

        return reStr;
    }

    /**
     * Remove duplicates from an array list
```

```
         *
         * @param arrayList
         * @return the cleaned arrayList
         */
        private ArrayList<String> removeDupFromArrayList(ArrayList<String>
            arrayList)
        {
            HashSet<String> hs = new HashSet<String>();
            hs.addAll(arrayList);
            arrayList.clear();
            arrayList.addAll(hs);
            return arrayList;

        }
    }
```

```
package approaches;

import java.io.IOException;
import java.net.Socket;

import Other.ApproachesThread;
import Other.Information;
import Other.ParseFile;
import Positioning.RadioMapMean;

public class BloomFilter extends Approach
{

    private String radioMap = null;


    public BloomFilter(int port, Socket clientSocket, String radioMap)
    {
        super(port, clientSocket);
        this.radioMap=radioMap;
    }

    public void run()
    {

        //Create Connection with client with port
        ApproachesThread decthread = null;
        try
        {
            decthread = new ApproachesThread(clientSocket);
        } catch (IOException e)
        {
            e.printStackTrace();
        }
        try
        {
            //Read the whole radio-map
            ParseFile radioMap = new ParseFile(this.radioMap);

            //RadioMapMean rmm = new RadioMapMean();
            //rmm.createRadioMap(radioMap.getDataArray());
            Bloom bl = new Bloom(radioMap.getDataArray(), 10, 4);
            decthread.selectAndSendRadioMap(radioMap,bl);
            Information.showData();
        } catch (IOException e)
        {
            e.printStackTrace();
        }
    }

}
```

Simulator Server code


```
package approaches;

import java.io.IOException;
import java.net.Socket;
import java.util.ArrayList;

import javax.sound.midi.MidiDevice.Info;

import org.omg.Dynamic.Parameter;

import Other.ApproachesThread;
import Other.Information;
import Other.Parameters;
import Other.ParseFile;
import Other.WriteRssLog;
import Positioning.RadioMapMean;

public class Centralized extends Approach
{

    public Centralized(int p1, Socket clientSocket)
    {
        super(p1,clientSocket);
    }

    @Override
    public void run()
    {
        RadioMapMean rmm = new RadioMapMean();
        System.out.println("ID: "+id);
        //Create Connection with client with port
        ApproachesThread decthread = null;
        try
        {
            decthread = new ApproachesThread(clientSocket);
        } catch (IOException e)
        {
            e.printStackTrace();
        }
        try
        {
            //Read the whole radio-map
            ParseFile radioMap = new ParseFile("radio-map.txt");


            rmm.createRadioMap(radioMap.getDataArray());

            decthread.incomingDataFromClient(rmm);
        } catch (IOException e)
        {
            e.printStackTrace();
        }

        //Write arrayList to file named test-data.txt
        WriteRssLog wrl = null;
        try
        {
            wrl = new WriteRssLog("test-data.txt");
        } catch (IOException e)
```



```
        {
            e.printStackTrace();
        }
        try
        {
        //  wrl.writeArrayListToFile(decthread.getSavedFile());
            decthread.closeConnections();
        } catch (IOException e)
        {
            e.printStackTrace();
        }

        //Time to position in server
        Parameters pr = null;
        try
        {
            pr = new Parameters();
        } catch (IOException e)
        {
            e.printStackTrace();
        }
        System.out.println("Time to position in Centralized: ");
        //Information.timeToPosition=decthread.getSavedFile().size()*pr.
            serverAlgExecution*rmm.getLocationRSS_HashMap().size();
        Information.showData();

    }

    /**
     * Method that convert the incoming arrayList to certain format.
     * This is used by other function to build the RadioMapMean data
         structure
     *
     * @param the arrayList we want to format
     * @return the final format ArrayList
     */

    private ArrayList<String> formatIncomingFile(ArrayList<String> file)
    {
        System.out.println("IncArray: "+file);
        String[] temp = file.get(0).split(", ");
        ArrayList<String> macAddresses = new ArrayList<String>();
        ArrayList<String> geolocations = new ArrayList<String>();
        ArrayList<String> rss = new ArrayList<String>();
        String geo = null;

        for(int i=1; i<temp.length; ++i)
        {
            if((i-1)%4 == 0)
            {
                geo=temp[i]+" ";
            }
            else if(i%4==0)
            {
                rss.add(temp[i]);
            }
            else if((i+1)%4==0)
            {
                macAddresses.add(temp[i]);
```

```
                }
                else if((i+2)%4 == 0)
                {
                    geo+=temp[i];
                    geolocations.add(geo);
                    geo=null;
                }
            }
            //Put in one line all macAddresses
            String macAddresses1="";
            for(int i=0; i<macAddresses.size(); ++i)
            {
                macAddresses1+=macAddresses.get(i);
            }
            //Put in other lines (in each line) geo+rss

            return null;
        }

    }
```

```
package Other;

public class CountTime
{
    private long start;
    private long elapsedTimeMillis;
    public CountTime()
    {
        start = System.currentTimeMillis();
    }

    public void endCount()
    {
        elapsedTimeMillis = System.currentTimeMillis()-start;
    }

    public float getSeconds()
    {
        return elapsedTimeMillis/1000F;
    }
}
```

```
package approaches;

import java.io.IOException;
import java.net.Socket;

import Other.ApproachesThread;
import Other.ParseFile;


public class Decentralized extends Approach implements Runnable
{

    private String radioMapStr;

    public Decentralized(int p1, Socket clientSocket, String radioMapStr)
    {
        super(p1, clientSocket);
        this.radioMapStr=radioMapStr;
    }

    @Override
    public void run()
    {
        System.out.println("ID: "+id);

        //Take radio-map and upload it to a simple structure
        ParseFile radioMap = null;
        try
        {
            radioMap = new ParseFile(this.radioMapStr);
        } catch (IOException e1)
        {
            e1.printStackTrace();
        }

        try
        {
        //Send back to client the constructed radio-map.txt
        ApproachesThread decthread = new ApproachesThread(clientSocket);

        //upload the radio-map
        decthread.setSendArrayList(radioMap.getDataArray());

        //Send the radio-map
        decthread.outgoingData();

        decthread.closeConnections();
        } catch (IOException e)
        {
            e.printStackTrace();
        }

        System.out.println("END");
    }

}
```

```
package Other;

public class Information
{
    public static int messages;
    public static float timeToPosition;
    public static float timeWholeProcedure;
    public static float bytesSendReceive;

    /* Is used only by bloom approach */
    public static float sumOfSizeOfRadioMapSmall;

    /**
     * Method to show the current information
     */
    public static void showData()
    {
        System.out.println("Messages: "+messages+" Whole time: "+
            timeWholeProcedure+"s Bytes: "+bytesSendReceive+" Time to
            position: "+timeToPosition);
    }

}
```

```
package Positioning;

public class LocDistance
{
    private double distance;
    private String location;

    public LocDistance(double distance, String location)
    {
        this.distance = distance;
        this.location = location;
    }

    public double getDistance()
    {
        return distance;
    }

    public String getLocation()
    {
        return location;
    }
}
```

```
package Positioning;
public class LogRecord
{

    private String bssid;
    private float rss;

    public LogRecord(String bssid, float rss)
    {
        super();
        this.bssid = bssid;
        this.rss = rss;
    }

    public String getBssid()
    {
        return bssid;
    }

    public float getRss()
    {
        return rss;
    }

    public String toString()
    {
        String str = new String();
        str = String.valueOf(bssid) + " " + String.valueOf(rss) + "\n";
        return str;
    }

}
```

```
package Other;
import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;


public class Parameters
{
    public static String port1;
    public static String port2;
    public static String choice;
    public static String radioMap;
    public static float serverAlgExecution;

    public Parameters() throws IOException
    {
        ArrayList<String> arrayList = new ArrayList<String>();
        FileInputStream fstream = null;
        try
        {
            fstream = new FileInputStream("parameters.txt");
        } catch (FileNotFoundException e)
        {
            e.printStackTrace();
        }
        // Get the object of DataInputStream
        DataInputStream in = new DataInputStream(fstream);
        BufferedReader br = new BufferedReader(new InputStreamReader(in));
        String strLine;
        // Read File Line By Line
        while ((strLine = br.readLine()) != null)
        {
            // Print the content on the console
            String[] tempArray = strLine.split(" ");
            arrayList.add(tempArray[1]);
        }
        in.close();
        port1=arrayList.get(0);
        port2=arrayList.get(1);
        choice=arrayList.get(2);
        radioMap=arrayList.get(3);
        serverAlgExecution=Float.valueOf(arrayList.get(4));
    }
}
```

```java
package Other;
import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;

public class ParseFile
{
    private ArrayList<String> dataArray = new ArrayList<String>();

    public ParseFile(String nameOfTheFile) throws IOException
    {
        FileInputStream fstream = null;
        try
        {
            fstream = new FileInputStream(nameOfTheFile);
        } catch (FileNotFoundException e)
        {
            e.printStackTrace();
        }
        // Get the object of DataInputStream
        DataInputStream in = new DataInputStream(fstream);
        BufferedReader br = new BufferedReader(new InputStreamReader(in));
        String strLine;
        // Read File Line By Line
        while ((strLine = br.readLine()) != null)
        {
            dataArray.add(strLine);
        }
        in.close();
    }

    public ArrayList<String> getDataArray()
    {
        return dataArray;
    }
}
```

```
package Positioning;
import java.util.ArrayList;
import java.util.HashMap;


public class RadioMapMean
{
    private ArrayList<String> MacAdressList = null; //Whole macAddresses
    private HashMap<String, ArrayList<String>> LocationRSS_HashMap = null;

    public RadioMapMean()
    {
        MacAdressList = new ArrayList<String>();
        LocationRSS_HashMap = new HashMap<String, ArrayList<String>>();
    }

    /**
     * Construct the radiomap
     *
     * @param incomingArrayList
     */
    public void createRadioMap(ArrayList<String> incomingArrayList)
    {
        //Take the first line and put it in MacAddressList
        String macAddresses = new String(incomingArrayList.get(0));
        macAddresses = macAddresses.replace(",", "");
        String [] temp = macAddresses.split(" ");

        for(int i=3; i<temp.length; ++i)
        {
            MacAdressList.add(temp[i]);
        }

        //Take the rest data
        for(int i=1; i<incomingArrayList.size(); ++i)
        {
            String str = incomingArrayList.get(i);
            str = str.replace(",", "");
            temp = str.split(" ");
            String key = temp[0]+" "+temp[1];
            ArrayList<String> rssValues = new ArrayList<String>();
            for(int j=2; j<temp.length; ++j)
            {
                rssValues.add(temp[j]);
            }

            LocationRSS_HashMap.put(key, rssValues);
        }
    }

    public HashMap<String, ArrayList<String>> getLocationRSS_HashMap()
    {
        return LocationRSS_HashMap;
    }

    public ArrayList<String> getMacAdressList()
    {
        return MacAdressList;
    }
```



```
}
```

Simulator Server code


```
package Other;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

import approaches.BloomFilter;
import approaches.Centralized;
import approaches.Decentralized;

public class Server
{
    private Socket clientSocket = null;
    private ServerSocket serverSocket = null;
    private PrintWriter out = null;
    private BufferedReader in = null;
    private ArrayList<String> savedFile = null;
    private ArrayList<String> sendArrayList = null;

    private final int maxNoOfThreads = 5;

    /**
     * Open the connection with client
     *
     * @throws IOException
     */
    public Server(int port, int approach, String radioMapName) throws
        IOException
    {
        ExecutorService execSvc = Executors.newFixedThreadPool
            (maxNoOfThreads);

        try
        {
            serverSocket = new ServerSocket(port);

        } catch (IOException e)
        {
            System.err.println("Could not listen on port.");
            System.exit(1);
        }

        while (true)
        {
            try
            {
                clientSocket = serverSocket.accept();
                switch (approach)
                {
                case 1:
                    execSvc.execute(new Decentralized(port, clientSocket,
                        radioMapName));
```



```
                    break;
                case 2:
                    execSvc.execute(new Centralized(port, clientSocket));
                    break;
                case 3:
                    // Bloom algorithm
                    execSvc.execute(new BloomFilter(port, clientSocket,
                        radioMapName));
                    break;
                default:
                    break;
                }

            } catch (IOException e)
            {
                System.err.println("Accept failed.");
                System.exit(1);
            }
        }
    }

    public Server(int port) throws IOException
    {
        serverSocket = new ServerSocket(port);
        clientSocket = serverSocket.accept();

    }

    public Socket getSecondSocket()
    {
        return clientSocket;
    }

    /**
     * If server wants to send a message to presure client to wait because
         of
     * processing of the server
     *
     * @param message
     */
    public void sendToClientAMessage(String message)
    {
        out.println(message);
    }

    /**
     * Incoming data from client. Here Server receives the rss-log.txt
     *
     * @throws IOException
     */
    public void incomingDataFromClient() throws IOException
    {
        String inputLine, outputLine;
        outputLine = "+OK READY";
        out.println(outputLine);

        while ((inputLine = in.readLine()) != null)
        {
            outputLine = "taken";
```

Simulator Server code


```
                out.println(outputLine);
                if (inputLine.equals("END"))
                    break;
                savedFile.add(inputLine);
            }
        }

        /**
         * Server is sending the final radio-map.txt to client
         *
         * @throws IOException
         */
        private void outgoingData() throws IOException
        {
            String inputLine;
            int index = 0;
            out.println("INCOMING RADIO-MAP");
            inputLine = in.readLine();
            if (inputLine.equals("+OK READY TO RECEIVE RADIO-MAP")) // Message
                from
            // client
            {

                while ((inputLine = in.readLine()) != null || (index !=
                    sendArrayList.size() - 1))
                {
                    out.println(sendArrayList.get(index));
                    if (index == sendArrayList.size() - 1)
                    {
                        break;
                    }
                    ++index;
                }
                out.println("END");
            }
        }

        /**
         * Just process the radio-map to send it to the client
         *
         * @param LocationRSS_HashMap
         *            Contains the geolocation and the respectively rss values
         *
         * @param macAddresses
         *            All the mac addresses that took from the radio-map
         *
         * @throws IOException
         */

        public void sendRadioMapToClient(HashMap<String, ArrayList<String>>
            LocationRSS_HashMap, HashSet<String> macAddresses) throws
            IOException
        {
            sendArrayList = new ArrayList<String>();
            String MacAddressesStr = "# Longitude, Latitude";
            for (String str : macAddresses)
            {
                MacAddressesStr += ", " + str;
            }
```



```
            sendArrayList.add(MacAddressesStr);
            MacAddressesStr = null;

            for (String key : LocationRSS_HashMap.keySet())
            {
                String tempStr = null;
                String[] temp = key.split(" ");
                tempStr = temp[0] + ", " + temp[1];

                ArrayList<String> rssValues = LocationRSS_HashMap.get(key);

                for (String str : rssValues)
                {
                    tempStr += ", " + str;
                }

                sendArrayList.add(tempStr);
            }
            // Method that sends the radio-map to client
            outgoingData();
        }

        /**
         *
         * @return the savedFile arraylist
         */

        public ArrayList<String> getSavedFile()
        {
            return savedFile;
        }

        /**
         * Close connections for second Constructor with one port
         *
         * @throws IOException
         */
        public void closeConnectionsSecond() throws IOException
        {
            clientSocket.close();
            serverSocket.close();
        }

        /**
         * Close all connections. For the main Constructor(port1,port2)
         *
         * @throws IOException
         */
        public void closeConnections() throws IOException
        {
            out.close();
            in.close();
            clientSocket.close();
        }
    }
```

Simulator Server code


```
package Other;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class StartServer
{
    private final static int maxNoOfThreads = 5;

    /**
     * @param args
     * @throws IOException
     */
    public static void main(String[] args) throws IOException
    {

        Parameters pr = new Parameters();

        if (pr.choice.equals("DEC"))
        {
            Server srv = new Server(Integer.valueOf(pr.port1), 1, pr.
                radioMap);

        } else if (pr.choice.equals("CEN"))
        {
            Server srv = new Server(Integer.valueOf(pr.port1), 2, "");

        } else if (pr.choice.equals("BLOOM"))
        {
            System.out.println("Bloom is starting");
            Server srv = new Server(Integer.valueOf(pr.port1), 3, pr.
                radioMap);
        } else
        {
            // Wrong choice
            System.exit(-1);
        }

    }

    public static String getString()
    {
        String userInput = null;
        try
        {
            BufferedReader in = new BufferedReader(new InputStreamReader
                (System.in));
            userInput = in.readLine();
        } catch (IOException e)
        {
            System.out.println("IOException has been caught");
        }
        return userInput;
    }
}
```

Simulator Server code


```
package Other;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.Writer;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;


public class WriteRadioMap
{
    private Writer output = null;
    private File file = null;
    public WriteRadioMap(String fileName) throws IOException
    {
        file = new File(fileName);
        output= new BufferedWriter(new FileWriter(file));
    }

    public void writeArrayListToFile(HashMap<String, ArrayList<String>>
        incomingMap, HashSet<String> MacAddressListDist) throws
        IOException
    {
        output.write("# Longitude, Latitude, ");

        //Write the mac address
        int count=0;
        for(String str : MacAddressListDist)
        {
            if(count == MacAddressListDist.size()-1)
            {
                output.write(str+"\n");
            }
            else
            {
                output.write(str+", ");
            }
            ++count;
        }

        //Write the location and rss values
        for(String str : incomingMap.keySet())
        {
            ArrayList<String> rssValues = incomingMap.get(str);

            //Separate the key to two geolocation
            String[] temp = str.split(" ");
            output.write(temp[0]+", "+temp[1]);
            for(String rss : rssValues)
            {
                output.write(", "+rss);
            }
            output.write("\n");
        }
        output.close();
    }
}
```


WriteRadioMap.java 28/5/12 3:10 μ.μ.