Ατομική Διπλωματική Εργασία

ΤΟ ΠΡΩΤΟΚΟΛΛΟ SMARTTRACE ΓΙΑ ΚΑΤΑΝΕΜΗΜΕΝΗ ΑΝΑΖΗΤΗΣΗ ΧΩΡΟ – ΧΡΟΝΙΚΩΝ ΤΡΟΧΙΩΝ

Κωνσταντίνος Κώστα

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2011

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Το Πρωτόκολλο SmartTrace για Κατανεμημένη Αναζήτηση Χώρο – Χρονικών Τροχιών

Κωνσταντίνος Κώστα

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 2011

Ευχαριστίες

Θα ήθελα αρχικά να ευχαριστήσω τον κύριο Δημήτρη Ζεϊναλιπούρ, ο οποίος μου προσέφερε αυτή την πολύ ενδιαφέρουσα και χρήσιμη στην κοινωνία, εργασία, καθώς και για την πλήρη κατανόηση και υποστήριξη που είχα από αυτόν, κατά τη διάρκεια της διπλωματικής μου εργασίας. Πρόκειται για ένα σπουδαίο καθηγητή και ερευνητή με πολλές γνώσεις σε όλους του τομείς της επιστήμης της Πληροφορικής, αλλά πάνω απ' όλα για ένα εξαίρετο άνθρωπο.

Θα ήθελα επίσης να ευχαριστήσω το Τμήμα της Πληροφορικής του Πανεπιστημίου Κύπρου για την αγορά και διάθεση όλων των συσκευών που ήταν απαραίτητα για την ολοκλήρωση της παρούσας διπλωματικής εργασίας.

Τέλος οφείλω να πω ένα μεγάλο ευχαριστώ στους φίλους μου εντός και εκτός Πανεπιστημίου οι οποίοι μου συμπαραστάθηκαν όλο αυτό τον καιρό αλλά και ένα τεράστιο ευχαριστώ στην οικογένειά μου και ιδιαίτερα στους γονείς μου Ερμογένη και Δέσποινα για την ηθική και οικονομική υποστήριξη που μου πρόσφεραν όχι μόνο κατά τη διάρκεια εκπόνησης της παρούσας διπλωματικής εργασίας αλλά και καθ' όλη τη διάρκεια των σπουδών μου.

Αναγνώριση

Επιδείξεις του Πρωτόκολλου SmartTrace και της διπλωματικής αυτής εργασιας έχουν παρουσιαστεί στα ακόλουθα συνέδρια:

- “SmartTrace: Finding Similar Trajectories in Smartphone Networks without Disclosing the Traces”, C. Costas, C. Laoudias, D. Zeinalipour-Yazti, D. Gunopulos, Demo at the 27th IEEE International Conference on Data Engineering (ICDE'11), ISBN 978-1-4244-8958-9, pp. 1288-1291, IEEE Press, April 11-16, Hannover, Germany, 2011, [Acceptance Rate: 31% (21/67)]

- “A Distributed Spatio-Temporal Similarity Search Framework using Android-based Smartphones (demo)”, C. Costa, C. Christoforou, G. Nicolaides, D. Papadiomidous, C. Laoudias, D. Zeinalipour-Yazti, The 9th Hellenic Data Management Symposium (HDMS 2010), Ayia Napa, Cyprus, June 30th - July 3rd, 2010.

Περίληψη

Αυτή η διπλωματική εργασία έχει εκπονηθεί στα πλαίσια του προπτυχιακού προγράμματος Πληροφορικής στο Πανεπιστήμιο Κύπρου. Στην παρούσα διπλωματική εργασία παρουσιάζεται ένα καινοτόμο πλαίσιο αναζήτησης ομοιότητας σε χώρο-χρονικά δεδομένα, χωρίς να αποκαλύπτεται κανένα στοιχείο για τους συμμετάσχοντες χρήστες. Το πλαίσιο που έχουμε υλοποιήσει, το οποίο ονομάζεται SmartTrace, εκμεταλλεύεται την ευκαιρία και την συμμέτοχη των χρηστών για να απαντηθούν άμεσα οι επερωτήσεις της μορφής: "*Βρες τα αντικείμενα τα οποία ακλουθούν μια κίνηση όμοια με ένα αντικείμενο Q, οπού Q είναι κάποια τροχιά εισόδου*". Το SmartTrace στηρίζεται στο αποκλειστικά μοντέλο αποθήκευσης δεδομένων, οπού χωρικά δεδομένα αποθηκεύονται τοπικά στην έξυπνη κινητή συσκευή για επίδοση και απόκρυψη πληροφοριών. Το SmartTrace μετά αναπτύσσει έναν αποδοτικό top-k αλγόριθμο επερώτησης, ο οποίος εκμεταλλεύεται τα κατανεμημένα μετρά ομοιότητας τροχείων, ανθεκτικό στον χωρικό και χρονικό θόρυβο, καταφέρνοντας να πάρουμε την πιο σχετική απάντηση με το Q γρήγορα και αποδοτικά.

Στην εν λόγω διπλωματική εργασία έχουμε την δυνατότητα επερώτησης με αποστολή τροχιών από συντεταγμένες (GPS coordinates) αλλά και από ασύρματα σημεία πρόσβασης συνδυασμένα με την ισχύ του σήματος (Wi-Fi access point with received signal strength). Επίσης έχουν υλοποιηθεί δυο τρόποι επερώτησης αντίστοιχα και για τις δυο πτυχές :

i. Αλληλεπιδραστικό μοντέλο επερώτησης, οπού οι συσκευές χειρίζονται από του χρήστες για να αναγνωρίσουν ποιος κινείται πιο όμοια με τον κόμβο που έχει κάνει την επερώτηση.
ii. Μοντέλο επερώτησης βάση αρχείου, οπού μπορεί να γίνει επερώτηση μεγάλης κλίμακας μέσω των αποθηκευμένων αρχείων με σκοπό την εύρεση των K πιο κοινών τροχιών γρήγορα και αποδοτικά.

Η όλη υλοποίηση έγινε πάνω σε συσκευές Android. Συγκεκριμένα πάνω σε HTC desire, HTC Wildfire και HTC Hero. Το Android SDK χρησιμοποιήθηκε για την υλοποίηση των αλγορίθμων και η αντικειμενοστραφείς γλωσσά προγραμματισμού Java για την υλοποίηση των εξυπηρετητών.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Υποκίνηση Εργασιας

Οι έξυπνες κινητές συσκευές έχουν εξαπλωθεί αρκετά και διαθέτουν χαρακτηριστικά που μπορούν να βοηθήσουν τον γεωγραφικό εντοπισμό καθώς και άλλα, συμπεριλαμβανόμενου της δυνατότητας σύνδεσης με το διαδίκτυο μέσω WLAN, WCDMA/UMTS(3G), HSPA(3.5G) και LTE/WiMAX(4G) δικτύων που έχουν φέρει επανάσταση στο χώρο των χώρο-χρονικών εφαρμογών και υπηρεσιών στις έξυπνες κινητές συσκευές .Το IMS research και η Comscore αναφέρουν ότι πάνω από 225 χιλιάδες πώλησης σε έξυπνες κινητές συσκευές έγιναν κατά τον Φεβρουάριο του 2010. Σύμφωνα με την έκθεση από την έρευνα της ομάδας της IDC που κυκλοφόρησε στις 7 του Φεβρουαρίου , τα έξυπνα τηλέφωνα έχουν υπερνικήσει τους προσωπικούς υπολογιστές, για πρώτη φορά. Σε παγκόσμιο επίπεδο, έχουν αποσταλεί 100,9 εκατομμύρια έξυπνα τηλέφωνα κατά τους τρεις τελευταίους μήνες του 2010, με 87 τοις εκατό πρόοδο από το προηγούμενο έτος. Ενώ για τα PCs βελτιώθηκε ελαφρά μόλις 3 τοις εκατό σε 92,1 εκατομμύρια.

Η έρευνα για την επεξεργασία χώρο-χρονικών έχει αρχίσει να αναπτύσσεται και να κινά το ενδιαφέρον για έρευνες και ανάπτυξη εφαρμογών και για διαφημιστικούς σκοπούς. Είναι ένα νέο σχετικά θέμα που βρίσκεται στον στόχο μεγάλων συγκροτημάτων που ασχολούνται με την διαχείριση δεδομένων καθώς και την εξέλιξη βάσεων δεδομένων.

1.2 Υπόβαθρο και σχετική δουλειά

Υπάρχουν πολλές καινοτόμες εφαρμογές που μπορούμε να βρούμε σε δίκτυο έξυπνων τηλεφώνων. Ένα παράδειγμα είναι η καιροσκοπική και η συμμετοχική αίσθηση [1], [2], [3], [5] οπού εφαρμογές μπορούν να ενεργούν ως κινητοί κόμβοι σε μια δεδομένη περιοχή για να παρέχουν πληροφορίες για την γειτνίαση χρησιμοποιώντας τις ικανότητες των αισθήσεων. Ακόμα ένα παράδειγμα είναι ο υπολογισμός της καθυστέρησης λόγω των κυκλοφοριακής κίνησης [5] χρησιμοποιώντας Wi-Fi σήματα που συλλέχτηκαν από έξυπνες κινητές συσκευές έναντι της υψηλής σε κόστος χρησιμοποίηση του GPS. Στην κοινωνική σελίδα Google Latitude ενεργοποιεί χρήστες για να εντοπίζει αυτούς και τα κοινωνικά δίχτυα που έχουν επισκεφτεί. Η Δεδομένη υπηρεσία ανέφερε πάνω από 3 εκατομμύρια εγγεγραμμένους χρήστες και πάνω από 1 εκατομμύριο ενεργούς χρήστες., πάρα την αμφιλεγόμενη ανησυχία για τα ιδιωτικά δεδομένα. Παρομοίως υπάρχουν πολλές εφαρμογές όπως το Foursquare, Gowalla και Loopt που έχουν την ανάλογη επιτυχία στον κόσμο των έξυπνων τηλεφώνων. Επίσης η ακαδημαϊκή έρευνα και πρόοδος σε αυτό τον τομέα βρίσκεται σε εξέλιξη.

Μπορούμε να ορίσουμε ότι ένα δίκτυο έξυπνων τηλεφώνων σαν "ένα σύνολο από έξυπνα τηλεφωνά που επικοινωνούν με ένα διακριτικό τρόπο, χωρίς ο χρήστης να αλληλοεπιδρά ρητά , με σκοπό να πραγματοποιηθεί μια συνεργασία ή κοινωνική εργασία".

Το πλαίσιο SmartTrace που προσφέρει την κατανεμημένη σύγκριση τροχείων και μπορεί να χρησιμοποιηθεί ως ένα ενδιάμεσο λογισμικό που προσφέρει υπηρεσίες σε ένα έξυπνο τηλέφωνο ούτος ώστε να μπορεί να απαντήσει επερωτήσεις της μορφής : *"Βρες τα αντικείμενα τα οποία ακολουθούν μια κίνηση όμοια με ένα αντικείμενο Q, όπου Q είναι κάποια τροχιά εισόδου."*. Υπάρχει αφθονία από εφαρμογές που μπορούν να

ενσωματώσουν το πλαίσιο αυτό όπως σε συστήματα έξυπνης μεταφοράς [4], [5], εφαρμογές κοινωνικής δικτύωσης [6], [7], συστήματα παρακολούθησης κάτοικων [8] και πολλές άλλες.

Ένα παράδειγμα επερωτήσεις που θα μπορούσε να απαντηθεί είναι *" Βρες αν υπάρχει ποδηλατοδρόμος από το Metropolitan Museum of Art στο Μανχάταν μέσω του central park στο Julliard School "* ή *" Βρες ποιες ζέβρες κινούνται πιο κοντές στην ζέβρα που λέγεται Abby πριν να τραυματιστεί"* [8]. Υπάρχουν ήδη υπηρεσίες για κεντρικοποιημένη αναζήτηση τροχείων όπως το GeoLife, GPS-Waypoints, ShareMyRoutes , και οι αντίστοιχες ακαδημαϊκές, που μπορούν να διεκπεραιώσουν αυτούς του τύπου επερωτήσεις. Όμως οι υπηρεσίες αυτές αποθηκεύουν τις τροχιές των χρηστών σε μια κεντρικοποιημένη ή σε μια cloud-based υποδομή.

Στο υποκεφάλαιο αυτό παρέχουμε σχετικές ερευνητικές εργασίες και για τα δύο, χωροχρονική επεξεργασία των ερωτημάτων και επεξεργασία για κατανεμημένα ερώτημα top-K, οι οποίες βρίσκονται στα θεμέλια του πλαισίου SmartTrace. Οι χωρο-χρονικές επερωτήσεις έχουν γίνει ένα κρίσιμο σημείο για έρευνα με την πάροδο των χρόνων [13], με την ανάπτυξη των αποτελεσματικών μεθόδων πρόσβασης [14],[15],[16],[17] και των μετρικών ομοιότητας, όπως Dynamic Time Warping (DTW) [18], Longest Common Subsequence (LCSS) [19], παραλλαγές των κανόνων Lp όπως Edit Distance with Real Penalty (ERP) [24] και Edit Distance on Real Sequences (EDR) [20]. Αυτές οι μετρήσεις έχουν προταθεί για την έξυπνη [21], ιστορική [15] και για πολύπλοκες χώρο-χρονικές επερωτήσεις [22]. Όλες αυτές οι τεχνικές, καθώς και τα πλαίσια για χωροχρονικά ερωτήματα [23], [24], [26], δουλεύουν σε ένα εντελώς κεντρικοποιημένο περιβάλλον . Το ίδιο ισχύει για τις online υπηρεσίες αναζήτησης τροχιών, όπως GeoLife, GPS-Waypoints, Sharemyroutes και τους ακαδημαϊκούς οργανισμούς [27], η οποία υποθέτουμε ότι τροχιές του χρήστη συναθροίζονται και να αποθηκεύονται σε κεντρικοποιημένη ή cloud-based υποδομή. Είναι σημαντικό να σημειώσουμε ότι για μια κεντρικοποιημένη ρύθμιση, ο ορισμός του προβλήματος είναι σημαντικά διαφορετικός, από τις αποκεντρωμένο σενάριο που θεωρούμε σε αυτό το έργο, όπως ο επεξεργαστής, που κάνει την επερώτηση, διατηρεί όλες τις τροχιές τοπικά και τα στατιστικά σε τοπικούς καταλόγους. Επιπλέον, σε μια κεντρικοποιημένη ρύθμιση του επεξεργαστή που κάνει την επερώτηση μπορεί να

χρησιμοποιήσει χωρικό ή χωροχρονικό δείκτη , όπως τα δέντρα τύπου R(π.χ., χρησιμοποιούνται σε [27] και [28]), STR-δένδρα ή TB-δένδρα [29], προκειμένου να επιταχυνθεί η ανάκτηση της απάντησης. Από την άλλη πλευρά, σε μια αποκεντρωμένη ρύθμιση κανένα από αυτά δεν έχει επιπλέον κόστος.

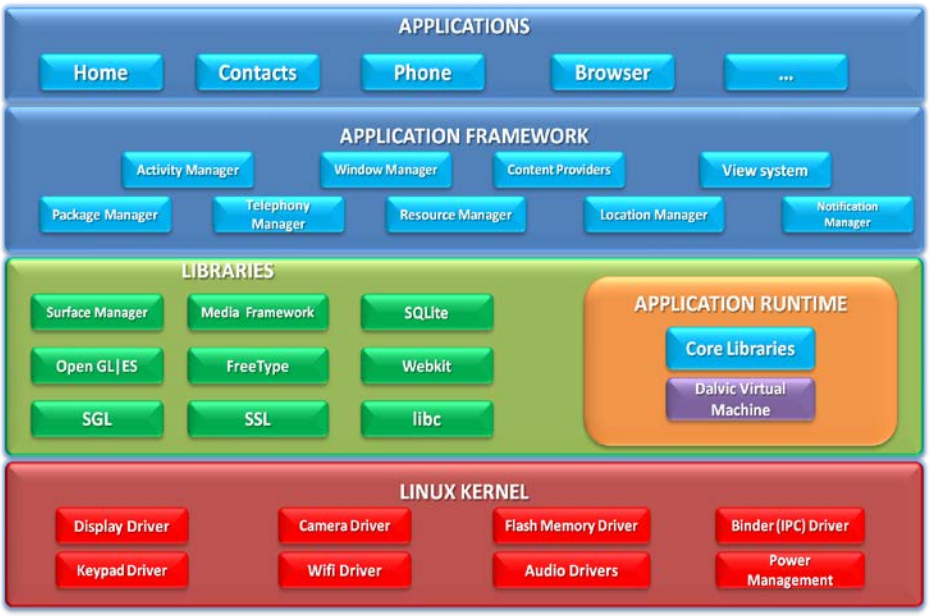
Σε προηγούμενη δουλειά [31], έχουν προετοιμάσει το έδαφος προς τις τεχνικές επεξεργασίας με ένα κατανεμημένο τρόπο (δηλαδή, χωρίς διήθηση κάθε γεωγραφικού σημείου του χρήστη σε μια κεντρική αρχή). Ωστόσο, και τα δυο ήταν τόσο απολυτά σε όσον αφορά την ενέργεια και τους χρονικούς περιορισμούς που προκύπτουν σε ένα δίκτυο έξυπνων τηλεφώνων αλλά και σε σχέση με θέματα που αφορούν την αποκάλυψη σημείων της τροχιάς (π.χ., υπέθεσαν ότι ο επεξεργαστή που κάνει την επερώτηση, μπορεί αυθαίρετα έχει πρόσβαση στις κατανεμημένες τροχιές). Το πιο σημαντικό, είναι ότι σε αυτή την προηγούμενη δουλειά υποτίθεται ότι τροχιές όπου κάθετα κατακερματισμένη n κατανεμημένες τοποθεσίες (δηλαδή, κάθε κατανεμημένη τοποθεσία κατέχει υποακολουθίες ενός ή περισσοτέρων τροχιών), ενώ αυτή η εργασία εστιάζεται στην οριζόντια κατακερματισμένη περίπτωση (δηλαδή, κάθε έξυπνο τηλέφωνο διαθέτει την πλήρη τροχιά σε τοπικό επίπεδο.)

Τα top-K ερωτήματα έχουν μελετηθεί σε διάφορα πλαίσια, συμπεριλαμβανομένων συστήματα ενδιάμεσου λογισμικού [32], βάσεις δεδομένων με πρόσβαση από το web [33] και επεξεργαστές ρευμάτων [34]. Μια εξαιρετική έρευνα για περιβάλλοντα σχεσιακών βάσεων δεδομένων εμφανίζεται στο [35]. Έχει αποδειχθεί σε πολυάριθμες μελέτες [36], [33], [37], ότι η επεξεργασία των top-K ερωτημάτων έχει νόημα μόνο εάν το κατηγόρημα K αναφέρεται σε ένα μικρό υποσύνολο της πλήρους απάντηση (π.χ., ως 1%). Για μεγαλύτερες τιμές του K, ο βελτιστοποίησης του επερωτήματος μπορεί να επιλέξει να ανακτήσει την πλήρη απάντηση. Το κύμα της κεντρικοποιημένης από αλγόριθμους επεξεργασίας top-K ερωτημάτων δεν ήταν επιτυχημένο λόγω των κατανεμημένων, δηλαδή ο TPUT [36] αλγόριθμος και ο TJA [37] αλγόριθμος. Σε όλα αυτά τα σενάρια, τα ερωτήματα αναφέρονται σε ακριβές σκορ, ενώ εμείς έχουμε επικεντρωθεί σε άνω όριο αποτελέσματα που εμφανίζονται στον πίνακα με τα σκορ.
Ενώ οι αλγόριθμοι για άνω όριο έχουν μελετηθεί στο [38], αυτοί είχαν τελείως διαφορετικές υποθέσεις και στηρίχθηκε σε ένα κεντρικοποιημένο μοντέλο υπολογισμού.

Το πρόβλημα της κατανεμημένης επεξεργασίας του top-K ερωτήματος με πιθανοτικές εγγυήσεις, και όχι ακριβές τιμές, μελετήθηκε σε KLEE [39] ακόμη οι απαντήσεις ήταν και πάλι κατά προσέγγιση. Το πρόβλημα της συνεχούς παροχής top-K προσεγγιστικών απαντήσεων σε ένα περιβάλλον πελάτη-εξυπηρετητή μελετήθηκε επίσης στο [34]. Σε όλες τις περιπτώσεις τα αποτελέσματα είναι κατά προσέγγιση και συνεχής πάνω από ένα απλό χαρακτηριστικό, Έτσι λειτουργούν πάνω από ατομικά χαρακτηριστικά (στήλες), ενώ η προσέγγιση μας είναι ακριβής και λειτουργεί οριζόντια πάνω από τις γραμμές.

1.3 Αρχιτεκτονική Android

Android είναι μια στοίβα λογισμικού για κινητές συσκευές που περιλαμβάνει ένα λειτουργικό σύστημα, ενδιάμεσο λογισμικό και τις βασικές εφαρμογές. Το Android SDK παρέχει τα εργαλεία και τα APIs για να αρχίσουν να αναπτύσσονται εφαρμογές για την πλατφόρμα Android χρησιμοποιώντας τη γλώσσα προγραμματισμού Java.



Σχήμα 1.1 : Αρχιτεκτονική Android

Android αποτελείται από πολλά αναγκαία και ανεξάρτητα μέρη συμπεριλαμβανομένων των ακόλουθων [11]:

- Το σχέδιο ή υπόδειγμα αναφοράς υλικού που περιγράφει τις ικανότητες που απαιτούνται από μια κινητή συσκευή, ώστε να υποστηρίξει το λογισμικό στοίβα
- Ένας Linux πυρήνας του λειτουργικού συστήματος που παρέχει τη διασύνδεση χαμηλού επιπέδου με το υλικό, τη διαχείριση μνήμης, και ελέγχου διεργασιών, όλα βελτιστοποιημένα για φορητές συσκευές
- Βιβλιοθήκες ανοιχτού κώδικα για την ανάπτυξη εφαρμογών, συμπεριλαμβανομένων SQLite, WebKit, OpenGL, και Media Manager
- Ο χρόνος εκτέλεσης είναι ο χρόνος που χρησιμοποιείται για να εκτελέσει και να φιλοξενήσει τις Android εφαρμογές, συμπεριλαμβανομένης της Dalvik εικονικής μηχανής και τις βιβλιοθήκες των βασικών ικανοτήτων που παρέχουν στο Android συγκεκριμένη λειτουργικότητα. Ο χρόνος εκτέλεσης είναι σχεδιασμένος για να ειναι μικρός και αποδοτικός για να χρησιμοποιείται από σε κινητές συσκευές.
- Ένα πλαίσιο εφαρμογών που εκθέτει πραγματικά σύστημα υπηρεσιών για το επίπεδο εφαρμογών, συμπεριλαμβανομένου τον διαχειριστή παραθύρων, content providers, τηλεφωνία και peer-to-peer υπηρεσίες.
- Ένα πλαίσιο για διεπαφή του χρήστη που χρησιμοποιείται για να φιλοξενήσει και να εκκινήσετε εφαρμογές.
- Προεγκατεστημένες εφαρμογές που αποστέλλονται ως μέρος της στοίβας.
- Μια πακέτο ανάπτυξης λογισμικού που χρησιμοποιείται για τη δημιουργία εφαρμογών, συμπεριλαμβανομένων των εργαλείων, plug-ins, και τεκμηρίωσης.

1.4 Περίγραμμα Εργασίας

Στο πρώτο κεφάλαιο παρουσιάσαμε το βασικό υπόβαθρο και την υποκίνηση που υπάρχει για δημιουργία πιο αποδοτικών λύσεων για σύγκριση τροχιών. Στο δεύτερο κεφάλαιο της εργασίας αυτής θα παρουσιάσουμε το μοντέλο συστήματος, τα χαρακτηριστικά που έχει το μοντέλο SmartTrace. Θα αναφερθούμε επίσης σε όλους τους τεχνικούς όρους, τους αντίστοιχους συμβολισμούς που θα χρησιμοποιηθούν καθώς επίσης και σε όλες τις παραδοχές που πρέπει να γίνουν. Στο τρίτο κεφάλαιο θα γίνει μια παρουσίαση του πρωτοκόλλου SmartTrace καθώς και περιγραφές όλων των φάσεων του πρωτοκόλλου. Επίσης θα υπάρχουν παραδείγματα για την χρήση του πρωτοκόλλου. Στο τέταρτο κεφάλαιο θα παρουσιάσουμε την υλοποιήσει του πλαισίου SmartTrace,

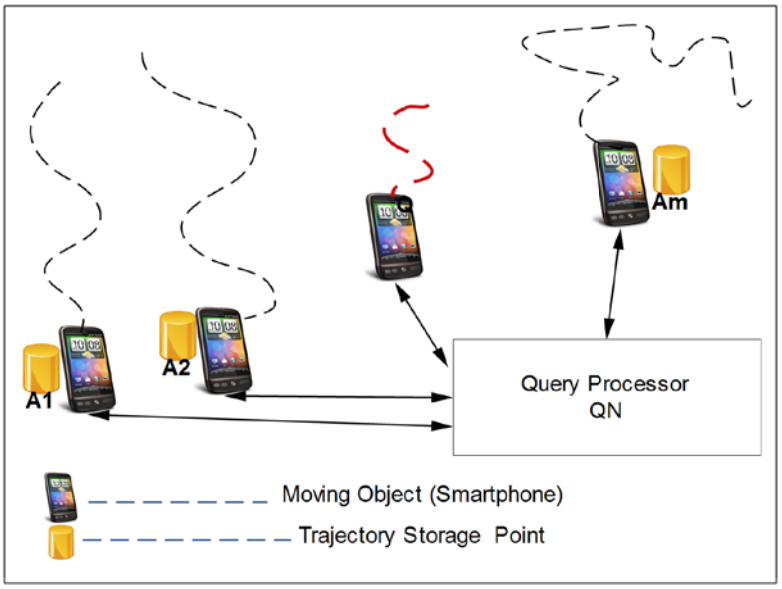
τους δυο εξυπηρετητές στην γλωσσά προγραμματισμού java και ένα πελάτη υλοποιημένο στην πλατφόρμα Android καθώς και το σύστημα διπροσωπίας τους. Στο πέμπτο κεφάλαιο θα γίνει μια παρουσίαση της
πειραματικής διαδικασίας που ακολουθήσαμε για την σύγκριση των αλγορίθμων ενώ στο έκτο θα παρουσιάζουμε και θα αναλύσουμε τα αποτελέσματα αυτής της διαδικασίας. Στο έβδομο και τελευταίο κεφάλαιο θα παραθέσουμε τα τελικά συμπεράσματα και τις επιπλέον βελτιστοποιήσεις που μπορούν να γίνουν. Στο Παράρτημα Α παρουσιάζουμε το σχετικό RFC για την σχετική χρησιμοποίηση του πρωτοκόλλου, στο Β τον πηγαίο κώδικα των εφαρμογών που έχουμε γράψει και τέλος τον συνοδευτικό ψηφιακό δίσκο.

Κεφάλαιο 2

Μοντέλο Συστήματος

2.1 Το μοντέλο του πλαισίου SmartTrace

Ας υποθέσουμε ότι { A_1, A_2,…, A_m} συνθέτει ένα σύνολο από m χρήστες κινητών συσκευών που κινούνται σε xy-πλαίσιο(βλέπε Σχήμα 2.). Σε κάθε διακριτή χρονική τιμή , αντικείμενο A_i ($\forall_i \leq m$) δημιουργώ μια χώρο-χρονική εγγραφή $A_{ij}(t_{ij}$, x_{ij} , y_{ij}) οπού το t_{ij} συνθέτει/αντικατοπτρίζει την προσωρινή διάσταση του A_i και x_{ij} , y_{ij} τις δυο χωρικές διαστάσεις του A_i. Κατά συνέπεια μπορούμε να πούμε ότι μια τροχεία μπορεί να εκφραστεί ως μια συνεχόμενη ακολουθία από l εγγραφές που έχουν την μορφή A_i=(A_{i1},A_{i2}, …, A_{il}) οπού l=|A_i| .



Σχήμα 2.1 : Αρχιτεκτονική συστήματος του πλαισίου SmartTrace.

Το αρχείο με τα καταγεγραμμένα σημεία είναι αποθηκευμένο τοπικά στο Smartphone (π.χ στην flash μνήμη του κινητού). Ένα Smartphone είναι κινητή συσκευή που λειτουργά με μπαταριά. Έχει περιορισμένους υπολογιστικούς πόρους και λειτουργιές δικτύωσης που είναι ακριβοί λόγω των συγκρούσεων στο επίπεδο MAC και των αναμεταδόσεων. Επιπρόσθετα η γραμμή είναι ασύμμετρη με την αποστολή δεδομένων (uplink) να είναι πολύ πιο αργότερο από την ανάκτηση δεδομένων (downlink).

Μπορούμε τώρα να πούμε ότι έχουμε ένα αυθαίρετο παράδειγμα από μια επερώτηση ομοιότητας $Q=(Q_1, Q_2,\ldots,Q_f)$, οπού $f << l$ (το f είναι ισοδύναμο με $|Q|$)., που σκοπό έχει την ανακάλυψη των K πιο σχετικών τροχείων με το Q, οπού K μια σταθερά που έχει εισηγείται από τον χρήστη. Η Q μπορεί να ξεκινήσει από ένα έξυπνο τηλέφωνο και να διαδοθεί στο κόμβο QN που θα απαντήσει. Η εύρεση μεταξύ Q και A_i ($i<m$), καλεί κάποιες ειδικευμένες μετρικές ομοιότητας που μπορούν να λειτουργήσουν κάτω από χωρικό και χρονικό θόρυβο.

Για παράδειγμα αν καμία από τις τροχιές δεν κινήθηκε σε κάποια προηγουμένη χρονική στιγμή ή παρουσιάζει μια ελάχιστη απόκλιση από την χωρική κίνηση τότε τα μετρικά πρέπει να μπορεί να ανταποκριθούν με ψηλά επίπεδα ομοιότητας. Τα μετρικά αυτά θα τα επεξηγήσουμε σε πιο κάτω κεφάλαιο. Σε αυτό το σημείο μπορούμε να υποθέσουμε ότι υπάρχει κάποια συνάρτηση FullM(Q, A_i), οπού συγκρίνει τις δυο

αυτές τροχιές με ακρίβεια αλλά με υψηλό υπολογιστικό κόστος. FullM(Q, A_i) επιστρέφει ένα σκορ από το διάστημα [0..1] (οπού 1 ο υψηλότερος δείκτης ομοιότητας). Σε ένα δίκτυο έξυπνων κινητών συσκευών η FullM(Q, Ai) μπορεί να διεκπεραιωθεί με κεντρικοποιημένο τρόπο (δηλαδή να στείλουν και τα m έξυπνα τηλεφωνά τις τροχιές τους στον εξυπηρετητή) ή αποκεντρικοποιημένο (δηλαδή να υπολογίσει η κάθε συσκευή τοπικά την FullM(Q, A_i) και να στείλει την βαθμολογία της στον εξυπηρετητή) τρόπο. Όπως θα παρουσιάσουμε στην αναλυτική και πειραματική ανάλυση η κεντρικοποιημένη προσέγγιση εκτελείται με πολύ άσχημα αποτελέσματα ως προς την ενέργεια και τον χρόνο απόκρισης αλλά έχουμε και προβλήματα με την ιδιωτικότητα των δεδομένων αφού ο κάθε χρήστης πρέπει να στείλει την τροχεία του στον QN κόμβο. Η αποκεντρικοποιημένη από την άλλη αποδίδει άσχημα ως προς την κατανάλωση ενέργειας λόγω του ότι εκτελεί ακριβές υπολογιστικά πράξης στην κάθε κινητή συσκευή που συμμετέχει στην αναζήτηση. Εμείς προτείνουμε την προσέγγιση SmartTrace η οποία αποδίδει πολύ καλά ως προς την κατανάλωση ενέργειας και του χρόνου απόκρισης αλλά και συγχρόνως δεν αποκαλύπτει την πλήρη τροχιά του χρηστή στο QN κόμβο.

2.2 Προκαταρτικά πλαισίου SmartTrace

Πρώτη διαπίστωση ότι η ομοιότητα ερώτημα Q κινείται από κάποια επερώτηση κόμβο QN (ή, εναλλακτικά, σε κάποιο έξυπνο τηλέφωνο που διαδίδεται την τροχεία του, Q προς QN). QN διαδίδει τότε το Q σε όλους ενεργούς χρήστες των έξυπνων κινητών συσκευών σε ένα προκαθορισμένο χωρικό όριο. Κατά την παραλαβή Q, κάθε υποψήφιο έξυπνο τηλέφωνο εκτελεί τοπικά μια ανέξοδη σε γραμμικό χρόνο συνάρτηση αντιστοίχισης. QN κατόπιν συλλέγει αυτά τα αποτελέσματα και να συγκεντρώνει σε ένα διάνυσμα άνω φράγματα UB = (ub_1,… , ub_m). Θα αναφερθώ στο UB-διάνυσμα που έχει κατασκευαστεί σχετικά με τον QN ως ΜΕΤΑΔΕΔΟΜΕΝΑ και με την πραγματική τροχεία αποθηκεύεται τοπικά σε κάθε έξυπνο τηλέφωνο ως δεδομένα. Προφανώς, τα ΔΕΔΟΜΕΝΑ είναι τάξεις μεγέθους μεγαλύτερο από τα ΜΕΤΑΔΕΔΟΜΕΝΑ, έτσι τα ΔΕΔΟΜΕΝΑ πρέπει να παραμείνετε στο έξυπνο τηλέφωνο κατά τη διάρκεια την ανάλυση του ερωτήματος. Στόχος μας είναι να εκμεταλλευτούμε με έξυπνο τρόπο τις βαθμολογίες των ΜΕΤΑΔΕΔΟΜΕΝΩΝ για τον προσδιορισμό των K υψηλότερων σε κατάταξη απαντήσεων χωρίς διάβασμα όλων των ΔΕΔΟΜΕΝΩΝ στο QN.

2.3 Λεπτομερής ανάλυση πλαισίου SmartTrace για εξωτερικούς χώρους

Σχήμα 2.2 : Η διεπαφή του πλαισίου SmartTrace σε κινητές συσκευές με λειτουργικό Android.

Ο SmartTrace αλγόριθμος είναι ένας επαναληπτικός αλγόριθμος για ανάκτηση των K πιο όμοιων τροχιών στην επερωτώμενη τροχεία Q. Ο σκοπός του όλου συστήματος είναι να έχει την μεγίστη απόδοση και από πλευράς χρόνου απόκρισης και από πλευρά ενέργειας, αλλά χωρίς να αποκαλύπτει ολόκληρες τις τροχιές στην απάντηση από τον QN κόμβο.

Σχήμα 2.3 : Παράδειγμα εκτέλεση για μια επερώτηση.

Σαν πρώτη φάση του αλγόριθμου SmartTrace, ο κόμβος QN αναθέτει σε όλους τους m κόμβους την εκτέλεση ενός υπολογισμού σε γραμμικό χρόνο για την Upper-Bounding

LCSS (M B E_Q, A_i) ($i \leq m$). Με αυτό τον τρόπο αποφεύγουμε την τεραστία ανάπτυξη της ακριβής σε υπολογιστικό κόστος συνάρτησης LCSS(Q, A_i) [9], που θα παρουσιάσουμε αμέσως μετά, η οποία εκτελεί τοπικά σε όλη την έκταση και στις δυο διαστάσεις χρόνο και χώρο για να μπορέσει να ξεπεράσει το εμπόδιο του χωρικού και χρονικού θορύβου στις τραχείες. Συγκεκριμένα, κάθε κόμβος συγκρίνει την τοπική τροχεία A_i για τον οριοθέτηση του φάκελου από την επερώτηση.

$$LCSS(M\ B\ E_Q,\ A_i) = \sum_{j=1}^{|A_i|} \begin{cases} 1 \text{ αν το } A_i[j] \text{ είναι στον φάκελο} \\ 0 \text{ διαφορετικά} \end{cases}$$

Στην δεύτερη φάση του αλγόριθμου SmartTrace, ο κόμβος QN άνακτα όλα αυτά τα άνω όρια (upper bounds) με φθίνουσα σειρά για το τοπικό διάνυσμα ΜΕΤΑΔΕΔΟΜΕΝΩΝ. Κάνοντας στο αυτό , ο QN μπορεί να έχει μια γρήγορη σύνοψη από τις τροχιές που είναι κοινές με τον Q.

Οι φάσεις 3 μέχρι 5 εκτελούνται επαναληπτικά μέχρι ο αλγόριθμος να συγκλίνει στην τελική απάντηση. Πιο λεπτομερώς, στην τρίτη φάση, ο κόμβος QN προσθέτει τις ταυτότητες των λ+1 αντικειμένων με την υψηλότερη βαθμολογία σε ένα σύνολο S. Αυτά τα αντικείμενα παρέχουν την πρώτη γραμμή από το υποψήφιο σύνολο για απάντηση, οπού τα αντικείμενα αυτά έχουν την υψηλότερη βαθμολογία LCSS (M B E_Q, A_i). Τα δεδομένα αντικείμενα θα αναλυθούν πιο προσέχτηκα στην επομένη φάση του αλγόριθμου για να αποφασιστεί το σωστό top-K σύνολο. Πρέπει να παρατηρήσουμε το γεγονός ότι τα αντικείμενα στο σύνολο S, δεν ξανά ορίζονται στο τελικό top-K αποτέλεσμα. Πιο συγκεκριμένα, είναι απόλυτος πιθανό κάποια αυθαίρετα αντικείμενα στο σύνολο S με υψηλή βαθμολόγια LCSS (M B E_Q, A_i) να έχουν χαμηλή LCSS(Q, A_i). Συνεπώς, ο αλγόριθμος μπορεί ακόμη να μην συγκλίνει στο επιθυμητό αποτέλεσμα.

Η παράμετρος λ , η οποία έχει προαναφερθεί στην προηγούμενη παράγραφο, εκφράζει μια αισιοδοξία της εφαρμογή μέσα στα όρια των ΜΕΤΑΔΕΔΟΜΕΝΩΝ. Με περισσότερη λεπτομέρεια , όταν το διάνυσμα των ΜΕΤΑΔΕΔΟΜΕΝΩΝ περιέχει πιο στενά όρια, τότε το λ μπορεί να έχει μικρή τιμή. Έτσι η παράμετρος καθορίζει ποσό επιθετική μπορεί να γίνει η εφαρμογή που θέλει να πάρει το top-K αποτέλεσμα. Μπορεί

εύκολα να αποδειχτεί ότι ο αλγόριθμος δεν μπορεί να ξεπεράσει O(m/λ) επαναλήψεις στην χειρίστη περίπτωση.

Στην τέταρτη φάση, QN ζητά από κάθε έξυπνο τηλέφωνο που ανήκει στο σύνολο S, να εκτελέσουν τον υπολογισμό η πλήρης βαθμολογίες (αν ένα έξυπνο τηλέφωνο έχει χρησιμοποιηθεί σε προηγούμενη επανάληψη δεν χρησιμοποιείται και πάλι). Ειδικότερα, ζητούμε από κάθε έξυπνο τηλέφωνο για να υπολογιστεί σε τοπικό επίπεδο FullM (Q, Ai), όπου Ai είναι αποθηκευμένα τοπικά, και μεταδίδουν μόνο την τιμή της FullM (Q, Ai) προς QN (δηλαδή, το αποκεντρωμένο τρόπο). Εναλλακτικά, θα μπορούσαμε να έχουμε ανακτήσει το σύνολο S στο κινητό και μετά να υπολογίζαμε FullM (Q, Ai) ∀Ai ∈ S (δηλαδή, η κεντρικοποιημένο τρόπο), ωστόσο αυτό θα παραβίαζε τόσο τον συντελεστή αποκάλυψης και θα υποβάθμιζε και τον χρόνο απόκρισης του αλγόριθμο σε επίπεδο ανάλογο με την κεντρικοποιημένο αλγόριθμο. Παρατηρήστε ότι η τέταρτη φάση του αλγορίθμου ισχύει μόνο για το στοιχεία του συνόλου S, σε αντίθεση με όλα τα στοιχεία m, έτσι αυτό είναι πραγματικά πολύ φθηνότερο από άποψη ενέργειας που καταναλώνεται για ένα έξυπνο τηλέφωνο ως | S | << m.

Στην περίπτωσή μας, FullM (Q, Ai) είναι η ομοιότητα LCSS, η οποία έχει χρησιμοποιηθεί εκτενώς σε πολλά ακολουθιακά προβλήματα μιας διάστασης, όπως ο αντιστοίχιση σειρές χαρακτήρων. Η προσαρμογή του LCSS σε δυο διαστάσεις χρησιμοποιώντας το L_∞ ορίζεται ως εξής: Δεδομένου των ακέραιων δ και ε, η μεγαλύτερη κοινή ομοιότητα υπό-ακολουθίας $LCSS_{\delta,\varepsilon}$ (A, B) μεταξύ δύο ακολουθιών A και B ορίζονται ως εξής:

$$LCSS_{\delta,\varepsilon}(A,B) = \begin{cases} 0 \text{ αν το A ή το B είναι κενό} \\ 1 + LCSS_{\delta,\varepsilon}(Tail(A), Tail(B)) \\ \qquad \text{και } |ay{:}l1 - by{:}l2| < \varepsilon \text{ και } |l1 - l2| < \delta \\ \max(LCSS,(Tail(A),B),LCSS,(A,Tail(B))) \text{ otherwise} \end{cases}$$

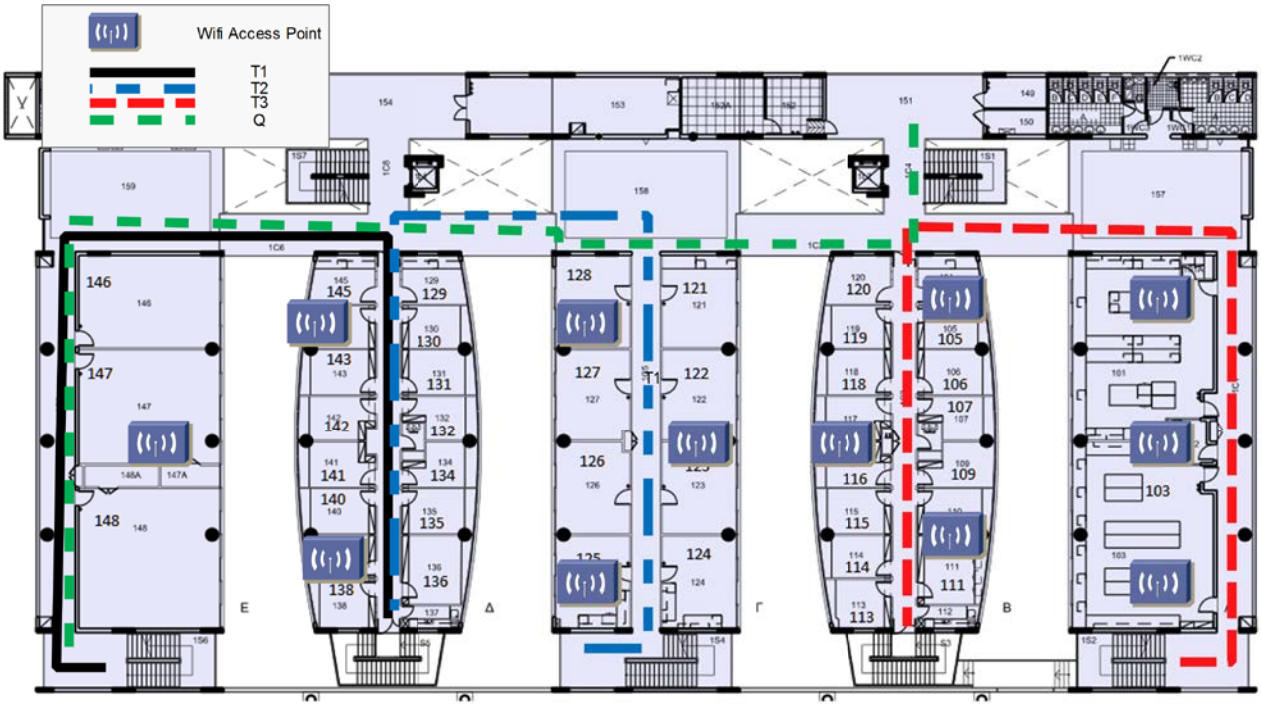
όπου δ και ε είναι ειδικές παραμέτρους της εφαρμογής που επιτρέπουν ευέλικτη αντιστοίχιση στο χρονικό (π.χ., αν έχουμε δύο πανομοιότυπες τροχιές, αλλά το πρώτο αντικείμενο κινείται νωρίτερα στο χρόνο) και στο χωρικό (π.χ., έχουμε δύο

πανομοιότυπες τροχιές, αλλά η πρώτη έχει κάποια μικρή απόκλιση στη χωρική περιοχή) πεδίο ορισμού, αντίστοιχα. Η LCSS ασχολείται με τα δύο προαναφερθέντα όρια της οικογένειας των Lp-Norm των αποστάσεων, επειδή αυτές τις περιπτώσεις απλά μειώθηκαν από την αντίστοιχη χωρίς στρέβλωση του μέτρου.

Στην πέμπτη φάση, θα καθορίσουμε αν ο αλγόριθμος έχει φτάσει την συνθήκη τερματισμού. Πιο συγκεκριμένα, ελέγχουμε αν το (λ+1) - οστό υψηλότερο UB είναι μικρότερο από την k- κοστή μεγαλύτερη τιμή πλήρης αντιστοίχισης. Εάν συμβαίνει αυτό, τότε μπορούμε να τερματίσουμε με ασφάλεια την εκτέλεσης του αλγορίθμου και να είμαστε σίγουροι ότι το σωστό top-K σύνολο έχει προσδιοριστεί. Αν αυτή η συνθήκη δεν πληρείται (δηλαδή, όταν το UB ενός αντικειμένου X είναι μεγαλύτερο από την K μεγαλύτερη τιμή πλήρης αντιστοίχισης Y), τότε είμαστε αναγκασμένη να εκτελέσουμε κι' άλλη επανάληψη αφού η απάντηση δεν είναι ντετερμινιστική δηλαδή, είτε X είτε Y μπορεί να είναι το K- οστό η απάντηση). Κατά συνέπεια, έχουμε αυξήσει το βήμα κατά λ επανάληψης, ώστε να εντοπίζει τους επόμενους λ υποψηφίους στον επόμενο γύρο.

Στην τελική φάση, η οποία εμφανίζεται μόνο μία φορά στο τέλος, ίσως σταλεί κάθε αντιστοιχισμένη ακολουθία A_i^{match} ($|A_i^{match}| << |A_\iota|$) στον QN, η οποία μπορεί να επιστραφεί στη συνέχεια στον χρήστη. Προσέξτε, ότι οι κόμβοι που επιλεχτήκαν S_i ($i \leq K$) μπορούν να επιλέξουν να μη μοιράζονται την αντιστοιχισμένη τροχεία ή να την μοιράζονται με βάση ορισμένο τοπικό προφίλ για αποκάλυψη πληροφοριών [10], προκειμένου να διατηρηθεί η k-ανωνυμίας και άλλα υψηλοτέρα σχήματα ανωνυμίας. Σε κάθε περίπτωση, ούτε QN ούτε την χρήστης που έκανε την επερώτηση θα δει ποτέ την πλήρη τροχεία των χρηστών που συμμετέχουν (βλέπε Σχήμα 2.2).

2.4 Λεπτομερής ανάλυση πλαισίου SmartTrace για εσωτερικούς χώρους



Σχήμα 2.4 : Λειτουργιά του πλαισίου SmartTrace σε εσωτερικό χώρο.

Σε αυτό το υποκεφάλαιο θα εξηγήσουμε την πλήρη λειτουργιά του αλγορίθμου SmartTrace σε εσωτερικό χώρο. Συγκεκριμένα θα δείξουμε πως ο SmartTrace αλγόριθμος εκμεταλλεύεται τα σήματα των ασύρματων σημείων πρόσβασης για να εκτελέσει αναζήτηση ομοιότητας(βλέπε Σχήμα 2.4).

Είναι γνωστό ότι τα σήματα από τον δορυφόρο που χρησιμοποιούνται από τους αποδεκτές GPS είναι εξασθενημένα ή παγιδευμένα μέσα σε ένα κτίριο και επιπλέον αποτυγχάνουν να παρέχουν μια ακριβής πληροφορία για την συγκεκριμένη τοποθεσία. Η τεραστία ανάπτυξη των κινητών συσκευών μαζί με το γεγονός ότι ο μέσος άνθρωπος ξοδεύει τις περισσότερες ώρες του , πάνω από 90% του συνολικού του χρόνου, σε εσωτερικό περιβάλλον κίνησαν το ενδιαφέρον για γεωγραφικές εφαρμογές εσωτερικούς χώρους .

Τα build-in Cell tower ή WLAN τεχνικές εύρεσης θέσης για έξυπνα τηλεφωνά δεν μπορούν να παρέχουν μια αρκετή καλή ακρίβεια για να υπολογίσουν της ομοιότητες μεταξύ των εσωτερικών χώρων . Για παράδειγμα , η "*My location*" υπηρεσία που προσφέρει το Google, μπορεί μόνο να παρέχει ακρίβεια περίπου στα 200 μετρά ενώ η

εύρεση θέση μέσω Cell Tower παρέχει ακρίβεια μεταξύ μερικών χιλιάδων χιλιόμετρων. Παρόλα αυτά οι δυο τεχνικές δεν μπορούν να συνεργαστούν για εσωτερικό χώρο σε αυτό το υποκεφάλαιο.

Μια χώρο-χρονικη τροχεία A_i ($i \leq m$), δίνεται από μια RSS τροχεία, είναι μια ακολουθία από l πολυδιάστατες εγγραφές του τύπου $\{a_1, a_2, \ldots, a_l\}$. Κάθε πλειάδα χαρακτηρίζεται από n διαστάσεις χώρου και μια χρονική, π.χ. $a_j(t_j, S_j^1, S_j^2, \ldots, S_j^n)$, $j \leq l$, οπού t_j δίνεται από στιγμιότυπα χρόνου οπού κάθε πλειάδα είχε δημιουργηθεί και S_j^k ($k \leq n$, $j \leq l$) είναι η τιμή του RSS από το K-οστό σημείο ασύρματης πρόσβασης(Access Point (AP)) υπολογιζόμενο από A_i στην t_j . Χρησιμοποιώντας το πιο πάνω μοντέλο. Η ομοιότητα μεταξύ δυο τροχείων μπορεί να υπολογιστεί χρησιμοποιώντας τις μετρικές του LCSS σε κάθε χαρακτηριστικό και προσθέτοντας τις γραμμικά..

Έχουμε κάνει πειράματα στο χώρο του πανεπιστήμιου όπως φαίνεται στο Σχήμα 2.4. Ο χρήστης που έχει κάνει την επερώτηση έχει την τροχεία Q και ζήτα τους 2 πιο ομοίους μαζί του. Η εφαρμογή επιστρέφει τον χρηστή με τροχεία T1 και τον χρηστή με τροχεία T2.

2.5 Επιπλέον Παραδοχές

Δεδομένου του γεγονότος ότι το μοντέλο του πλαισίου SmartTrace που αναφέραμε την παράγραφο 2.1 αφορά κινητές συσκευές. Οι κινητές αυτές συσκευές συνήθως είναι έξυπνα τηλεφωνά τα οποία υποστηρίζουν διάφορες υπηρεσίες μέσω τον αισθητήρων που διαθέτουν. Έτσι πρέπει να αποδεχτούμε ότι όσα αναφέρονται στις επόμενες παραγράφους 2.2, 2.3 και 2.4 απαιτούν από την κινητή συσκευή να έχει τουλάχιστον τον απαραίτητο εξοπλισμό σε υλικό, πιο συγκεκριμένα Παγκόσμιο Σύστημα Εντοπισμού γνωστό ως GPS και ασύρματη κάρτα διαδικτύου γνώστη ως Wi-Fi. Επίσης το αρχείο με τα καταγεγραμμένα σημεία είναι αποθηκευμένο τοπικά στο Smartphone (π.χ στην flash μνήμη του κινητού). Ένα Smartphone είναι κινητή συσκευή που λειτουργά με μπαταριά άρα πρέπει να έχουμε πάντα υπόψη μας την κατανάλωση ενέργειας. Έχει περιορισμένους υπολογιστικούς πόρους και λειτουργιές δικτύωσης που είναι ακριβοί λόγω των συγκρούσεων στο επίπεδο MAC και των αναμεταδόσεων.

Επιπρόσθετα η γραμμή είναι ασύμμετρη με το uplink να είναι πολύ πιο αργότερο από το downlink.

2.6 Πίνακας Συμβολισμών και Παραδοχών

Συμβολισμός	Περιγραφή
Q	Η επερωτώμενη τροχεία.
QN	Ο κόμβος που επεξεργάζεται την επερώτηση.
A_i	Μια συνεχόμενη ακολουθία από l εγγραφές από χώρο-χρονικα σημεία
A_i^{match}	Η πιο κοινή τροχεία με την Q.
MAC	Media Access Control.
AP	Access Point
RSS	Received Signal Strength.
GPS	Global Positioning System.
STP	SmartTrace Protocol
S	Εξυπηρετητής (Server)
C	Πελάτης (Client)
Cs	Πελάτης που κάνει την επερώτηση (Client Searcher)
Cp	Πελάτης που συμμετέχει στην επερώτηση (Client Participant)

Πίνακας 2.1

Παραδοχές
1. $l=\|A_i\|$
2. $\|A_i^{match}\| << \|A_i\|$

Πίνακας 2.2

Κεφάλαιο 3

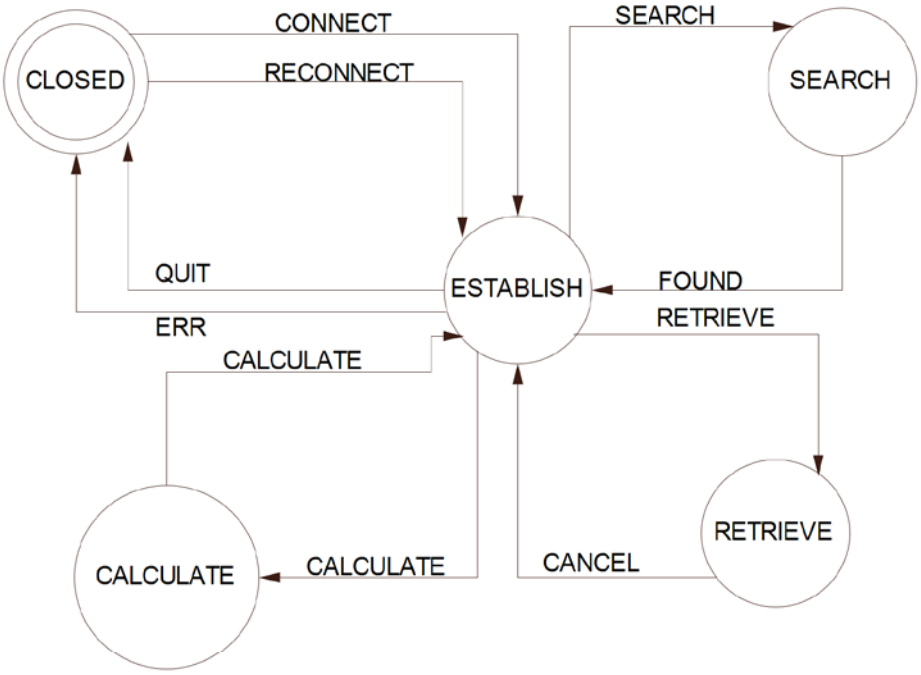
Πρωτόκολλο SmartTrace

3.1 Διάγραμμα καταστάσεων

Όπως μπορούμε να δούμε το πρωτόκολλο αποτελείται από πέντε βασικές καταστάσεις (βλέπε Σχήμα 3.1). Οι ροές εισόδου και εξόδου δείχνουν τις μετακινήσεις μεταξύ των καταστάσεων του κάθε τμήματος. Το διάγραμμα καταστάσεων φανερώνει όλες τις δυνατές καταστάσεις οπού ο πελάτης και ο εξυπηρετητής μπορούν να βρίσκονται ανά πασά στιγμή.

Η κάθε φάση του πρωτοκόλλου επεξηγείται με περισσότερες λεπτομερές σε αυτό το υποκεφάλαιο. Το πρωτόκολλο μπορεί να εφαρμοστεί και να υποστηριχτεί από διάφορες πλατφόρμες με οποιοδήποτε λειτουργικό σύστημα αφού οι συναλλαγές μεταξύ πελάτη εξυπηρετητή γίνεται σε καθαρό κείμενο. Στο πρωτόκολλο υπάρχουν κάποια θέματα ασφάλειας αφού τα δεδομένα δεν είναι κρυπτογραφημένα. Μπορεί να επιλυθεί με την εισαγωγή και συνεργασία του TLS (Transport Layer Security) πρωτοκόλλου. Δηλαδή τα δεδομένα να αποστέλλονται μέσω του TLS έτσι τα δεδομένα μπορούν να

αποστέλλονται κρυπτογραφημένα και τα ευαίσθητα δεδομένα του χρηστή να είναι ασφαλές,



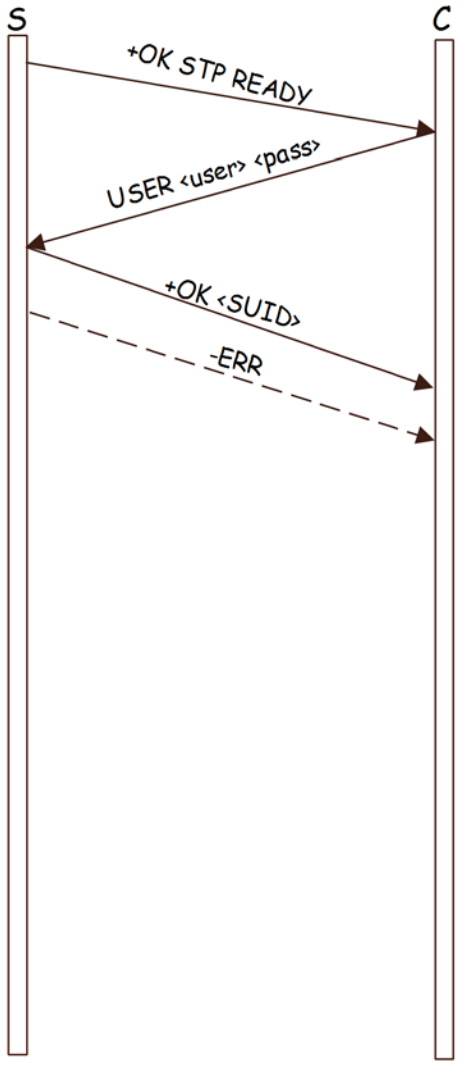
Σχήμα 3.1 : Διάγραμμα καταστάσεων πλαισίου SmartTrace

Με την εισαγωγή του TLS μπορούμε να επιτύχουμε την εμπιστευτικότητα και την ακμαιότητα που αρμόζει σε ένα πρωτόκολλο που μεταχειρίζεται προσωπικά δεδομένα που αφορούν τον χρηστή.

3.2 Πρωτόκολλο SmartTrace: CLOSED STATE

Όλοι οι πελάτες αρχικοποιούνται με αυτή την κατάσταση. Η κατάσταση αυτή δείχνει τη δημιουργία της σύνδεσης TCP μεταξύ του πελάτη και του STP εξυπηρετητή. Όταν ένας πελάτης αρχίσει να χρησιμοποιεί το πρωτόκολλο και συνδεθεί μαζί με τον εξυπηρετητή βρίσκεται στην κατάσταση.

3.3 Πρωτόκολλο SmartTrace: ESTABLISH STATE



Σχήμα 3.2 : Ανταλλαγή μηνυμάτων για την κατάσταση ESTABLISH

Μόλις η σύνδεση TCP έχει εγκαθιδρυθεί από έναν πελάτη STP, ο STP εξυπηρετητής μια γραμμή χαιρετισμού. Αυτή μπορεί να είναι οποιαδήποτε θετική απόκριση. Ένα παράδειγμα θα μπορούσε να είναι:

S: +OK STP READY

Το STP session είναι τώρα στην φάση ESTABLISH. Ο πελάτης τώρα πρέπει να αναγνωριστεί και να επικυρωθεί από τον εξυπηρετητή STP. Δύο πιθανοί μηχανισμοί για να γίνει αυτό περιγράφονται στο έγγραφο αυτό (βλέπε Παράρτημα Α).

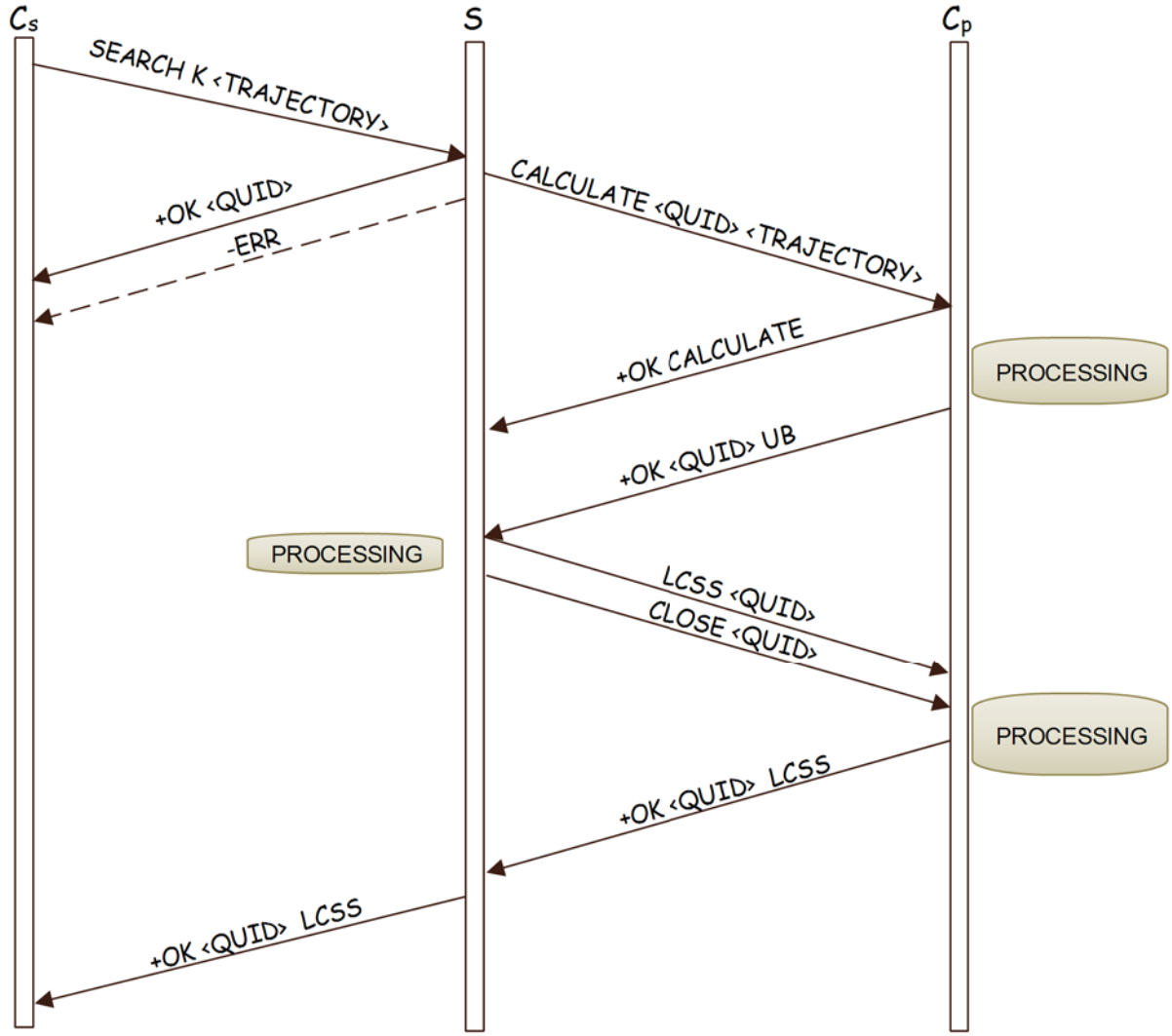
Μόλις ο εξυπηρετητής STP έχει καθορίσει την επικύρωση του πελάτη τότε μπορεί να του δώσει πρόσβαση και να μπορέσει να χρησιμοποιήσει το πλαίσιο SmartTrace . Αν ο πελάτης δώσει την εντολή SEARCH στο STP session ή θα μεταβεί στην κατάσταση

SEARCH ή θα λάβει ένα μήνυμα λάθους αν ο εξυπηρετητής δεν μπορεί να τον εξυπηρέτηση την δεδομένη χρονική στιγμή.

Ο πελάτης πρέπει να δώσει σωστά στοιχειά ούτος σώστε ο εξυπηρετητής να τον επικυρώσει. Τα στοιχειά αυτά στο παρόν στάδιο μεταφέρονται υπό μορφή καθαρού κείμενου μέσω του δικτύου.

Μετά την επιστροφή μιας αρνητικής ένδειξης κατάστασης, ο εξυπηρετητής μπορεί να κλείσει την σύνδεση. Εάν ο εξυπηρετητής δεν κλείσει την σύνδεση, ο πελάτης μπορεί να δώσει είτε μια νέα εντολή ελέγχου ταυτότητας και να αρχίσει εκ νέου, ή την πελάτης μπορεί να δώσει την εντολή QUIT. Μετά την εξακρίβωση την επαλήθευση του χρηστή ο διακομιστείς STP στείλει ένα μοναδικό αριθμό που αντιπροσωπεύει την ταυτότητα του Session.

3.4 Πρωτόκολλο SmartTrace: SEARCH STATE



Σχήμα 3.3: Ανταλλαγή μηνυμάτων για την κατάσταση SEARCH

Μόλις ο πελάτης έχει την σύνδεση και να εγκριθεί, είναι στην κατάσταση ESTABLISH. Θα μπορούσε να μεταβεί στην κατάσταση SEARCH ή στην κατάσταση CALCULATE με το να στείλει “SEARCH” ή αποστολή “+OK CALCULATE” μήνυμα λόγω της μετάδοσης του πελάτη που κάνει την επερώτηση, αντίστοιχα. Όταν ο πελάτης είναι σε κατάσταση SEARCH αυτός θα πρέπει να περιμένει για μια απάντηση από τον εξυπηρετητή STP. Τότε περιμένει για το σκορ LCSS. Αν ο πελάτης θέλει να κάνει ένα ερώτημα χρησιμοποιώντας το πρωτόκολλο SmartTrace πρέπει να στείλει το ερώτημα

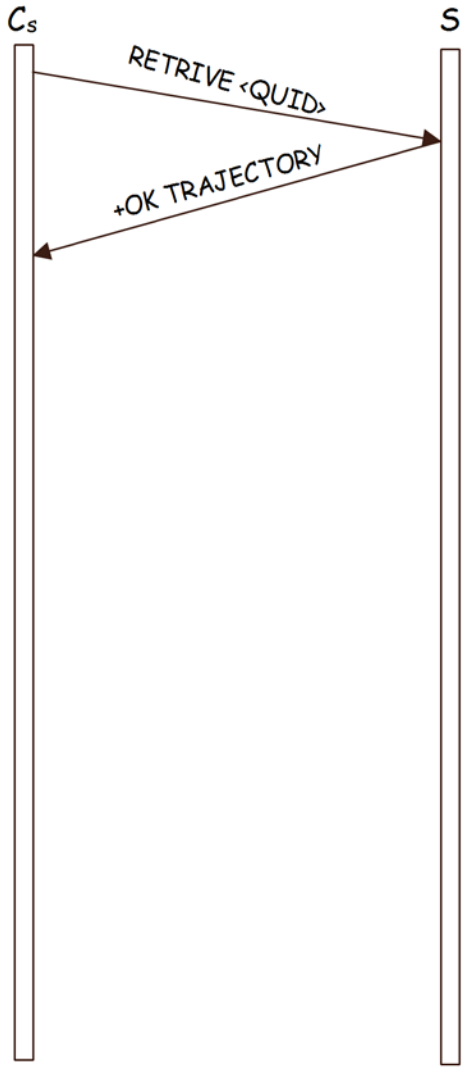
χρησιμοποιώντας την εντολή SEARCH. Ο χρόνος απόκρισης του το επερωτήματος είναι με βάση το χρόνο της επεξεργασία του κάθε πελάτη (βλέπε Σχήμα 3.3).

3.5 Πρωτόκολλο SmartTrace: CALCULATE STATE

Όταν ο πελάτης κάνει το επερώτημα στον εξυπηρετητή STP ενημερώνει τους ήδη συνδεδεμένους πελάτες για τον υπολογισμό της βαθμολογίας UB και στη συνέχεια το σκορ LCSS (βλέπε Σχήμα 3.3). Εδώ ο κάθε χρήστης που είναι συνδεδεμένος εκτελεί όπως έχουμε περιγράψει και πιο πάνω του δυο αλγορίθμους τάξεως του O(n) και $O(n^2)$ αντίστοιχα.

Ο χρόνος απόκριση εξαρτάτε από την υπολογιστική δύναμη του λιγότερου ισχυρού επεξεργαστή για την φάση του άνω ορίου UB καθώς και για την φάση του LCSS. Αν το έξυπνο τηλέφωνο καλεστεί για να υπολογίσει το LCSS σκορ πρέπει ο εξυπηρετητής να περιμένει τον συγκεκριμένο πελάτη να υπολογίσει και να αποστείλει το σκορ. Μετά μπορεί ο ίδιος να προχωρήσει και να κάνει την ταξινόμηση με βάση αυτό το σκορ.

3.6 Πρωτόκολλο SmartTrace: RETRIEVE STATE



Σχήμα 3.4: Ανταλλαγή μηνυμάτων για την κατάσταση RETRIEVE

Όταν ο πελάτης τελειώσει με την εντολή SEARCH, μπορεί να εκτελέσει την RETRIEVE αν θέλει να ανάκτηση την τροχεία που έχει αντιστοιχηθεί (βλέπε Σχήμα 3.4). Σε περίπτωση οπού χρηστής, που έχει την A_i^{match}, δεν θέλει να την μοιραστεί τότε αποστέλλεται η λέξη PRIVATE αντί της τραχείας.

3.7 Πίνακας Επιστρεφόντων Σφαλμάτων

Αριθμός	Περιγραφή
0	Υπάρχει κάποιο λάθος στον εξυπηρετητή το οποίο οφείλεται αποκλειστικά σε αυτό άσχετα με ποιος πελάτης είναι συνδεδεμένος την παρούσα στιγμή.
401	Ο πελάτης έχει δώσει λάθος στοιχειά και δεν μπορεί ο εξυπηρετητής να τον επικυρώσει.
511	Ο εξυπηρετητής είναι απασχολημένος την δεδομένη στιγμή.
512	Δεν μπορεί κανένας από τους πελάτες να υπολογίσει τους αλγορίθμους.
513	Δεν υπάρχουν οι απαραίτητη πελάτες για να ολοκληρωθεί η επερώτηση
514	Η μορφή της βαθμολογία του άνω φράγματος είναι λανθασμένη
515	Η μορφή της βαθμολογία του LCSS είναι λανθασμένη

Πίνακας 3.3

Κεφάλαιο 4

Υλοποίηση Πρωτόκολλου SmartTrace

4.1 Εξυπηρετητής SmartTrace σε JAVA

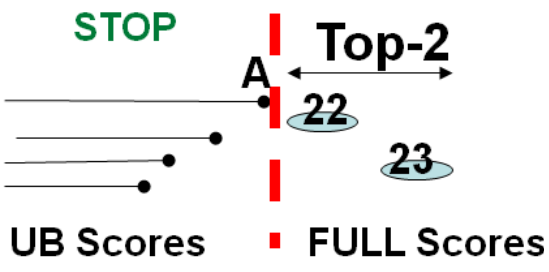
Ο εξυπηρετητής έχει γραφτεί στην γλωσσά προγραμματισμού JAVA. Με τη χρήση της Java, μπορεί να τρέξει σε πολλές διαφορετικές πλατφόρμες. Αυτό σημαίνει ότι δεν χρειάζεται να υποχρεώσει τον καθένα που χρησιμοποιεί το πλαίσιο να χάσει χρόνο για την ανάπτυξη μια διαφορετική έκδοση του λογισμικού για κάθε πλατφόρμα. Σε πολλές περιπτώσεις, η εικονική μηχανή Java μπορεί να διασφαλίσει ότι μια γραμμένη λάθος εφαρμογή δε μπορεί να δημιουργήσει προβλήματα στο υπόλοιπο περιβάλλον του υπολογιστή. Έχει την δυνατότητα για να αναπτύξουμε ένα πολυνηματίκο πρόγραμμα για να εκτελεί διάφορες λειτουργίες ταυτόχρονα στο πλαίσιο ενός προγράμματος. Στην Java, ο πολυνηματίκος προγραμματισμός έχει ενσωματωθεί ομαλά σε αυτήν, πάρα σε άλλες γλώσσες, το συγκεκριμένες διαδικασίες του λειτουργικού συστήματος πρέπει να κληθούν για να μπορέσει ενεργοποιήσουν τον παραλληλισμό μέσω νημάτων.

Επειδή τα προγράμματα γραμμένα σε γλώσσα Java τρέχουν σε μια εικονική μηχανή, τρέχει κάπως αργά σε σύγκριση με άλλα προγράμματα. Ωστόσο, είναι απίθανο να είναι αισθητά και ενοχλητικά αργό σχετικά γρήγορους υπολογιστές σήμερα. Αυτό δε επηρεάζει πολύ την ταχύτατα του εξυπηρετητή αφού ο ίδιος εξαρτάτε από τους πελάτες που είναι συνδεδεμένη σε αυτόν. Πιο συγκριμένα ο κάθε πελάτης όπως έχουμε προαναφέρει πρέπει να κάνει κάποιους υπολογισμούς έτσι στον καθένα ξεχωριστά ο χρόνος εκτέλεσης των αλγορίθμων διαφέρει από μια κινητή συσκευή σε άλλη.

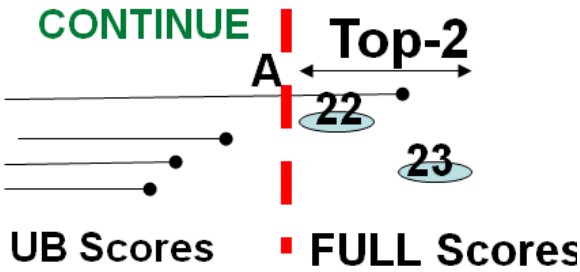
Σε αυτή την υλοποίηση τα πάντα δημιουργούνται δυναμικά. Δε κρατάμε καταστάσεις και πληροφορίες για οποιοδήποτε πελάτη εξασφαλίζοντας πλήρη ιδιωτικότητα και ανωνυμία για τον καθένα.

Ο εξυπηρετητής είναι υπεύθυνος την διαχείριση των πελατών καθώς και για την σωστή ρύθμιση όλων των βαθμολογίων μέχρι να συγκλίνει στην απάντηση. Έτσι ο εξυπηρετείς εφαρμόζει τον αλγόριθμο SmartTrace.

Ο αλγόριθμος είναι επαναληπτικός και χωρίζεται σε πέντε βήματα. Έχει σαν είσοδο την τροχιά επερώτησης Q, τις m πιθανές τροχιές, την επιλογή αποτελέσματος K (K << m) και το αυξητικό βήμα επανάληψης λ. έχει σαν έξοδο τις K τροχιές που είναι πιο κοινές με την Q.



Σχήμα 4.1 : Η συνθήκη τερματισμού ισχύει.



Σχήμα 4.2 : Η συνθήκη τερματισμού δεν ισχύει.

Στον εξυπηρετητή :

Βήμα 1. Υπολογισμός Upper Bound (UB): Οδήγησε κάθε ένα από τα m smartphones για να εκτελέσει έναν υπολογισμό γραμμικού χρόνου $LCSS(MBE_Q,A_i)$ ($i \leq m$).

Βήμα 2. Συλλογή των UB: Παράλαβε τα UBS όλων των τροχιών m που συμμετέχουν στο ερώτημα και να προσθέσετε τα αποτελέσματα στο διάνυσμα ΜΕΤΑΔΕΔΟΜΕΝΩΝ που αποθηκεύονται στον QN. Τα ΜΕΤΑΔΕΔΟΜΕΝΑ ταξινομούνται με φθίνουσα σειρά με βάση τη βαθμολογία UB.

Βήμα 3. Πλήρης Υπολογισμός: Ρώτησε τους $\lambda + 1$ ($\lambda \geq K$) υψηλότερους UB κόμβου για να υπολογίσουν τον LCSS(Q,Ai) και μετά να στείλουν τα λ αποτελέσματα πίσω.

Βήμα 4. Συνθήκη τερματισμού: Αν το επόμενο υψηλότερο UB είναι μικρότερο από το K μεγαλύτερο full match τότε σταματά (βλέπε Σχήμα 4.1) διαφορετικά πήγαινε στο βήμα 3 για να αναγνωριστούν τα επόμενα λ υποψήφια (βλέπε Σχήμα 4.2).

Βήμα 5. Μεταφορά της τροχιάς: Αν η συνθήκη τερματισμού ισχύει, μπορούμε να αποστείλουμε την αντίστοιχη τροχιά που είναι όμοια στο QN βασισμένο σε μια τοπική πολιτική ιδιωτικότητας των δεδομένων.

Όταν ολοκληρωθεί ο αλγόριθμος μπορούμε να συγκλίνουμε στην απάντηση και να ικανοποιήσουμε την επερώτηση.

4.2 Πελάτης SmartTrace σε Android

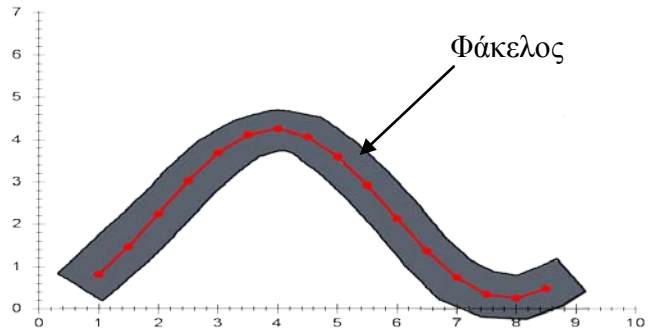
Σύμφωνα με τα πιο πάνω σε συνδυασμό της μεγάλης εξαπλώσεις του λειτουργικού συστήματος Android ήταν η καλύτερη επιλογή για ανάπτυξη της εφαρμογής μας σε αυτό το περιβάλλον.

Ο πελάτης που έχει υλοποιηθεί είναι γραμμένος σε γλωσσά προγραμματισμού JAVA με το SDK του Android. Με το Android μπορούσαμε εύκολα να έχουμε την πρόσβαση σε πολλά κομμάτια υλικού που θέλαμε να έχουμε όπως GPS και Wi-Fi. Ο πελάτης έχει υλοποιημένες δυο σημαντικές λειτουργιές που χρειάζονται για την λειτουργία του πρωτοκόλλου, το αλγόριθμο για το άνω φράγμα (Upper Bound) τον αλγόριθμο για την

μεγαλύτερη κοινή ακολουθία (Longest Common Subsequence) που θα επεξηγήσουμε πιο κάτω.

4.2.1 Ο αλγόριθμος UB

Ο αλγόριθμος για το άνω φράγμα (Upper Bound) πρέπει να είναι πολύ εύκολος υπολογιστικά έτσι η έξυπνη κινητή συσκευή να μην έχει μεγάλη επίπτωση στο χρόνο και ενέργεια που πρέπει να σπαταλήσει. Στο Σχήμα 4.3 φαίνεται η βασική ιδέα του αλγόριθμου, που είναι η δημιουργεί ενός φάκελου γύρο από την τροχεία όπως φαίνεται στο σχήμα.



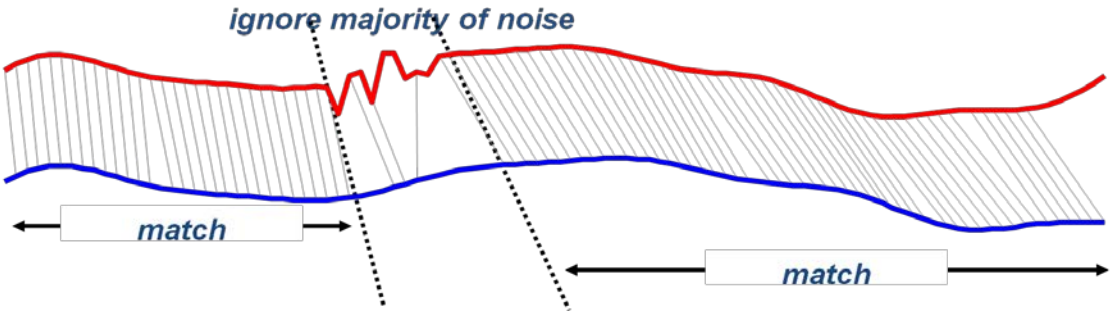
Σχήμα 4.3 : Φάκελος για αλγόριθμο του άνω φράγματος.

Ο αλγόριθμος του άνω φράγματος υπολογίζεται με χειρίστη περίπτωση O(n) οπού n ο αριθμός των σημείων της τροχιάς.

Στην περίπτωση οπού ο αλγόριθμος χρησιμοποιείται στην πτυχή του RSS δεν χρειάζεται να γίνει οποιοσδήποτε φάκελος αλλά μια πρώτη εκτίμηση για τα ποια AP μπορεί να λαμβάνει η έξυπνη κινητή συσκευή δηλαδή το ποσοστό ποσά από όλα τα AP που λαμβάνει η επερωτωμένη τροχεία.

4.2.2 Ο αλγόριθμος LCSS

Η μεγαλύτερη κοινή ακολουθία (LCS), το πρόβλημα είναι να βρεθεί η μεγαλύτερη κοινή ακολουθία σε όλες τις ακολουθίες σε ένα σύνολο ακολουθιών. Έτσι έχουμε τροποποιήσει τον αλγόριθμο αυτό για να λειτουργεί αποδοτικά με γεωγραφικές συντεταγμένες.



Σχήμα 4.4 : Διάγραμμα σύγκρισης δυο τροχείων.

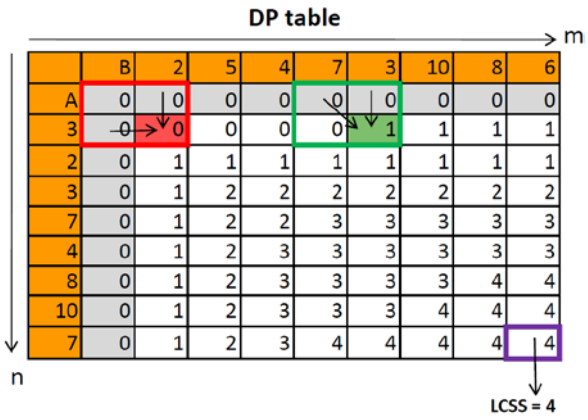
Ο LCSS αλγόριθμος μπορεί να ικανοποιήσει ερωτήματα κάτω από θόρυβο. Ο αλγόριθμος χωρίζεται σε δυο φάσεις. Η πρώτη φάση είναι η κατασκευή του DP πίνακα και η δεύτερη, η κατασκευή του LCSS μονοπατιού. Στην υλοποίηση του πελάτη δεν χρειάζεται η δεύτερη φάση του αλγόριθμου αλλά μόνο ο υπολογισμός του LCSS σκορ.

```
int A[] = {3,2,5,7,4,8,10,7};
int B[] = {2,5,4,7,3,10,8,6};
int L[n+1][m+1]; // DP Table

// Initialize first column and row of the DP Table
for (i=0;i<n+1;i++)          L[i][0] = 0;
for (j=0;j<m+1;j++)          L[0][j] = 0;

for (i=1;i<n+1;i++) {
        for (j=1;j<m+1;j++) {
                if (A[i-1] == B[j-1]) {
                   L[i][j] = L[i-1][j-1] + 1;
                } else {
                   L[i][j] = max(L[i-1][j], L[i][j-1]);
                }
        }
```

Σχήμα 4.5 : Κώδικας για υλοποίηση του LCSS



	B	2	5	4	7	3	10	8	6
A	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	1	1	1	1
2	0	1	1	1	1	1	1	1	1
3	0	1	2	2	2	2	2	2	2
7	0	1	2	2	3	3	3	3	3
4	0	1	2	3	3	3	3	3	3
8	0	1	2	3	3	3	3	4	4
10	0	1	2	3	3	3	4	4	4
7	0	1	2	3	4	4	4	4	4

Σχήμα 4.6 : Πίνακας για εύρεση του σκορ του LCSS και μονοπατιού.

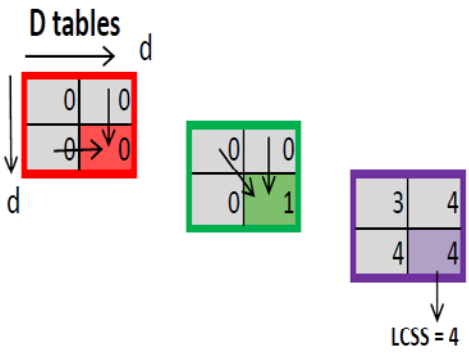
Από ότι μπορούμε να επισημάνουμε από τον κώδικα (βλέπε Σχήμα 4.5) ο αλγόριθμος είναι επαναληπτικός με χρόνο χειρίστης περιπτώσεις O(|A|*|B|).

Λόγω της χωρητικότητας της μνήμης έπρεπε να γίνουν κάποιες σημαντικές αλλαγές ούτως ώστε να μπορέσουμε να τρέξουμε τους αλγορίθμους χωρίς να χρειαζόμαστε τεραστία μνήμη (π. χ για 10000γεωγραφικα σημεία χρειαζόμαστε 763Mb = 0,75 Gb). Έτσι έχουμε τον LCSS με ένα κινητό παράθυρο έναντι του σταθερού πίνακα n x m. Για να επιτευχτεί εισάγουμε την έννοια του δ που έχουμε ήδη αναφέρει σε θεωρητικό επίπεδο.

```
int A[] = {3,2,5,7,4,8,10,7};
int B[] = {2,5,4,7,3,10,8,6};

double[][] previous= new double[d+1][d+1];
for (int i = 1; i < A.size() + 1; i++) {
        for (int j = i - d; j < i + d; j++) {
                if (j <= 0)
                        continue;
                if (j > B.size())
                        break;
        if (A[i-1] == B[j-1])
        {               previous[i%(d+1)][j%(d+1)] = previous[(i-1)%(d+1)][(j-1)%(d+1)] + 1;
        }
         else
        {               previous[i%(d+1)][j%(d+1)] = Math.max(previous[(i-1)%(d+1)][j%(d+1)],
                                                              previous[i%(d+1)][(j-1)%(d+1)]);
        }
    }
}
```

Σχήμα 4.7 : Κώδικας βελτιωμένης λύσης του LCSS.



Σχήμα 4.8 : Μεταβάσεις του πίνακα D.

Σε αντίθεση με τον αλγόριθμο LCSS που έχουμε αναφέρει η νέα υλοποίηση παρέχει μια πολύ πιο γρήγορη εκτέλεση του αφού περιορίζει την μετακίνηση των σημείων στο χρόνο. Όπως φαίνεται στο Σχήμα 4.7 η πολυπλοκότητα του αλγορίθμου μειώνεται καθώς και επαναλήψεις που χρειάζονται. Στο Σχήμα 4.8 μπορούμε να δούμε πως ο πίνακας αλλάζει κατά την εκτέλεση του LCSS.

Στην περίπτωση του RSS μετατρέπουμε τα πολυδιάστατα σημεία από AP σε σημεία μιας διάστασης. Με την μετατροπή αυτή μπορούμε να χρησιμοποιήσουμε τον αλγόριθμο LCSS όπως περιγράφηκε και επεξηγήθηκε πιο πάνω.

4.2.3 Δομή αρχείου XML για τα σφάλματα

Έχουμε υλοποιήσει με XML ένα αρχείο που υποδηλώνει των αριθμών των σφάλματα (Error code). Έχουμε επιλέξει την XML αφού προσφέρει την αυτόεπεξηγηματική περιγραφή των δεδομένων καθώς και την εύκολη επέκταση των κωδίκων των σφαλμάτων με απλώς την εισαγωγή νέων σφαλμάτων στο αρχείο XML που βρίσκεται στον πελάτη. Μπορούμε να το δούμε ένα παράδειγμα για το πώς μπορούμε να κατασκευάσουμε το συγκεκριμένο αρχείο στο Σχήμα 4.9.

```
    <error>
        <number>512</number>
        <description>None Client is calculating</description>
    </error>
    <error>
        <number>513</number>
        <description>You are the only online user. The system requires at least 2 connections to operate</desc
    </error>
    <error>
        <number>514</number>
        <description>Wrong format of UpperBound</description>
    </error>
        <error>
        <number>515</number>
        <description>Wrong format of UpperBound</description>
    </error>
</errors>
```

Σχήμα 4.9

4.3 GUI SmartTrace

Λόγω του ότι η εφαρμογή έχει αναπτυχτεί σε έξυπνες συσκευές με λειτουργικό σύστημα Android, έπρεπε να δημιουργηθεί μια διεπαφή έτσι ώστε ο χρήστης να έχει την δυνατότητα να μπορεί εύκολα να αλλάξει δεδομένα και παραμέτρους για την σωστή λειτούργει του πρωτοκόλλου SmartTrace. Σε αυτό το υποκεφάλαιο θα αναλύσουμε όλες τις πτυχές της διεπαφής. Η διεπαφή έχει αχολογάει και επανασχεδιαστεί πάρα πολλές φορές αφού έπρεπε να ήταν απολυτά λειτουργική και εύκολη για ένα νέο χρηστή.

Σχήμα 4.10 : Εφαρμογή στην

Σχήμα 4.11 : Μενού της

Σχήμα 4.12 : Επιλογή για τη

πρώτη κατάσταση της. εφαρμογής. ιδιωτικότητα των δεδομένων

Η εφαρμογή έχει δυο βασικές πτυχές. Η λειτουργιά της εφαρμογής σε εξωτερικό χώρο χρησιμοποιώντας GPS συντεταγμένες (βλέπε Σχήμα 4.13) και σε εσωτερικό χώρο χρησιμοποιώντας Wi-Fi αντίστοιχα. Ο χρήστης μπορεί να μεταβεί από τη λειτουργία της εφαρμογής σε εξωτερικό χώρο σε λειτουργία εσωτερικού χώρου με το πάτημα ενός κουμπιού.

Στην περίπτωση του κομματιού της εφαρμογής για εξωτερικό χώρο χρησιμοποιούμε Google Maps για να μπορέσουμε να δώσουμε μια ρεαλιστική εντύπωση στον χρηστή για το ότι η εφαρμογή λειτουργεί πραγματικά. Επίσης ο χρήστης μπορεί να ζωγραφίσει την τροχιά του μέσω ενός αρχείου η λαμβάνοντας σημεία από το GPS.

Στο Σχήμα 4.13 φαίνονται μερικές από τις δυνατότητες τις εφαρμογής υπάρχουν πέντε κουμπιά, το πρώτο αριστερά με το σχήμα του σπιτιού είναι το κουμπί που είναι υπεύθυνο για να σχεδιάζει την τροχεία αλλά και καθοδήγα τον χρηστή στην αρχή ή στο τέλος της . Το αμέσως επόμενο κουμπί καθορίζει το μέγεθος της τροχιάς που θα εμφανίζεται στον χάρτη. Διπλά του βρίσκεται το κουμπί που καθορίζει το χρώμα της τροχεία. Σε περίπτωση οπού ο χρήστης θέλει να αναγνωρίζει την αρχή και το τέλος της τροχιάς τούτο μπορεί να χρησιμοποιήσει το δεύτερο κουμπί από τα δεξιά το οποίο τοποθετεί δυο καρφίτσες σε διάφορα σχήματα στην αρχή και στο τέλος της τροχιάς.

Στο Σχήμα 4.11 φαίνεται το μενού το οποίο επιτρέπει στον χρηστή να κάνει διάφορες λειτουργιές. Στο μενού με το πρώτο κουμπί πάνω αριστερά ο πελάτης μπορεί να κάνει καταγραφή των σημείων που παίρνει μέσω του GPS. Το διπλανό του κουμπί είναι υπεύθυνο για την σύνδεση/αποσύνδεση του πελάτη στον εξυπηρετητή. Σε περίπτωση που ο πελάτης είναι συνδεδεμένος με τον εξυπηρετητή το κουμπί έχει κόκκινο χρώμα, εάν το πατήσει ο χρήστης γίνετε μπλε και ο χρήστης αποσυνδέεται από τον εξυπηρετητή. Το πάνω δεξιά κουμπί με το σχήμα του κεραυνού είναι υπεύθυνο να ξεκινήσει την επερώτηση σύμφωνα με το πρωτόκολλο SmartTrace που έχουμε προαναφέρει. Το κουμπί “Share” ανοίγει ένα μενού προτιμήσεων στο οποίο μπορείς να θέσεις το όνομα που θα εμφανίζεται στην όλη επεξεργασία των συναλλαγών του πρωτοκόλλου. Επίσης μπορείς να επιλέξεις αν θέλεις να μοιραστείς την τροχεία σου (βλέπε Σχήμα 4.12).

Σχήμα 4.13 : Online Επιλογές.

Σχήμα 4.14 : Offline Επιλογές.

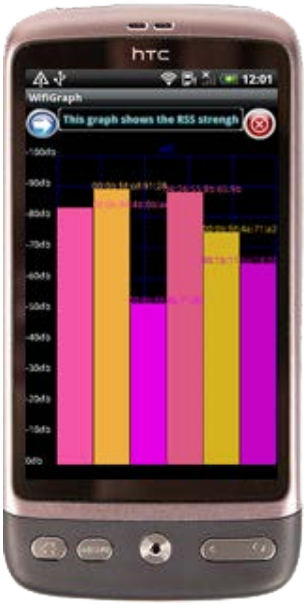
Σχήμα 4.15 : Περιηγητής αρχείων.

Σχήμα 4.16 : Γενικές πληροφορίες

Όταν ο χρήστης πατήσει το μεσαίο κουμπί στο κάτω μέρος του μενού (βλέπε Σχήμα 4.11) τότε ανοίγει το μενού επιλογών όπως φαίνεται στο Σχήμα 4.12 και Σχήμα 4.13. Υπάρχει πλήθος επιλογών γι'αυτό το λόγω είναι χωρισμένα σε κατηγορίες όπως έχουμε αναφέρει στην εισαγωγή ο πελάτης έχει την δυνατότητα να είναι σε online mode δηλαδή δυναμικά να μαζεύει συντεταγμένες και να τις καταγράφει σε ένα αρχείο δικής του επιλογής. Επίσης υπάρχει η επιλογή για το κάθε ποσό να λαμβάνει σήματα από GPS. Η δεύτερη κατηγορία είναι σε offline mode υπάρχει η επιλογή να διαλέξει ο χρήστης ένα αρχείο το οποίο έχει την σωστή δομή έτσι ώστε να μπορεί να αναγνωστεί σωστά από το πρόγραμμα. Για να μπορεί ο χρήστης εύκολα να διαλέξει ένα αρχείο που θέλει μπορεί να ανοίξει περιηγητή αρχείων που έχουμε υλοποιήσει στα πλαίσια της εφαρμογής.

Το δεξιά κουμπί στο κάτω μέρος του μενού μεταφέρει τον χρηστή σε μια νέα οθόνη οπού μπορεί να δει όλες τις συναλλαγές που έχει κάνει σε ένα Session. Επίσης υπάρχουν επιλογές για αλλαγή του mode του χάρτη όπως και για την αρχικοποιήσει του. Επίσης υπάρχουν περαιτέρω πληροφορίες για όλη την εφαρμογή.

Σχήμα 4.17 : Επιλογή μετάβασης σε RSS λειτουργιάς.



Σχήμα 4.18 : Λειτουργιά RSS.

Σχήμα 4.19 : Αποτέλεσμα επερώτησης RSS.

Στο Σχήμα 4.17 φαίνεται το ποσό εύκολα μπορεί κάποιος να αλλάξει από GPS κατάσταση σε Wi-Fi κατάσταση. Ο πελάτης μπορεί να πατήσει το βέλος στο κέντρο της οθόνης και έτσι να μπορέσει να αλλάξει κατάσταση αντίστοιχα στην νέα οθόνη του Wi-Fi (βλέπε Σχήμα 4.18). Λόγω του ότι ο χρήστης δεν μπορεί να ζωγραφίσει την τροχεία του το μήνυμα που θα εμφανιστεί θα περιέχει μόνο την βαθμολογία και το όνομα του ομοιότερης τροχιάς.

Κεφάλαιο 5

Πειραματική Μεθοδολογία

5.1 Πειραματική Υποδομή

Σε αυτό το κεφάλαιο θα αναφέρουμε την όλη υποδομή που έχει γίνει για να μπορούμε να τρέξουμε κάποια πειράματα. Στόχος είναι να επικυρώσουμε την διαισθητική και θεωρητική ανάλυση, οπού το SmartTrace είναι το πιο αποδοτικό και οικονομικό σε κατανάλωση ενέργειας καθώς και χρόνου απόκρισης. Σε αυτό το κεφαλαίο θα δείξουμε πως έγινε η σύγκριση μεταξύ της κεντρικοποιημένης, αποκεντρικοποιημένη και SmartTrace προσέγγισης.

Λόγω του ότι τα πειράματα πρέπει να γίνονται χωρίς να αλλάζει η κύρια δομή του πρωτοκόλλου SmartTrace, έχουμε υλοποιήσει ακόμα δυο εξυπηρετητές ένα για την κεντρικοποιημένη και ένα για την αποκεντρικοποιημένη περίπτωση που ακλουθούν το πρωτόκολλο με κάποιες τροποποιήσεις.

Σχήμα 5.22 : Παράδειγμα εκτέλεσης PowerTutor

Για να μπορέσουμε να πάρουμε της αναγκαίες μετρήσεις για την αξιολόγηση των αποτελεσμάτων χρησιμοποιήσαμε την εφαρμογή Power Tutor [30]. Η συγκεκριμένη εφαρμογή μας επιτρέπει να δούμε πόση ενέργεια σπαταλάτε και που (βλέπε Σχήμα 5.).

5.2 GUI πειραματικής εφαρμογής

Σχήμα 5.20 : Επιλογές πειραματικής εφαρμογής.

Σχήμα 5.21 : Επιλογή αλγορίθμου.

Έχουμε δημιουργήσει μια διεπαφή για να μπορούμε εύκολα να αλλάζουμε τις παραμέτρους κάθε πειραματικής μελέτης. Όπως φαίνεται στο Σχήμα 5.20 και Σχήμα 5.21 ο αναλυτής-ερευνητής μπορεί να επιλέξει την διεύθυνση (ip) και την θύρα (port) οπού ο συγκεκριμένος πελάτης θα συνδεθεί. Επίσης υπάρχει η παράμετρος k την οποία έχουμε προαναφέρει στην επεξήγηση του αλγορίθμου και πρωτοκόλλου SmartTrace. Διπλά από την παράμετρο k υπάρχει ένα πεδίο στο οποίο μπορείς να διαλέξεις ποσά νήματα (threads) θα δημιουργηθούν, για να δημιουργηθούν οι εικονικοί πελάτες για την πειραματική μελέτη. Ακόμη υπάρχουν τρία(3) πεδία που αντιστοιχούν σε τρία αρχεία. Το πρώτο αντιστοιχείται στο αρχείο που καταγράφει όλες τις πληροφορίες (log file) που σχετίζονται με την εφαρμογή και την σύνδεση στον εξυπηρετητή. Το επόμενο είναι το αρχείο έχει ήδη μια καταγεγραμμένη τροχεία. Έτσι οι πελάτες που θα δημιουργηθούν μπορούν να διαβάσουν το αρχείο και να φορτώσουν την συγκεκριμένη τροχεία. Το επόμενο πεδίο έχει ακριβώς ρολό με το προηγούμενο με την μόνη διάφορα ότι ο πελάτης που θα κάνει την επερώτηση θα διαβάσει το αρχείο. Τα δυο κουμπιά στο κάτω μέρος έχουν την ευθύνη να αρχικοποιήσουν και να ξεκινήσουν τους εικονικούς πελάτες που προαναφέραμε.

Σημαντικό είναι να αναφέρουμε ότι υπάρχει μια λίστα (βλέπε Σχήμα 5.21) οπού μπορούμε να επιλέξουμε τον αλγόριθμο που θα θέλαμε να αναλύσουμε. Βασική προϋπόθεση είναι ότι οι εξυπηρετητές που έχουμε αναφέρει στην αρχή αυτού του κεφαλαίου τρέχουν σε κάποιο μηχάνημα. Κάθε αλγόριθμος αντικατοπτρίζει μικρές παραλλαγές του πρωτοκόλλου SmartTrace και χρειάζεται ο αντίστοιχος εξυπηρετητής που μπορεί να συνδεθεί και να επικοινωνήσει παίρνοντας από όλες τις φάσεις του πρωτοκόλλου.

5.2.1 Οι Συσκευές που Χρησιμοποιήθηκαν

Έχουμε κάνει την πειραματική μελέτη με πέντε (5) κινητά τηλεφωνά HTC Desire. Το κάθε έξυπνο τηλέφωνο μπορεί να υποστηρίξει τουλάχιστον είκοσι (20) παράλληλα νήματα. Έτσι μπορούμε να δημιουργήσουμε είκοσι (20) ιδεατούς πελάτες σε κάθε συσκευή.

HTC DESIRE	
	***Storage:*** **Internal phone storage:** 4 GB **RAM:** 768 MB **Expansion slot:** microSD™ memory card (SD 2.0 compatible) ***CPU Processing Speed:*** 1 GHz ***Platform:*** Android™ 2.2 (Froyo) with HTC Sense™ ***Wi-Fi®:*** IEEE 802.11 b/g/n
Ubuntu Server	
	***Storage:*** **RAM:** 4 GB ***CPUs number:*** 2 ***CPU:*** Intel(R) Xeon(R) CPU E5620 @ 2.40GHz ***Cache Size:*** 12288 KB ***OS:*** Ubuntu 11.04 Server Edition

Σχήμα 5.22 : Συσκευές που χρησιμοποιήθηκαν για την πειραματική μελέτη.

5.2.2 Περιγραφή των Datasets

Χρησιμοποιήθηκαν δυο μεγάλα και γνωστά σύνολα δεδομένων. Το πρώτο ονομάζεται GeoLife GPS Trajectories. Πρόκειται για ένα σύνολο δεδομένων από τροχιές GPS που

συλλέγονται (Microsoft Research Asia) στα πλαίσια του σχεδίου GeoLife με 165 χρήστες σε μια περίοδο άνω των δύο ετών (από το Απρίλιος 2007 - Αύγουστος 2009). Αυτό το σύνολο δεδομένων είναι καταγραμμένο από ένα ευρύ φάσμα από υπαίθριες κινήσεις των χρηστών, καθώς όχι μόνο η ζωή ρουτίνες όπως πάω σπίτι και να πάνε στην εργασία τους αλλά και κάποια ψυχαγωγία και αθλητικές δραστηριότητες, όπως τα ψώνια, τα αξιοθέατα, τραπεζαρία, πεζοπορία και ποδηλασία. Ως εκ τούτου, το σύνολο δεδομένων μπορεί να χρησιμοποιηθεί σε πολλούς ερευνητικούς τομείς, όπως η εξόρυξη πρότυπο της κινητικότητας, των χρηστών της αναγνώρισης δραστηριότητας, location-based κοινωνικά δίκτυα, και τη σύσταση θέσης. Παρακαλούμε να αναφερθούν τα ακόλουθα δύο έγγραφα όταν χρησιμοποιούν αυτό το σύνολο δεδομένων GPS.[8],[12]. Το άλλο ονομάζεται Oldenburg και είναι ένα ρεαλιστικό σύνολο δεδομένων που περιλαμβάνει 2.000 τροχιές αυτοκινήτων κινούνται στην πόλη του Oldenburg[40]. Το μέσο μήκος κάθε τροχιάς είναι 11, 731 ± 7, 193 σημεία, ενώ το μέγιστο μήκος τροχιάς είναι 42.500 σημεία. Κάθε ερώτημα για το παραπάνω σύνολο δεδομένων που παράγεται με την προσθήκη θορύβου μέσω της παρεμβολής Gaussian σε ένα τμήμα μιας τυχαίας επιλεγμένης τροχιάς στο σύνολο δεδομένων. Αυτό δημιούργησε αποκλίσεις στο σχήμα (σημεία) της επερωτώμενης τροχιάς σε σύγκριση με την αρχική τροχιά.

5.3 Πειραματική Διαδικασία

Πρώτα έχουμε γράψει τον απαραίτητο κώδικα για τους τρεις (3) εξυπηρετητές και πελάτες που μπορούν να επιλεχτούν μέσω της διεπαφής που έχουμε αναφέρει πιο πάνω. Έχουμε ελέγξει πολλές επαναλαμβανόμενες φορές τον κώδικα για την ορθότητα και την εκτέλεση του πάνω στις συσκευές. Επίσης έχουμε ελέγξει όλες της φάσεις του πρωτοκόλλου έτσι ώστε να μην έχουμε περιπτώσεις αδιέξοδου. Όσον αφορά τους εξυπηρετητές ελέγξαμε τον συγχρονισμό για την παράλληλη εκτέλεση των νημάτων που δημιουργούνται στους εξυπηρετητές.

Ο χρόνος απόκρισης αποθηκεύεται σε ένα αρχείο που υπάρχει στον φάκελο της κάθε περίπτωσης. Ο χρόνος υπολογίζεται από την στιγμή που κάποιος κάνει κάποια επερώτηση και τελειώνει μόνο όταν απαντηθεί η επερώτηση. Τα πειράματα γίνονται επαναληπτικά

Η ενέργεια μετρήθηκε με την βοήθεια του εργαλείου power tutor το οποίο δημιουργεί ένα αρχείο που συσχετίζει την κάθε διεργασία/δραστηριότητα (activity) με το ποσό ενέργειας για CPU, OLED, WIFI, GPS κ.τ.λ.

Κεφάλαιο 6

Πειραματικά Αποτελέσματα

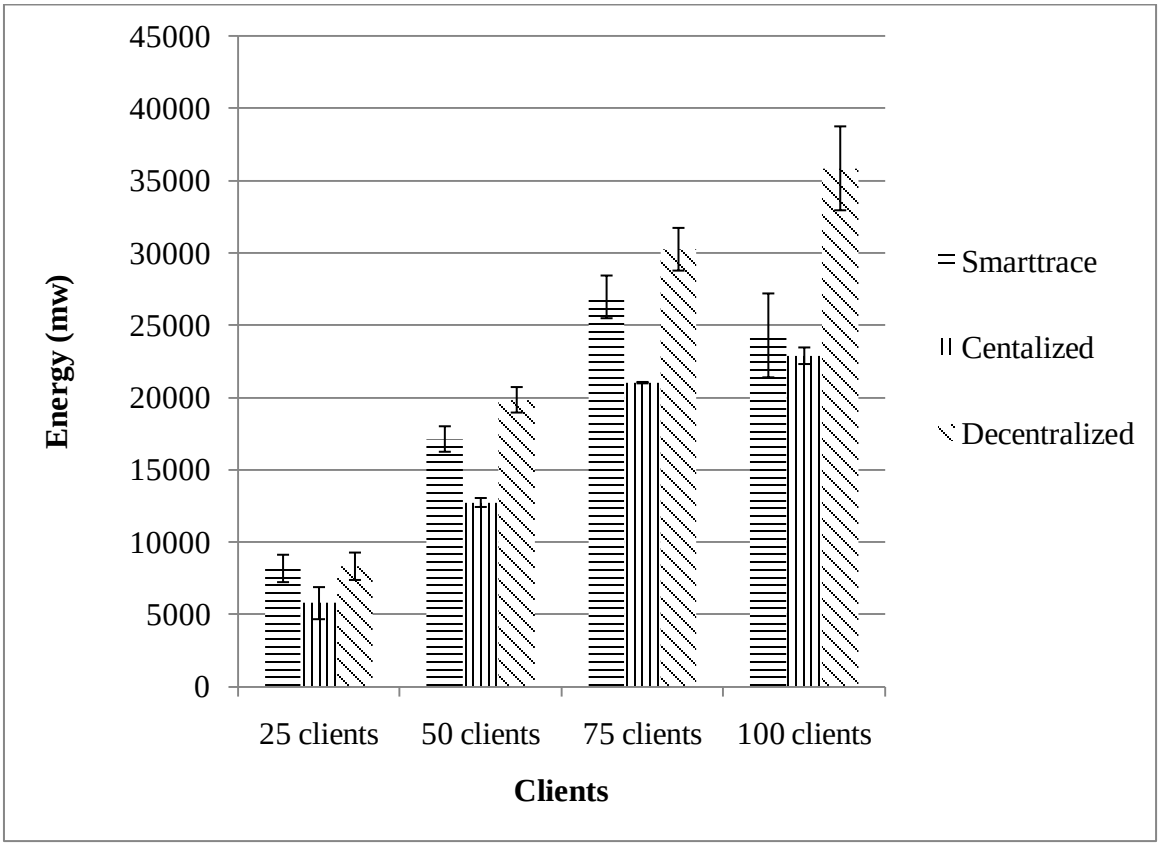
6.1 Εισαγωγή

Όπως έχουμε προαναφέρει έχουμε κάνει τα πειράματα που ακλουθούν για να ανακαλύψουμε αν όντως το πλαίσιο SmartTrace είναι πιο αποδοτικό σε ενέργεια καθώς και χρόνο απόκρισης από άλλους αλγορίθμου σύγκρισης χώρο - χρονικών δεδομένων. Η πειραματική μελέτη έγινε πάνω σε (5) πέντε έξυπνα τηλεφωνά HTC Desire με πολλαπλά νήματα που αντίστοιχο αριθμό από πελάτες.

Η συγκρίσεις των αλγορίθμων έχουν γίνει με διάφορα μεγέθη τροχείων για να μπορέσουμε να διαπιστώσουμε ποτέ και πως ο αλγόριθμος έχει την απόδοση που διαισθητικά και θεωρητικά υπολογίζουμε να έχει. Όπως έχουμε αναφέρει πιο πάνω σε ένα πιο αφαιρετικό επίπεδο μπορούμε να προβλέψουμε ότι ο αλγόριθμος SmartTrace σίγουρα θα είναι πολύ καλύτερος από το Decentralized αλγόριθμο σε κάθε περίπτωση ενώ για τον αλγόριθμο Centralized υπάρχουν περιπτώσεις (π.χ. Όταν η τροχεία είναι μικρή) που θα είναι ελαφρά καλύτερος.

6.2 Πειραματικά Αποτέλεσμα για ενέργεια

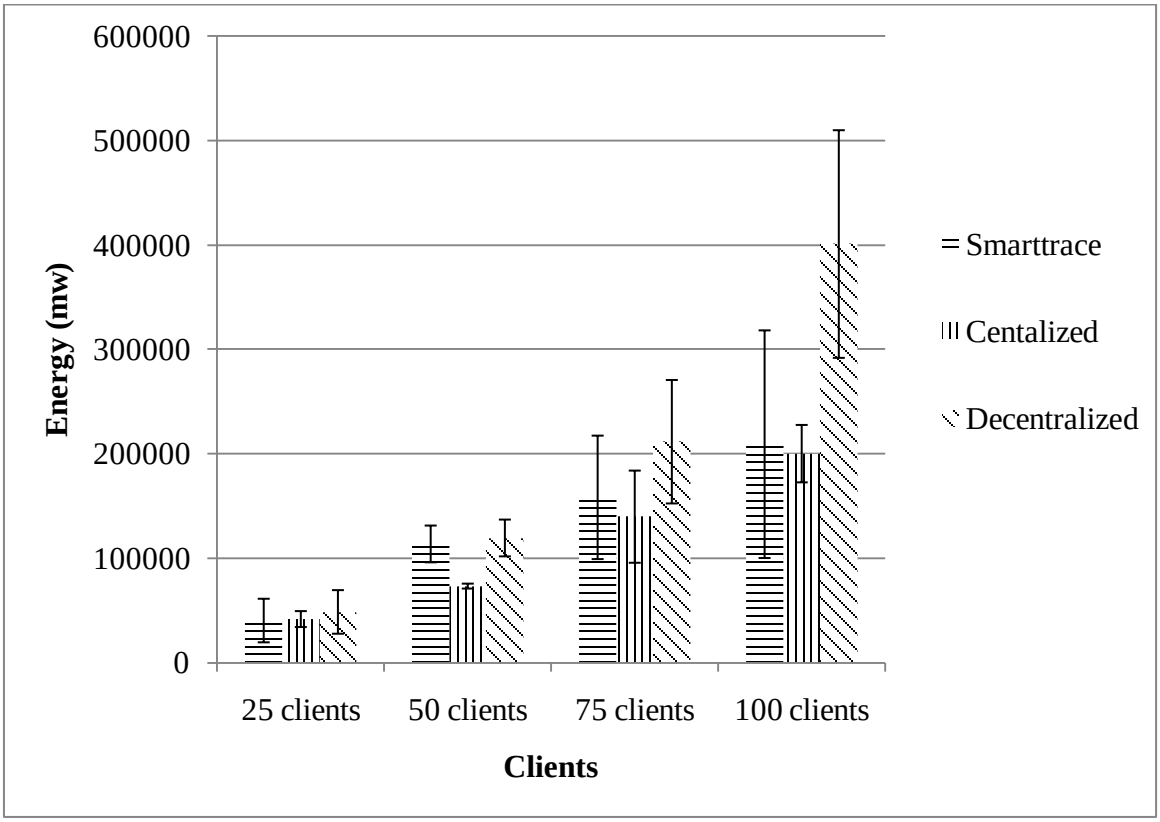
6.2.1 Τροχιά με 100 γεωγραφικά σημεία, δ=max(|A|,|B|), e=0.001, k=1



Σχήμα 6.1

Στην πιο πάνω γραφική παράσταση (Σχήμα 6.1) μπορούμε να δούμε την σχέση του κάθε αλγορίθμου με την ενέργεια που καταναλώνει ο πελάτης. Για να έχουμε πιο ακριβής αποτελέσματα έχουμε επαναλάβει το πείραμα τρεις (3) φορές και έχουμε υπολογίσει την τυπική απόκλιση.

6.2.2 Τροχιά με 500 γεωγραφικά σημεία, δ=max(|A|,|B|), e=0.001, k=1

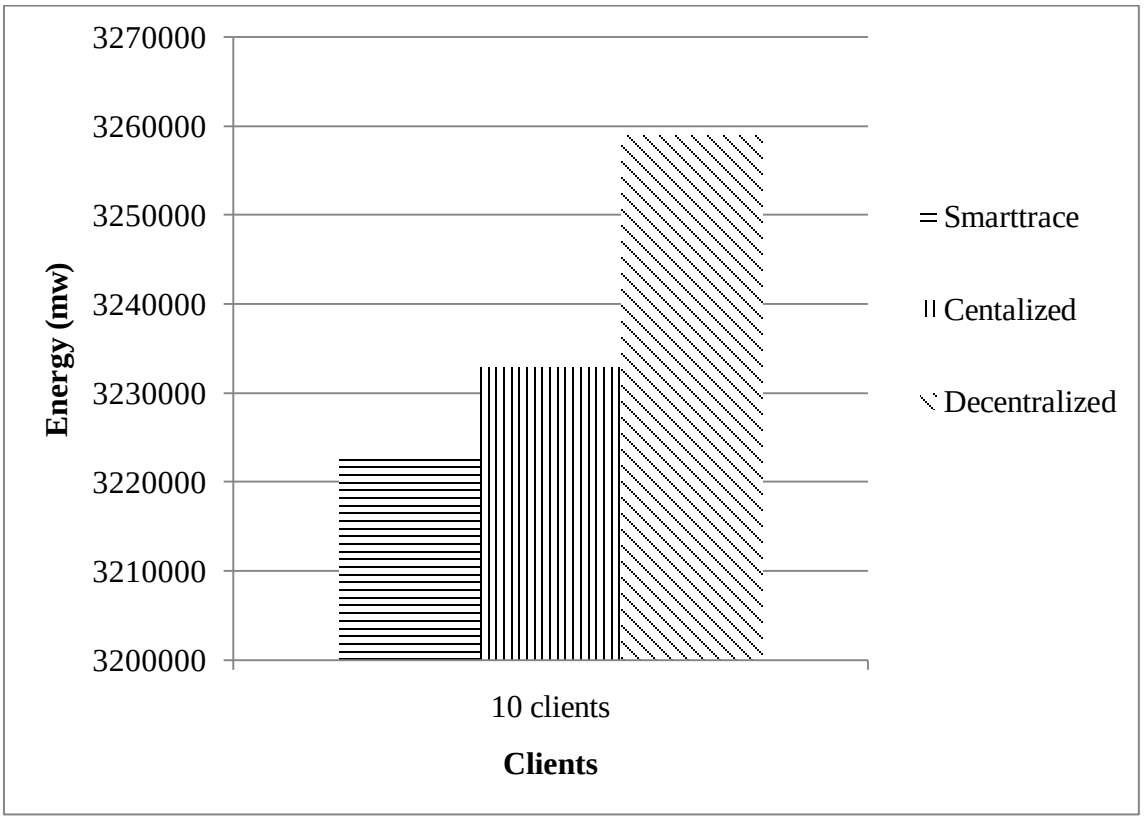


Σχήμα 6.2

Στην πιο πάνω γραφική παράσταση (Σχήμα 6.2) μπορούμε να δούμε την σχέση του κάθε αλγορίθμου με την ενέργεια που καταναλώνει ο πελάτης. Για να έχουμε πιο ακριβής αποτελέσματα έχουμε επαναλάβει το πείραμα πέντε (5) φορές και έχουμε υπολογίσει την τυπική απόκλιση.

Από τις δυο γραφικές (βλέπε Σχήμα 6.1 και Σχήμα 6.2) μπορούμε να είμαστε σίγουροι ότι ο αλγόριθμος SmartTrace είναι πολύ καλύτερος από τον αποκεντρωμένο αλγόριθμο. Αυτό οφείλεται στο ότι στον αποκεντρωμένο αλγόριθμο όλες οι έξυπνες κινητές συσκευές πρέπει να υπολογίσουν τον δύσκολο υπολογιστικά αλγόριθμο, έτσι καταναλώνουν ενέργεια λόγω επεξεργασίας (CPU energy). Ο κεντρικοποιημένος αλγόριθμος φαίνεται να έχει καλύτερα αποτελέσματα και από του δυο αυτό συμβαίνει λόγω του ότι η τροχιά είναι πολύ μικρή έτσι το κόστος μεταφοράς σε ενέργεια είναι μικρότερο από το κόστος υπολογισμού.

6.2.1 Τροχιά με 10000 γεωγραφικά σημεία, δ=300, e=0.001, k=1

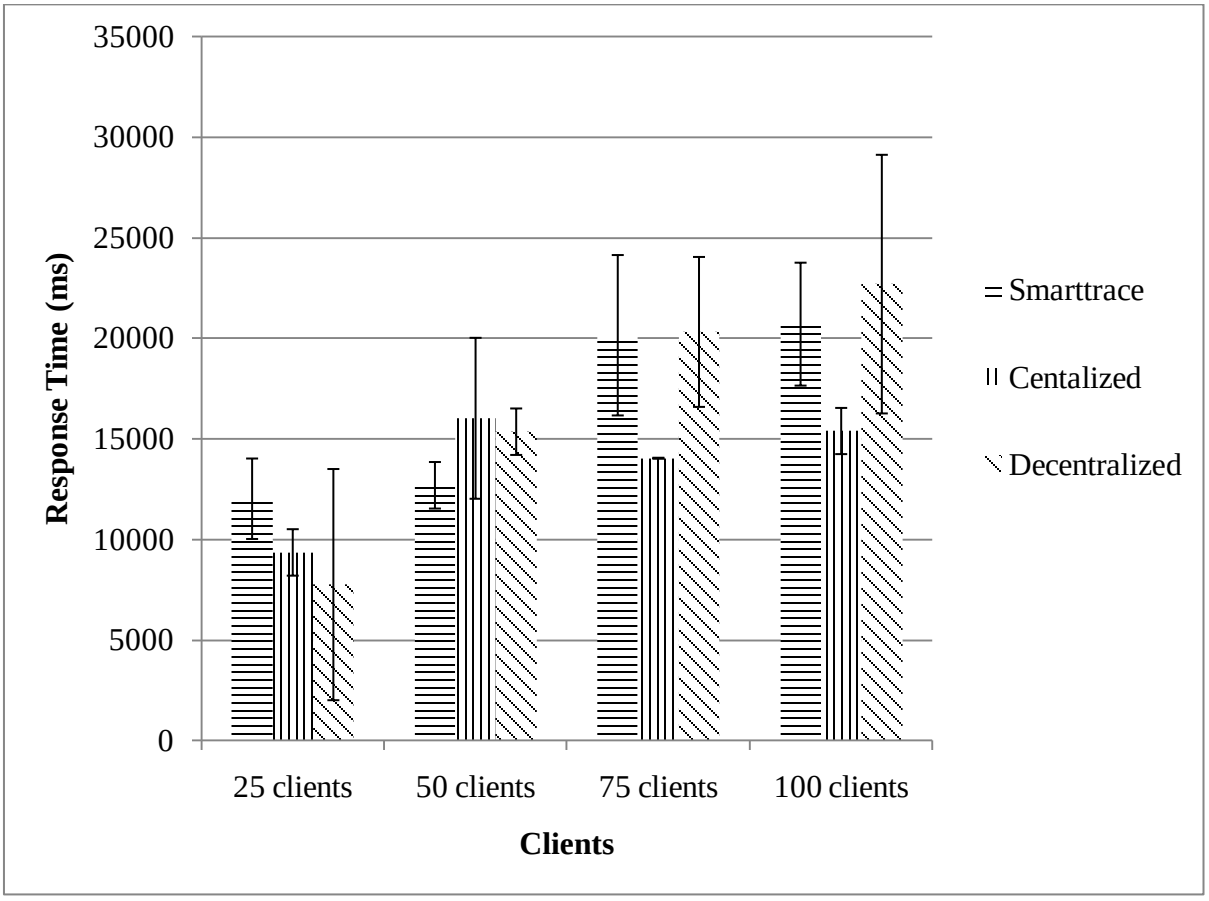


Σχήμα 6.3

Στο Σχήμα 6.3 βλέπουμε ότι όντως ο αλγόριθμος SmartTrace συμπεριφέρεται πολύ καλύτερα από τους δυο κλασικούς αλγορίθμους, κεντρικοποιημένο και αποκεντρικοποιημένο ως προς την ενεργεία. Άρα ο τελικός αλγόριθμος που έχουμε αναπτύξει δουλεύει πολύ καλά και πιο αποδοτικά αν η τροχιά ή τα δεδομένα που αποστέλλονται έχουν αρκετά μεγάλο μέγεθος. Αυτό συμβαίνει διότι η επεξεργασία που γίνεται στο κινητό πρέπει να είναι μικρότερη από το κόστος μεταφοράς για να μπορέσει ο αλγόριθμος να δείξει την ωφελιμότητα του.

6.3 Πειραματικά Αποτέλεσμα για χρόνο απόκρισης

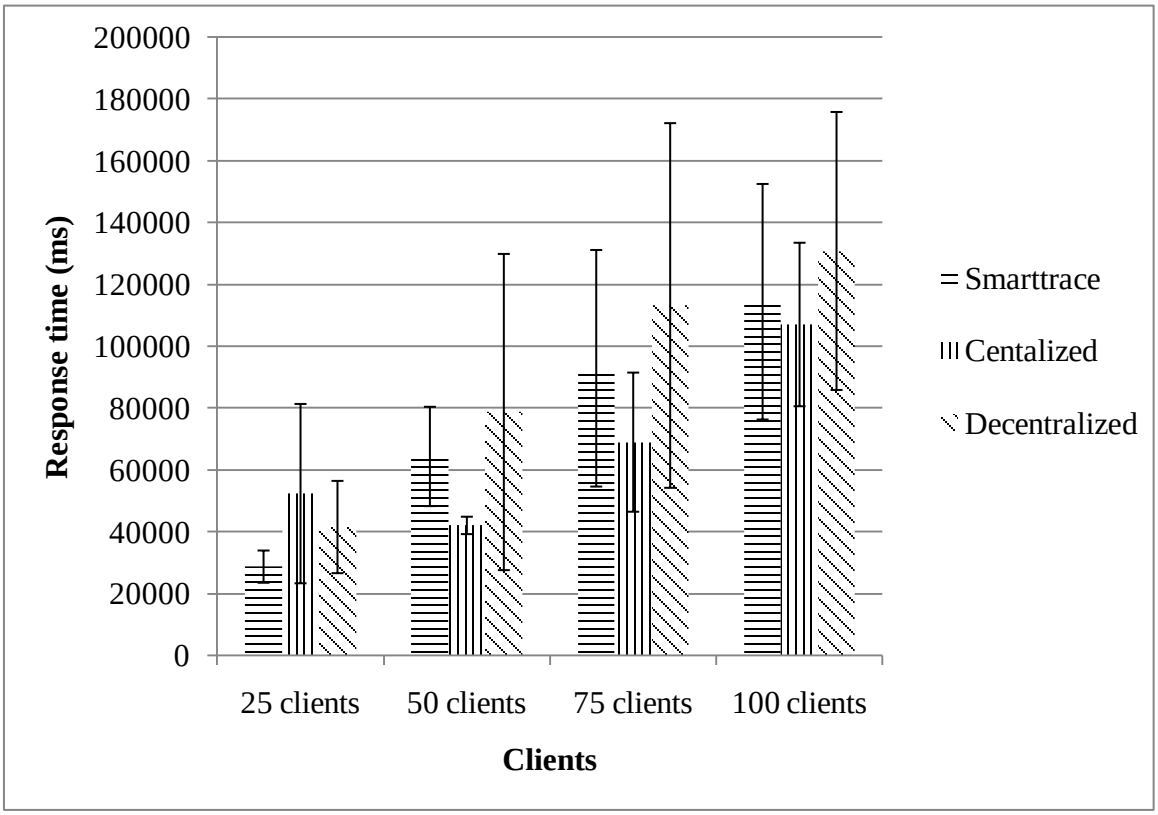
6.3.1 Τροχιά με 100 γεωγραφικά σημεία, δ=max(|A|,|B|), e=0.001, k=1



Σχήμα 6.4

Στην πιο πάνω γραφική παράσταση (Σχήμα 6.4) μπορούμε να δούμε την σχέση του κάθε αλγορίθμου με την ενέργεια που καταναλώνει ο πελάτης. Για να έχουμε πιο ακριβής αποτελέσματα έχουμε επαναλάβει το πείραμα τρεις (3) φορές και έχουμε υπολογίσει την τυπική απόκλιση.

6.3.2 Τροχιά με 500 γεωγραφικά σημεία, δ=max(|A|,|B|), e=0.001, k=1

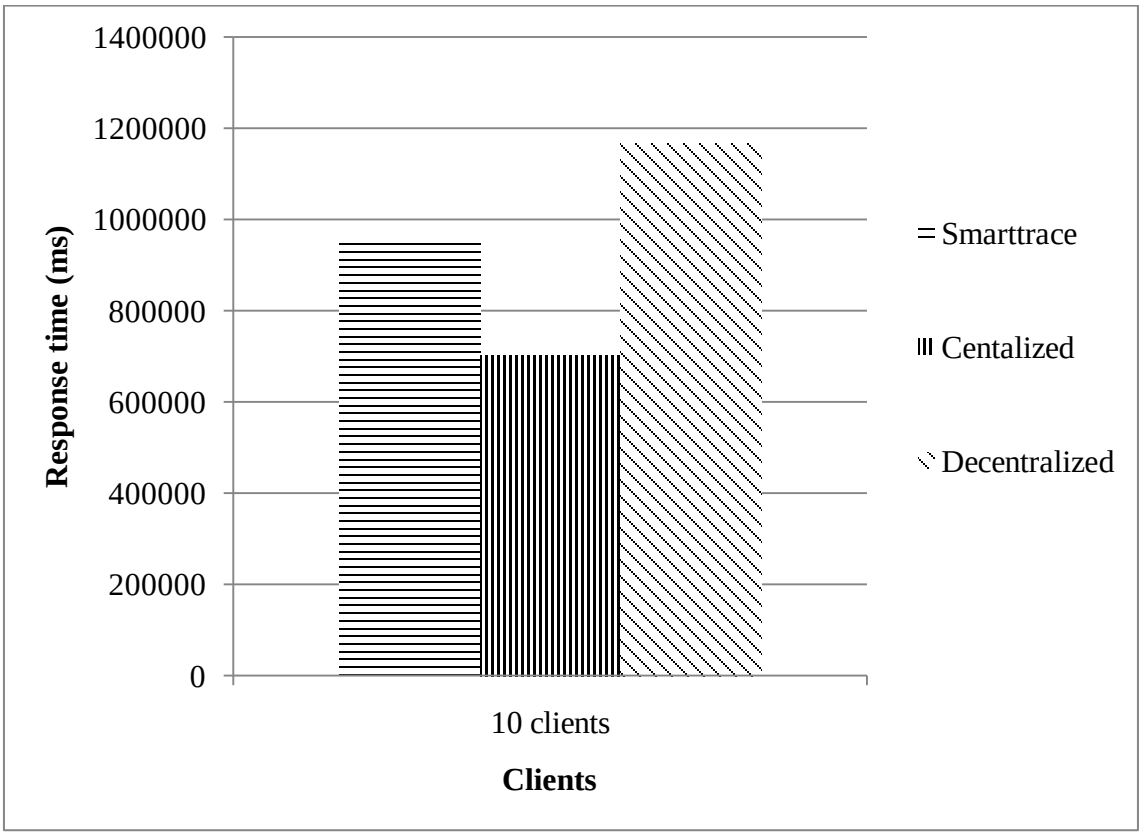


Σχήμα 6.5

Στην πιο πάνω γραφική παράσταση (Σχήμα 6.5) μπορούμε να δούμε την σχέση του κάθε αλγορίθμου με την ενέργεια που καταναλώνει ο πελάτης. Για να έχουμε πιο ακριβής αποτελέσματα έχουμε επαναλάβει το πείραμα πέντε (5) φορές και έχουμε υπολογίσει την τυπική απόκλιση.

Από τις δυο γραφικές (βλέπε Σχήμα 6.4 και Σχήμα 6.5 Σχήμα 6.2) μπορούμε να είμαστε σίγουροι ότι ο αλγόριθμος SmartTrace είναι πολύ καλύτερος από τον αποκεντρωμένο αλγόριθμο. Αυτό οφείλεται στο ότι στον αποκεντρωμένο αλγόριθμο όλες οι έξυπνες κινητές συσκευές πρέπει να υπολογίσουν τον δύσκολο υπολογιστικά αλγόριθμο, έτσι καταναλώνουν το χρόνο λόγω επεξεργασίας (CPU time). Ο κεντρικοποιημένος αλγόριθμος φαίνεται να έχει καλύτερα αποτελέσματα και από του δυο αυτό συμβαίνει λόγω του ότι η τροχιά είναι πολύ μικρή έτσι το κόστος μεταφοράς είναι μικρότερο από το κόστος υπολογισμού έτσι αυτό μειώνει το χρόνο απόκρισης.

6.3.3 Τροχιά με 10000 γεωγραφικά σημεία, δ=300, e=0.001, k=1



Σχήμα 6.6

Στο Σχήμα 6.6Σχήμα 6.3 βλέπουμε ότι όντως ο αλγόριθμος SmartTrace συμπεριφέρεται πολύ καλύτερα από τον αποκεντρικοποιημένο ως προς τον χρόνο απόκρισης. Αυτό δεν συμβαίνει με το κεντρικοποιημένο αλγόριθμο αφού το πείραμα για αυτή την γραφική παράσταση έγινε μόνο με δέκα πελάτες. Έτσι ο χρόνος εκτέλεσης και μεταφοράς της τροχιά ήταν πολύ μικρός. Έτσι πρέπει να γίνουν περαιτέρω πειράματα για να εξακριβωθεί πόσοι είναι οι απαραίτητοι πελάτες που πρέπει να συμμετάσχουν στην επερώτηση για να αποδίδει ο αλγόριθμος SmartTrace.

Κεφάλαιο 7

Συμπεράσματα και Μελλοντικές Επεκτάσεις

7.1 Συμπεράσματα

Σύμφωνα με την μελέτη που έχουμε κάνει έχουμε κατάληξη το κόστος της μεταφοράς μεγάλων τροχιών είναι μεγαλύτερο από τον κατανεμημένο των αλγορίθμων που έχουμε αναπτύξει στα πλαίσια της παρούσας διπλωματικής εργασιας. Έχουμε δοκιμάσει τους κεντρικοποιημένους και αποκεντρικοποιημένους αλγορίθμους παρουσιάζοντας τα αντίστοιχα πειραματικά αποτελέσματα. Βασικά μετρικά σύγκρισης είναι η ενέργεια και ο χρόνος απόκρισης, αυτές οι μετρικές επιλέχτηκαν γιατί ένα έξυπνο κινητό τηλέφωνο πρέπει να μην καταναλώνει μεγάλα ποσοστά ενέργειας αφού οι πόροι του είναι περιορισμένοι. Επίσης για τον ίδιο λόγο δεν πρέπει να έχει μεγάλους υπολογισμούς συνεχώς. Ο χρόνος απόκρισης είναι ακόμα μια πολύ σημαντική μετρική αφού ο χρήστης του κάθε συστήματος έχει ένα όριο που αντικατοπτρίζει τον χρόνο που μπορεί να περιμένει χωρίς να λάβει την απάντηση.

Έχουμε αποδείξει μαθηματικά ότι ο αλγόριθμος με την καλύτερη απόδοση είναι ο SmartTrace με την βασική πάντα προϋπόθεση ότι οι τροχιές είναι αρκετά μεγάλες κατά προσέγγιση 100000. Φυσικά οι δυο άλλοι αλγόριθμοι πάσχουν από θέματα ιδιωτικότητας των δεδομένων. Η τοποθεσία κάποιου άτομου είναι πολύ ευαίσθητα δεδομένα και πρέπει να μην αποκαλύπτεται εύκολα και πρέπει να διατηρούνται με

κάποιο είδος μυστικότητας. Όλα αυτές τις λειτουργιές της παρέχει το πλαίσιο SmartTrace έναντι των δυο άλλων προσεγγίσεων.

7.2 Μελλοντικές Επεκτάσεις

Το πλαίσιο SmartTrace έχει υλοποιηθεί σε έξυπνα κινητά τηλεφωνά με Android. Μπορούμε να εφαρμόσουμε το πλαίσιο και πρωτόκολλο σε άλλες πλατφόρμες π.χ. Windows 7 phone, i-phone κ. τ. λ. Επίσης δεν χρειάζεται απαραίτητα να είναι κινητό τηλέφωνο αφού υπάρχει ποικιλία από πλατφόρμες που χρησιμοποιών Wi-Fi, GPS και άλλες τεχνολογίες που χρειάζονται για να μπορέσει το πρωτόκολλο να έχει την ανταλλαγή των μηνυμάτων που είναι απαραίτητη.

Μια πολύ σοβαρή αλλαγή που μπορεί να γίνει στο όλο πλαίσιο είναι η μετατροπή του πρωτοκόλλου από μορφή κείμενου σε bytecode πρωτόκολλο έτσι η απόδοση του πλαισίου SmartTrace αυξάνεται λόγο του το κόστος μεταφοράς μειώνεται και .

Ανάπτυξη μιας τελικής εφαρμογής για τον αλγόριθμό μας και να ανάπτυξη του εκτελέσιμου APK στο Google Market για να αποκτήσουμε περαιτέρω εμπειρία με αυτό. Πιθανόν επίσης να αναπτύξει έναν πελάτη για το iPhone συσκευές

Βιβλιογραφία

[1]. Tathagata Das, Prashanth Mohan, Venkata N. Padmanabhan, Ramachandran Ramjee, Asankhaya Sharma, "*PRISM: Platform for Remote Sensing using Smartphones*" In MobilSys, 2010

[2]. Martin Azizyan, Ionut Constandache, Romit Roy Choudhury, *"SurroundSense: Mobile Phone Localization via Ambience Fingerprinting"* In MibiCom'09

[3]. Andrew T. Campbell, Shane B. Eisenman, Nicholas D. Lane, Emiliano Miluzzo, Ronald A. Peterson, "*PeopleCentric Urban Sensing"* In WICON, 2006

[4]. Ting Liu, Christopher M. Sadler, Pei Zhang, "*Margaret Martonosi,Implementing Software on Resource-Constrained Mobile Sensors: Experiences with Impala and ZebraNet*" In MobilSys, 2004

[5]. Emre Sarigol, Oriana Riva, Gustavo Alonso, *"A Tuple Space for Social Networking on Mobile Phones"*

[6]. Arvind Thiagarajan, Lenin Ravindranath, Katrina LaCurts, Sivan Toledo Jakob Eriksson ,*"VTrack: Accurate, Energy-aware Road Traffic Delay Estimation Using Mobile Phones*", In SenSys, 2009

[7]. Yu Zheng, Like Liu, Longhao Wang, Xing Xie, "*Learning Transportation Mode from Raw GPS Data for Geographic Applications on the Web*", In WWW'08

[8]. Yu Zheng, Lizhu Zhang, Xing Xie, Wei-Ying Ma, *"Mining Interesting Locations and Travel Sequences* from. *GPS Trajectories*.", In WWW'09

[9]. Zeinalipour-Yazti D., Lin S., Gunopulos D., *"Distributed Spatio-Temporal Similarity Search"* In CIKM, 2006.

[10]. Chow C-Y., Mokbel M.F., Aref W.G., *"Casper*: Query Processing for Location Services without Compromising Privacy"* In ACM TODS 34(4), pp. 1-48, 2009.

[11]. Wiley Publishing, Inc.10475, Crosspoint Boulevard, Indianapolis, IN 46256 ,2009, *"Professional Android Application Development"* .978-0-470-34471-2

[12]. Yu Zheng, Quannan Li, Yukun Chen, Xing Xie. *"Understanding Mobility Based on GPS Data"*. In Proceedings of ACM conference on Ubiquitous Computing (UbiComp 2008), Seoul, Korea. ACM Press: 312–321.

[13]. Al-Aha K. , Snodgrass R., Soo M., *"Bibliography on Spatiotemporal Databases"*, In SIGMOD Record, 22(1), 59–67, 1993.

[14]. Kollios G., Gunopulos D., Tsotras V.J., Delis A., Hadjieleftheriou M., *"Indexing Animated Objects Using Spatiotemporal Access Methods,"* In TKDE 13(5), 2001.

[15]. Vlachos M., Hadjieleftheriou M., Gunopulos D., Keogh E.,*"Indexing multi-dimensional time-series with support for multiple distance measures,"* In SIGKDD, 2003.

[16]. Ni J., Ravishankar C.V. *"Indexing Spatio-Temporal Trajectories with Efficient Polynomial Approximations,"* In em TKDE 19(5), 2007.

[17]. Xia T., Zhang D., Kanoulas E., Du Y., *"On Computing Top-t Most Influential Spatial Sites,"* In VLDB, 2005.

[18]. Berndt D., Clifford J., *"Using Dynamic Time Warping to Find Patterns in Time Series,"* In KDD, 1994.

[19]. Das G., Gunopulos D., Mannila H., *"Finding Similar Time Series,"* In PKDD, 1997.

[20]. Chen L., Ozsu T., Oria V., *"Robust and Fast Similarity Search for Moving Object Trajectories,"* In SIGMOD, 2005.

[21]. Tao Y., Sun J., Papadias D., *"Analysis of Predictive Spatio-Temporal Queries,"* In ACM TODS 28(4), 2003.

[22]. Hadjieleftheriou M., Kollios G., Bakalov P., Tsotras V.J., *"Complex Spatio-Temporal Pattern Queries,"* In VLDB, 2005.

[23]. Bakalov P., Hadjieleftheriou M., Tsotras V., *"Time Relaxed Spatiotemporal Trajectory Joins,"* In GIS, 2005.

[24]. Chen L., Ng R.-T., *"On The Marriage of Lp-norms and Edit Distance,"* In VLDB, 2004.

[25]. Vieira M., Bakalov P., Tsotras V., *"On-Line Discovery of Flock Patterns in Spatio-Temporal Data,"* In GIS, 2009.

[26]. Jeung H., Lung Yiu M., Zhou X., Jensen C.S., Tao Shen H., *"Discovery of convoys in trajectory databases,"* In PVLDB 1(1), 2008.

[27]. Chen Z., Shen H-T., Zhou X., Zheng Y., Xie X. *"Searching trajectories by locations: an efficiency study,"* In SIGMOD, 2010.

[28]. Frentzos E., Gratsias K., Theodoridis Y., *"Index-based Most Similar Trajectory Search,"* In ICDE 2007.

[29]. Pfoser D., Jensen C. S., and Theodoridis, Y., Novel *"Approaches to the Indexing of Moving Object Trajectories"*, In VLDB, 2000.

[30]. PowerTutor Tool, http://powertutor.org/.

[31]. Zeinalipour-Yazti D., Lin S., Gunopulos D., *"Distributed Spatio- Temporal Similarity Search,"* In CIKM, 2006.

[32]. Fagin R., Lotem A., Naor M., "Optimal Aggregation Algorithms For Middleware," In PODS, 2001.

[33]. Bruno N., Gravano L., Marian A., "Evaluating Top-K Queries OverWeb Accessible Databases," In ICDE, 2002.

[34]. Babcock B., Olston C., "Distributed Top-K Monitoring", In SIGMOD'03

[35]. Ilyas I.F., Beskales G., and Soliman M.A., "A Survey of Top-k Query Processing Techniques in Relational Database Systems", In ACM Computing Surveys 40(4), 2008.

[36]. Cao P., Wang Z., "Efficient Top-K Query Calculation in Distribute Networks," In PODC'04, pp. 206-215, 2004.

[37]. Zeinalipour-Yazti D. et. al., "Finding the K highest-ranked answers in a distributed network," In ComNet 53(9), 1431-1449, 2009.

[38]. Guo L., Amer-Yahia S., Ramakrishnan R., Shanmugasundaram J., Srivastava U., Vee E. "Efficient top-k processing over query-dependent functions," In PVLDB 1(1) 1044-1055 (2008).

[39]. Michel S., Triantafillou P., Weikum G., "KLEE: A Framework for Distributed Top-K Query Algorithms," In VLDB, 2005.

[40]. Brinkhoff T., "A Framework for Generating Network-Based Moving Objects," In GeoInformatica, 6(2), 2002.

Παράρτημα Α

Category: Standards Track Jan 2011

SmartTrace Protocol (STP)

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Table of Contents

1. Introduction

On certain types of smaller nodes in the Internet it is often impractical to maintain a message transport system (MTS). For example, a workstation(in these case smart phones) may not have sufficient resources (cycles, disk space) in order to permit a SMTP server [RFC821] and associated local smart trace system to be kept resident and continuously running. Similarly, it may be expensive (or impossible) to keep a cell phone interconnected to an IP-style network for long amounts of time (the node is lacking the resource known as "connectivity").

Despite this, it is often very useful to be able to run a smartTrace on these smaller nodes, and they often support a user agent (UA) to aid the tasks of smartTrace handling. To solve this problem, a node which can support an MTS entity offers a smartTrace communication service to these less endowed nodes. The SmartTrace protocol is intended to permit a workstation to dynamically access a service on a server host in a useful fashion. Usually, this means that the smartTrace protocol is used to allow a workstation to retrieve mail that the server is holding for it.

For the remainder of this memo, the term "client host" refers to a host making use of the SmartTrace service, while the term "serve host" refers to a host which offers the SmartTrace service.

2. A Short Digression

This memo does not specify how a client host uses SmartTrace services into the transport system, although a method consistent with the philosophy of this memo is presented here:

When the user agent on a client host wishes to enter a message into the transport system, it establishes an SMTP connection to its relay host and sends all info for the search query to it. This relay host could be, and need to be, the SmartTrace server host for the client host. Of course, the relay host must accept any query's information and process them correctly with a first level of privacy.

3. Basic Operation

Initially, the server host starts the SmartTrace service by listening on TCP port 65000 and 65001. When a client host wishes to make use of the service, it establishes a TCP connection with the server host. When the connection is established, the SmartTrace server sends a greeting. The client and SmartTrace server then exchange commands and responses(respectively) until the connection is closed or aborted.

Commands in the STP consist of a case-insensitive keyword, possibly followed by one or more arguments. All commands are terminated by a CRLF pair. Keywords and arguments consist of printable ASCII characters. Keywords and arguments are each separated by a single SPACE character. Keywords are three or four characters long. Each argument may be up to 40 characters long.

Responses in the STP consist of a status indicator and a keyword possibly followed by additional information. All responses are terminated by a CRLF pair. Responses may be up to 512 characters long, including the terminating CRLF. There are currently two status indicators: positive ("+OK") and negative ("-ERR"). Servers MUST send the "+OK" and "-ERR" in upper case.

Responses to certain commands are multi-line. In these cases, which are clearly indicated below, after sending the first line of the response and a CRLF, any additional lines are sent, each terminated by a CRLF pair. When all lines of the response have been sent, a

final line is sent, consisting of a termination octet (decimal code 046, ".") and a CRLF pair. If any line of the multi-line response begins with the termination octet, the line is "byte-stuffed" by pre-pending the termination octet to that line of the response. Hence a multi-line response is terminated with the five octets "CRLF.CRLF". When examining a multi-line response, the client checks to see if the line begins with the termination octet. If so and if octets other than CRLF follow, the first octet of the line (the termination octet) is stripped away. If so and if CRLF immediately follows the termination character, then the response from the POP server is ended and the line containing ".CRLF" is not considered part of the multi-line response.

A STP session progresses through a number of states during its lifetime. Once the TCP connection has been opened and the STP server has sent the greeting, the session enters the ESTABLISH state. In this state, the client must identify itself to the STP server. Once the client has successfully done this, the server acquires resources associated with the client. In this state, the client requests actions on the part of the STP server. When the client has issued the QUIT command, the session enters to CLOSED

state. In this state, the STP server releases any resources acquired during the ESTABLISH state and says goodbye. The TCP connection is then closed.

A server MUST respond to an unrecognized, unimplemented, or syntactically invalid command by responding with a negative status indicator. A server MUST respond to a command issued when the session is in an incorrect state by responding with a negative status indicator. There is no general method for a client to distinguish between a server which does not implement an optional command and a server which is unwilling or unable to process the command.

A STP server MAY have an inactivity autologout timer. Such a timer MUST be of at least 10 minutes' duration. The receipt of any command from the client during that interval should suffice to reset the autologout timer. When the timer expires, the session does NOT enter the UPDATE state--the server should close the TCP connection without removing any messages or sending any response to the client.

4. The CLOSED State

All clients start in this state. This state indicates the creation of the TCP connection between the Client and the STP Server.

5. The ESTABLISH State

Once the TCP connection has been opened by a STP client, the STP server issues a one line greeting. This can be any positive response. An example might be:

S: +OK STP READY

The STP session is now in the ESTABLISH state. The client must now identify and authenticate itself to the STP server. Two possible mechanisms for doing this are described in this document, the USER and PASS command combination. Both mechanisms are described later in this document.

Once the STP server has determined through the use of any authentication command that the client should be given access to use the smartTrace framework, the STP server then maintain a persistence connection between the client and the Server. If the client gives the SEARCH command the STP session either goes in the SEARCH state or send back an error message if the Server is busy.

After returning a negative status indicator, the server may close the connection. If the server does not close the connection, the client may either issue a new authentication command and start again, or the client may issue the QUIT command. After the authentication is verified the STP server send a unique number that represent the session id.

USER <user name> <password>

Arguments:

User name that is represent the client name and the password that match the username.

Restrictions:

may only be given in the ESTABLISH state after the STP greeting.

Discussion:

To authenticate using the USER and PASS command combination, the client must first issue the USER command. If the STP server responds with a positive status indicator ("+OK"), then the client has complete the authentication phase, or the QUIT command to terminate the STP session. If the STP server responds with a negative status indicator ("-ERR") to the USER command, then the client may either issue a new authentication command or may issue the QUIT command.

Possible Responses:

+OK session id

-ERR error no

Here is the summary for the QUIT command when used in the ESTABLISH state:

QUIT

Arguments: none

Restrictions: none

Possible Responses:

+OK

Examples:

C: QUIT

S: +OK Goodbye

Here is the summary of fully transaction in ESTABLISH state:

Examples:

S: +OK STP READY

C: USER <user name> <password>

S: +OK <SUID>

6. The SEARCH State

Once the client is establish the connection and authorized, he is in ESTABLISH state. He could move to SEARCH state or CALCULATE state by Sending “SEARCH” or sending “+OK CALCULATE” message due the broadcast of the searcher, respectively. When the client is in SEARCH state he should wait for an acknowledgment from the STP Server. Then he waits for the LCSS score.

If the client want to make a query using the SmartTrace Protocol should send the query using the SEARCH command. The response time of the query is based on the process time of each client.

Here is the summary of fully transaction in SEARCH state:

Examples:

C: SEARCH K <trajectory>

S: +OK <quid>

S: <quid> <LCSS score>

Arguments:

For the SEARCH command the searcher should send how many trajectories he want to retrieve (K).

The search trajectory that it will be sends for calculations

Restrictions:

The client gets positive acknowledgement only if there is a number of clients that are connected and greater than the required number.

Discussion:

If the clients leave during the calculation the response that will send it will be negative.

Possible Responses:

+OK <quid>

-ERR <error number>

7. The CALCULATE State

When the client makes the query the STP Server to notify the connected
clients to calculate the UB score and then the LCSS score.

Here is the summary of fully transaction in CALCULATE state:

Examples:

S: CALCULATE <quid> <searcher trajectory>
C: +OK CALCULATE
C: +OK <quid> <UB score>
S: LCSS <quid>
C: +OK <quid> <LCSS score>

Arguments:

Calculate command needs the quid of the searcher and the searcher trajectory.

Restrictions:

may only be given in the CALCULATE state.

Discussion:

In this state the client just calculate values that are used by the STP Server to find the most similar trajectory. The client should use specific algorithms for the calculation so the equality among the users is balanced.

Examples:

……

C: +OK <quid> <UB score>

S: LCSS <quid>

C: +OK <quid> <LCSS score>

Restrictions:

The upper bound algorithm and LCSS algorithm should use the correct parameter (e) so everyone have equivalent calculations.

8. The RETRIEVE State

The searcher after the SEARCH state he can navigate to RETRIEVE state.

Here is the summary of fully transaction in CALCULATE state:

Examples:

C: RETRIEVE <quid>

S: +OK <Trajectory>

Arguments:

RETRIEVE command needs the quid of the searcher.

Restrictions:

may only be given in the SEARCH state.

Discussion:

In this state the client just retrieve the set of top k-trajectories that match with the query.

9. Scaling and Operational Considerations

Since the STP protocol is under developing in the future it will Definitely have much more option that could be used to an implementation of a STP server. The STP server can give information about your of similar movement in a building with RSS similarity or GPS similarity for wide uncovered area.

10. STP Command Summary

Minimal STP client Commands:

USER <name> <pass> valid in the ESTABLISH state
QUIT

SEARCH <sid> valid in the SEARCH state
RETRIEVE <qid> valid in the RETRIEVE state

STP Server Commands:

CALCULATE <trajectory> valid in the CALCULATE state

LCSS <quid> valid in the CALCULATE state

STP Replies:

+OK
-ERR

STP client Replies:

+OK
-ERR

11. Example STP Session

S:Server started on costantinos-laptop with ip:/12.34.56.78
port :65000
Please give the minimum require number of clients at bottom of the window

You gave wrong clients's number.
The default number (3) will be used

S:+OK READY
C: USER Daniel
S: +OK 6bb4d422-9a35-4fb9-925d-a1899e70d8c1
S:+OK READY
C: USER Christopher
S: +OK d4614054-552a-4664-872b-632327b4ec97
S:+OK READY
C: USER Lauren
S: +OK c1c0da2c-fbba-425c-910a-405063f853c1
C: SEARCH 00:1b:11:6a:18:0f,-66&00:0b:fd:4a:71:d2,-45&
00:0b:fd:4a:71:a2,-75&36:26:55:8b:65:9b,-77&00:0e:84:4b:0b:ff,-85&
00:0b:fd:4a:71:89,-85&00:0b:fd:f3:ab:56,-87&00:0e:84:4b:0b:c0,-87&
00:0b:fd:cd:91:28,-83&00:0b:fd:cd:91:30,-88&00:0b:fd:4a:71:d1,-89&

00:0b:fd:4a:71:95,-89&00:0e:84:4b:0b:ec,-91&00:0b:fd:4a:71:ab,-92&#&

00:1b:11:6a:18:0f,-55&00:0b:fd:4a:71:d2,-49&00:0b:fd:4a:71:a2,-74&

00:0e:84:4b:0b:ff,-80&36:26:55:8b:65:9b,-83&00:0e:84:4b:0b:dc,-89&

00:0b:fd:4a:71:b2,-89&00:0b:fd:4a:71:ce,-90&00:0b:fd:4a:71:ab,-91&

00:0b:fd:4a:71:95,-88#&

S: CALCULATE 41cf26ac-23bc-46ef-9f17-316f3ca5e4ad

S: CALCULATE 41cf26ac-23bc-46ef-9f17-316f3ca5e4ad

C: +OK CALCULATE

C: +OK CALCULATE

C: +OK quid 19.612499312202146

C: +OK quid 53.577678943323

S: 41cf26ac-23bc-46ef-9f17-316f3ca5e4ad

S: 41cf26ac-23bc-46ef-9f17-316f3ca5e4ad

C: +OK quid 82.79329291006508 00:0b:fd:cd:91:30,-88&00:0b:fd:4a:71:d1

C: +OK quid 57.02839552664549 00:0b:fd:4a:71:d2,-49&00:0b:fd:4a:71:a2

11. Message Format

All messages transmitted during a STP session are assumed to conform to the standard for the format of Internet text messages [RFC822].

It is important to note that the octet count for a message on the server host may differ from the octet count assigned to that message due to local conventions for designating end-of-line. Each point is divide by a special character (e.g '&','#') and the end of line with the CRLF character.

12. Security Considerations

It is conjectured that use of the SUID and QUID provides origin identification and replay protection for a STP session.

Further, note that as the length of the shared secret increases, so does the difficulty of deriving it.

Servers that answer -ERR to the USER command are giving potential attackers clues about which names are valid.
Use of the PASS command sends passwords in the clear over the network.

Use of the RETRIEVE,SEARCH,CALCULATE commands sends data in the clear
over the network.
Otherwise, security issues are not discussed in this memo.

Παράρτημα Β

```java
/*
 * (C) Copyright University of Cyprus. 2010-2011.
 *
 * SmartTrace API
 *
 * @version         : 1.0
 * @author : Costantinos Costa(costa.costantinos@gmail.com)
 * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
 * Computer Science Department , University of Cyprus
 *
 *
 */
package SmartTrace.apk;

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

import android.app.ListActivity;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.os.Environment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

public class AndroidFileBrowser extends ListActivity {
    // Enum For The Display Mode You Want
    private enum DISPLAYMODE {
        ABSOLUTE, RELATIVE;
    }

    private final DISPLAYMODE displayMode = DISPLAYMODE.ABSOLUTE;
    // All The Files In The Trajectory
    private List<String> directoryEntries = new ArrayList<String>();
    private File currentDirectory;
    private File homeDirectory;
    private File file;
    private Toast toast;
    static boolean flagMainOrwifi = true;

    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        // THE SAME TOAST AS THE PREVIOUS
        toast = Toast.makeText(AndroidFileBrowser.this, "", Toast.LENGTH_LONG);
        View textView = toast.getView();
        LinearLayout lay = new LinearLayout(AndroidFileBrowser.this);
        lay.setOrientation(LinearLayout.HORIZONTAL);
        ImageView view = new ImageView(AndroidFileBrowser.this);
        view.setImageResource(R.drawable.info);
        lay.addView(view);
        lay.addView(textView);
        toast.setView(lay);
        // BROWSE TO HOME DIRECTORY THAT WAS SAVE BEFORE
        browseToHome();

    }

    // This Function Browses To The Root-Directory Of The File-System.
    private void browseToHome() {
        if (flagMainOrwifi)
            currentDirectory = new File(makePath(MainActivity.custFilePath));
        else
            currentDirectory = new File(makePath(WifiGraph.custFilePath));
```



```java
        if (currentDirectory == null || isMount())
            currentDirectory = new File("/");

        homeDirectory = currentDirectory.getAbsoluteFile();
        toast.setText("You will browse from directory :\n" + homeDirectory.getAbsoluteFile());
        toast.show();
        browseTo(currentDirectory);

    }

    // check that the sdcard is mounted
    static public boolean isMount() {
        String state = Environment.getExternalStorageState();
        if (Environment.MEDIA_MOUNTED.equals(state)) {
            return true;

        }
        return false;
    }

    private String makePath(String path) {
        if (path.length() == 0) {
            return "/";
        }
        File check = new File(path);
        if (!check.exists()) {
            return "/";
        }
        if (check.isDirectory())
            return path;
        else {
            return check.getParent();
        }
    }

    // This Function Browses Up One Level According To The Field:
    // Current directory

    private void browseTo(final File aDirectory) {
        if (aDirectory.isDirectory()) {
            this.currentDirectory = aDirectory;
            fill(aDirectory.listFiles());
        } else {
            toast.setText("Browse...");
            toast.show();
        }
    }

    private void fill(File[] files) {
        this.directoryEntries.clear();
        // Add the "~" == "home directory"
        // And the ".." == 'Up one level'
        this.directoryEntries.add(getString(R.string.homeDir));
        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: // On relative Mode, we have to add the current-path to
            // the beginning
            int currentPathStringLenght = this.currentDirectory.getAbsolutePath().length();
            for (File file : files) {
                this.directoryEntries
                        .add(file.getAbsolutePath().substring(currentPathStringLenght));
            }
            break;
        }

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

```
147
148         this.setListAdapter(directoryList);
149     }
150
151     @Override
152     protected void onListItemClick(ListView l, View v, int position, long id) {
153         // TODO Auto-generated method stub
154         file = new File(directoryEntries.get(position));
155         if (file.isDirectory())
156             if (file.canRead()) {
157                 // Two Cases : If The Item Is The ".." Or "."
158                 if (file.getName().equals(getString(R.string.parentDir))) {
159                     browseTo(currentDirectory.getParentFile());
160                 } else if (file.getName().equals(getString(R.string.homeDir))) {
161                     browseTo(homeDirectory);
162                 } else
163                     browseTo(file);
164             } else {
165                 Intent i = new Intent(AndroidFileBrowser.this, Permision.class);
166                 startActivity(i);
167             }
168         else
169             Stop();
170
171         super.onListItemClick(l, v, position, id);
172
173     }
174
175     void Stop() {
176         if (flagMainOrwifi)
177             EditTextBrowser.txb.setText(file.getAbsolutePath());
178         else
179             wEditTextBrowser.txb.setText(file.getAbsolutePath());
180         flagMainOrwifi = true;
181         finish();
182     }
183
184     // Public Inner Class Which Help Me To Put Png Image For Each File That Is
185     // Different Type
186     public class MyCustomAdapter extends ArrayAdapter<String> {
187         List<String> myList;
188
189         public MyCustomAdapter(Context context, int textViewResourceId, List<String> objects) {
190
191             super(context, textViewResourceId, objects);
192             myList = objects;
193             // TODO Auto-generated constructor stub
194         }
195
196         @Override
197         public View getView(int position, View convertView, ViewGroup parent) {
198             // TODO Auto-generated method stub
199             // super.getView(position, convertView, parent);
200             View row = convertView;
201             // Check If Row Is Null
202             if (row == null) {
203                 // Make New LayoutInflater
204                 LayoutInflater vi = (LayoutInflater) getSystemService(Context.LAYOUT_INFLATER_SERVICE);
205                 row = vi.inflate(R.layout.file_row, parent, false);
206             }
207             TextView label = (TextView) row.findViewById(R.id.file);
208             String stringFile = myList.get(position).toString();
209             // Change The Symbol Of The Home Directory
210             if (stringFile.equals(getString(R.string.homeDir)))
211                 label.setText("~");
212
213             else
214                 label.setText(stringFile);
215
216             File file = new File(stringFile);
217             ImageView icon = (ImageView) row.findViewById(R.id.icon);
218             if (file.isDirectory())
219                 icon.setImageResource(R.drawable.directory);
```



```
220             else
221                 icon.setImageResource(FindDrawable(stringFile));
222             return row;
223         }
224
225         // Find The Right Icon For Each File Type
226         private int FindDrawable(String file) {
227             if (file.endsWith(".txt")) {
228                 return R.drawable.txt;
229             } else if (file.endsWith(".pdf")) {
230                 return R.drawable.pdf;
231             } else if (file.endsWith(".apk")) {
232                 return R.drawable.apk;
233             } else if (file.endsWith(".png") || file.endsWith(".gif")) {
234                 return R.drawable.png;
235             } else if (file.endsWith(".gif")) {
236                 return R.drawable.gif;
237             } else if (file.endsWith(".jpg")) {
238                 return R.drawable.jpg;
239             } else if (file.endsWith(".rar")) {
240                 return R.drawable.rar;
241             } else if (file.endsWith(".zip")) {
242                 return R.drawable.zip;
243             } else if (file.endsWith(".gz")) {
244                 return R.drawable.gz;
245             } else if (file.endsWith(".mp3") || file.endsWith(".wav") || file.endsWith(".amp")) {
246                 return R.drawable.sound;
247             } else if (file.endsWith(".mp4") || file.endsWith(".avi") || file.endsWith(".flv")) {
248                 return R.drawable.video;
249             } else
250                 return R.drawable.txt;
251         }
252
253     }
254 }
255
```

```
1  /*
2   * (C) Copyright University of Cyprus. 2010-2011.
3   *
4   * SmartTrace Server API
5   *
6   * @version         : 1.0
7   * @author : Costantinos Costa(costa.costantinos@gmail.com)
8   * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
9   * Computer Science Department , University of Cyprus
10  *
11  *
12  */
13 package cs.ucy.ac.cy;
14
15 import java.io.BufferedReader;
16 import java.io.BufferedWriter;
17 import java.io.DataOutputStream;
18 import java.io.File;
19 import java.io.FileReader;
20 import java.io.IOException;
21 import java.io.InputStream;
22 import java.io.InputStreamReader;
23 import java.io.OutputStream;
24 import java.net.Socket;
25 import java.util.ArrayList;
26 import android.content.Context;
27 import android.os.Handler;
28 import android.os.Message;
29 import android.widget.Toast;
30
31 public class CentralizeConnect extends Thread {
32
33     enum state {
34     CLOSED, ESTABLISH, CALCULATE, SEARCH, RETRIEVE;
35     }
36
37     myPoint[] trajectory = null;
38     private File inFile = null;
39     BufferedReader reader = null;
40     Toast toast;
41     private BufferedWriter writer = null;
42     private DataOutputStream out;
43     private BufferedReader in;
44     private String serverAddress = new String();
45     private String epsilon = new String();
46     private String usr = new String();
47     private Socket connection = null;
48     private String ClientTRAJ;
49     private double lcss;
50     private String SSID;
51     private String QUID;
52     private String LOG;
53     private String INFILE;
54     String host = new String();
55     String buf = new String();
56     String[] traj = null;
57     InputStream inStream = null;
58     OutputStream outStream = null;
59     private String psw = "";
60     private String LCSS;
61     private int port = 65000;
62     private String serverPort = "65000";
63     private ArrayList<myPoint> ClientTrajectory= new ArrayList<myPoint>();
64     private ArrayList<myPoint> ServerTrajectory;
65     static state curState = state.CLOSED;
66     static Handler messageHandler;
67
68     private String rmPREFIX(String msg) {
69     String[] sArr = msg.split(" ");
70     String temp = new String();
71     for (int i = 1; i < sArr.length; i++)
72         temp += sArr[i] + " ";
73     return temp;
```



```
74     }
75
76     // FUNCTION FOR THE MESSAGES
77     static public void Send(String msg, OutputStream out) throws IOException {
78     out.write((msg + Messages.CRLF).getBytes());
79     out.flush();
80     }
81
82     CentralizeConnect(Context c, boolean flag, String username, String port, String ip, String log,
       String in,
83         String e) {
84     // set username
85 //  SmartTraceThreaded.txtMsg.setText("Trying to Connect to " + ip + ":"
86 //      + port);
87 //  usr = username;
88     serverAddress = ip;
89     serverPort = port;
90     epsilon = e;
91     LOG=log;
92     INFILE=in;
93
94     // READ FROM FILE
95     readFromFile();
96     // initialized the message handler for the messages from the thread
97     messageHandler = new Handler() {
98
99         public void handleMessage(Message msg) {
100        super.handleMessage(msg);
101        Message msg1 = new Message();
102        switch (msg.arg1) {
103        // These means that the client want to use smartTrace search
104        // We create new thread so the GUI doesn't block
105        case Messages.SEARCH:
106
107            // // TODO Auto-generated method stub
108            if (!curState.equals(state.ESTABLISH)) {
109            switch (curState) {
110            case CALCULATE:
111                msg1.arg1 = Messages.STATECAL;
112                break;
113            case SEARCH:
114                msg1.arg1 = Messages.STATESER;
115                break;
116            case RETRIEVE:
117                msg1.arg1 = Messages.STATERET;
118                break;
119
120            default:
121                msg1.arg1 = Messages.SERVER_ERR;
122                break;
123            }
124
125            return;
126            }
127
128            try {
129            Send(("SEARCH " + SmartTraceThreaded.parK + " " + SSID
130                + " " + ClientTRAJ), out);
131            } catch (IOException e) {
132            // send a massage to main thread
133            SmartTraceThreaded.terminal += "Client send : SEARCH "
134                + SSID + " Client TRAJECTORY \n";
135            }
136            // write everything to the transaction string
137            synchronized (SmartTraceThreaded.terminal) {
138            SmartTraceThreaded.terminal += "Client send : SEARCH "
139                + SSID + " Client TRAJECTORY \n";
140
141            }
142
143            break;
144        case Messages.QUIT:
145
```

```
            if (connection == null) {
            // appClose();
            return;
            }
            try {
            Send("QUIT", out);
            connection.close();
            } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
            }
            synchronized (SmartTraceThreaded.terminal) {
            SmartTraceThreaded.terminal += "Client send : QUIT \n";

            }
            appClose();
            return;

        default:

            break;
        }

        }
    };
    this.setUncaughtExceptionHandler(new UncaughtExceptionHandler() {

        @Override
        public void uncaughtException(Thread thread, Throwable ex) {
        // TODO Auto-generated method stub

//      SmartTraceThreaded.txtMsg.setText(SmartTraceThreaded.terminal);
        }
    });
    }

    public void run() {

    // just connect to the server
    serverConnect();
    }

    // function for the ESTABLISH STATE
    private boolean ESTABLISH() throws IOException {
    curState = state.ESTABLISH;
    // GET THE +OK READY FROM SERVER
    buf = in.readLine();
    SmartTraceThreaded.terminal += "Server Response: " + buf + "\n";
    // Wait For Proceeding
    // Get The Session Id
    buf = "USER " + usr + " " + psw;
    // SEND A +OK REPLY and user-name TO SERVER
    Send(buf, out);
    SmartTraceThreaded.terminal += "Client Send: " + buf + "\n";
    // GET THE +OK or -ERR FROM SERVER
    buf = in.readLine();
    SmartTraceThreaded.terminal += "Server Response: " + buf + "\n";
    if (buf.startsWith("+OK")) {
        // Set SessionID
        SSID = rmPREFIX(buf);
        return true;
    }
    return false;

    }

    // Function For the SEARCH STATE
    private boolean SEARCH() throws IOException {
    curState = state.SEARCH;
    // Read The Message To Continue
    SmartTraceThreaded.terminal += "Server Response: " + buf + "\n";
    // Set SessionID
    QUID = rmPREFIX(buf);
```



```
    // Read The Message To Continue
    buf = in.readLine();
    synchronized (SmartTraceThreaded.terminal) {
        SmartTraceThreaded.terminal += "Server Response: " + buf + "\n";
    }
    if (buf.contains("Goodbye"))
        return false;
    LCSS = rmPREFIX(buf);
    return true;
    }

    // Function For the CALCULATE STATE
    private boolean CALCULATE() throws IOException {
    if (!buf.startsWith("CALCULATE"))
        return true;
    curState = state.CALCULATE;
    synchronized (SmartTraceThreaded.terminal) {
        SmartTraceThreaded.terminal += "Server Response: "
            + "CALCULATE queryTrajectory" + "\n";
    }// toast.show();
    try {
        QUID = buf.split(" ")[1];
    } catch (Exception e) {
        // TODO: handle exception
        return false;
    }
    String query = rmPREFIX(buf);
    buf = "+OK CALCULATE";
    Send(buf, out);
    synchronized (SmartTraceThreaded.terminal) {
        SmartTraceThreaded.terminal += "Client send :" + buf + "\n";
    }
    // split the points
    traj = query.split("\t");
    if (traj == null) {
        connection.close();
        return true;
    }

//  ServerTrajectory = new ArrayList<myPoint>();
//  String[] tmppnt;
//  for (int j = 0; j < traj.length; j++) {
//      tmppnt = traj[j].split(" ");
//      // synchronized (SmartTraceThreaded.terminal) {
//      // SmartTraceThreaded.terminal += traj[j] + "\n";
//      // }
//      if (tmppnt != null)
//
//      if (tmppnt.length > 1)
//          try {
//          ServerTrajectory.add(new myPoint(Double
//              .parseDouble(tmppnt[0]), Double
//              .parseDouble(tmppnt[1])));
//          ClientTrajectory.add(new myPoint(Double
//              .parseDouble(tmppnt[0]), Double
//              .parseDouble(tmppnt[1])));
//          } catch (Exception e) {
//          // TODO: handle exception
//          continue;
//          }
//  }


    // CALCULATE THE UB

    // SEND THE UB TO SERVER
    double eps = 0.1;
    try {
        eps = Double.parseDouble(epsilon);
    } catch (Exception e) {
        // TODO: handle exception
    }
//  Send the whole trajectory to the server
```

```
292     buf = "+OK " + QUID + " "
293         + ClientTRAJ;
294     Send(buf, out);
295     synchronized (SmartTraceThreaded.terminal) {
296         SmartTraceThreaded.terminal += "Client send :" + buf + "\n";
297     }
298     // Wait for the Server to send "LCSS" OR "CLOSE"
299 //  buf = in.readLine();
300 //  synchronized (SmartTraceThreaded.terminal) {
301 //      SmartTraceThreaded.terminal += "Server Response: " + buf + ".\n";
302 //  }
303 //  // IF LCSS SEND THE Coordinates[] ARRAY
304 //  if (buf.startsWith("LCSS")) {
305 //      synchronized (SmartTraceThreaded.terminal) {
306 //      SmartTraceThreaded.terminal += "Calculate LCCS and send it to server...\n";
307 //      }
308 //      lcss = mobileLCSS.LCSS(ServerTrajectory, ClientTrajectory, eps);
309 //      ClientTRAJ = "";
310 //      for (int i = 0; i < ClientTrajectory.size(); i++) {
311 //      ClientTRAJ += ClientTrajectory.get(i).getX() + " "
312 //          + ClientTrajectory.get(i).getY() + "\t";
313 //      }
314 //
315 //      buf = "+OK " + QUID + " " + lcss + " " + ClientTRAJ;
316 //      Send(buf, out);
317 //      synchronized (SmartTraceThreaded.terminal) {
318 //      SmartTraceThreaded.terminal += "Client send :" + buf + "\n";
319 //      }
320 //  }
321     return true;
322
323     }
324
325     // Function For the CALCULATE STATE
326     private boolean RETRIEVE() throws IOException {
327     curState = state.RETRIEVE;
328     // send RETRIEVE message to the server
329     Send("RETRIEVE " + QUID, out);
330     SmartTraceThreaded.terminal += "Client send :" + "RETRIEVE" + "\n";
331     buf = in.readLine();
332     if (!buf.startsWith("+OK"))
333         return true;
334     buf = rmPREFIX(buf);
335     SmartTraceThreaded.terminal += "Server response : [" + buf.split("\t")
336         + "]\n";
337     return true;
338
339     }
340
341     // Connect to server to use the upper bound
342     private void serverConnect() {
343
344     try {
345         // ACCEPT FROM SERVER
346         host = serverAddress;
347         try {
348         port = Integer.parseInt(serverPort);
349         } catch (Exception e) {
350         // TODO: handle exception
351         port = 80;
352         }
353         connection = new Socket(host, port);
354         synchronized (SmartTraceThreaded.terminal) {
355         SmartTraceThreaded.terminal += "Socket created.\nConnecting to server on "
356             + host + "...\n";
357
358         }
359         // INITIALIZE THE IN AND OUT STREAMS
360         inStream = connection.getInputStream();
361         in = new BufferedReader(new InputStreamReader(inStream));
362         outStream = connection.getOutputStream();
363         out = new DataOutputStream(outStream);
364     } catch (IOException e) {
```



```
365 //      SmartTraceThreaded.txtMsg.setText(SmartTraceThreaded.terminal);
366         return;
367     }
368     // --------------------------------ESTABLISH-STATE---------------------------------------
369     try {
370         // If the ESTABLISH-STATE is wrong
371         if (!ESTABLISH()) {
372 //      SmartTraceThreaded.txtMsg.setText(SmartTraceThreaded.terminal);
373         appClose();
374         return;
375         }
376     } catch (IOException e) {
377         synchronized (SmartTraceThreaded.terminal) {
378         SmartTraceThreaded.terminal += "Error Connection: "
379             + e.getMessage();
380         }
381         appClose();
382 //      SmartTraceThreaded.txtMsg.setText(SmartTraceThreaded.terminal);
383         return;
384     }
385     // --------------------------------ESTABLISH-STATE---------------------------------------
386
387     boolean once=true;
388     while (once) {
389 //      once=false;
390
391         // the thread is send a search query
392         // and the server answer with an +OK or -ERR message
393         curState = state.ESTABLISH;
394         try {
395         buf = in.readLine();
396         } catch (IOException e) {
397         synchronized (SmartTraceThreaded.terminal) {
398             SmartTraceThreaded.terminal += "Error Connection: "
399                 + e.getMessage();
400             // toast.show();
401         }
402         break;
403         }
404
405         // --------------------------------SEARCH-STATE---------------------------------------
406         if (buf.startsWith("+OK")) {
407         if (buf.contains("Goodbye")) {
408             SmartTraceThreaded.terminal += "\n+OK Goodbye\n";
409             break;
410         }
411         try {
412             if (!SEARCH())
413             break;
414         } catch (IOException e) {
415             synchronized (SmartTraceThreaded.terminal) {
416             SmartTraceThreaded.terminal += "Error Connection: "
417                 + e.getMessage();
418             // toast.show();
419             }
420             break;
421         } // end try
422         try {
423             if (!RETRIEVE())
424             break;
425         } catch (IOException e) {
426             synchronized (SmartTraceThreaded.terminal) {
427             SmartTraceThreaded.terminal += "Error Connection: "
428                 + e.getMessage();
429             // toast.show();
430             }
431             break;
432         } // end try
433           // search Done;
434         continue;
435
436         } else if (buf.startsWith("-ERR")) {
437         Message msg1 = new Message();
```

```
438         msg1.arg1 = Messages.SERVER_ERR;
439         try {
440             msg1.arg2 = Integer.parseInt(buf.split(" ")[1]);
441         } catch (Exception e) {
442             // TODO: handle exception
443             msg1.arg2 = 0;
444         }
445         SmartTraceThreaded.terminal += "Server Busy\nPlease try again";
446         continue;
447         }
448         // -
449         // ------------------------------SEARCH-STATE-----------------------------------------
450         // ------------------------------CALCULATE-STATE--------------------------------------
451         try {
452         if (!CALCULATE())
453             break;
454         } catch (IOException e) {
455         synchronized (SmartTraceThreaded.terminal) {
456             SmartTraceThreaded.terminal += "Error Connection: "
457                 + e.getMessage();
458             // toast.show();
459         }
460         break;
461         } // end try
462           // ------------------------------CALCULATE-STATE--------------------------------------
463
464     }// end while
465
466     // ELSE CLOSE THE PROGRAM
467     // ending and closing the application
468     try {
469         connection.close();
470         synchronized (SmartTraceThreaded.terminal) {
471         SmartTraceThreaded.terminal += "Done.\nConnection closed.\n";
472         // toast.show();
473         }
474     } catch (IOException e) {
475         synchronized (SmartTraceThreaded.terminal) {
476         SmartTraceThreaded.terminal += "Error Connection: "
477             + e.getMessage();
478         // toast.show();
479         }
480         appClose();
481 //      SmartTraceThreaded.txtMsg.setText(SmartTraceThreaded.terminal);
482         return;
483     } // end try
484 //  SmartTraceThreaded.txtMsg.setText(SmartTraceThreaded.terminal);
485     return;
486
487     }
488
489     /**
490      *
491      */
492     void appClose() {
493     // ending and closing the application
494     try {
495         connection.close();
496     } catch (IOException e) {
497     }
498     }
499
500     void readFromFile() {
501     ClientTrajectory.clear();
502     ClientTRAJ = "";
503     // get the Latitude from file and store it to an array
504     try {
505         String line;
506         String[] temp;
507         myPoint temppoint;
508         inFile = new File(INFILE);
509         reader = new BufferedReader(new FileReader(inFile));
510         int i = 0;
```



```
511         while ((line = reader.readLine()) != null) {
512         temp = line.split("\t");
513         temppoint = new myPoint(Double.parseDouble(temp[0]),
514             Double.parseDouble(temp[1]));
515         ClientTrajectory.add(temppoint);
516         ClientTRAJ += ClientTrajectory.get(i).getX() + " "
517             + ClientTrajectory.get(i++).getY() + "\t";
518         }
519
520     } catch (Exception e) {
521         synchronized (SmartTraceThreaded.terminal) {
522         SmartTraceThreaded.terminal += "Exception while reading from file:\n"
523             + e.getMessage();
524         }
525     }
526
527     try {
528         reader.close();
529     } catch (IOException e) {
530         // TODO Auto-generated catch block
531         synchronized (SmartTraceThreaded.terminal) {
532         SmartTraceThreaded.terminal += "Exception while closing file:\n"
533             + e.getMessage();
534         }
535     }
536
537     }
538
539 }
540
```

```
/*
 * (C) Copyright University of Cyprus. 2010-2011.
 *
 * Centralize Server API
 *
 * @version         : 1.0
 * @author : Costantinos Costa(costa.costantinos@gmail.com)
 * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
 * Computer Science Department , University of Cyprus
 *
 *
 */

import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.*;

public class Client {
    static String arrName[] = { "Michael", "Christopher", "Matthew", "Joshua", "Andrew", "David",
            "Justin", "Daniel", "James", "Robert", "John", "Joseph", "Ryan", "Nicholas",
            "Jonathan", "William", "Brandon", "Anthony", "Kevin", "Eric", "Jessica", "Ashley",
            "Amanda", "Sarah", "Jennifer", "Brittany", "Stephanie", "Samantha", "Nicole",
            "Elizabeth", "Lauren", "Megan", "Tiffany", "Heather", "Amber", "Melissa", "Danielle",
            "Emily", "Rachel", "Kayla" };
    private static final String CRLF = "\r\n";
    static BufferedReader in;
    static BufferedReader console;
    static DataOutputStream out;
    Socket socket;

    static public void Send(String msg, OutputStream out) throws IOException {
        out.write((msg + CRLF).getBytes());
        out.flush();
    }

    Client(String ip) throws UnknownHostException, IOException {
        socket = new Socket(ip, 8888);

        in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
        console = new BufferedReader(new InputStreamReader(System.in));
        out = new DataOutputStream(socket.getOutputStream());
    }

    public static void main(String[] args) {
        try {
            Random r = new Random();

            if (args.length > 0)
                new Client(args[0]);
            else
                new Client("127.0.0.1");

            String message = in.readLine();
            System.out.print("Receive : ");
            System.out.println(message);
            System.out.print("Enter response: ");
            String response = "USER " + arrName[r.nextInt(arrName.length)];// console.readLine()
    // +
    // "\n";
            System.out.println(response);
            Send(response, out);
            if (response.equals("QUIT"))
                return;
            message = in.readLine();
            System.out.print("Receive : ");
            System.out.println(message);
            while (true) {
                System.out.print("Do you want to search [y/n]: ");
```



```
                response = console.readLine();
                if (!response.equals("y") && !response.equals("\n"))
                    continue;
                System.out.print("Enter response: ");
                response = "SEARCH 1 2\t2 3\t4 5\t";//
                System.out.println(response);
                Send(response, out);

                message = in.readLine();
                System.out.print("Receive : ");
                System.out.println(message);
                if (message.startsWith("-ERR"))
                    continue;
                message = in.readLine() ;
                System.out.print("Receive : ");
                System.out.println(message);
                System.out.print("Do you want to RETRIEVE [y/n]: ");
                response = console.readLine() ;

                if (!response.equals("y")&& !response.equals("\n")) {
                    Send("NO RETRIEVE", out);

                    continue;
                } else {
                    Send("RETRIEVE", out);
                    message = in.readLine();
                    System.out.println("Receive : " + message);
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```
/*
 * (C) Copyright University of Cyprus. 2010-2011.
 *
 * Centralize Server API
 *
 * @version         : 1.0
 * @author : Costantinos Costa(costa.costantinos@gmail.com)
 * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
 * Computer Science Department , University of Cyprus
 *
 *
 */
import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.*;

public class ClientConnect {

    private static final String CRLF = "\r\n";
    static String arrName[] = { "Michael", "Christopher", "Matthew", "Joshua", "Andrew", "David",
            "Justin", "Daniel", "James", "Robert", "John", "Joseph", "Ryan", "Nicholas",
            "Jonathan", "William", "Brandon", "Anthony", "Kevin", "Eric", "Jessica", "Ashley",
            "Amanda", "Sarah", "Jennifer", "Brittany", "Stephanie", "Samantha", "Nicole",
            "Elizabeth", "Lauren", "Megan", "Tiffany", "Heather", "Amber", "Melissa", "Danielle",
            "Emily", "Rachel", "Kayla" };
    static BufferedReader in;
    static BufferedReader console;
    static DataOutputStream out;
    Socket socket;

    static public void Send(String msg, OutputStream out) throws IOException {
        out.write((msg + CRLF).getBytes());
        out.flush();
    }

    ClientConnect(String ip) throws UnknownHostException, IOException {
        socket = new Socket(ip, 8888);

        in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
        console = new BufferedReader(new InputStreamReader(System.in));
        out = new DataOutputStream(socket.getOutputStream());
    }

    public static void main(String[] args) {
        try {

            Random r = new Random();
            if (args.length > 0)
                new ClientConnect(args[0]);
            else
                new ClientConnect("127.0.0.1");
            System.out.print("Receive : ");
            String message = in.readLine();

            System.out.println(message);
            System.out.print("Enter response: ");
            String response = "USER " + arrName[r.nextInt(arrName.length)];// console.readLine()
    // +
    // "\n";
            System.out.println(response);
            Send(response, out);
            if (response.equals("QUIT"))
                return;
            message = in.readLine() ;
            System.out.print("Receive : ");
            System.out.println(message);
            while (true) {
                message = in.readLine() ;
```



```
                System.out.print("Receive for SEARCH: ");
                System.out.println(message);
                // if(response.startsWith("RETRIEVE")){
                // response = "1 2\t3 4";// Upper Bound
                // System.out.println(response);
                // out.println(response);
                //
                // }

                System.out.print("Enter response: ");
                response = "+OK CALCULATE";//
                System.out.println(response);
                Send(response, out);
                System.out.print("Enter response: ");
                response = console.readLine();
                if (!response.equals("QUIT"))
                    response = "+OK quid " + response;// Upper Bound
                System.out.println(response);
                Send(response, out);
                if (response.equals("QUIT"))
                    return;
                message = in.readLine();
                System.out.print("Receive : ");
                System.out.println(message);
                if (message.startsWith("CLOSE"))
                    continue;

                System.out.print("Enter response: ");
                response = console.readLine() ;
                if (!response.equals("QUIT"))
                    response = "+OK quid " + response + " 1 2\t3 4";// lcss
                System.out.println(response);
                Send(response, out);
                if (response.equals("QUIT"))
                    return;

            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```
1  /*
2   * (C) Copyright University of Cyprus. 2010-2011.
3   *
4   * Centralize Server API
5   *
6   * @version         : 1.0
7   * @author : Costantinos Costa(costa.costantinos@gmail.com)
8   * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
9   * Computer Science Department , University of Cyprus
10  *
11  *
12  */
13 import java.awt.Color;
14 import javax.swing.JTextPane;
15 import javax.swing.text.AttributeSet;
16 import javax.swing.text.SimpleAttributeSet;
17 import javax.swing.text.StyleConstants;
18 import javax.swing.text.StyleContext;
19
20 public class ColorPane extends JTextPane {
21
22     /**
23      *
24      */
25     private static final long serialVersionUID = 1L;
26
27
28
29
30     public  void append(Color c, String s) { // better
31                                                 // implementation--uses
32         // StyleContext
33         this.setEditable(true);
34         StyleContext sc = StyleContext.getDefaultStyleContext();
35         AttributeSet aset = sc.addAttribute(SimpleAttributeSet.EMPTY, StyleConstants.Foreground, c);
36
37         int len = getDocument().getLength(); // same value as
38         // getText().length();
39         setCaretPosition(len); // place caret at the end (with no selection)
40         setCharacterAttributes(aset, false);
41         replaceSelection(s); // there is no selection, so inserts at caret
42         this.setEditable(false);
43     }
44
45 }
```

```
/*
 * (C) Copyright University of Cyprus. 2010-2011.
 *
 * Centralize Server API
 *
 * @version         : 1.0
 * @author : Costantinos Costa(costa.costantinos@gmail.com)
 * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
 * Computer Science Department , University of Cyprus
 *
 *
 */
class Consumer implements Runnable {

    public void run() {
        synchronized (concurrency.NAME) {
        while (concurrency.NAME < 5000) {


                concurrency.NAME++;

            }

        }
    }
}

class Producer implements Runnable {

    public void run() {
        synchronized (concurrency.NAME) {

        while (concurrency.NAME < 5000) {


                concurrency.NAME++;
                if(concurrency.NAME>1000){
                    System.out.println("i will return");
                return;
                 }



            }
        }
    }
}

public class concurrency {

    public static Integer NAME = 0;


    public static void main(String[] args) {
        Thread one = new Thread(new Producer());
        Thread two = new Thread(new Consumer());
        one.start();
        two.start();
        try {
            one.join();
            two.join();
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        System.out.println("Final="+concurrency.NAME);

    }

}
```

```
/*
 * (C) Copyright University of Cyprus. 2010-2011.
 *
 * SmartTrace Server API
 *
 * @version         : 1.0
 * @author : Costantinos Costa(costa.costantinos@gmail.com)
 * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
 * Computer Science Department , University of Cyprus
 *
 *
 */
package SmartTrace.apk;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.Socket;
import java.util.ArrayList;
import android.content.Context;
import android.os.Handler;
import android.os.Message;

public class Connect extends Thread {

    enum state {
        CLOSED, ESTABLISH, CALCULATE, SEARCH, RETRIEVE;
    }

    private BufferedWriter writer = null;
    private DataOutputStream out;
    private BufferedReader in;
    private String serverAddress = new String();
    private String epsilon = new String();
    private String usr = new String();
    private Socket connection = null;
    private String ClientTRAJ;
    private double lcss;
    private String SSID;
    private String QUID;
    String host = new String();
    String buf = new String();
    String[] traj = null;
    InputStream inStream = null;
    OutputStream outStream = null;
    private String psw = "";
    private String LCSS;
    private int port = 80;
    private String serverPort = "80";
    static state curState = state.CLOSED;
    static Handler messageHandler;

    private String rmPREFIX(String msg) {
        String[] sArr = msg.split(" ");
        String temp = new String();
        for (int i = 1; i < sArr.length; i++)
            temp += sArr[i] + " ";
        return temp;
    }

    // FUNCTION FOR THE MESSAGES
    static public void Send(String msg, OutputStream out) throws IOException {
        out.write((msg + Messages.CRLF).getBytes());
        out.flush();
    }

    Connect(Context c, boolean flag, String username) {
        // set username
        usr = username;
```



```

        // initialized the message handler for the messages from the thread
        messageHandler = new Handler() {

            public void handleMessage(Message msg) {
                super.handleMessage(msg);
                Message msg1 = new Message();
                switch (msg.arg1) {
                // These means that the client want to use smartTrace search
                // We create new thread so the GUI doesn't block
                case Messages.SEARCH:

                    // // TODO Auto-generated method stub
                    if (!curState.equals(state.ESTABLISH)) {
                        switch (curState) {
                        case CALCULATE:
                            msg1.arg1 = Messages.STATECAL;
                            break;
                        case SEARCH:
                            msg1.arg1 = Messages.STATESER;
                            break;
                        case RETRIEVE:
                            msg1.arg1 = Messages.STATERET;
                            break;

                        default:
                            msg1.arg1 = Messages.SERVER_ERR;
                            break;
                        }

                        MainActivity.messageHandler.sendMessage(msg1);
                        return;
                    }

                    // change the trajectory to string
                    ClientTRAJ = "";
                    for (int i = 0; i < MainActivity.ClientTrajectory.size(); i++) {
                        ClientTRAJ += MainActivity.ClientTrajectory.get(i).getX() + " "
                                + MainActivity.ClientTrajectory.get(i).getY() + "\t";
                    }

                    String temp = MainActivity.preferences.getString("parK", "1");
                    try{
                    MainActivity.K = Integer.parseInt(temp);
                    }catch (Exception e) {
                        // TODO: handle exception
                        MainActivity.K=1;
                    }
                    try {
                        Send(("SEARCH "+MainActivity.K+" " + SSID + " " + ClientTRAJ), out);
                    } catch (IOException e) {
                        // send a massage to main thread
                        msg1.arg1 = Messages.OTHER;
                        MainActivity.messageHandler.sendMessage(msg1);
                    }
                    // write everything to the transaction string
                    synchronized (More.terminal) {
                        More.terminal += "Client send : SEARCH " + SSID + " Client TRAJECTORY \n";

                    }
                    // Send message to the main thread that you are searching
                    msg1.arg1 = Messages.SEARCH;
                    MainActivity.messageHandler.sendMessage(msg1);
                    // }
                    // }).start();
                    break;
                case Messages.QUIT:

                    if (connection == null) {
                        // appClose();
                        return;
                    }
                    try {
```

```
                    Send("QUIT", out);
                    connection.close();
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
                synchronized (More.terminal) {
                    More.terminal += "Client send : QUIT \n";

                }
                Message msg11 = new Message();
                msg11.arg1 = Messages.QUIT;
                MainActivity.messageHandler.sendMessage(msg11);
                appClose();
                return;

            default:

                break;
            }

        }
    };
    this.setUncaughtExceptionHandler(new UncaughtExceptionHandler() {

        @Override
        public void uncaughtException(Thread thread, Throwable ex) {
            // TODO Auto-generated method stub
            Message msg = new Message();
            msg.arg1 = Messages.OTHER;
            MainActivity.messageHandler.sendMessage(msg);
            More.terminal += "Client(" + thread.getId() + ").Connection ERROR :\n"
                    + ex.getLocalizedMessage();
            // toast.show();
        }
    });
}

public void run() {
    // Check for any change in preferences
    // Check if the user-name is set
    if (!MainActivity.isUsernameSet) {
        Message msg = new Message();
        msg.arg1 = Messages.SERVER_ERR;
        msg.arg2 = 0;
        MainActivity.messageHandler.sendMessage(msg);
        return;
    }
    serverAddress = MainActivity.preferences.getString("serverAdd", "n/a");
    if (serverAddress.equals("n/a") || serverAddress.equals("")) {
        Message msg = new Message();
        msg.arg1 = Messages.INVALID_ADDRESS;
        MainActivity.messageHandler.sendMessage(msg);
        return;
    }

    // serverPort
    serverPort = MainActivity.preferences.getString("serverPort", "65000");
    if (serverPort.equals("n/a") || serverPort.equals("")) {
        Message msg = new Message();
        msg.arg1 = Messages.INVALID_ADDRESS;
        MainActivity.messageHandler.sendMessage(msg);
        return;
    }
    // Check for any change in preferences
    epsilon = MainActivity.preferences.getString("epsilon", "n/a");
    if (epsilon.equals("n/a") || epsilon.equals("")) {
        Message msg = new Message();
        msg.arg1 = Messages.DEFAULT_EPSILON;
        MainActivity.messageHandler.sendMessage(msg);
        // toast.show();
        epsilon = "2";
    }
```



```
    // close the gps
    if (MainActivity.GpsStarted) {
        try {
            if (writer != null)
                writer.close();
        } catch (Exception e) {
        }
        MainActivity.locationManager.removeUpdates(MainActivity.locationListener);
        MainActivity.GpsStarted = false;
    }
    serverConnect();
}

// function for the ESTABLISH STATE
private boolean ESTABLISH() throws IOException {
    curState = state.ESTABLISH;
    // GET THE +OK READY FROM SERVER
    buf = in.readLine();
    More.terminal += "Server Response: " + buf + "\n";
    // Wait For Proceeding
    // Get The Session Id
    buf = "USER " + usr + " " + psw;
    // SEND A +OK REPLY and user-name TO SERVER
    Send(buf, out);
    More.terminal += "Client Send: " + buf + "\n";
    // GET THE +OK or -ERR FROM SERVER
    buf = in.readLine();
    More.terminal += "Server Response: " + buf + "\n";
    if (buf.startsWith("+OK")) {
        // Set SessionID
        SSID = rmPREFIX(buf);
        return true;
    }
    return false;

}

// Function For the SEARCH STATE
private boolean SEARCH() throws IOException {
    curState = state.SEARCH;
    // Read The Message To Continue
    More.terminal += "Server Response: " + buf + "\n";
    // Set SessionID
    QUID = rmPREFIX(buf);
    // Read The Message To Continue
    buf = in.readLine();
    synchronized (More.terminal) {
        More.terminal += "Server Response: " + buf + "\n";
    }
    if (buf.contains("Goodbye"))
        return false;
    LCSS = rmPREFIX(buf);
    return true;
}

// Function For the CALCULATE STATE
private boolean CALCULATE() throws IOException {
    if (!buf.startsWith("CALCULATE"))
        return true;
    curState = state.CALCULATE;
    synchronized (More.terminal) {
        More.terminal += "Server Response: " + "CALCULATE queryTrajectory" + "\n";
    }// toast.show();
    try {
        QUID = buf.split(" ")[1];
    } catch (Exception e) {
        // TODO: handle exception
        return false;
    }
    String query = rmPREFIX(buf);
    buf = "+OK CALCULATE";
    Send(buf, out);
    synchronized (More.terminal) {
```

```
293             More.terminal += "Client send :" + buf + "\n";
294         }
295         // split the points
296         traj = query.split("\t");
297         if (traj == null) {
298             connection.close();
299             return true;
300         }
301
302         MainActivity.ServerTrajectory = new ArrayList<myPoint>();
303         String[] tmppnt;
304         for (int j = 0; j < traj.length; j++) {
305             tmppnt = traj[j].split(" ");
306             // synchronized (More.terminal) {
307             // More.terminal += traj[j] + "\n";
308             // }
309             if (tmppnt != null)
310
311                 if (tmppnt.length > 1)
312                     try {
313                         MainActivity.ServerTrajectory.add(new myPoint(
314                                 Double.parseDouble(tmppnt[0]), Double.parseDouble(tmppnt[1])));
315                     } catch (Exception e) {
316                         // TODO: handle exception
317                         continue;
318                     }
319         }
320
321         // CALCULATE THE UB
322         Message msg = new Message();
323         msg.arg1 = Messages.UPBOUND;
324         MainActivity.messageHandler.sendMessage(msg);
325         // SEND THE UB TO SERVER
326         double eps = 0.1;
327         try {
328             eps = Double.parseDouble(epsilon);
329         } catch (Exception e) {
330             // TODO: handle exception
331         }
332         buf = "+OK " + QUID + " "
333                 + UB.upperBound(MainActivity.ServerTrajectory, MainActivity.ClientTrajectory, eps);
334         Send(buf, out);
335         synchronized (More.terminal) {
336             More.terminal += "Client send :" + buf + "\n";
337         }
338         // Wait for the Server to send "LCSS" OR "CLOSE"
339         buf = in.readLine();
340         synchronized (More.terminal) {
341             More.terminal += "Server Response: " + buf + ".\n";
342         }
343         // IF LCSS SEND THE Coordinates[] ARRAY
344         if (buf.startsWith("LCSS")) {
345             synchronized (More.terminal) {
346                 More.terminal += "Calculate LCCS and send it to server...\n";
347             }
348             lcss = mobileLCSS.LCSS(MainActivity.ServerTrajectory, MainActivity.ClientTrajectory,
349                     eps);
350             ClientTRAJ = "";
351             for (int i = 0; i < MainActivity.ClientTrajectory.size(); i++) {
352                 ClientTRAJ += MainActivity.ClientTrajectory.get(i).getX() + " "
353                         + MainActivity.ClientTrajectory.get(i).getY() + "\t";
354             }
355             if (MainActivity.privacy)
356                 buf = "+OK " + QUID + " " + lcss + " " + "PRIVATE";
357             else
358                 buf = "+OK " + QUID + " " + lcss + " " + ClientTRAJ;
359             Send(buf, out);
360             synchronized (More.terminal) {
361                 More.terminal += "Client send :" + buf + "\n";
362             }
363         }
364         return true;
365
```



```
366     }
367
368     // Function For the CALCULATE STATE
369     private boolean RETRIEVE() throws IOException {
370         curState = state.RETRIEVE;
371         // send RETRIEVE message to the server
372         Send("RETRIEVE " + QUID, out);
373         More.terminal += "Client send :" + "RETRIEVE" + "\n";
374         buf = in.readLine();
375         if (!buf.startsWith("+OK"))
376             return true;
377         buf = rmPREFIX(buf);
378         if (buf.contains("PRIVATE"))
379             MainActivity.QueryTrajectories = null;
380         else {
381             String[] temp=buf.split("&");
382             MainActivity.QueryTrajectories = new String[ temp.length][];
383             int i=0;
384             for (String ss : temp) {
385                 MainActivity.QueryTrajectories[i] = ss.split("\t");
386                     i++;
387             }
388
389         }
390         More.terminal += "Server response : [" + buf.split("\t") + "]\n";
391         return true;
392
393     }
394
395     private void searchDone() {
396         // TODO Auto-generated method stub
397         // close the connection
398         Message smsg = new Message();
399         smsg.arg1 = Messages.ENDSRCH;
400         smsg.arg2 = (int) (Double.parseDouble(LCSS) * 100);
401         MainActivity.messageHandler.sendMessage(smsg);
402     }
403
404     // Connect to server to use the upper bound
405     private void serverConnect() {
406         // open the coordinates file
407         if (MainActivity.ClientTrajectory.size() == 0) {
408             Message msg = new Message();
409             msg.arg1 = Messages.FIRST_PLOT;
410             MainActivity.messageHandler.sendMessage(msg);
411             return;
412         }
413         try {
414             // ACCEPT FROM SERVER
415             host = serverAddress;
416             try {
417                 port = Integer.parseInt(serverPort);
418             } catch (Exception e) {
419                 // TODO: handle exception
420                 port = 80;
421             }
422
423             connection = new Socket(host, port);
424             synchronized (More.terminal) {
425                 More.terminal += "Socket created.\nConnecting to server on " + host + "...\n";
426             }
427             // INITIALIZE THE IN AND OUT STREAMS
428             inStream = connection.getInputStream();
429             in = new BufferedReader(new InputStreamReader(inStream));
430             outStream = connection.getOutputStream();
431             out = new DataOutputStream(outStream);
432         } catch (IOException e) {
433             return;
434         }
435         // --------------------------------ESTABLISH-STATE---------------------------------------
436         try {
437             // If the ESTABLISH-STATE is wrong
438             if (!ESTABLISH()) {
```

```
439                 appClose();
440                 return;
441             }
442         } catch (IOException e) {
443             synchronized (More.terminal) {
444                 More.terminal += "Error Connection: " + e.getMessage();
445             }
446             appClose();
447             return;
448         }
449         // ------------------------------ESTABLISH-STATE---------------------------------------
450
451         while (true) {
452
453             // the thread is send a search query
454             // and the server answer with an +OK or -ERR message
455             curState = state.ESTABLISH;
456             try {
457                 buf = in.readLine();
458             } catch (IOException e) {
459                 synchronized (More.terminal) {
460                     More.terminal += "Error Connection: " + e.getMessage();
461                     // toast.show();
462                 }
463                 break;
464             }
465
466             // ------------------------------SEARCH-STATE-----------------------------------------
467             if (buf.startsWith("+OK")) {
468                 if (buf.contains("Goodbye")) {
469                     More.terminal += "\n+OK Goodbye\n";
470                     break;
471                 }
472                 try {
473                     if (!SEARCH())
474                         break;
475                 } catch (IOException e) {
476                     synchronized (More.terminal) {
477                         More.terminal += "Error Connection: " + e.getMessage();
478                         // toast.show();
479                     }
480                     break;
481                 } // end try
482                 try {
483                     if (!RETRIEVE())
484                         break;
485                 } catch (IOException e) {
486                     synchronized (More.terminal) {
487                         More.terminal += "Error Connection: " + e.getMessage();
488                         // toast.show();
489                     }
490                     break;
491                 } // end try
492                 searchDone();
493                 continue;
494
495             } else if (buf.startsWith("-ERR")) {
496                 Message msg1 = new Message();
497                 msg1.arg1 = Messages.SERVER_ERR;
498                 try {
499                     msg1.arg2 = Integer.parseInt(buf.split(" ")[1]);
500                 } catch (Exception e) {
501                     // TODO: handle exception
502                     msg1.arg2 = 0;
503                 }
504                 MainActivity.messageHandler.sendMessage(msg1);
505                 More.terminal += "Server Busy\nPlease try again";
506                 continue;
507             }
508             // -
509             // ------------------------------SEARCH-STATE-----------------------------------------
510             // ------------------------------CALCULATE-STATE--------------------------------------
511             try {
```



```
512                 if (!CALCULATE())
513                     break;
514             } catch (IOException e) {
515                 synchronized (More.terminal) {
516                     More.terminal += "Error Connection: " + e.getMessage();
517                     // toast.show();
518                 }
519                 break;
520             } // end try
521             // ------------------------------CALCULATE-STATE---------------------------------------
522
523         }// end while
524
525         // ELSE CLOSE THE PROGRAM
526         // ending and closing the application
527         try {
528             connection.close();
529             synchronized (More.terminal) {
530                 More.terminal += "Done.\nConnection closed.\n";
531                 // toast.show();
532             }
533         } catch (IOException e) {
534             synchronized (More.terminal) {
535                 More.terminal += "Error Connection: " + e.getMessage();
536                 // toast.show();
537             }
538             appClose();
539             return;
540         } // end try
541         Message mmsg = new Message();
542         mmsg.arg1 = Messages.ENDCONN;
543         MainActivity.messageHandler.sendMessage(mmsg);
544         return;
545
546     }
547
548     /**
549      *
550      */
551     void appClose() {
552         // ending and closing the application
553         try {
554             connection.close();
555         } catch (IOException e) {
556         }
557         Message msg = new Message();
558         msg.arg1 = Messages.OTHER;
559         MainActivity.messageHandler.sendMessage(msg);
560     }
561
562 }
563
```

```
1  /*
2   * (C) Copyright University of Cyprus. 2010-2011.
3   *
4   * SmartTrace Server API
5   *
6   * @version         : 1.0
7   * @author : Costantinos Costa(costa.costantinos@gmail.com)
8   * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
9   * Computer Science Department , University of Cyprus
10  *
11  *
12  */
13 package cs.ucy.ac.cy;
14
15 import java.io.BufferedReader;
16 import java.io.BufferedWriter;
17 import java.io.DataOutputStream;
18 import java.io.File;
19 import java.io.FileReader;
20 import java.io.IOException;
21 import java.io.InputStream;
22 import java.io.InputStreamReader;
23 import java.io.OutputStream;
24 import java.net.Socket;
25 import java.util.ArrayList;
26 import android.content.Context;
27 import android.os.Handler;
28 import android.os.Message;
29 import android.util.Log;
30 import android.widget.Toast;
31
32 public class DecentralizeConnect extends Thread {
33
34     enum state {
35     CLOSED, ESTABLISH, CALCULATE, SEARCH, RETRIEVE;
36     }
37
38     myPoint[] trajectory = null;
39     private File inFile = null;
40     BufferedReader reader = null;
41     Toast toast;
42     private BufferedWriter writer = null;
43     private DataOutputStream out;
44     private BufferedReader in;
45     private String serverAddress = new String();
46     private String epsilon = new String();
47     private String usr = new String();
48     private Socket connection = null;
49     private String ClientTRAJ;
50     private double lcss;
51     private String SSID;
52     private String QUID;
53     private String LOG;
54     private String INFILE;
55     String host = new String();
56     String buf = new String();
57     String[] traj = null;
58     InputStream inStream = null;
59     OutputStream outStream = null;
60     private String psw = "";
61     private String LCSS;
62     private int port = 65000;
63     private String serverPort = "65000";
64     private ArrayList<myPoint> ClientTrajectory= new ArrayList<myPoint>();
65     private ArrayList<myPoint> ServerTrajectory;
66     static state curState = state.CLOSED;
67 //    static Handler messageHandler;
68
69     private String rmPREFIX(String msg) {
70     String[] sArr = msg.split(" ");
71     String temp = new String();
72     for (int i = 1; i < sArr.length; i++)
73         temp += sArr[i] + " ";
```



```
74     return temp;
75     }
76
77     // FUNCTION FOR THE MESSAGES
78     static public void Send(String msg, OutputStream out) throws IOException {
79     out.write((msg + Messages.CRLF).getBytes());
80     out.flush();
81     }
82
83     DecentralizeConnect(Context c, boolean flag, String username, String port, String ip, String log,
       String in,
84         String e) {
85     // set username
86 //  SmartTraceThreaded.txtMsg.setText("Trying to Connect to " + ip + ":"
87 //          + port);
88     usr = username;
89     serverAddress = ip;
90     serverPort = port;
91     epsilon = e;
92     LOG=log;
93     INFILE=in;
94     // initialized the message handler for the messages from the thread
95 //  messageHandler = new Handler() {
96 //
97 //      public void handleMessage(Message msg) {
98 //      super.handleMessage(msg);
99 //      Message msg1 = new Message();
100 //     switch (msg.arg1) {
101 //     // These means that the client want to use smartTrace search
102 //     // We create new thread so the GUI doesn't block
103 //     case Messages.SEARCH:
104 //
105 //         // // TODO Auto-generated method stub
106 //         if (!curState.equals(state.ESTABLISH)) {
107 //         switch (curState) {
108 //         case CALCULATE:
109 //             msg1.arg1 = Messages.STATECAL;
110 //             break;
111 //         case SEARCH:
112 //             msg1.arg1 = Messages.STATESER;
113 //             break;
114 //         case RETRIEVE:
115 //             msg1.arg1 = Messages.STATERET;
116 //             break;
117 //
118 //         default:
119 //             msg1.arg1 = Messages.SERVER_ERR;
120 //             break;
121 //         }
122 //
123 //         return;
124 //         }
125 //
126 //         try {
127 //         Send(("SEARCH " + SmartTraceThreaded.parK + " " + SSID
128 //             + " " + ClientTRAJ), out);
129 //         } catch (IOException e) {
130 //         // send a massage to main thread
131 //         SmartTraceThreaded.terminal += "Client send : SEARCH "
132 //             + SSID + " Client TRAJECTORY \n";
133 //         }
134 //         // write everything to the transaction string
135 //         synchronized (SmartTraceThreaded.terminal) {
136 //         SmartTraceThreaded.terminal += "Client send : SEARCH "
137 //             + SSID + " Client TRAJECTORY \n";
138 //
139 //         }
140 //
141 //         break;
142 //     case Messages.QUIT:
143 //
144 //         if (connection == null) {
145 //         // appClose();
```

```
146 //          return;
147 //          }
148 //          try {
149 //          Send("QUIT", out);
150 //          connection.close();
151 //          } catch (IOException e) {
152 //          // TODO Auto-generated catch block
153 //          e.printStackTrace();
154 //          }
155 //          synchronized (SmartTraceThreaded.terminal) {
156 //          SmartTraceThreaded.terminal += "Client send : QUIT \n";
157 //
158 //          }
159 //          appClose();
160 //          return;
161 //
162 //      default:
163 //
164 //          break;
165 //      }
166 //
167 //      }
168 //  };
169
170     // READ FROM FILE
171     readFromFile();
172
173
174     this.setUncaughtExceptionHandler(new UncaughtExceptionHandler() {
175
176         @Override
177         public void uncaughtException(Thread thread, Throwable ex) {
178         // TODO Auto-generated method stub
179
180 //      SmartTraceThreaded.txtMsg.setText(SmartTraceThreaded.terminal);
181         }
182     });
183     }
184
185     public void run() {
186
187     // just connect to the server
188     serverConnect();
189     }
190
191     // function for the ESTABLISH STATE
192     private boolean ESTABLISH() throws IOException {
193     curState = state.ESTABLISH;
194     // GET THE +OK READY FROM SERVER
195     buf = in.readLine();
196     SmartTraceThreaded.terminal += "Server Response: " + buf + "\n";
197     // Wait For Proceeding
198     // Get The Session Id
199     buf = "USER " + usr + " " + psw;
200     // SEND A +OK REPLY and user-name TO SERVER
201     Send(buf, out);
202     SmartTraceThreaded.terminal += "Client Send: " + buf + "\n";
203     // GET THE +OK or -ERR FROM SERVER
204     buf = in.readLine();
205     SmartTraceThreaded.terminal += "Server Response: " + buf + "\n";
206     if (buf.startsWith("+OK")) {
207         // Set SessionID
208         SSID = rmPREFIX(buf);
209         return true;
210     }
211     return false;
212
213     }
214
215     // Function For the SEARCH STATE
216     private boolean SEARCH() throws IOException {
217     curState = state.SEARCH;
218     // Read The Message To Continue
```



```
219     SmartTraceThreaded.terminal += "Server Response: " + buf + "\n";
220     // Set SessionID
221     QUID = rmPREFIX(buf);
222     // Read The Message To Continue
223     buf = in.readLine();
224     synchronized (SmartTraceThreaded.terminal) {
225         SmartTraceThreaded.terminal += "Server Response: " + buf + "\n";
226     }
227     if (buf.contains("Goodbye"))
228         return false;
229     LCSS = rmPREFIX(buf);
230     return true;
231     }
232
233     // Function For the CALCULATE STATE
234     private boolean CALCULATE() throws IOException {
235     if (!buf.startsWith("CALCULATE"))
236         return true;
237     curState = state.CALCULATE;
238     synchronized (SmartTraceThreaded.terminal) {
239         SmartTraceThreaded.terminal += "Server Response: "
240             + "CALCULATE queryTrajectory" + "\n";
241     }// toast.show();
242     try {
243         QUID = buf.split(" ")[1];
244     } catch (Exception e) {
245         // TODO: handle exception
246         return false;
247     }
248     String query = rmPREFIX(buf);
249     buf = "+OK CALCULATE";
250     Send(buf, out);
251     synchronized (SmartTraceThreaded.terminal) {
252         SmartTraceThreaded.terminal += "Client send :" + buf + "\n";
253     }
254     // split the points
255     traj = query.split("\t");
256     if (traj == null) {
257         connection.close();
258         return true;
259     }
260
261     ServerTrajectory = new ArrayList<myPoint>();
262     String[] tmppnt;
263     for (int j = 0; j < traj.length; j++) {
264         tmppnt = traj[j].split(" ");
265         // synchronized (SmartTraceThreaded.terminal) {
266         // SmartTraceThreaded.terminal += traj[j] + "\n";
267         // }
268         if (tmppnt != null)
269
270         if (tmppnt.length > 1)
271             try {
272             ServerTrajectory.add(new myPoint(Double
273                 .parseDouble(tmppnt[0]), Double
274                 .parseDouble(tmppnt[1])));
275             } catch (Exception e) {
276             // TODO: handle exception
277             continue;
278             }
279     }
280
281
282
283     // CALCULATE THE UB
284
285     // SEND THE UB TO SERVER
286     double eps = 0.1;
287     try {
288         eps = Double.parseDouble(epsilon);
289     } catch (Exception e) {
290         // TODO: handle exception
291     }
```

```
292 //  buf = "+OK " + QUID + " "
293 //      + UB.upperBound(ServerTrajectory, ClientTrajectory, eps);
294 //  Send(buf, out);
295 //  synchronized (SmartTraceThreaded.terminal) {
296 //      SmartTraceThreaded.terminal += "Client send :" + buf + "\n";
297 //  }
298 //  // Wait for the Server to send "LCSS" OR "CLOSE"
299 //  buf = in.readLine();
300 //  synchronized (SmartTraceThreaded.terminal) {
301 //      SmartTraceThreaded.terminal += "Server Response: " + buf + ".\n";
302 //  }
303 //  // IF LCSS SEND THE Coordinates[] ARRAY
304 //  if (buf.startsWith("LCSS")) {
305         synchronized (SmartTraceThreaded.terminal) {
306         SmartTraceThreaded.terminal += "Calculate LCCS and send it to server...\n";
307         }
308         lcss = mobileLCSS.LCSS(ServerTrajectory, ClientTrajectory, eps,SmartTraceThreaded.delta);
309       //  lcss=0;
310         ClientTRAJ = "";
311         for (int i = 0; i < ClientTrajectory.size(); i++) {
312         ClientTRAJ += ClientTrajectory.get(i).getX() + " "
313             + ClientTrajectory.get(i).getY() + "\t";
314         }
315
316         buf = "+OK " + QUID + " " + lcss + " " + ClientTRAJ;
317         Send(buf, out);
318         synchronized (SmartTraceThreaded.terminal) {
319         SmartTraceThreaded.terminal += "Client send :" + buf + "\n";
320         }
321 //  }
322     return true;
323
324     }
325
326     // Function For the CALCULATE STATE
327     private boolean RETRIEVE() throws IOException {
328     curState = state.RETRIEVE;
329     // send RETRIEVE message to the server
330     Send("RETRIEVE " + QUID, out);
331     SmartTraceThreaded.terminal += "Client send :" + "RETRIEVE" + "\n";
332     buf = in.readLine();
333     if (!buf.startsWith("+OK"))
334         return true;
335     buf = rmPREFIX(buf);
336     SmartTraceThreaded.terminal += "Server response : [" + buf.split("\t")
337         + "]\n";
338     return true;
339
340     }
341
342     // Connect to server to use the upper bound
343     private void serverConnect() {
344
345     try {
346         // ACCEPT FROM SERVER
347         host = serverAddress;
348         try {
349         port = Integer.parseInt(serverPort);
350         } catch (Exception e) {
351         // TODO: handle exception
352         port = 80;
353         }
354         connection = new Socket(host, port);
355         synchronized (SmartTraceThreaded.terminal) {
356         SmartTraceThreaded.terminal += "Socket created.\nConnecting to server on "
357             + host + "...\n";
358
359         }
360         // INITIALIZE THE IN AND OUT STREAMS
361         inStream = connection.getInputStream();
362         in = new BufferedReader(new InputStreamReader(inStream));
363         outStream = connection.getOutputStream();
364         out = new DataOutputStream(outStream);
```



```
365     } catch (IOException e) {
366 //      SmartTraceThreaded.txtMsg.setText(SmartTraceThreaded.terminal);
367         return;
368     }
369     // --------------------------------ESTABLISH-STATE---------------------------------------
370     try {
371         // If the ESTABLISH-STATE is wrong
372         if (!ESTABLISH()) {
373 //      SmartTraceThreaded.txtMsg.setText(SmartTraceThreaded.terminal);
374         appClose();
375         return;
376         }
377     } catch (IOException e) {
378         synchronized (SmartTraceThreaded.terminal) {
379         SmartTraceThreaded.terminal += "Error Connection: "
380             + e.getMessage();
381         }
382         appClose();
383 //      SmartTraceThreaded.txtMsg.setText(SmartTraceThreaded.terminal);
384         return;
385     }
386     // --------------------------------ESTABLISH-STATE---------------------------------------
387
388     boolean once=true;
389     while (once) {
390 //      once=false;
391         // the thread is send a search query
392         // and the server answer with an +OK or -ERR message
393         curState = state.ESTABLISH;
394         try {
395         buf = in.readLine();
396         } catch (IOException e) {
397         synchronized (SmartTraceThreaded.terminal) {
398             SmartTraceThreaded.terminal += "Error Connection: "
399                 + e.getMessage();
400             // toast.show();
401         }
402         break;
403         }
404
405         // --------------------------------SEARCH-STATE---------------------------------------
406         if (buf.startsWith("+OK")) {
407         if (buf.contains("Goodbye")) {
408             SmartTraceThreaded.terminal += "\n+OK Goodbye\n";
409             break;
410         }
411         try {
412             if (!SEARCH())
413             break;
414         } catch (IOException e) {
415             synchronized (SmartTraceThreaded.terminal) {
416             SmartTraceThreaded.terminal += "Error Connection: "
417                 + e.getMessage();
418             // toast.show();
419             }
420             break;
421         } // end try
422         try {
423             if (!RETRIEVE())
424             break;
425         } catch (IOException e) {
426             synchronized (SmartTraceThreaded.terminal) {
427             SmartTraceThreaded.terminal += "Error Connection: "
428                 + e.getMessage();
429             // toast.show();
430             }
431             break;
432         } // end try
433           // search Done;
434         continue;
435
436         } else if (buf.startsWith("-ERR")) {
437         Message msg1 = new Message();
```

```java
/*
 * (C) Copyright University of Cyprus. 2010-2011.
 *
 * SmartTrace Server API
 *
 * @version         : 1.0
 * @author : Costantinos Costa(costa.costantinos@gmail.com)
 * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
 * Computer Science Department , University of Cyprus
 *
 *
 */
package SmartTrace.apk;

import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Point;

import com.google.android.maps.GeoPoint;
import com.google.android.maps.MapView;
import com.google.android.maps.Overlay;
import com.google.android.maps.Projection;

public class DrawPathOverlay extends Overlay {
    static int widthOfPath;
    static int color=Color.RED;
    private GeoPoint gp1;
    private GeoPoint gp2;
    private int mcolor=0;
    public DrawPathOverlay(GeoPoint gp1, GeoPoint gp2,int mcolor) {
        this.gp1 = gp1;
        this.gp2 = gp2;

        if(mcolor==0)
            this.mcolor=color;
        else
            this.mcolor=mcolor;
    }

    public boolean draw(Canvas canvas, MapView mapView, boolean shadow,
            long when) {
        Projection projection = mapView.getProjection();
        if (shadow == false) {
            Paint paint = new Paint();
            paint.setAntiAlias(true);
            Point point = new Point();
            projection.toPixels(gp1, point);
            paint.setColor(mcolor);
            Point point2 = new Point();
            projection.toPixels(gp2, point2);
            paint.setStrokeWidth(widthOfPath);
            canvas.drawLine((float) point.x, (float) point.y, (float) point2.x,
                    (float) point2.y, paint);
        }
        return super.draw(canvas, mapView, shadow, when);
    }

    public void draw(Canvas canvas, MapView mapView, boolean shadow) {
        // TODO Auto-generated method stub

        super.draw(canvas, mapView, shadow);
    }

}
```

```java
/*
 * (C) Copyright University of Cyprus. 2010-2011.
 *
 * SmartTrace Server API
 *
 * @version         : 1.0
 * @author : Costantinos Costa(costa.costantinos@gmail.com)
 * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
 * Computer Science Department , University of Cyprus
 *
 *
 */
package SmartTrace.apk;

import SmartTrace.apk.R;
import android.app.Activity;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.text.TextUtils;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

public class EditTextBrowser extends Activity {

    final public static String PATH = "path";
    Button btnOk;
    Button btnCancel;
    Button btnBrowse;
    private SharedPreferences mPrefs;
    public static EditText txb;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.dialog_layout);
        btnOk = (Button) findViewById(R.id.btnOk);
        btnCancel = (Button) findViewById(R.id.btnCancel);
        btnBrowse = (Button) findViewById(R.id.btnBrowse);
        txb = (EditText) findViewById(R.id.txbFile);
        txb.setScrollContainer(true);
        btnCancel.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                Stop();
            }
        });
        btnBrowse.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                Intent i = new Intent(EditTextBrowser.this,
                        AndroidFileBrowser.class);
                AndroidFileBrowser.flagMainOrwifi=true;
                startActivity(i);
            }
        });

        mPrefs = PreferenceManager.getDefaultSharedPreferences(this);
        MainActivity.custFilePath = mPrefs.getString(PATH, "/sdcard/");

        txb.setText(MainActivity.custFilePath);
        btnOk.setOnClickListener(new OnClickListener() {

            @Override
```



```java
            public void onClick(View v) {
                // TODO Auto-generated method stub
                MainActivity.custFilePath = txb.getText().toString();
                Stop();
            }
        });
    }

    @Override
    protected void onRestoreInstanceState(Bundle savedInstanceState) {
        MainActivity.custFilePath = savedInstanceState.getString(PATH);
        if (!TextUtils.isEmpty(MainActivity.custFilePath)) {
            // Update lbResult
            EditTextBrowser.txb.setText(savedInstanceState.getString(PATH));
        }
        super.onRestoreInstanceState(savedInstanceState);
    }

    @Override
    protected void onSaveInstanceState(Bundle outState) {
        /*************************************************
         * Here we do _temporary_ save all critical data, like our
         * selectedProduct.
         *************************************************/
        outState.putString(PATH, MainActivity.custFilePath);
        super.onSaveInstanceState(outState);
    }

    void Stop() {
        SharedPreferences.Editor ed = mPrefs.edit();
        ed.putString(PATH, MainActivity.custFilePath);
        ed.commit();
        finish();
    }

}

```

```
/*
 * (C) Copyright University of Cyprus. 2010-2011.
 *
 * SmartTrace Server API
 *
 * @version         : 1.0
 * @author : Costantinos Costa(costa.costantinos@gmail.com)
 * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
 * Computer Science Department , University of Cyprus
 *
 *
 */
package SmartTrace.apk;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Paint.Align;
import android.view.View;

/**
 *
 *
 * @author Costantinos
 *
 */
//GraphView creates a scaled line or bar graph with x and y axis labels.
public class GraphView extends View {

    public static boolean BAR = true;
    public static boolean LINE = false;

    private Paint paint;
    private float[] values;
    private String[] horlabels;
    private String[] verlabels;
    private String title;
    private boolean type;

    public GraphView(Context context, float[] values, String title, String[] horlabels,
            String[] verlabels, boolean type) {
        super(context);
        if (values == null)
            values = new float[0];
        else
            this.values = values;
        if (title == null)
            title = "";
        else
            this.title = title;
        if (horlabels == null)
            this.horlabels = new String[0];
        else
            this.horlabels = horlabels;
        if (verlabels == null)
            this.verlabels = new String[0];
        else
            this.verlabels = verlabels;
        this.type = type;
        paint = new Paint();
    }

    @Override
    protected void onDraw(Canvas canvas) {
        float border = 20;
        float horstart = border * 2;
        float height = getHeight();
        float width = getWidth() - 1;
        float max = -100f; //getMax();
        float min = 0f;// getMin();
        float diff = max - min;
        float graphheight = height - (2 * border);
```



```
        float graphwidth = width - (2 * border);

        paint.setTextAlign(Align.LEFT);
        int vers = verlabels.length - 1;
        for (int i = 0; i < verlabels.length; i++) {
            paint.setColor(Color.BLUE);
            float y = ((graphheight / vers) * i) + border;
            canvas.drawLine(horstart, y, width, y, paint);
            paint.setColor(Color.WHITE);
            canvas.drawText(verlabels[i], 0, y, paint);
        }
        int hors = horlabels.length - 1;
        for (int i = 0; i < horlabels.length; i++) {
            paint.setColor(Color.BLUE);
            float x = ((graphwidth / hors) * i) + horstart;
            canvas.drawLine(x, height - border, x, border, paint);
        }

        paint.setTextAlign(Align.CENTER);
        canvas.drawText(title, (graphwidth / 2) + horstart, border - 4, paint);

        if (max != min) {
            paint.setColor(Color.RED);
            if (type == BAR) {
                float datalength = values.length;
                float colwidth = (width - (2 * border)) / datalength;
                for (int i = 0; i < values.length; i++) {
                    paint.setColor(Color.RED ^ (255 << 5) *(i + 1) * 75);

                    float val = values[i] - min;
                    float rat = Math.abs(val / diff);
                    float h = graphheight * rat;
                    canvas.drawRect((i * colwidth) + horstart, (border - h) + graphheight,
                            ((i * colwidth) + horstart) + (colwidth - 1), height - (border - 1),
                            paint);

                }
                for (int i = 0; i < values.length; i++) {
                    float val = values[i] - min;
                    float rat = val / diff;
                    float h = graphheight * rat;
                    // Toast.makeText(getContext(),
                    // "val="+val+"rat="+rat+"h="+h,Toast.LENGTH_LONG).show();
                    paint.setTextAlign(Align.CENTER);
                    if (i == horlabels.length - 1)
                        paint.setTextAlign(Align.RIGHT);
                    if (i == 0)
                        paint.setTextAlign(Align.LEFT);
                    paint.setColor(Color.RED ^ (255 <<5) *(i + 1) * 75);

                    canvas.drawText(horlabels[i], ((i * colwidth) + horstart) + (colwidth - 1),
                            (border - h) + graphheight, paint);

                }
            } else {
                float datalength = values.length;
                float colwidth = (width - (2 * border)) / datalength;
                float halfcol = colwidth / 2;
                float lasth = 0;
                for (int i = 0; i < values.length; i++) {
                    float val = values[i] - min;
                    float rat = val / diff;
                    float h = graphheight * rat;
                    if (i > 0)
                        canvas.drawLine(((i - 1) * colwidth) + (horstart + 1) + halfcol,
                                (border - lasth) + graphheight, (i * colwidth) + (horstart + 1)
                                        + halfcol, (border - h) + graphheight, paint);
                    lasth = h;
                }
            }
        }
    }

```

```
147 //  private float getMax() {
148 //      float largest = values[0];
149 //      for (int i = 0; i < values.length; i++)
150 //          if (values[i] > largest)
151 //              largest = values[i];
152 //      return largest;
153 //  }
154 //
155 //  private float getMin() {
156 //      float smallest = values[0];
157 //      for (int i = 0; i < values.length; i++)
158 //          if (values[i] < smallest)
159 //              smallest = values[i];
160 //      return smallest;
161 //  }
162
163 }
164
```

```java
/*
 * (C) Copyright University of Cyprus. 2010-2011.
 *
 * SmartTrace Server API
 *
 * @version         : 1.0
 * @author : Costantinos Costa(costa.costantinos@gmail.com)
 * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
 * Computer Science Department , University of Cyprus
 *
 *
 */
package SmartTrace.apk;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.DialogInterface.OnClickListener;
import android.content.SharedPreferences.Editor;
import android.graphics.Color;
import android.graphics.drawable.Drawable;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.net.ConnectivityManager;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.preference.PreferenceManager;
import android.provider.Settings;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.Toast;

import com.google.android.maps.GeoPoint;
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapController;
import com.google.android.maps.MapView;
import com.google.android.maps.Overlay;
import com.google.android.maps.OverlayItem;

public class MainActivity extends MapActivity {

    // Server and Client Trajectory
    public volatile static ArrayList<myPoint> ClientTrajectory;
```



```java
    public volatile static ArrayList<myPoint> ServerTrajectory;
    // the dialog that is use for drawing the path
    public volatile static ProgressDialog dialog;
    // flag that is used to now if the data comes from
    // a trace or the GPS
    public volatile static boolean useTrace;
    public volatile static boolean GpsStarted;
    public volatile static boolean privacy = false;
    // String that keep the previous file
    public volatile static boolean AlreadyRead = false;
    public volatile static String previous = "";

    public volatile static String custFilePath = "";
    public volatile static String outpFileloc = new String();
    // flags for the map
    public volatile static boolean pin = false;
    public volatile static boolean sat = false;
    public volatile static boolean traffic = false;
    public volatile static boolean strview = false;
    public volatile static int kindofdraw = R.drawable.flag;
    public volatile static String[][] QueryTrajectories = null;

    public volatile static SharedPreferences preferences;
    public volatile static LocationManager locationManager = null;
    public volatile static LocationListener locationListener = null;
    public volatile static boolean ploted;

    public volatile static Handler messageHandler;
    // start and end point for double clicking the home button
    public static ArrayList<GeoPoint> startpoints;
    public static ArrayList<GeoPoint> endpoints;

    public static boolean srchdn;
    public static int next;
    public static int startend;

    protected LinearLayout linearLayout;

    private int Density = 2;
    protected double platidute;
    protected double plongitude;
    // clicks from all the buttons
    public static int clicks;
    public static int clckcol;
    public static int clckpin;
    // variables that is using to flag something
    private boolean append;
    private boolean connected;

    // private variables that only are used here
    private MapView mapView;
    private BufferedWriter writer = null;
    private BufferedReader reader = null;
    private File outFile = null;
    private File inFile = null;
    private List<Overlay> mapOverlays;
    private Drawable drawable;
    private MapController myMC = null;
    private GeoPoint geoPoint = null;
    private MyItemizedOverlay itemizedOverlay;
    private Toast warn;
    private Toast info;
    private Connect con;
    /**
     * All the buttons are private
     */
    private ImageButton btnHome;
    private ImageButton btnClose;
    private ImageButton btnNext;
    // these button need to have a restore function
    private volatile static ImageButton btnCol;

    protected static void restoreCol() {
```

```
147         btnCol.setBackgroundResource(R.drawable.colred);
148     }
149
150     // these button need to have a restore function
151     private volatile static ImageButton btnPin;
152
153     protected static void restorePin() {
154         btnPin.setBackgroundResource(R.drawable.none);
155     }
156
157     // these buttons need to have a restore function
158     private volatile static ImageButton btnWid;
159
160     protected static void restoreWid() {
161         btnWid.setBackgroundResource(R.drawable.width1);
162
163     }
164
165     protected String username = new String();
166     protected static boolean isUsernameSet = false;
167     protected static int K;
168
169     int key = 0;
170
171     private boolean record = false;
172
173     @Override
174     public void onCreate(Bundle savedInstanceState) {
175         // start the main activity
176
177         super.onCreate(savedInstanceState);
178         setContentView(R.layout.map);
179         final PopUp popup = (PopUp) findViewById(R.id.popupGpswindow);
180         popup.setVisibility(View.GONE);
181         popup.setTitle("Do you want to switch to wifi mode?");
182         final Button btn = (Button) findViewById(R.id.show_popup_button);
183         final Button btnCancel = (Button) findViewById(R.id.btnCancel);
184         final Button btnOk = (Button) findViewById(R.id.btnOk);
185         btnCancel.setOnClickListener(new View.OnClickListener() {
186
187             @Override
188             public void onClick(View v) {
189                 // TODO Auto-generated method stub
190                 popup.setVisibility(View.GONE);
191                 btn.setBackgroundResource(R.drawable.leftarrow);
192             }
193         });
194
195         btnOk.setOnClickListener(new View.OnClickListener() {
196
197             @Override
198             public void onClick(View v) {
199                 // TODO Auto-generated method stub
200                 Intent i = new Intent(MainActivity.this, WifiGraph.class);
201                 startActivity(i);
202                 if (con != null) {
203                     con.appClose();
204                 }
205                 appClose();
206             }
207         });
208
209         btn.setOnClickListener(new View.OnClickListener() {
210             @Override
211             public void onClick(View arg0) {
212                 if (key == 0) {
213                     key = 1;
214                     popup.setVisibility(View.VISIBLE);
215                     btn.setBackgroundResource(R.drawable.selectorrigth);
216                 } else if (key == 1) {
217                     key = 0;
218
219                     popup.setVisibility(View.GONE);
```



```
220                     btn.setBackgroundResource(R.drawable.selectorleft);
221                 }
222             }
223         });
224         // initialized Toast messages
225         initMessage();
226         // initialize variables
227         initVariables();
228         // initialize controls
229         initControls(this);
230
231         // initialized the message handler for the messages from the thread
232         messageHandler = new Handler() {
233             public void handleMessage(Message msg) {
234                 super.handleMessage(msg);
235
236                 // Connection mycon;
237                 switch (msg.arg1) {
238                 case Messages.FIRST_PLOT:
239                     warn.setText("Please plot your trajectory first.");
240                     warn.show();
241                     connected = false;
242                     // searched = false;
243                     break;
244                 case Messages.CONNECT:
245                     info.setText("Connecting.");
246                     info.show();
247                     break;
248                 case Messages.INVALID_ADDRESS:
249                     warn.setText("Invalid Address.");
250                     warn.show();
251                     connected = false;
252                     // searched = false;
253                     break;
254                 case Messages.SEARCH:
255                     warn.setText("You are Searching.");
256                     warn.show();
257                     break;
258                 case Messages.DEFAULT_EPSILON:
259                     info.setText("Using default epsilon(0.001).");
260                     info.show();
261                     break;
262                 case Messages.LCSS:
263                     info.setText("Calculating LCSS.!");
264                     info.show();
265                     // PlotSmrt(msg.arg2);
266                     break;
267                 case Messages.UPBOUND:
268                     info.setText("Calculating UB.!");
269                     info.show();
270                     break;
271                 case Messages.ENDSRCH:
272                     info.setText("Searching done!");
273                     info.show();
274                     if (QueryTrajectories == null)
275                         PlotSmrtPrivate(msg.arg2);
276                     else
277                         PlotSmrt(msg.arg2);
278                     // connected = false;
279                     // searched = false;
280                     break;
281                 case Messages.ENDCONN:
282                     info.setText("Connection done!");
283                     info.show();
284                     connected = false;
285                     break;
286                 case Messages.STATECAL:
287                     warn.setText("You are in calculate state.\nPlease try again!");
288                     warn.show();
289                     break;
290                 case Messages.STATERET:
291                     warn.setText("You are in retrieval state.\nPlease try again!");
292                     warn.show();
```

```
293                     break;
294                 case Messages.STATESER:
295                     warn.setText("You are already searching.\nPlease try again!");
296                     warn.show();
297                     break;
298                 case Messages.SERVER_ERR:
299                     warn.setText(Messages.errors.get(msg.arg2));
300                     warn.show();
301                     break;
302                 default:
303                     warn.setText("The Connection was closed.");
304                     warn.show();
305                     connected = false;
306                     // searched = false;
307                     break;
308                 }
309 
310             }
311         };
312         // set The UserName
313         if (!setUsername()) {
314 
315             final AlertDialog.Builder alert = new AlertDialog.Builder(this);
316             final EditText input = new EditText(this);
317             alert.setView(input);
318             alert.setMessage("Please write your Screen Name");
319             alert.setPositiveButton("Ok", new DialogInterface.OnClickListener() {
320                 public void onClick(DialogInterface dialog, int whichButton) {
321                     username = input.getText().toString().trim();
322                     Editor editor = preferences.edit();
323                     editor.putString("usrName", username);
324                     editor.commit();
325                     warn.setText(username);
326                     warn.show();
327                 }
328             });
329 
330             alert.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
331                 public void onClick(DialogInterface dialog, int whichButton) {
332                     dialog.cancel();
333                 }
334             });
335             alert.show();
336 
337         }
338     }
339 
340     private void initMessage() {
341         // TODO Auto-generated method stub
342         // initialized errors list
343         Messages.errors = new HashMap<Integer, String>();
344         fillHashMapError(Messages.errors);
345 
346         // initialize toast for warnings
347         warn = Toast.makeText(MainActivity.this, "", Toast.LENGTH_LONG);
348         View wifitextView = warn.getView();
349         LinearLayout wifilay = new LinearLayout(MainActivity.this);
350         wifilay.setOrientation(LinearLayout.HORIZONTAL);
351         ImageView wifiview = new ImageView(MainActivity.this);
352         wifiview.setImageResource(R.drawable.warning);
353         wifilay.addView(wifiview);
354         wifilay.addView(wifitextView);
355         warn.setView(wifilay);
356         // initialize toast for informations
357         info = Toast.makeText(MainActivity.this, "", Toast.LENGTH_LONG);
358         View textView = info.getView();
359         LinearLayout lay = new LinearLayout(MainActivity.this);
360         lay.setOrientation(LinearLayout.HORIZONTAL);
361         ImageView view = new ImageView(MainActivity.this);
362         view.setImageResource(R.drawable.info);
363         lay.addView(view);
364         lay.addView(textView);
365         info.setView(lay);
```



```
366     }
367 
368     @Override
369     protected void onStart() {
370         // TODO Auto-generated method stub
371         super.onStart();
372 
373         // The only time that a sample is draw
374         if (MainActivity.custFilePath.equals(getString(R.string.sdcard))) {
375             InputStream inStream = getResources().openRawResource(R.raw.trajectory_sample);
376             readFromSampleFile(inStream);
377             myMC = mapView.getController();
378             myMC.setZoom(15);
379             info.setText("This is a sample trajectory");
380             info.show();
381         }
382         // We should redraw the map due the process dialog
383         startMap();
384         getPref();
385     }
386 
387     @Override
388     protected void onResume() {
389         // TODO Auto-generated method stub
390         super.onResume();
391         // We should redraw the map due the process dialog
392         startMap();
393         getPref();
394     }
395 
396     @Override
397     protected boolean isRouteDisplayed() {
398         // TODO Auto-generated method stub
399         return false;
400     }
401 
402     /* Creates the menu items */
403     public boolean onCreateOptionsMenu(Menu menu) {
404 
405         menu.add(0, R.id.record, 0, "Record").setIcon(R.drawable.recorded);
406         menu.add(0, R.id.Cons, 1, "Connect").setIcon(R.drawable.conser);
407         menu.add(0, R.id.SmrtTrc, 2, "Smart Trace").setIcon(R.drawable.smart);
408         menu.add(0, R.id.Share, 3, "Share").setIcon(R.drawable.share);
409         // menu.add(0, R.id.OnOff, 3,
410         // "Online/Offline").setIcon(R.drawable.offgps);
411         // menu.add(0, R.id.mapMode, 0, "Map Mode").setIcon(R.drawable.mode);
412         menu.add(0, R.id.preferences, 4, "Settings").setIcon(R.drawable.prefer);
413         menu.add(0, R.id.more, 5, "More ...").setIcon(R.drawable.more);
414         return true;
415     }
416 
417     public boolean onMenuOpened(int featureId, Menu menu) {
418         // TODO Auto-generated method stub
419         MenuItem item = menu.getItem(1);
420         MenuItem itemrecord = menu.getItem(0);
421         if (!connected) {
422             item.setIcon(R.drawable.conser);
423             item.setTitle("Connect");
424         } else {
425             item.setIcon(R.drawable.disser);
426             item.setTitle("Disconnect");
427         }
428 
429         if (!record) {
430             itemrecord.setIcon(R.drawable.recorded);
431             itemrecord.setTitle("Record");
432         } else {
433             itemrecord.setIcon(R.drawable.record);
434             itemrecord.setTitle("Recording...");
435         }
436         return super.onMenuOpened(featureId, menu);
437     }
438 
```

```
439     // This method is called once the menu is selected
440     public boolean onOptionsItemSelected(MenuItem item) {
441         Intent i = null;
442         Message msg = null;
443         switch (item.getItemId()) {
444         // We have only 5 menu option
445         case R.id.OnOff:
446             // Launch OnOffPreferences activity
447             i = new Intent(MainActivity.this, OnOffPreferences.class);
448             startActivity(i);
449             con = null;
450             break;
451         // Smart trace request to Server with new thread
452         case R.id.SmrtTrc:
453 
454             if (!connected) {
455                 warn.setText("You should connect first!");
456                 warn.show();
457                 break;
458             }
459             msg = new Message();
460             msg.arg1 = Messages.SEARCH;
461             Connect.messageHandler.sendMessage(msg);
462 
463             break;
464         case R.id.Cons:
465             // Connect to Server with new thread
466             final ConnectivityManager connMgr = (ConnectivityManager) this
467                     .getSystemService(Context.CONNECTIVITY_SERVICE);
468             final android.net.NetworkInfo wifi = connMgr
469                     .getNetworkInfo(ConnectivityManager.TYPE_WIFI);
470 
471             if (!connected) {
472                 if (setUsername()) {
473                     if (wifi.isAvailable())
474                         con = new Connect(getApplicationContext(), false, username);
475                     else {
476                         gpsWifiAlert("Wifi");
477                         break;
478                     }
479                 } else {
480                     warn.setText("The username is not set yet!!!");
481                     warn.show();
482                     break;
483                 }
484                 More.terminal = "";
485                 connected = true;
486 
487             } else {
488                 msg = new Message();
489                 msg.arg1 = Messages.QUIT;
490                 Connect.messageHandler.sendMessage(msg);
491                 connected = false;
492             }
493 
494             break;
495         case R.id.more:
496             // Launch more activity
497             i = new Intent(MainActivity.this, More.class);
498             startActivity(i);
499             // con = null;
500             break;
501         case R.id.Share:
502             // Launch Preferences activity
503             i = new Intent(MainActivity.this, Preferences.class);
504             startActivity(i);
505             info.setText("Here are the application's preferences.");
506             info.show();
507             // con = null;
508             break;
509 
510         case R.id.preferences:
511             // Launch Preferences activity
```



```
512             i = new Intent(MainActivity.this, OnOffPreferences.class);
513             startActivity(i);
514             info.setText("Here are the application's settings.");
515             info.show();
516             // con = null;
517             break;
518         case R.id.record:
519 
520             if (useTrace) {
521                 warn.setText("You are using a trace file.\nTo collect point go in online mode");
522                 warn.show();
523                 break;
524             }
525             // Launch ModeOfMap activity
526             if (!record) {
527                 if (!GpsStarted) {
528                     Density = (int) preferences.getInt("Density", 1);
529                     Density = (int) Math.pow(10, Density);
530                     startGps();
531                 }
532                 record = true;
533             } else {
534                 if (GpsStarted)
535                     stopGps();
536                 record = false;
537             }
538             break;
539 
540         }
541         return true;
542     }
543 
544     @Override
545     public void onOptionsMenuClosed(Menu menu) {
546         // TODO Auto-generated method stub
547         super.onOptionsMenuClosed(menu);
548         if (con != null) {
549             con.start();
550             con = null;
551         }
552     }
553 
554     void initVariables() {
555 
556         // initialize important variables
557         connected = false;
558         // searched = false;
559         con = null;
560         ClientTrajectory = new ArrayList<myPoint>();
561         QueryTrajectories = new String[10][];
562         startpoints = new ArrayList<GeoPoint>();
563         endpoints = new ArrayList<GeoPoint>();
564         startpoints.clear();
565         endpoints.clear();
566         ClientTrajectory.clear();
567         clicks = 0;
568         clckcol = 0;
569         clckpin = 0;
570         next = 0;
571         ploted = false;
572         srchdn = false;
573         previous = new String("");
574         DrawPathOverlay.widthOfPath = 2;
575 
576     }
577 
578     // initialize controls
579     private void initControls(MainActivity googleMaps) {
580         btnPin = (ImageButton) findViewById(R.id.btnPin);
581         btnPin.setBackgroundResource(R.drawable.none);
582         btnCol = (ImageButton) findViewById(R.id.btnCol);
583         btnCol.setBackgroundResource(R.drawable.colred);
584         btnClose = (ImageButton) findViewById(R.id.btnclose);
```

```
585         btnClose.setBackgroundResource(R.drawable.close);
586         btnClose.setAdjustViewBounds(true);
587         btnWid = (ImageButton) findViewById(R.id.btnWid);
588         btnWid.setBackgroundResource(R.drawable.width1);
589         btnHome = (ImageButton) findViewById(R.id.btnHome);
590         btnHome.setBackgroundResource(R.drawable.selector_home);
591         btnNext = (ImageButton) findViewById(R.id.btnNext);
592         btnNext.setBackgroundResource(R.drawable.selector_next);
593
594         preferences = PreferenceManager.getDefaultSharedPreferences(this);
595
596         // initialize the input file
597         custFilePath = preferences.getString(EditTextBrowser.PATH, "/sdcard/");
598         // initialized the map view
599         mapView = (MapView) findViewById(R.id.mapview);
600
601         myMC = mapView.getController();
602         myMC.setZoom(8);
603         // initialize the density and gps variable
604         platidute = 0.0;
605         plongitude = 0.0;
606         GpsStarted = false;
607         locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
608         locationListener = new MyLocationListener();
609         // On click listener
610         // Plot the trajectory on the
611         btnHome.setOnClickListener(new Button.OnClickListener() {
612             public void onClick(View v) {
613                 // if we use a trace we should read the trace first
614                 if (useTrace) {
615                     if (!previous.equals(custFilePath) || !ploted) {
616
617                         dialog = ProgressDialog.show(MainActivity.this, "",
618                                 "Loading. Please wait...", true);
619                         dialog.show();
620
621                         Intent i = new Intent(MainActivity.this, PlotPath.class);
622                         startActivity(i);
623                         ++startend;
624                     }
625                 } else if (record || !ploted) {
626                     // start drawing on the map
627                     startMap();
628                     ploted = true;
629                 }
630                 // end-start of the trajectory
631                 startend = (++startend) % 2;
632                 // go to end or start point
633                 goToPoint();
634
635             }
636         });
637         btnPin.setOnClickListener(new Button.OnClickListener() {
638             public void onClick(View v) {
639                 clckpin = (++clckpin % 4);
640                 switch (clckpin) {
641                 case 0:
642                     MainActivity.pin = false;
643                     btnPin.setBackgroundResource(R.drawable.none);
644                     break;
645                 case 1:
646                     MainActivity.pin = true;
647                     MainActivity.kindofdraw = R.drawable.pin;
648                     btnPin.setBackgroundResource(R.drawable.thumbtack);
649
650                     break;
651                 case 2:
652                     MainActivity.pin = true;
653                     MainActivity.kindofdraw = R.drawable.arrow;
654                     btnPin.setBackgroundResource(R.drawable.btnarr);
655                     break;
656                 case 3:
657                     MainActivity.pin = true;
```



```
658                     MainActivity.kindofdraw = R.drawable.flag;
659                     btnPin.setBackgroundResource(R.drawable.btnflag);
660                     break;
661                 default:
662                     break;
663                 }
664                 startMap();
665             }
666         });
667         btnCol.setOnClickListener(new Button.OnClickListener() {
668             public void onClick(View v) {
669                 clckcol = (++clckcol % 6);
670                 switch (clckcol) {
671                 case 0:
672                     DrawPathOverlay.color = Color.RED;
673                     btnCol.setBackgroundResource(R.drawable.colred);
674                     break;
675                 case 1:
676                     DrawPathOverlay.color = Color.GREEN;
677                     btnCol.setBackgroundResource(R.drawable.colgreen);
678                     break;
679                 case 2:
680                     DrawPathOverlay.color = Color.BLUE;
681                     btnCol.setBackgroundResource(R.drawable.colblue);
682                     break;
683                 case 3:
684                     DrawPathOverlay.color = Color.YELLOW;
685                     btnCol.setBackgroundResource(R.drawable.colyel);
686                     break;
687                 case 4:
688                     DrawPathOverlay.color = Color.BLACK;
689                     btnCol.setBackgroundResource(R.drawable.colblk);
690                     break;
691                 case 5:
692                     DrawPathOverlay.color = Color.GREEN / 256;
693                     btnCol.setBackgroundResource(R.drawable.color);
694                     break;
695                 default:
696                     break;
697                 }
698                 startMap();
699             }
700         });
701         btnWid.setOnClickListener(new Button.OnClickListener() {
702             public void onClick(View v) {
703                 clicks = (++clicks % 3);
704                 switch (clicks) {
705                 case 0:
706                     btnWid.setBackgroundResource(R.drawable.width1);
707                     break;
708                 case 1:
709                     btnWid.setBackgroundResource(R.drawable.width2);
710                     break;
711                 case 2:
712                     btnWid.setBackgroundResource(R.drawable.width3);
713                     break;
714                 default:
715                     break;
716                 }
717                 DrawPathOverlay.widthOfPath = clicks * 2 + 2;
718                 startMap();
719             }
720
721         });
722         btnClose.setOnClickListener(new Button.OnClickListener() {
723             public void onClick(View v) {
724                 btnClose.setBackgroundResource(R.drawable.disclose);
725                 AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
726                 builder.setMessage("Are you sure you want to exit?").setCancelable(false)
727                         .setPositiveButton("Yes", new OnClickListener() {
728
729                             @Override
730                             public void onClick(DialogInterface dialog, int which) {
```

```
731                             // TODO Auto-generated method stub
732                             btnClose.setBackgroundResource(R.drawable.close);
733                             appClose();
734
735                         }
736                     }).setNegativeButton("No", new OnClickListener() {
737
738                         @Override
739                         public void onClick(DialogInterface dialog, int which) {
740                             // TODO Auto-generated method stub
741                             btnClose.setBackgroundResource(R.drawable.close);
742                             return;
743
744                         }
745                     });
746             AlertDialog alert = builder.create();
747             alert.show();
748         }
749
750     });
751     btnNext.setOnClickListener(new Button.OnClickListener() {
752         public void onClick(View arg0) {
753             // TODO Auto-generated method stub
754             if (startpoints.size() > 0)
755                 next = (++next % startpoints.size());
756             if (startend == 0) {
757                 if (startpoints.size() > 0)
758                     myMC.setCenter(startpoints.get(next));
759             } else {
760                 if (endpoints.size() > 0)
761                     myMC.setCenter(endpoints.get(next));
762             }
763         }
764
765     });
766 }
767
768 protected void goToPoint() {
769     // TODO Auto-generated method stub
770     if (!ClientTrajectory.isEmpty() && startpoints.size() > 0 && endpoints.size() > 0) {
771         if (startend == 0) {
772             geoPoint = startpoints.get(0);
773         } else {
774             geoPoint = endpoints.get(0);
775         }
776     } else {
777         geoPoint = new GeoPoint(34667805, 33080183);
778         myMC.setZoom(15);
779     }
780     // the next button should be other trajectory but not mine
781     next = 0;
782     // put the center of the map the first coordinate
783     myMC.setCenter(geoPoint);
784 }
785
786 private void appClose() {
787     if (GpsStarted) {
788         Editor editor = preferences.edit();
789         editor.putString("custFile", outpFileloc);
790         editor.commit();
791         try {
792             if (writer != null)
793                 writer.close();
794         } catch (Exception e) {
795             warn.setText("writer Exception:" + e);
796             warn.show();
797         }
798
799         locationManager.removeUpdates(locationListener);
800     }
801     finish();
802     System.exit(0);
803 }
```



```
804
805 private class MyLocationListener implements LocationListener {
806
807     public void onLocationChanged(Location loc) {
808         double latidute = 0.0;
809         double longitude = 0.0;
810         if (loc != null) {
811             try {
812                 latidute = loc.getLatitude();
813                 longitude = loc.getLongitude();
814                 if (!((int) (latidute * Density) - (int) (platidute * Density) == 0 && (int)
(longitude * Density)
815                         - (int) (plongitude * Density) == 0)) {
816                     platidute = latidute;
817
818                     plongitude = longitude;
819                     myPoint temppoint = new myPoint(latidute, longitude);
820                     ClientTrajectory.add(temppoint);
821                     writer.append(latidute + "\t" + longitude + "\n");
822                     warn.setText(
823
824                     "Latitude (X): " + latidute + ",Longitude(Y)=" + longitude);
825                     warn.show();
826                 }
827             } catch (Exception e) {
828                 warn.setText("Exception in onLocationChanged(): " + e.getMessage());
829                 warn.show();
830             }
831         }
832     }
833
834     public void onProviderDisabled(String provider) {
835     }
836
837     public void onProviderEnabled(String provider) {
838     }
839
840     public void onStatusChanged(String provider, int status, Bundle extras) {
841     }
842
843 }
844
845 void readFromFile() {
846     if (AlreadyRead && previous.equals(inFile.getName()))
847         return;
848     // it's the same file
849     previous = inFile.getName();
850
851     // get the Latitude from file and store it to an array
852     try {
853         String line;
854         String[] temp;
855         myPoint temppoint;
856         while ((line = reader.readLine()) != null) {
857             temp = line.split("\t");
858             temppoint = new myPoint(Double.parseDouble(temp[0]), Double.parseDouble(temp[1]));
859             ClientTrajectory.add(temppoint);
860         }
861
862     } catch (Exception e) {
863         warn.setText("Exception while reading from file:\n" + e.getMessage());
864         warn.show();
865     }
866     AlreadyRead = true;
867 }
868
869 void startMap() {
870     startpoints.clear();
871     endpoints.clear();
872     mapView.clearDisappearingChildren();
873     mapView.removeAllViews();
874     mapView.clearAnimation();
875     mapView.setBuiltInZoomControls(true);
```

```
876         mapView.setSatellite(sat);
877         mapView.setTraffic(traffic);
878         mapView.setStreetView(strview);
879         mapOverlays = mapView.getOverlays();
880         mapOverlays.clear();
881         drawable = this.getResources().getDrawable(kindofdraw);
882         // for the map
883         // STARTING POINT
884         if (ClientTrajectory == null || ClientTrajectory.size() <= 0) {
885             return;
886         }
887         GeoPoint startGP = new GeoPoint((int) (ClientTrajectory.get(0).getX() * 1E6),
888                 (int) (ClientTrajectory.get(0).getY() * 1E6));
889         startpoints.add(startGP);
890         geoPoint = startGP;
891         // myMC.setCenter(geoPoint);
892         // myMC.setZoom(15);
893         mapView.getOverlays().add(new DrawPathOverlay(startGP, startGP, 0));
894         // NAVIGATE THE PATH
895         GeoPoint gp1;
896         GeoPoint gp2 = startGP;
897         for (int i = 0; i < ClientTrajectory.size(); i++) {
898             gp1 = gp2;
899             // watch out! For GeoPoint, first:latitude, second:longitude
900             gp2 = new GeoPoint((int) (ClientTrajectory.get(i).getX() * 1E6),
901                     (int) (ClientTrajectory.get(i).getY() * 1E6));
902             mapView.getOverlays().add(new DrawPathOverlay(gp1, gp2, 0));
903         }
904         endpoints.add(gp2);
905         // END POINT
906         mapView.getOverlays().add(new DrawPathOverlay(gp2, gp2, 0));
907         itemizedOverlay = new MyItemizedOverlay(drawable, getApplicationContext());
908         if (pin) {
909             OverlayItem overlayStart = new OverlayItem(startGP, "", "");
910             itemizedOverlay.addOverlay(overlayStart);
911             OverlayItem overlayEnd = new OverlayItem(gp2, "", "");
912             itemizedOverlay.addOverlay(overlayEnd);
913             mapOverlays.add(itemizedOverlay);
914         }
915         if (srchdn)
916             searchMap();
917         mapView.getController().animateTo(startGP);
918         mapView.setBuiltInZoomControls(true);
919         mapView.displayZoomControls(true);
920
921     }
922
923     void getPref() {
924         useTrace = preferences.getBoolean("OffGps", false);
925         append = preferences.getBoolean("Append", false);
926         privacy = preferences.getBoolean("PrivacyTraj", false);
927         Density = (int) preferences.getInt("Density", 1);
928         Density = (int) Math.pow(10, Density);
929     }
930
931     void stopGps() {
932
933         GpsStarted = false;
934         // Close gps
935         custFilePath = outpFileloc;
936         SharedPreferences.Editor ed = preferences.edit();
937         ed.putString(EditTextBrowser.PATH, custFilePath);
938         ed.commit();
939         // EditTextBrowser.txb.setText(custFilePath);
940         try {
941             if (writer != null)
942                 writer.close();
943         } catch (Exception e) {
944             warn.setText("writer Exception:" + e);
945             warn.show();
946         }
947
948         locationManager.removeUpdates(locationListener);
```



```
949         GpsStarted = false;
950         warn.setText(
951
952         "GPS tracking stoped...\nClick Connect to server to upload data.");
953         warn.show();
954
955     }
956
957     private void gpsWifiAlert(final String sett) {
958
959         final AlertDialog.Builder builder = new AlertDialog.Builder(this);
960         builder.setMessage("Your " + sett + " seems to be disabled, do you want to enable it?")
961                 .setCancelable(false).setPositiveButton("Yes",
962                         new DialogInterface.OnClickListener() {
963                             @Override
964                             public void onClick(DialogInterface dialog, int which) {
965                                 // TODO Auto-generated method stub
966                                 if (sett.equals("GPS"))
967                                     startActivity(new Intent(
968                                             Settings.ACTION_LOCATION_SOURCE_SETTINGS));
969                                 else
970                                     startActivity(new Intent(Settings.ACTION_WIFI_SETTINGS));
971
972                             }
973                         }).setNegativeButton("No", new DialogInterface.OnClickListener() {
974                     public void onClick(DialogInterface dialog, int which) {
975                         dialog.cancel();
976                     }
977                 });
978         final AlertDialog alert = builder.create();
979         alert.show();
980     }
981
982     void startGps() {
983         final LocationManager manager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
984
985         if (!manager.isProviderEnabled(LocationManager.GPS_PROVIDER)) {
986             gpsWifiAlert("GPS");
987         }
988
989         if (useTrace) {
990             warn.setText("Uncheck the use trace file check box.\n(menu preferences)");
991             warn.show();
992
993             return;
994         }
995         GpsStarted = true;
996         ClientTrajectory.clear();
997         outpFileloc = preferences.getString("outpFile", "n/a");
998
999         if (outpFileloc.equals("n/a") || outpFileloc.equals("")) {
1000            warn.setText("Provide path/filename for output file.\n(menu preferences)");
1001            warn.show();
1002            return;
1003        }
1004        // initialize the coordinates file
1005        try {
1006            outFile = new File(outpFileloc);
1007            writer = new BufferedWriter(new FileWriter(outFile, append));
1008        } catch (Exception e) {
1009            warn.setText(
1010
1011            "Error when opening output file:\n" + e.getMessage());
1012            warn.show();
1013        }
1014
1015        try {
1016            locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0,
1017                    locationListener);
1018            GpsStarted = true;
1019        } catch (Exception e) {
1020            warn.setText("Unexpected Error on GPS service:\n" + e.getMessage());
1021            warn.show();
```

```
1022        }
1023        warn.setText("GPS tracking started...\nCollecting Position Information...");
1024        warn.show();
1025
1026    }
1027
1028    void searchMap() {
1029
1030        mapView = (MapView) findViewById(R.id.mapview);
1031        mapView.setBuiltInZoomControls(true);
1032
1033        mapView.setSatellite(sat);
1034        mapView.setTraffic(traffic);
1035        mapView.setStreetView(strview);
1036        // for the map
1037        // STARTING POINT
1038        String[] tmppnt;
1039
1040        int length = QueryTrajectories.length;
1041        info.setText("K=" + QueryTrajectories.length);
1042        info.show();
1043        for (int k = 0; k < length; k++) {
1044            mapOverlays = mapView.getOverlays();
1045            tmppnt = QueryTrajectories[k][0].split(" ");
1046            GeoPoint startGP;
1047            try {
1048                startGP = new GeoPoint((int) (Double.parseDouble(tmppnt[0]) * 1E6), (int) (Double
1049                        .parseDouble(tmppnt[1]) * 1E6));
1050            } catch (Exception e) {
1051                // TODO: handle exception
1052                startGP = new GeoPoint(0, 0);
1053
1054            }
1055            startpoints.add(startGP);
1056            myMC = mapView.getController();
1057            geoPoint = startGP;
1058            // myMC.setCenter(geoPoint);
1059            // myMC.setZoom(15);
1060            mapView.getOverlays().add(
1061                    new DrawPathOverlay(startGP, startGP, DrawPathOverlay.color ^ (256 << 5)
1062                            * (k + 1) * 75));
1063            // NAVIGATE THE PATH
1064            GeoPoint gp1;
1065            GeoPoint gp2 = startGP;
1066            for (int i = 0; i < QueryTrajectories[k].length; i++) {
1067                gp1 = gp2;
1068                try {
1069                    tmppnt = QueryTrajectories[k][i].split(" ");
1070                    // watch out! For GeoPoint, first:latitude, second:longitude
1071                    gp2 = new GeoPoint((int) (Double.parseDouble(tmppnt[0]) * 1E6), (int) (Double
1072                            .parseDouble(tmppnt[1]) * 1E6));
1073                } catch (Exception e) {
1074                    continue;
1075                }
1076                mapView.getOverlays().add(
1077                        new DrawPathOverlay(gp1, gp2, DrawPathOverlay.color ^ (256 << 5) * (k + 1)
1078                                * 75));
1079            }
1080            endpoints.add(gp2);
1081            // END POINT
1082            mapView.getOverlays()
1083                    .add(
1084                            new DrawPathOverlay(gp2, gp2, DrawPathOverlay.color ^ (256 << 5)
1085                                    * (k + 1) * 75));
1086            // mapView.getController().animateTo(startGP);
1087
1088        }
1089
1090        mapView.setBuiltInZoomControls(true);
1091        mapView.displayZoomControls(true);
1092
1093    }
1094
```



```
1095    boolean setUsername() {
1096        // Check for any change in preferences
1097        username = MainActivity.preferences.getString("usrName", "");
1098        if (username.length() <= 0 || username.contains(" \t\n\b\"\'){}[];")) {
1099
1100            isUsernameSet = false;
1101            return false;
1102        } else {
1103            isUsernameSet = true;
1104            return true;
1105        }
1106
1107    }
1108
1109    void PlotSmrt(int lcss) {
1110        AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
1111        builder.setMessage("LCSS = " + lcss / 100.0 + "!\nDo you want to plot the trajectory?")
1112                .setCancelable(false).setPositiveButton("Yes", new OnClickListener() {
1113                    @Override
1114                    public void onClick(DialogInterface dialog, int which) {
1115                        // TODO Auto-generated method stub
1116                        srchdn = true;
1117                        startMap();
1118
1119                    }
1120                }).setNegativeButton("No", new OnClickListener() {
1121
1122                    @Override
1123                    public void onClick(DialogInterface dialog, int which) {
1124                        // TODO Auto-generated method stub
1125                        btnClose.setBackgroundResource(R.drawable.close);
1126                        srchdn = false;
1127                        return;
1128
1129                    }
1130                });
1131        AlertDialog alert = builder.create();
1132        alert.show();
1133
1134    }
1135
1136    void PlotSmrtPrivate(int lcss) {
1137        AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
1138        builder.setMessage("LCSS = " + lcss / 100.0 + "!\nBut The trajectory is private")
1139                .setNeutralButton("OK", new OnClickListener() {
1140
1141                    @Override
1142                    public void onClick(DialogInterface dialog, int which) {
1143                        // TODO Auto-generated method stub
1144                        btnClose.setBackgroundResource(R.drawable.close);
1145                        srchdn = false;
1146                        return;
1147
1148                    }
1149                });
1150        AlertDialog alert = builder.create();
1151        alert.show();
1152
1153    }
1154
1155    void readFromSampleFile(InputStream in) {
1156        if (AlreadyRead && MainActivity.previous.equals(MainActivity.custFilePath))
1157            return;
1158        MainActivity.ClientTrajectory.clear();
1159        MainActivity.ploted = true;
1160        // it's the same file
1161        MainActivity.previous = MainActivity.custFilePath;
1162
1163        InputStreamReader isr = new InputStreamReader(in);
1164
1165        BufferedReader br = new BufferedReader(isr);
1166
1167        // get the Latitude from file and store it to an array
```

```
1168        try {
1169            String line;
1170            String[] temp;
1171            myPoint temppoint;
1172            while ((line = br.readLine()) != null) {
1173                temp = line.split("\t");
1174                temppoint = new myPoint(Double.parseDouble(temp[0]), Double.parseDouble(temp[1]));
1175                MainActivity.ClientTrajectory.add(temppoint);
1176            }
1177
1178        } catch (Exception e) {
1179            warn.setText("Exception while reading from file:\n" + e.getMessage());
1180            warn.show();
1181        }
1182        try {
1183            br.close();
1184            isr.close();
1185        } catch (IOException e) {
1186            // TODO Auto-generated catch block
1187            e.printStackTrace();
1188        }
1189
1190    }
1191
1192    void fillHashMapError(HashMap<Integer, String> errors) {
1193        try {
1194
1195            DocumentBuilderFactory docBuilderFactory = DocumentBuilderFactory.newInstance();
1196            DocumentBuilder docBuilder = docBuilderFactory.newDocumentBuilder();
1197            Document doc = docBuilder.parse(getResources().openRawResource(R.raw.errors));
1198
1199            // normalize text representation
1200            doc.getDocumentElement().normalize();
1201
1202            NodeList listOfErrors = doc.getElementsByTagName("error");
1203
1204            for (int s = 0; s < listOfErrors.getLength(); s++) {
1205
1206                Node firstErrorNode = listOfErrors.item(s);
1207                if (firstErrorNode.getNodeType() == Node.ELEMENT_NODE) {
1208
1209                    Element firstErrorElement = (Element) firstErrorNode;
1210
1211                    // -------
1212                    NodeList numberList = firstErrorElement.getElementsByTagName("number");
1213                    Element numberElement = (Element) numberList.item(0);
1214
1215                    NodeList textFNList = numberElement.getChildNodes();
1216                    String number = ((Node) textFNList.item(0)).getNodeValue().trim();
1217
1218                    // -------
1219                    NodeList descriptionList = firstErrorElement
1220                            .getElementsByTagName("description");
1221                    Element descriptionElement = (Element) descriptionList.item(0);
1222
1223                    NodeList textLNList = descriptionElement.getChildNodes();
1224                    String description = ((Node) textLNList.item(0)).getNodeValue().trim();
1225
1226                    errors.put(Integer.parseInt(number), description);
1227
1228                }// end of if clause
1229
1230            }// end of for loop with s var
1231
1232        } catch (Exception e) {
1233            // TODO: handle exception
1234            Toast.makeText(getBaseContext(), "Error: " + e.getMessage(), Toast.LENGTH_SHORT).show();
1235        }
1236    }
1237 }
```

```java
package SmartTrace.apk;

import java.util.HashMap;

public class Messages {
    // define variable for events
    public static final int FIRST_PLOT = 0;
    public static final int CONNECT = 1;
    public static final int INVALID_ADDRESS = 2;
    public static final int SEARCH = 3;
    public static final int DEFAULT_EPSILON = 4;
    public static final int LCSS = 5;
    public static final int UPBOUND = 6;
    public static final int ENDSRCH = 7;
    public static final int ENDCONN = 8;
    public static final int OTHER = 9;
    public static final int SERVER_ERR = 10;
    public static final int STATECAL = 11;
    public static final int STATESER = 12;
    public static final int STATERET = 13;
    public static final int QUIT = 14;
    public static final String CRLF = "\r\n";
    public static  HashMap<Integer, String> errors;
}

```

```
1  /*
2   * (C) Copyright University of Cyprus. 2010-2011.
3   *
4   *Centralize Server API
5   *
6   * @version         : 1.0
7   * @author : Costantinos Costa(costa.costantinos@gmail.com)
8   * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
9   * Computer Science Department , University of Cyprus
10  *
11  *
12  */
13
14 import java.io.BufferedReader;
15 import java.io.File;
16 import java.io.FileReader;
17 import java.io.IOException;
18 import java.util.ArrayList;
19
20 public class mobileLCSS {
21
22     private ArrayList<myPoint> ClientTrajectory = new ArrayList<myPoint>();
23     private ArrayList<myPoint> ServerTrajectory = new ArrayList<myPoint>();
24
25     static final String file1 = "client";
26     static final String file2 = "server";
27     static final int e = 5;
28
29     static double LCSS(ArrayList<myPoint> server, ArrayList<myPoint> client, double e, int de) {
30         // end here
31         // PHASE 1
32 int d=3000;
33         ArrayList<myPoint> serverTraj;
34         ArrayList<myPoint> clientTraj;
35
36         if (server.size() > client.size()) {
37             serverTraj = client;
38             clientTraj = server;
39
40         } else {
41             serverTraj = server;
42             clientTraj = client;
43
44         }
45         double[][] previous= new double[d+1][d+1];
46
47         // Initialize first column and row to assist the DP Table
48
49         for (int i = 1; i < clientTraj.size() + 1; i++) {
50         //  previousLeft = cur;
51             for (int j = i - d; j < i + d; j++) {
52
53                 if (j <= 0)
54                     continue;
55                 if (j > serverTraj.size())
56                     break;
57                 if (containInFolder(clientTraj.get(i - 1).getX(), serverTraj.get(j - 1).getX(), e)
58                         && containInFolder(clientTraj.get(i - 1).getY(), serverTraj.get(j - 1)
59                                 .getY(), e)) {
60                     previous[i%(d+1)][j%(d+1)] = previous[(i-1)%(d+1)][(j-1)%(d+1)] + 1;
61                 } else
62                     previous[i%(d+1)][j%(d+1)] = Math.max(previous[(i-1)%(d+1)][j%(d+1)], previous[i%(d+ ↙
   1)][(j-1)%(d+1)]);
63
64             }
65         }
66         // int k, l;
67         // PHASE 2
68         /*
69          * ////////////////////////////////////////////////////////////////////////
70          * In this phase we can find the path on array L that shows the common
71          * values between the trajectory
72          *// ////////////////////////////////////////////////////////////////////////
```



```
73          // k = clientTraj.size();
74          // l = serverTraj.size();
75          //
76          // while (true) {
77          // if ((k == 0) || (l == 0))
78          // break;
79          //
80          // // Match
81          // if (containInFolder(clientTraj.get(k - 1).getX(),serverTraj.get(l -
82          // 1).getX(),e)
83          // &&containInFolder(clientTraj.get(k - 1).getY(),serverTraj.get(l -
84          // 1).getY(),e))
85          // {
86          // // Move to L[i-1][j-1] in next round
87          // k--;
88          // l--;
89          //
90          // } else {
91          // // Move to max { L[k][l-1],L[k-1][l] } in next round
92          // if (L[k][l - 1] >= L[k - 1][l])
93          // l--;
94          // else
95          // k--;
96          //
97          // }
98          //
99          // }
100         // for (int i = 0; i < L.length; i++) {
101         // for (int j = 0; j < L.length; j++) {
102         // System.out.print("["+L[i][j]+"]");
103         // }
104         // System.out.println();
105         // }
106         //
107
108         double result = previous[d][d];
109
110         return ((result / (double) (Math.min(clientTraj.size(), serverTraj.size()))) * 100);
111     }
112
113     private static boolean containInFolder(double valueC, double valueS, double e) {
114         if ((valueC - e) < valueS && (valueC + e) > valueS)
115             return true;
116         return false;
117
118     }
119
120     public void readFromFile(String custFilePath, ArrayList<myPoint> Trajectory) {
121
122         String line = "";
123
124         File inFile = new File(custFilePath);
125         BufferedReader reader = null;
126         if (!inFile.exists()) {
127             System.out
128                     .println("Provide path/filename for input file that exists.\n(menu preferences)\n");
129
130             return;
131         }
132         try {
133             reader = new BufferedReader(new FileReader(inFile));
134         } catch (IOException e1) {
135             // TODO Auto-generated catch block
136             e1.printStackTrace();
137         }
138
139         // get the Latitude from file and store it to an array
140         try {
141             String[] temp;
142             myPoint temppoint;
143             while ((line = reader.readLine()) != null) {
144                 temp = line.split("\t");
145                 temppoint = new myPoint(Double.parseDouble(temp[0]), Double.parseDouble(temp[1]));
```

```
146                 Trajectory.add(temppoint);
147             }
148
149         } catch (Exception e) {
150             System.out.println("Exception while reading from file:\n" + e.getMessage());
151         }
152     }
153
154     public static void main(String[] args) {
155         mobileLCSS m = new mobileLCSS();
156
157         m.readFromFile("./traces/" + file1 + ".txt", m.ClientTrajectory);
158         m.readFromFile("./traces/" + file2 + ".txt", m.ServerTrajectory);
159
160         System.out.println("LCSS = "
161                 + (((int) (mobileLCSS.LCSS(m.ClientTrajectory, m.ServerTrajectory, e, 1) * 100)))
162                 / 100.0);
163
164     }
165
166 }
167
```

```
/*
 * (C) Copyright University of Cyprus. 2010-2011.
 *
 * SmartTrace Server API
 *
 * @version         : 1.0
 * @author : Costantinos Costa(costa.costantinos@gmail.com)
 * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
 * Computer Science Department , University of Cyprus
 *
 *
 */
package SmartTrace.apk;

import SmartTrace.apk.R;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.graphics.Color;
import android.os.Bundle;
import android.text.method.ScrollingMovementMethod;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class More extends Activity {
    public static String terminal = "";
    private TextView textView;
    private Button btnClear;
    private Button btnAbout;
    private Button btnBack;
    private Button btncon;
    private Button btnMod;
    final CharSequence[] items = { "Map", "Satellite", "Traffic", "Street View" };
    AlertDialog alert;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.more);
        textView = (TextView) findViewById(R.id.textView);
        textView.setMovementMethod(new ScrollingMovementMethod());
        textView.setText("");
        btnClear = (Button) findViewById(R.id.clear);
        btnClear.setBackgroundResource(R.drawable.selector_clear);
        btnAbout = (Button) findViewById(R.id.about);
        btnMod = (Button) findViewById(R.id.mode);
        btnMod.setBackgroundResource(R.drawable.selector_mode);
        btnAbout.setBackgroundResource(R.drawable.selector_about);
        btnBack = (Button) findViewById(R.id.back);
        btnBack.setBackgroundResource(R.drawable.selector_back);
        btncon = (Button) findViewById(R.id.btncon);
        btncon.setBackgroundResource(R.drawable.selector_terminal);
        initComp();
    }

    private void initComp() {
        textView.setScrollContainer(true);




        btnClear.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                textView.setTextColor(Color.WHITE);
                textView.setText("Clearing...\n\n");
                MainActivity.pin = false;
                MainActivity.sat = false;
                MainActivity.traffic = false;
                MainActivity.strview = false;
                DrawPathOverlay.color=Color.RED;
```



```
                // initialize important variables
                MainActivity.QueryTrajectories = new String[10][];
                MainActivity.startpoints.clear();
                MainActivity.endpoints.clear();
                MainActivity.ClientTrajectory.clear();
                MainActivity.clicks = 0;
                MainActivity.clckcol = 0;
                MainActivity.clckpin = 0;
                MainActivity.next = 0;
                MainActivity.ploted = false;
                MainActivity.srchdn = false;
                MainActivity.previous = new String("");
                MainActivity.restoreCol();
                MainActivity.restorePin();
                MainActivity.restoreWid();
                DrawPathOverlay.widthOfPath = 2;
                finish();
            }
        });
        btnAbout.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                textView.setTextColor(0xFF00EE76);
                textView.setText("\n\nThe application created by...\n\n" );
                textView.append("\tCosta Costantinos\n");
                textView.append("\tZeinalipour Demetrios\n");
                textView.append("\tLaoudias Christos\n");
                textView.append("\n\t\tThank you\n");


            }
        });
        btnBack.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                finish();
            }
        });
        btncon.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                textView.setTextColor(Color.WHITE);
                if(terminal.equals(""))
                    textView.setText("None transaction yet...");
                    else
                textView.setText(terminal);
            }
        });
        btnMod.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                AlertDialog.Builder builder = new AlertDialog.Builder(v.getContext());
                builder.setTitle("Select Mode");
                builder.setIcon(R.drawable.mode);
                builder.setSingleChoiceItems(items, -1,
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog, int item) {
                                Toast.makeText(getApplicationContext(), items[item],
                                        Toast.LENGTH_SHORT).show();
                                switch (item) {
                                case 0:
                                    MainActivity.strview = false;
                                    MainActivity.traffic = false;
                                    MainActivity.sat = false;
                                    break;
                                case 1:
                                    MainActivity.strview = false;
                                    MainActivity.traffic = false;
                                    MainActivity.sat = true;
                                    break;
                                case 2:
                                    MainActivity.strview = false;
                                    MainActivity.traffic = true;
                                    MainActivity.sat = false;
                                    break;
                                case 3:
                                    MainActivity.strview = true;
```

```
                                MainActivity.traffic = false;
                                MainActivity.sat = false;
                                break;
                            default:
                                break;
                            }
                            dialog.dismiss();
                        }
                        });
                ////////////////////////////////////////////////
                alert = builder.create();
                alert.show();
            }

        });

}
}
```

```java
/*
 * (C) Copyright University of Cyprus. 2010-2011.
 *
 * SmartTrace Server API
 *
 * @version         : 1.0
 * @author : Costantinos Costa(costa.costantinos@gmail.com)
 * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
 * Computer Science Department , University of Cyprus
 *
 *
 */
package SmartTrace.apk;

import java.util.ArrayList;
import android.content.Context;
import android.graphics.drawable.Drawable;
import android.widget.Toast;
import com.google.android.maps.ItemizedOverlay;
import com.google.android.maps.OverlayItem;

public class MyItemizedOverlay extends ItemizedOverlay<OverlayItem> {

    private ArrayList<OverlayItem> mOverlays = new ArrayList<OverlayItem>();
    private Context mContext;

    public MyItemizedOverlay(Drawable defaultMarker, Context context) {
        super(boundCenterBottom(defaultMarker));
        mContext = context;

        // TODO Auto-generated constructor stub
    }

    @Override
    protected OverlayItem createItem(int i) {
        // TODO Auto-generated method stub
        return mOverlays.get(i);

    }

    @Override
    public int size() {
        // TODO Auto-generated method stub
        return mOverlays.size();
    }

    public void addOverlay(OverlayItem overlay) {

        mOverlays.add(overlay);
        populate();
    }

    @Override
    protected boolean onTap(int pIndex) {
        OverlayItem item = mOverlays.get(pIndex);
        Toast.makeText(mContext, "(" + item.routableAddress() + ")",
                Toast.LENGTH_LONG).show();
        return true;

    }
    public MyItemizedOverlay(Context context, Drawable defaultMarker) {
        super(boundCenterBottom(defaultMarker));
        mContext = context;
    }

}
```

```
1  /*
2   * (C) Copyright University of Cyprus. 2010-2011.
3   *
4   * Centralize Server API
5   *
6   * @version         : 1.0
7   * @author : Costantinos Costa(costa.costantinos@gmail.com)
8   * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
9   * Computer Science Department , University of Cyprus
10  *
11  *
12  */
13 public class myPoint {
14     private Double x;
15     private Double y;
16
17     myPoint(Double x, Double y) {
18         this.x = x;
19         this.y = y;
20     }
21
22     public Double getX() {
23         return x;
24     }
25
26     public Double getY() {
27         return y;
28     }
29
30 }
31
```

```java
/*
 * (C) Copyright University of Cyprus. 2010-2011.
 *
 * SmartTrace Server API
 *
 * @version         : 1.0
 * @author : Costantinos Costa(costa.costantinos@gmail.com)
 * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
 * Computer Science Department , University of Cyprus
 *
 *
 */
package SmartTrace.apk;

import SmartTrace.apk.R;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.preference.CheckBoxPreference;
import android.preference.EditTextPreference;
import android.preference.Preference;
import android.preference.PreferenceActivity;
import android.preference.PreferenceManager;
import android.preference.Preference.OnPreferenceClickListener;

public class OnOffPreferences extends PreferenceActivity {
    /** Called when the activity is first created. */
    String ListPreference;
    String editTextPreference;
    String ringtonePreference;
    String secondEditTextPreference;
    String customPref;
    SharedPreferences prefs;
    CheckBoxPreference onPref;
    CheckBoxPreference offPref;
    EditTextPreference etpUser;
    Preference OffCategory;
    Preference OnCategory;
    Preference InputFile;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.onoffprefers);
        // android:key="OffCategory"
        prefs = PreferenceManager.getDefaultSharedPreferences(getBaseContext());
        OffCategory = (Preference) findPreference("OffCategory");
        OnCategory = (Preference) findPreference("OnCategory");
        InputFile = (Preference) findPreference("custFile");
        onPref = (CheckBoxPreference) findPreference("OnGps");

        // when a click event is creating
        offPref = (CheckBoxPreference) findPreference("OffGps");
        offPref.setOnPreferenceClickListener(new OnPreferenceClickListener() {
            public boolean onPreferenceClick(Preference preference) {
                checkOff();
                return true;
            }

        });

        onPref.setOnPreferenceClickListener(new OnPreferenceClickListener() {
            public boolean onPreferenceClick(Preference preference) {
                checkOn();
                return true;
            }

        });

        InputFile.setOnPreferenceClickListener(new OnPreferenceClickListener() {

            @Override
            public boolean onPreferenceClick(Preference preference) {
                // TODO Auto-generated method stub
```



```java
                // Launch OnOffPreferences activity
                Intent i = new Intent(OnOffPreferences.this, EditTextBrowser.class);
                startActivity(i);
                return false;
            }
        });
    }

    @Override
    protected void onStart() {
        // TODO Auto-generated method stub
        super.onStart();
        checkOff();
        checkOn();
    }

    public void checkOff() {
        if (offPref.isChecked()) {
            offPref.setChecked(true);
            OffCategory.setEnabled(true);
            onPref.setChecked(false);
            OnCategory.setEnabled(false);
            onPref.setEnabled(true);
        } else {
            onPref.setChecked(true);
            OnCategory.setEnabled(true);
            offPref.setChecked(false);
            OffCategory.setEnabled(false);
            offPref.setEnabled(true);
        }
    }

    public void checkOn() {
        if (onPref.isChecked()) {
            onPref.setChecked(true);
            OnCategory.setEnabled(true);
            offPref.setChecked(false);
            OffCategory.setEnabled(false);
            offPref.setEnabled(true);
        } else {
            offPref.setChecked(true);
            OffCategory.setEnabled(true);
            onPref.setChecked(false);
            OnCategory.setEnabled(false);
            onPref.setEnabled(true);
        }
    }
}

```

```
1  /*
2   * (C) Copyright University of Cyprus. 2010-2011.
3   *
4   * SmartTrace Server API
5   *
6   * @version         : 1.0
7   * @author : Costantinos Costa(costa.costantinos@gmail.com)
8   * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
9   * Computer Science Department , University of Cyprus
10  *
11  *
12  */
13 package SmartTrace.apk;
14
15 import SmartTrace.apk.R;
16 import android.app.Activity;
17 import android.os.Bundle;
18 import android.view.View;
19 import android.view.View.OnClickListener;
20 import android.widget.ImageButton;
21
22 public class Permision extends Activity {
23     @Override
24     protected void onCreate(Bundle savedInstanceState) {
25     // TODO Auto-generated method stub
26     super.onCreate(savedInstanceState);
27     setContentView(R.layout.perdin);
28     ImageButton img = (ImageButton) findViewById(R.id.Img);
29     img.setBackgroundResource(R.drawable.error);
30     ImageButton back = (ImageButton) findViewById(R.id.btnback);
31     back.setBackgroundResource(R.drawable.selector_back);
32     back.setOnClickListener(new OnClickListener() {
33         @Override
34         public void onClick(View v) {
35         // TODO Auto-generated method stub
36         finish();
37         }
38     });
39     }
40
41     @Override
42     protected void onPause() {
43     // TODO Auto-generated method stub
44     super.onPause();
45     finish();
46     }
47 }
48
```

```
/*
 * (C) Copyright University of Cyprus. 2010-2011.
 *
 * SmartTrace Server API
 *
 * @version         : 1.0
 * @author : Costantinos Costa(costa.costantinos@gmail.com)
 * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
 * Computer Science Department , University of Cyprus
 *
 *
 */
package SmartTrace.apk;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;

import SmartTrace.apk.R;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.Toast;

public class PlotPath extends Activity {
    private boolean AlreadyRead = false;
    myPoint[] trajectory = null;
    private File inFile = null;
    BufferedWriter writer = null;
    BufferedReader reader = null;
    private String ClientTRAJ;
    Toast toast;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.splash);
        toast = Toast.makeText(PlotPath.this, "", Toast.LENGTH_LONG);
        View textView = toast.getView();
        LinearLayout lay = new LinearLayout(PlotPath.this);
        lay.setOrientation(LinearLayout.HORIZONTAL);
        ImageView view = new ImageView(PlotPath.this);
        view.setImageResource(R.drawable.warning);
        lay.addView(view);
        lay.addView(textView);
        toast.setView(lay);

    }

    @Override
    protected void onStart() {
        // TODO Auto-generated method stub
        super.onStart();
        // check for any change in preferences
        if (MainActivity.ClientTrajectory == null)
            MainActivity.ClientTrajectory = new ArrayList<myPoint>();
        if (!MainActivity.useTrace) {
            if (MainActivity.ClientTrajectory.size() != 0) {
                appClose();
            } else {
                toast.setText("Press GPS button to collect coordinates");
                toast.show();
                appClose();
            }
            return;
        }
        try {
```



```
            inFile = new File( MainActivity.custFilePath);
            if (!inFile.exists()) {
                toast
                        .setText("Provide path/filename for input file that exists.\n(menu preferences)\n");
                toast.show();
                appClose();
                return;
            }
            if ( MainActivity.custFilePath.equals("n/a") ||  MainActivity.custFilePath.equals("")) {
                toast.setText("Using default trajectory file (GPS data).");
                toast.show();
                inFile = new File( MainActivity.outpFileloc);
            } else {
                toast.setText("Using trajectory file: " +  MainActivity.custFilePath + ".");
                toast.show();
                appClose();
            }

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

        } catch (Exception e) {
            toast.setText("Error when opening file:" + e.getMessage());
            toast.show();
            appClose();
        }
        // save the file for check if it is already processed
        readFromFile();
        appClose();
    }

    void readFromFile() {
        if (AlreadyRead && MainActivity.previous.equals(MainActivity.custFilePath))
            return;
        MainActivity.ClientTrajectory.clear();
        MainActivity.ploted=true;
        // it's the same file
        MainActivity.previous = MainActivity.custFilePath;
        ClientTRAJ = "";
        // get the Latitude from file and store it to an array
        try {
            String line;
            String[] temp;
            myPoint temppoint;
            int i = 0;
            while ((line = reader.readLine()) != null) {
                temp = line.split("\t");
                temppoint = new myPoint(Double.parseDouble(temp[0]), Double
                        .parseDouble(temp[1]));
                MainActivity.ClientTrajectory.add(temppoint);
                ClientTRAJ += MainActivity.ClientTrajectory.get(i).getX() + " "
                        + MainActivity.ClientTrajectory.get(i++).getY() + "\t";
            }

        } catch (Exception e) {
            toast.setText("Exception while reading from file:\n"
                    + e.getMessage());
            toast.show();
        }
        AlreadyRead = true;

        try {
            reader.close();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            toast.setText("Exception while closing file:\n"
                + e.getMessage());
        toast.show();
        }

    }

    void appClose() {
```

```
146         MainActivity.dialog.dismiss();
147         finish();
148     }
149 }
150
```

```java
/*
 * (C) Copyright University of Cyprus. 2010-2011.
 *
 * SmartTrace Server API
 *
 * @version         : 1.0
 * @author : Costantinos Costa(costa.costantinos@gmail.com)
 * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
 * Computer Science Department , University of Cyprus
 *
 *
 */
package SmartTrace.apk;


import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.RectF;
import android.graphics.Paint.Style;
import android.util.AttributeSet;
import android.view.Gravity;
import android.widget.LinearLayout;
import android.widget.TextView;


public class PopUp extends LinearLayout
{
    private Paint   innerPaint, borderPaint ;

    public PopUp(Context context, AttributeSet attrs) {
        super(context, attrs);
            init();
    }

    public PopUp(Context context) {
        super(context);
            init();
    }

    private void init() {

        innerPaint = new Paint();
        innerPaint.setARGB(255, 75, 75, 75); //gray
        innerPaint.setAntiAlias(true);
        innerPaint.setAlpha(0);
        borderPaint = new Paint();
        borderPaint.setARGB(255, 255, 255, 255);
        borderPaint.setAntiAlias(true);
        borderPaint.setStyle(Style.STROKE);
        borderPaint.setStrokeWidth(2);
    }

    public void setInnerPaint(Paint innerPaint) {
        this.innerPaint = innerPaint;
    }
    public void setTitle(String s) {
        TextView popTitle=(TextView)findViewById(R.id.txtTitle);
        popTitle.setText(s);
      popTitle.setGravity(Gravity.CENTER_HORIZONTAL);
    }
    public void setBorderPaint(Paint borderPaint) {
        this.borderPaint = borderPaint;
    }

    @Override
    protected void dispatchDraw(Canvas canvas) {

        RectF drawRect = new RectF();
        drawRect.set(0,0, getMeasuredWidth(), getMeasuredHeight());

        canvas.drawRoundRect(drawRect, 5, 5, innerPaint);
        canvas.drawRoundRect(drawRect, 5, 5, borderPaint);
```



```java

        super.dispatchDraw(canvas);
    }
}
```

```java
/*
 * (C) Copyright University of Cyprus. 2010-2011.
 *
 * SmartTrace Server API
 *
 * @version         : 1.0
 * @author : Costantinos Costa(costa.costantinos@gmail.com)
 * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
 * Computer Science Department , University of Cyprus
 *
 *
 */
package SmartTrace.apk;

import SmartTrace.apk.R;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.preference.EditTextPreference;
import android.preference.Preference;
import android.preference.PreferenceActivity;
import android.preference.Preference.OnPreferenceChangeListener;
import android.view.View;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.Toast;

public class Preferences extends PreferenceActivity {
    private Toast toast;
    private EditTextPreference etpUser;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.prefers);
        etpUser= (EditTextPreference) findPreference("usrName");
        toast = Toast.makeText(this, "", Toast.LENGTH_LONG);
        View textView = toast.getView();
        LinearLayout lay = new LinearLayout(this);
        lay.setOrientation(LinearLayout.HORIZONTAL);
        ImageView view = new ImageView(this);
        view.setImageResource(R.drawable.warning);
        lay.addView(view);
        lay.addView(textView);
        toast.setView(lay);
        etpUser.setOnPreferenceChangeListener(new OnPreferenceChangeListener() {

            @Override
            public boolean onPreferenceChange(Preference preference, Object newValue) {
                // TODO Auto-generated method stub
                //
                // Check for any change in preferences

                String username = newValue.toString();
                        if (username.length() <= 0 || username.contains(" \t\n\b\"\'(){}[];")) {
                            toast.setText("Invalid Username " + username);
                            toast.show();
                            SharedPreferences.Editor ed = preference.getEditor();
                            ed.putString("usrName", "");
                            ed.commit();
                            toast.setText("Invalid Username");
                            toast.show();
                        } else {
                            MainActivity.isUsernameSet = true;
                            toast.setText("Thank you " + username);
                            toast.show();
                        }
                return true;
            }
        });
    }

}
```


74

```
/*
 * (C) Copyright University of Cyprus. 2010-2011.
 *
 * SmartTrace Server API
 *
 * @version         : 1.0
 * @author : Costantinos Costa(costa.costantinos@gmail.com)
 * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
 * Computer Science Department , University of Cyprus
 *
 *
 */
package cs.ucy.ac.cy;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.Socket;
import java.util.ArrayList;

import android.content.Context;
import android.os.Handler;
import android.os.Message;
import android.widget.Toast;

public class Search extends Thread {

    enum state {
    CLOSED, ESTABLISH, CALCULATE, SEARCH, RETRIEVE;
    }

    myPoint[] trajectory = null;
    private File inFile = null;
    BufferedReader reader = null;
    Toast toast;
    private BufferedWriter writer = null;
    private DataOutputStream out;
    private BufferedReader in;
    private String serverAddress = new String();
    private String epsilon = new String();
    private String usr = new String();
    private Socket connection = null;
    private String ClientTRAJ;
    private double lcss;
    private String SSID;
    private String QUID;
    private String LOG;
    private String INFILE;
    String host = new String();
    String buf = new String();
    String[] traj = null;
    InputStream inStream = null;
    OutputStream outStream = null;
    private String psw = "";
    private String k="1";
    private String LCSS;
    private int port = 65000;
    private String serverPort = "65000";
    private ArrayList<myPoint> ClientTrajectory = new ArrayList<myPoint>();
    private ArrayList<myPoint> ServerTrajectory;
    static state curState = state.CLOSED;
    static Handler messageHandler;

    private String rmPREFIX(String msg) {
    String[] sArr = msg.split(" ");
    String temp = new String();
    for (int i = 1; i < sArr.length; i++)
```



```
        temp += sArr[i] + " ";
    return temp;
    }

    // FUNCTION FOR THE MESSAGES
    static public void Send(String msg, OutputStream out) throws IOException {
    out.write((msg + Messages.CRLF).getBytes());
    out.flush();
    }

    Search(Context c, boolean flag,String k, String username, String port, String ip,
        String log, String in, String e) {
    // set username
//  SmartTraceThreaded.txtMsg.setText("Trying to Connect to " + ip + ":"
//      + port);
    usr = username;
    serverAddress = ip;
    serverPort = port;
    epsilon = e;
    LOG = log;
    INFILE = in;
    this.k=k;
    // initialized the message handler for the messages from the thread
    messageHandler = new Handler() {

        public void handleMessage(Message msg) {
        super.handleMessage(msg);
        Message msg1 = new Message();
        switch (msg.arg1) {
        // These means that the client want to use smartTrace search
        // We create new thread so the GUI doesn't block
        case Messages.SEARCH:

            // // TODO Auto-generated method stub
            if (!curState.equals(state.ESTABLISH)) {
            switch (curState) {
            case CALCULATE:
                msg1.arg1 = Messages.STATECAL;
                break;
            case SEARCH:
                msg1.arg1 = Messages.STATESER;
                break;
            case RETRIEVE:
                msg1.arg1 = Messages.STATERET;
                break;

            default:
                msg1.arg1 = Messages.SERVER_ERR;
                break;
            }

            return;
            }

            try {
            Send(("SEARCH " + SmartTraceThreaded.parK + " " + SSID
                + " " + ClientTRAJ), out);
            } catch (IOException e) {
            // send a massage to main thread
            SmartTraceThreaded.terminal += "Client send : SEARCH "
                + SSID + " Client TRAJECTORY \n";
            }
            // write everything to the transaction string
            synchronized (SmartTraceThreaded.terminal) {
            SmartTraceThreaded.terminal += "Client send : SEARCH "
                + SSID + " Client TRAJECTORY \n";

            }

            break;
        case Messages.QUIT:

            if (connection == null) {
```

```
147             // appClose();
148             return;
149             }
150             try {
151             Send("QUIT", out);
152             connection.close();
153             } catch (IOException e) {
154             // TODO Auto-generated catch block
155             e.printStackTrace();
156             }
157             synchronized (SmartTraceThreaded.terminal) {
158             SmartTraceThreaded.terminal += "Client send : QUIT \n";
159
160             }
161             appClose();
162             return;
163
164         default:
165
166             break;
167         }
168
169         }
170     };
171     this.setUncaughtExceptionHandler(new UncaughtExceptionHandler() {
172
173         @Override
174         public void uncaughtException(Thread thread, Throwable ex) {
175         // TODO Auto-generated method stub
176
177     //  SmartTraceThreaded.txtMsg.setText(SmartTraceThreaded.terminal);
178         }
179     });
180     }
181
182     public void run() {
183
184     // just connect to the server
185     serverConnect();
186     }
187
188     // function for the ESTABLISH STATE
189     private boolean ESTABLISH() throws IOException {
190     curState = state.ESTABLISH;
191     // GET THE +OK READY FROM SERVER
192     buf = in.readLine();
193     SmartTraceThreaded.terminal += "Server Response: " + buf + "\n";
194     // Wait For Proceeding
195     // Get The Session Id
196     buf = "USER " + usr + " " + psw;
197     // SEND A +OK REPLY and user-name TO SERVER
198     Send(buf, out);
199     SmartTraceThreaded.terminal += "Client Send: " + buf + "\n";
200     // GET THE +OK or -ERR FROM SERVER
201     buf = in.readLine();
202     SmartTraceThreaded.terminal += "Server Response: " + buf + "\n";
203     if (buf.startsWith("+OK")) {
204         // Set SessionID
205         SSID = rmPREFIX(buf);
206         return true;
207     }
208     return false;
209
210     }
211
212     // Function For the SEARCH STATE
213     private boolean SEARCH() throws IOException {
214     curState = state.SEARCH;
215     // Read The Message To Continue
216     SmartTraceThreaded.terminal += "Server Response: " + buf + "\n";
217     // Set SessionID
218     QUID = rmPREFIX(buf);
219     // Read The Message To Continue
```



```
220     buf = in.readLine();
221     synchronized (SmartTraceThreaded.terminal) {
222         SmartTraceThreaded.terminal += "Server Response: " + buf + "\n";
223     }
224     if (buf.contains("Goodbye"))
225         return false;
226     LCSS = rmPREFIX(buf);
227     return true;
228     }
229
230     // Function For the CALCULATE STATE
231     private boolean CALCULATE() throws IOException {
232     if (!buf.startsWith("CALCULATE"))
233         return true;
234     curState = state.CALCULATE;
235     synchronized (SmartTraceThreaded.terminal) {
236         SmartTraceThreaded.terminal += "Server Response: "
237             + "CALCULATE queryTrajectory" + "\n";
238     }// toast.show();
239     try {
240         QUID = buf.split(" ")[1];
241     } catch (Exception e) {
242         // TODO: handle exception
243         return false;
244     }
245     String query = rmPREFIX(buf);
246     buf = "+OK CALCULATE";
247     Send(buf, out);
248     synchronized (SmartTraceThreaded.terminal) {
249         SmartTraceThreaded.terminal += "Client send :" + buf + "\n";
250     }
251     // split the points
252     traj = query.split("\t");
253     if (traj == null) {
254         connection.close();
255         return true;
256     }
257
258     ServerTrajectory = new ArrayList<myPoint>();
259     String[] tmppnt;
260     for (int j = 0; j < traj.length; j++) {
261         tmppnt = traj[j].split(" ");
262         // synchronized (SmartTraceThreaded.terminal) {
263         // SmartTraceThreaded.terminal += traj[j] + "\n";
264         // }
265         if (tmppnt != null)
266
267         if (tmppnt.length > 1)
268             try {
269             ServerTrajectory.add(new myPoint(Double
270                 .parseDouble(tmppnt[0]), Double
271                 .parseDouble(tmppnt[1])));
272             // ClientTrajectory.add(new myPoint(Double
273             // .parseDouble(tmppnt[0]), Double
274             // .parseDouble(tmppnt[1])));
275             } catch (Exception e) {
276             // TODO: handle exception
277             continue;
278             }
279     }
280
281     // CALCULATE THE UB
282
283     // SEND THE UB TO SERVER
284     double eps = 0.1;
285     try {
286         eps = Double.parseDouble(epsilon);
287     } catch (Exception e) {
288         // TODO: handle exception
289     }
290     buf = "+OK " + QUID + " "
291         + UB.upperBound(ServerTrajectory, ClientTrajectory, eps);
292     Send(buf, out);
```

```
293     synchronized (SmartTraceThreaded.terminal) {
294         SmartTraceThreaded.terminal += "Client send :" + buf + "\n";
295     }
296     // Wait for the Server to send "LCSS" OR "CLOSE"
297     buf = in.readLine();
298     synchronized (SmartTraceThreaded.terminal) {
299         SmartTraceThreaded.terminal += "Server Response: " + buf + ".\n";
300     }
301     // IF LCSS SEND THE Coordinates[] ARRAY
302     if (buf.startsWith("LCSS")) {
303         synchronized (SmartTraceThreaded.terminal) {
304         SmartTraceThreaded.terminal += "Calculate LCCS and send it to server...\n";
305         }
306         lcss = mobileLCSS.LCSS(ServerTrajectory, ClientTrajectory, eps,1);
307         ClientTRAJ = "";
308         for (int i = 0; i < ClientTrajectory.size(); i++) {
309         ClientTRAJ += ClientTrajectory.get(i).getX() + " "
310             + ClientTrajectory.get(i).getY() + "\t";
311         }
312 
313         buf = "+OK " + QUID + " " + lcss + " " + ClientTRAJ;
314         Send(buf, out);
315         synchronized (SmartTraceThreaded.terminal) {
316         SmartTraceThreaded.terminal += "Client send :" + buf + "\n";
317         }
318     }
319     return true;
320 
321     }
322     // Function For the CALCULATE STATE
323     private boolean RETRIEVE() throws IOException {
324     curState = state.RETRIEVE;
325     // send RETRIEVE message to the server
326     Send("RETRIEVE " + QUID, out);
327     SmartTraceThreaded.terminal += "Client send :" + "RETRIEVE" + "\n";
328     buf = in.readLine();
329     if (!buf.startsWith("+OK"))
330         return true;
331     buf = rmPREFIX(buf);
332     SmartTraceThreaded.terminal += "Server response : [" + buf.split("\t")
333         + "]\n";
334     return true;
335 
336     }
337 
338     // Connect to server to use the upper bound
339     private void serverConnect() {
340 
341     try {
342         // ACCEPT FROM SERVER
343         host = serverAddress;
344         try {
345         port = Integer.parseInt(serverPort);
346         } catch (Exception e) {
347         // TODO: handle exception
348         port = 80;
349         }
350         connection = new Socket(host, port);
351         synchronized (SmartTraceThreaded.terminal) {
352         SmartTraceThreaded.terminal += "Socket created.\nConnecting to server on "
353             + host + "...\n";
354 
355         }
356         // INITIALIZE THE IN AND OUT STREAMS
357         inStream = connection.getInputStream();
358         in = new BufferedReader(new InputStreamReader(inStream));
359         outStream = connection.getOutputStream();
360         out = new DataOutputStream(outStream);
361     } catch (IOException e) {
362     //    SmartTraceThreaded.txtMsg.setText(SmartTraceThreaded.terminal);
363         return;
364     }
365     // -------------------------------ESTABLISH-STATE----------------------------------------
```



```
366     try {
367         // If the ESTABLISH-STATE is wrong
368         if (!ESTABLISH()) {
369         ///SmartTraceThreaded.txtMsg.setText(SmartTraceThreaded.terminal);
370         appClose();
371         return;
372         }
373 
374     } catch (IOException e) {
375         synchronized (SmartTraceThreaded.terminal) {
376         SmartTraceThreaded.terminal += "Error Connection: "
377             + e.getMessage();
378         }
379         appClose();
380 //      SmartTraceThreaded.txtMsg.setText(SmartTraceThreaded.terminal);
381         return;
382     }
383     // -------------------------------ESTABLISH-STATE----------------------------------------
384     boolean once=true;
385     while (once) {
386         once=false;
387         // SEARCH CLIENT
388         // READ FROM FILE
389         readFromFile();
390         try {
391         Send(("SEARCH "+k+" " + SSID + " " + ClientTRAJ), out);
392         } catch (IOException e1) {
393         // TODO Auto-generated catch block
394         //e1.printStackTrace();
395         }
396         // the thread is send a search query
397         // and the server answer with an +OK or -ERR message
398         curState = state.ESTABLISH;
399         try {
400         buf = in.readLine();
401         } catch (IOException e) {
402         synchronized (SmartTraceThreaded.terminal) {
403             SmartTraceThreaded.terminal += "Error Connection: "
404                 + e.getMessage();
405             // toast.show();
406         }
407         break;
408         }
409 
410         // -------------------------------SEARCH-STATE----------------------------------------
411         if (buf.startsWith("+OK")) {
412         if (buf.contains("Goodbye")) {
413             SmartTraceThreaded.terminal += "\n+OK Goodbye\n";
414             break;
415         }
416         try {
417             if (!SEARCH())
418             break;
419         } catch (IOException e) {
420             synchronized (SmartTraceThreaded.terminal) {
421             SmartTraceThreaded.terminal += "Error Connection: "
422                 + e.getMessage();
423             // toast.show();
424             }
425             break;
426         } // end try
427         try {
428             if (!RETRIEVE())
429             break;
430         } catch (IOException e) {
431             synchronized (SmartTraceThreaded.terminal) {
432             SmartTraceThreaded.terminal += "Error Connection: "
433                 + e.getMessage();
434             // toast.show();
435             }
436             break;
437         } // end try
438           // search Done;
```

```
439         continue;
440
441         } else if (buf.startsWith("-ERR")) {
442         Message msg1 = new Message();
443         msg1.arg1 = Messages.SERVER_ERR;
444         try {
445             msg1.arg2 = Integer.parseInt(buf.split(" ")[1]);
446         } catch (Exception e) {
447             // TODO: handle exception
448             msg1.arg2 = 0;
449         }
450         SmartTraceThreaded.terminal += "Server Busy\nPlease try again";
451         continue;
452         }
453         // -
454         // ------------------------------SEARCH-STATE-----------------------------------------
455         // ------------------------------CALCULATE-STATE--------------------------------------
456         try {
457         if (!CALCULATE())
458             break;
459         } catch (IOException e) {
460         synchronized (SmartTraceThreaded.terminal) {
461             SmartTraceThreaded.terminal += "Error Connection: "
462                 + e.getMessage();
463             // toast.show();
464         }
465         break;
466         } // end try
467           // ------------------------------CALCULATE-STATE--------------------------------------
468
469     }// end while
470
471     // ELSE CLOSE THE PROGRAM
472     // ending and closing the application
473     try {
474         connection.close();
475         synchronized (SmartTraceThreaded.terminal) {
476         SmartTraceThreaded.terminal += "Done.\nConnection closed.\n";
477         // toast.show();
478         }
479     } catch (IOException e) {
480         synchronized (SmartTraceThreaded.terminal) {
481         SmartTraceThreaded.terminal += "Error Connection: "
482             + e.getMessage();
483         // toast.show();
484         }
485         appClose();
486     //    SmartTraceThreaded.txtMsg.setText(SmartTraceThreaded.terminal);
487         return;
488     } // end try
489 //  SmartTraceThreaded.txtMsg.setText(SmartTraceThreaded.terminal);
490     return;
491
492     }
493
494     /**
495      *
496      */
497     void appClose() {
498     // ending and closing the application
499     try {
500         connection.close();
501     } catch (IOException e) {
502     }
503     }
504
505     void readFromFile() {
506     ClientTrajectory.clear();
507     ClientTRAJ = "";
508     // get the Latitude from file and store it to an array
509     try {
510         String line;
511         String[] temp;
```



```
512         myPoint temppoint;
513         inFile = new File(INFILE);
514         reader = new BufferedReader(new FileReader(inFile));
515         int l = 0;
516         while ((line = reader.readLine()) != null) {
517         temp = line.split("\t");
518         temppoint = new myPoint(Double.parseDouble(temp[0]),
519             Double.parseDouble(temp[1]));
520         ClientTrajectory.add(temppoint);
521         ClientTRAJ += temp[0] + " " + temp[1] + "\t";
522         }
523
524     } catch (Exception e) {
525         synchronized (SmartTraceThreaded.terminal) {
526         SmartTraceThreaded.terminal += "Exception while reading from file:\n"
527             + e.getMessage();
528         }
529     }
530
531     try {
532         reader.close();
533     } catch (IOException e) {
534         // TODO Auto-generated catch block
535         synchronized (SmartTraceThreaded.terminal) {
536         SmartTraceThreaded.terminal += "Exception while closing file:\n"
537             + e.getMessage();
538         }
539     }
540
541     }
542
543 }
544
```

```java
/*
 * (C) Copyright University of Cyprus. 2010-2011.
 *
 * SmartTrace Server API
 *
 * @version         : 1.0
 * @author : Costantinos Costa(costa.costantinos@gmail.com)
 * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
 * Computer Science Department , University of Cyprus
 *
 *
 */
package SmartTrace.apk;

import android.content.Context;
import android.content.SharedPreferences;
import android.content.res.TypedArray;
import android.graphics.Color;
import android.preference.Preference;
import android.text.Layout;
import android.util.AttributeSet;
import android.view.Gravity;
import android.view.View;
import android.view.ViewGroup;
import android.widget.LinearLayout;
import android.widget.SeekBar;
import android.widget.TextView;
import android.widget.SeekBar.OnSeekBarChangeListener;

public class SeekBarPreference extends Preference implements
        OnSeekBarChangeListener {

    public static int maximum = 6;
    public static int interval = 1;

    private float oldValue = 3;
    private TextView monitorBox;
    private TextView view;
    private SeekBar bar;
    private Layout layout;

    public SeekBarPreference(Context context) {
        super(context);
    }

    public SeekBarPreference(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    public SeekBarPreference(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
    }

    @Override
    protected View onCreateView(ViewGroup parent) {
        // TODO Auto-generated method stub
        LinearLayout layout = new LinearLayout(getContext());

        LinearLayout.LayoutParams params1 = new LinearLayout.LayoutParams(
                LinearLayout.LayoutParams.WRAP_CONTENT,
                LinearLayout.LayoutParams.WRAP_CONTENT);
        params1.gravity = Gravity.LEFT;
        params1.weight = 1.0f;
        LinearLayout.LayoutParams params2 = new LinearLayout.LayoutParams(160,
                LinearLayout.LayoutParams.WRAP_CONTENT);
        params2.gravity = Gravity.RIGHT;

        LinearLayout.LayoutParams params3 = new LinearLayout.LayoutParams(30,
                LinearLayout.LayoutParams.WRAP_CONTENT);
        params3.gravity = Gravity.CENTER;

        layout.setPadding(15, 5, 5, 5);
        layout.setOrientation(LinearLayout.HORIZONTAL);
```



```java
        view = new TextView(getContext());
        view.setText(getTitle());
        view.setTextSize(20);
        if (this.isEnabled())
            view.setTextColor(Color.WHITE);
        else
            view.setTextColor(Color.GRAY);
        view.setGravity(Gravity.LEFT);
        view.setLayoutParams(params1);
        bar = new SeekBar(getContext());
        bar.setMax(maximum);
        bar.setProgress((int) this.oldValue);
        bar.setLayoutParams(params2);
        bar.setOnSeekBarChangeListener(this);
        this.monitorBox = new TextView(getContext());
        this.monitorBox.setTextColor(Color.WHITE);
        this.monitorBox.setLayoutParams(params3);
        this.monitorBox.setPadding(2, 5, 5, 2);
        this.monitorBox.setText(bar.getProgress() + "");
        layout.addView(view);
        layout.addView(bar);
        layout.addView(this.monitorBox);
        layout.setId(android.R.id.widget_frame);
        return layout;
    }

    @Override
    public void onProgressChanged(SeekBar seekBar, int progress,
            boolean fromUser) {

        progress = Math.round(((float) progress) / interval) * interval;

        if (!callChangeListener(progress)) {
            seekBar.setProgress((int) this.oldValue);
            return;
        }
        seekBar.setProgress(progress);
        this.oldValue = progress;
        this.monitorBox.setText(progress + "");
        updatePreference(progress);

        notifyChanged();
    }

    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {
    }

    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {
    }

    @Override
    protected Object onGetDefaultValue(TypedArray ta, int index) {
        // TODO Auto-generated method stub

        int dValue = (int) ta.getInt(index, 50);

        return validateValue(dValue);
    }

    @Override
    protected void onSetInitialValue(boolean restoreValue, Object defaultValue) {

        int temp = restoreValue ? getPersistedInt(50) : (Integer) defaultValue;

        if (!restoreValue)
            persistInt(temp);

        this.oldValue = temp;
    }

    private int validateValue(int value) {
```

```
147
148         if (value > maximum)
149             value = maximum;
150         else if (value < 0)
151             value = 0;
152         else if (value % interval != 0)
153             value = Math.round(((float) value) / interval) * interval;
154
155         return value;
156     }
157
158     private void updatePreference(int newValue) {
159
160         SharedPreferences.Editor editor = getEditor();
161         editor.putInt(getKey(), newValue);
162         editor.commit();
163     }
164
165     @Override
166     public void onDependencyChanged(Preference dependency,
167             boolean disableDependent) {
168         if (view != null)
169             if (disableDependent)
170                 view.setTextColor(Color.GRAY);
171             else
172                 view.setTextColor(Color.WHITE);
173         // TODO Auto-generated method stub
174         super.onDependencyChanged(dependency, disableDependent);
175         if (this.layout != null) {
176             this.view.setEnabled(!disableDependent);
177             this.bar.setEnabled(!disableDependent);
178             this.monitorBox.setEnabled(!disableDependent);
179         }
180     }
181 }
182
```

```
/*
 * (C) Copyright University of Cyprus. 2010-2011.
 *
 * Centralize Server API
 *
 * @version         : 1.0
 * @author : Costantinos Costa(costa.costantinos@gmail.com)
 * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
 * Computer Science Department , University of Cyprus
 *
 *
 */
import java.awt.Color;
import java.awt.EventQueue;
import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.InetAddress;
import java.net.Socket; //import java.net.SocketException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Set;
import java.util.UUID;
import java.lang.Thread;

enum state {
    CLOSED, ESTABLISH, CALCULATE, SEARCH, RETRIEVE;
}

public class ServeClient extends Thread {
    // private static final int MINITES = 1200000;
    double UB = -1.0;
    InetAddress ip;
    boolean Oktraj = false;
    private Socket connection;
    String usr;
    String pass;
    String responseUsr;
    synServer syn;
    synServer synUB;
    synServer synSearch;
    // The server thread has a state in any time
    state curState = state.CLOSED;
    private OutputStream outStream;
    private DataOutputStream outDataStream;
    private InputStream inStream;
    private BufferedReader inDataStream;
    private String buf;
    private ColorPane logDisplay;
    private boolean quit = false;// VARIABLE THAT INDICATES IF THE STATES ARE
    private String[] traj;
    private double LCSS;

    // CORRECTLY

    private void updateGUI(final Color c, final String s) {
    if (!Server.graphics) {
        if (c.equals(Color.RED))
        System.err.print(s);
        else
        System.out.print(s);
        return;
    }
    EventQueue.invokeLater(new Runnable() {

        public void run() {

        logDisplay.append(c, s);

        }
```



```

    });
    yield();

    }

    private void init() {
    // INITIALIZED VARIABLES

    UB = -1.0;
    Oktraj = false;
    quit = false;// VARIABLE THAT INDICATES IF THE STATES ARE CORRECTLY
    curState = state.ESTABLISH;
    // In case the thread is the searcher
    if (Server.threadSearch == this) {
        Server.synSearch.unpause("Server.synSearch.unpause();");
    }
    }

    // FUNCTION FOR THE OK MESSAGES
    public boolean checkIfQuit(String msg) throws IOException {
    if (msg == null) {
        connection.close();
        quit = true;
        return true;
    }
    if (msg.equals("QUIT")) {
        Send(SendOK("Goodbye"), outDataStream);
        connection.close();
        quit = true;
        return true;
    }
    return false;
    }

    // FUNCTION FOR THE OK MESSAGES
    static public String SendOK(String msg) {
    return "+OK " + msg;
    }

    // FUNCTION FOR THE ERROR MESSAGES
    static public String SendERR(String msg) {
    return "-ERR " + msg;
    }

    // FUNCTION FOR THE OK k-Trajectories
    static public String SendTraj(String[] msg) {
    String conMsg = "";
    for (int i = 0; i < msg.length; i++) {
        conMsg += msg[i];
        if (i < msg.length - 1)
        conMsg += "&";
    }
    System.out.println("--------->Message length=" + msg.length);
    return "+OK " + conMsg;

    }

    // FUNCTION FOR THE MESSAGES
    static public void Send(String msg, OutputStream out) throws IOException {
    out.write((msg + Server.CRLF).getBytes());
    out.flush();
    }

    // FUNCTION FOR THE CREATION OF SESSIONID
    static public String CreateSessionId() {
    return UUID.randomUUID().toString();
    }

    // DEFAULT CONSTRUCTOR
    public ServeClient(Socket connection, ColorPane aModel) {
    this.connection = connection;
    // try {
```

```
147     // this.connection.setSoTimeout(MINITES);
148     // } catch (SocketException e) {
149     // // TODO Auto-generated catch block
150     // e.printStackTrace();
151     // }
152     this.logDisplay = aModel;
153     }
154
155     private String rmPREFIX(String msg) {
156     String[] sArr = msg.split(" ");
157     String temp = new String();
158     for (int i = 1; i < sArr.length; i++)
159         temp += sArr[i] + " ";
160     return temp;
161     }
162
163     // FUNCTION FOR THE ESTABLISH STATE
164     // ALL THE EXCEPTIONS IN THIS FUNCTION ARE HANDLE IN THE MAIN FUNCTION
165
166     private boolean ESTABLISH() throws IOException {
167     // SEND TO CLIENT +OK MESSAGE
168     Send(SendOK("STP READY"), outDataStream);
169     // CHECK IF THE PASSWORD AND USER-NAME IS CORRECT
170     buf = inDataStream.readLine();
171     if (Server.DEBUG)
172         System.out.println("ESTABLISH:Server Send:" + SendOK("READY"));
173     // CHECK THAT IS QUIT COMMAND
174     if (checkIfQuit(buf))
175         return false;
176     usr = rmPREFIX(buf);
177     if (Server.DEBUG)
178         System.out.println("(" + getId() + ")ESTABLISH:Client Response: "
179             + buf);
180     // CHECK USER NAME
181     if (checkPass(usr)) {
182
183         Server.SID = CreateSessionId();
184         Send(SendOK(Server.SID), outDataStream);
185         if (Server.DEBUG)
186         System.out.println("(" + getId() + ")ESTABLISH:Server Send: "
187             + SendOK(Server.SID));
188         // THE CLIENT IS ACCEPTABLE
189         synchronized (Server.ClientList) {
190         Server.ClientList.put(this, -1.0);
191         }
192         curState = state.ESTABLISH;
193         return true;
194     } else {
195         Send(SendERR("401"), outDataStream);
196
197         if (Server.DEBUG)
198         System.out.println("(" + getId() + ")ESTABLISH:Server Send: "
199             + SendERR("401"));
200         return false;
201     }
202
203     }
204
205     // FUNCTION FOR THE SEARCH STATE
206     private boolean SEARCH() throws IOException {
207     // CHECK THAT IS QUIT COMMAND
208     if (checkIfQuit(buf))
209         return false;
210
211     if (Server.DEBUG)
212         System.out.println("(" + getId() + ")SEARCH:Client Response: "
213             + buf);
214     if (!buf.startsWith("SEARCH")) {
215         return true;
216     }
217     curState = state.SEARCH;
218     // THE TRAJECTORY IS ALREADY SET
219     if (Server.Trajset) {
```



```
220         Send(SendERR("511"), outDataStream);
221         // STOP THE WHOLE CONNECTION
222         return true;
223     } else {
224         synchronized (Server.ClientList) {
225         if (Server.ClientList.size() < Server.clients) {
226             Send(SendERR("513"), outDataStream);
227             // System.out.println("SEARCH:Server Send: " + SendERR(""));
228             // STOP THE QUERY NOT ENOUGH CLIENTS
229             return true;
230         }
231         synchronized (Server.Trajset) {
232             Server.Trajset = true;
233         }
234
235         }
236         synchronized (Server.QID) {
237         Server.threadSearch = this;
238         Server.QID = CreateSessionId();
239         }
240         synchronized (Server.QueryTrajectory) {
241         try {
242             // remove the search keyword
243             buf = rmPREFIX(buf);
244             Server.K = Integer.parseInt(buf.split(" ")[0]);
245         } catch (Exception e) {
246             // TODO: handle exception
247             e.printStackTrace();
248             Server.K = 1;
249         }
250         Server.QueryTrajectory = rmPREFIX(buf);
251
252         }
253         Send(SendOK(Server.QID), outDataStream);
254
255         // ---------------------- BROADCAST TO ALL TO MOVE IN CALCULATE STATE
256         Set<ServeClient> set;
257         synchronized (Server.ClientList) {
258         set = Server.ClientList.keySet();
259         }
260         Iterator<ServeClient> iter = set.iterator();
261         buf = "CALCULATE " + Server.QID + " " + Server.QueryTrajectory;
262         // BROADCAST "+PROC" TO ALL CLIENT
263         while (iter.hasNext()) {
264
265         ServeClient temp;
266         synchronized (Server.ClientList) {
267             temp = iter.next();
268         }
269         // DON'T SEND TO THIS CONNECTION
270         // IT IS THE CURRENT ONE
271         if (temp.connection.equals(this.connection))
272             continue;
273         OutputStream toutStream;
274         DataOutputStream toutDataStream;
275         toutStream = temp.connection.getOutputStream();
276         toutDataStream = new DataOutputStream(toutStream);
277         try {
278             if (temp.curState == state.RETRIEVE
279                 || temp.curState == state.SEARCH) {
280             continue;
281             }
282             Send(buf, toutDataStream);
283             synchronized (Server.countUBCalls) {
284             Server.countUBCalls++;
285             }
286         } catch (IOException e) {
287
288             continue;
289         }
290         // Very Important Don't Close The ToutDataStream
291         // The handle is close the main object
292
```

```
293         if (Server.DEBUG)
294             System.out.println("(" + getId()
295                 + ")SEARCH:Server Send to " + temp.usr + " : "
296                 + buf);
297         }
298
299         updateGUI(Color.BLACK, "The trajectory of ");
300         updateGUI(Color.BLUE, usr);
301         updateGUI(Color.BLACK, " : " + Server.QueryTrajectory + "\n");
302
303         // LOCK AND UNLOCK OF SHARE VARIABLES
304
305     }
306     synSearch.pause("Client.synSearch.pause");
307     synchronized (Server.LCSSList) {
308         try {
309         if (Server.LCSSList.size() > 0) {
310             // Make response equals with the minimum of Clients number
311             // and Parameter K
312             Server.ResponseTrajectory = new String[Math.min(
313                 Server.LCSSList.size(), Server.K)];
314             // Fill the response K array
315             for (int i = 0; i < Server.LCSSList.size() && i < Server.K; i++) {
316             Server.ResponseTrajectory[i] = Server.LCSSList.get(i).thread.responseUsr;
317             }
318             Send(Server.QID + " " + Server.LCSSList.get(0).UBLCSS,
319                 outDataStream);
320             if (Server.DEBUG)
321             System.out.println("K=" + Server.K + "...(" + getId()
322                 + ")SEARCH:Server Send: " + Server.QID + " "
323                 + Server.LCSSList.get(0).UBLCSS);
324         } else {
325             Send(SendERR("512"), outDataStream);
326             if (Server.DEBUG)
327             System.out.println("(" + getId()
328                 + ")SEARCH:Server Send: " + SendERR(""));
329
330         }
331         } catch (IOException e) {
332         // TODO Auto-generated catch block
333         // return false;
334
335         }
336     }
337     Server.synSearch.unpause("Server.synSearch.unpause();");
338     return true;
339     }
340
341     // FUNCTION FOR THE CALCULATE STATE
342     private boolean CALCULATE() throws InterruptedException, IOException {
343         // CHECK THAT IS QUIT COMMAND
344         if (checkIfQuit(buf))
345             return false;
346         // IF MESSAGE IS NOT DEFERMENT WITH "+OK CALCULATE" THEN STOP
347         if (!buf.equals("+OK CALCULATE")) {
348
349             return true;
350         }
351         curState = state.CALCULATE;
352         // System.out.println(getId() + "Clients are " + Server.countClients);
353         while (!Server.Trajset) {
354             Thread.sleep(500);
355         }
356
357         // System.out.println("("+getId() +")CALCULATE:Server.trajectory" +
358         // Server.QueryTrajectory + "\n");
359         // WAIT FOR THE UPPER BOUND with traj
360         try {
361             buf = inDataStream.readLine();
362         } catch (IOException e) {
363             // We should decrease the number of UB calls to wait less answers
364             synchronized (Server.countUBCalls) {
365                 Server.countUBCalls--;
```



```
366             }
367
368             return false;
369         }
370         // CHECK THAT IS QUIT COMMAND
371         if (checkIfQuit(buf)) {
372             // We should decrease the number of UB calls to wait less answers
373             synchronized (Server.countUBCalls) {
374                 Server.countUBCalls--;
375                 if (Server.DEBUG)
376                     System.err.println("Server.countUBCalls(" + Server.countUBCalls + ")");
377             }
378             return false;
379         }
380
381 buf=rmPREFIX(buf);
382 buf=rmPREFIX(buf);
383 //split the points
384 traj = buf.split("\t");
385 if (traj == null) {
386     connection.close();
387     return true;
388 }
389
390 ArrayList<myPoint> ClientTrajectory = new ArrayList<myPoint>();
391 String[] tmppnt;
392 for (int j = 0; j < traj.length; j++) {
393     tmppnt = traj[j].split(" ");
394     // synchronized (SmartTraceThreaded.terminal) {
395     // SmartTraceThreaded.terminal += traj[j] + "\n";
396     // }
397     if (tmppnt != null)
398
399     if (tmppnt.length > 1)
400         try {
401         ClientTrajectory.add(new myPoint(Double
402             .parseDouble(tmppnt[0]), Double
403             .parseDouble(tmppnt[1])));
404         } catch (Exception e) {
405         // TODO: handle exception
406         continue;
407         }
408 }
409
410 //split the points
411 traj = Server.QueryTrajectory.split("\t");
412 if (traj == null) {
413     connection.close();
414     return true;
415 }
416
417 ArrayList<myPoint> ServerTrajectory = new ArrayList<myPoint>();
418 for (int j = 0; j < traj.length; j++) {
419     tmppnt = traj[j].split(" ");
420     // synchronized (SmartTraceThreaded.terminal) {
421     // SmartTraceThreaded.terminal += traj[j] + "\n";
422     // }
423     if (tmppnt != null)
424
425     if (tmppnt.length > 1)
426         try {
427         ServerTrajectory.add(new myPoint(Double
428             .parseDouble(tmppnt[0]), Double
429             .parseDouble(tmppnt[1])));
430         } catch (Exception e) {
431         // TODO: handle exception
432         continue;
433         }
434 }
435
436         // WAIT FOR THE COMPUTATION OF THE MAIN PROCESSES
437
438         // Calculate THE Lcss
```

```
439
440             LCSS=   mobileLCSS.LCSS(ServerTrajectory, ClientTrajectory, 0.001,1000);
441
442         if (Server.DEBUG)
443             System.out.println("(" + getId() + ")CALCULATE: Client Response: " + buf);
444
445         ip = this.connection.getInetAddress();
446         // System.out.println("CALCULATE:Server.countLCSSClients=" +
447         // Server.countLCSSClients);
448         synchronized (Server.LCSSList) {
449             Server.LCSSList.add(new ThreadUB(LCSS, this));
450         }
451         if (Server.DEBUG)
452             System.out.println("CALCULATE:[*]LCSS=" + LCSS);
453         // UPDATE GRAPHICS
454         updateGUI(Color.BLACK, "(");
455         updateGUI(Color.BLUE, usr);
456         updateGUI(Color.BLACK, ip + ") LCSS=");
457         updateGUI(Color.BLACK, LCSS + "\n");
458         //
459
460         return true;
461     }
462
463     // FUNCTION FOR THE RETRIEVE STATE
464     private boolean RETRIEVE() throws IOException, InterruptedException {
465     if (!buf.startsWith("RETRIEVE"))
466         return true;
467     curState = state.RETRIEVE;
468     long    time=0;
469     // SEND RETRIEVE MASSAGE TO THE CLIENT THAT HAVE THE BEST LCSS-SCORE
470     Send(SendTraj(Server.ResponseTrajectory), outDataStream);
471     time = System.currentTimeMillis();
472         System.out
473         .println(" Retrieval Time : " + time);
474     responseUsr = "";
475     if (Server.DEBUG)
476         System.out
477             .println("(" + getId() + ")RETRIEVE: Server Send: " + buf);
478     return true;
479
480     }
481
482     // function of the thread
483     public void run() {
484     try {
485         // INITIALIZE THE TRAJECTORIES;
486         syn = new synServer();
487         synUB = new synServer();
488         synSearch = new synServer();
489         try {
490         outStream = connection.getOutputStream();
491         inStream = connection.getInputStream();
492         } catch (IOException e) {
493         // TODO Auto-generated catch block
494         e.printStackTrace();
495         return;
496         }
497
498         inDataStream = new BufferedReader(new InputStreamReader(inStream));
499         outDataStream = new DataOutputStream(outStream);
500         // --------------------------------ESTABLISH-STATE---------------------------------------
501         try {
502         // If the ESTABLISH-STATE is wrong
503
504         if (!ESTABLISH())
505             return;
506         } catch (IOException e) {
507         synchronized (Server.ClientList) {
508             Server.ClientList.remove(this);
509         }
510         try {
511             connection.close();
```



```
512         } catch (IOException e1) {
513             // TODO Auto-generated catch block
514             e1.printStackTrace();
515         }
516         return;
517         }
518         init();
519         while (!quit) {
520         // --------------------------------ESTABLISH-STATE---------------------------------------
521         // The main read is determine the next state
522         // Read The Message To Continue
523         try {
524             buf = inDataStream.readLine();
525             if (Server.DEBUG)
526             System.out.println("CHOOSE TYPE OF CLIENT");
527         } catch (IOException e) {
528             if (Server.DEBUG)
529             System.out.println("----------------4---------------");
530             break;
531         }
532
533         // --------------------------------SEARCH-STATE-----------------------------------------
534         try {
535             if (!SEARCH()) {
536
537             break;
538
539             }
540         } catch (IOException e) {
541             if (Server.DEBUG)
542             System.out.println("----------------3---------------");
543             break;
544         }
545         // --------------------------------SEARCH-STATE-----------------------------------------
546         // --------------------------------CALCULATE-STATE--------------------------------------
547
548         try {
549             if (!CALCULATE())
550             break;
551         } catch (IOException e) {
552             if (Server.DEBUG)
553             System.out.println("----------------2---------------");
554             break;
555
556         }
557         // --------------------------------CALCULATE-STATE--------------------------------------
558         try {
559             // if (!search)
560             if (!RETRIEVE())
561             break;
562         } catch (IOException e) {
563             if (Server.DEBUG)
564             System.out.println("----------------1---------------");
565             break;
566         }
567         // if (!search)
568         // syn.pause("syn.pause();");
569         init();
570         }
571
572         // CLOSE THE CONNECTION AFTER THE QUIT COMMAND SEND OR THE
573         // CONNECTION WAS CLOSED
574         try {
575         connection.close();
576         } catch (IOException e) {
577         // TODO Auto-generated catch block
578         e.printStackTrace();
579         }
580         removeClient();
581     } catch (InterruptedException e) {
582         // TODO Auto-generated catch block
583         e.printStackTrace();
584     }
```

```
585     // finally {
586     // removeClient();
587     //
588     // }
589     }
590
591     private void removeClient() {
592     synchronized (Server.ClientList) {
593         Server.ClientList.remove(this);
594         if (Server.DEBUG)
595         System.err.println("Server.ClientList.size() = "
596             + Server.ClientList.size());
597
598     }
599
600     synchronized (Server.UBList) {
601         if (Server.UBList.remove(this))
602         // synchronized (Server.countUBCalls) {
603         // Server.countUBCalls--;
604         // System.err.println("Server.countUBCalls = " +
605         // Server.countUBCalls);
606         // }
607         if (Server.DEBUG)
608             System.err.println("Server.UBList.size() = "
609                 + Server.UBList.size());
610
611     }
612     if (Server.DEBUG)
613         System.err.println("Before:Server.LCSSList.size() = "
614             + Server.LCSSList.size());
615     synchronized (Server.LCSSList) {
616         if (Server.LCSSList.remove(this))
617         // synchronized (Server.countLCSSCalls) {
618         // Server.countLCSSCalls--;
619         // System.err.println("Server.countLCSSCalls = " +
620         // Server.countLCSSCalls);
621         // }
622         if (Server.DEBUG)
623             System.err.println("After:Server.LCSSList.size() = "
624                 + Server.LCSSList.size());
625     }
626
627     updateGUI(Color.LIGHT_GRAY, "User " + usr + " has left.\n");
628
629     // // IF THE CLIENT PASS THE LCSS CALCULATION
630     // synchronized (Server.LCSSthread) {
631     // Server.LCSSthread.remove(this);
632     // }
633     // CHECK IF THE THREADSEARCH ISN'T NULL AND CHECK IF IS THE CURRENT
634     if (Server.threadSearch != null)
635         if (Server.threadSearch.equals(this.connection)) {
636         synchronized (Server.QueryTrajectory) {
637             Server.Trajset = false;
638         }
639         }
640     if (Server.DEBUG)
641         System.err
642             .println("++++++++++++++++++++++++++++++++++++++++++++++++");
643     // THREAD RETURN
644     }
645
646     private boolean checkPass(String userpass) {
647     return true;
648     }
649 }
650
```

```
/*
 * (C) Copyright University of Cyprus. 2010-2011.
 *
 * Centralize Serverr API
 *
 * @version         : 1.0
 * @author : Costantinos Costa(costa.costantinos@gmail.com)
 * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
 * Computer Science Department , University of Cyprus
 *
 *
 */
import java.net.*;
import java.io.*;
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.JScrollPane;
import javax.swing.UIManager;
import javax.swing.UnsupportedLookAndFeelException;

public class Server {

    volatile static String SID = "";
    volatile static String QID = "";

    /**
     *
     */
    // static final Lock lock = new ReentrantLock();
    // static final Condition write = lock.newCondition();
    static final boolean DEBUG = true;

    private static final long serialVersionUID = 1L;
    private static  int port = 65000;
    // interval time for checking when the client are left
    private static  int interval = 2000;
    volatile static synServer synSearch;
    public static final String CRLF = "\r\n";

    // TODO: Reset the clients to 3
    volatile static int clients = 2;
    volatile static ServeClient threadSearch = null;
    // private synServer lock;
    volatile static Integer countUBCalls = 0;
    volatile static String QueryTrajectory = "";
    volatile static String[] ResponseTrajectory =null;
    volatile static Boolean Trajset = false;
    volatile static Hashtable<ServeClient, Double> ClientList;
    volatile ColorPane serverDisplay;
    volatile static JScrollPane scrollPane;
    volatile static ArrayList<ThreadUB> LCSSList;
    volatile static ArrayList<ThreadUB> UBList;
    ServerSocket listenSocket;
    Socket connection;
    OutputStream outStream;
    DataOutputStream outDataStream;
    TextField filetext;
    String message;
    volatile private synServer synClient;
    // boolean flag = true;
    volatile static Integer countLCSSCalls = 0;
    static boolean graphics;
    volatile static Integer K=1;

    public Server() {

        // initialize variables
        synSearch = new synServer();
        synClient = new synServer();
        LCSSList = new ArrayList<ThreadUB>();
        LCSSList.clear();
        UBList = new ArrayList<ThreadUB>();
```



```
        UBList.clear();
        ClientList = new Hashtable<ServeClient, Double>();
        ClientList.clear();

        //initialize all the parameters
        initPrams();

        // choose if you want to continue with grahics

        if (!graphics)
            return;

        class graphicDisplay extends Frame implements WindowListener, KeyListener {

            /**
             *
             */
            private static final long serialVersionUID = 1L;

            public graphicDisplay() {
                // TODO Auto-generated constructor stub
                super("Server");

                try {
                    UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
                } catch (ClassNotFoundException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (InstantiationException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (IllegalAccessException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (UnsupportedLookAndFeelException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
                serverDisplay = new ColorPane();
                scrollPane = new JScrollPane();

                serverDisplay.setBorder(new javax.swing.border.TitledBorder(null,
                        "Server is running", javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
                        javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font(
                                "Tahoma", 1, 16), new java.awt.Color(0, 102, 102)));
                // scrollPane.add(logDisplay);
                add(BorderLayout.CENTER, scrollPane);
                scrollPane.getViewport().setBackground(Color.white);
                scrollPane.getViewport().add(serverDisplay);
                filetext = new TextField();
                filetext.addKeyListener(this);
                add(BorderLayout.PAGE_END, filetext);
                addWindowListener(this);
                setSize(650, 400);
                setVisible(true);

            }

            public void windowActivated(WindowEvent e) {
            }

            public void windowDeactivated(WindowEvent e) {
            }

            public void windowOpened(WindowEvent e) {
            }

            public void windowClosed(WindowEvent e) {
            }

            public void windowClosing(WindowEvent e) {
                setVisible(false);
                this.dispose();
```

```
147                 System.exit(0);
148             }
149
150             public void windowIconified(WindowEvent e) {
151             }
152
153             public void windowDeiconified(WindowEvent e) {
154             }
155
156             /** Handle the key typed event from the text field. */
157             public void keyTyped(KeyEvent e) {
158                 if (e.getKeyChar() == '\n' && filetext.isEditable()) {
159
160                     if (!isIntNumber(filetext.getText())) {
161                         updateGUI(Color.RED,
162                                 "You gave wrong clients's number.\nThe default number (3) will be used\n");
163                     } else {
164                         clients = Integer.parseInt(filetext.getText());
165                         if (clients < 3) {
166                             updateGUI(Color.RED,
167                                     "You gave invalid clients's number.\nThe default number (3) will be used\n");
168                             clients = 3;
169                         } else
170
171                             updateGUI(Color.BLACK, "You gave " + clients + " clients\n");
172                     }
173                     filetext.setText("");
174                     filetext.setText("The server has started!");
175                     filetext.setEditable(false);
176                     synClient.unpause("synClient.unpause();");
177
178                 }
179             }
180
181             @Override
182             public void keyPressed(KeyEvent arg0) {
183                 // TODO Auto-generated method stub
184
185             }
186
187             @Override
188             public void keyReleased(KeyEvent arg0) {
189                 // TODO Auto-generated method stub
190             }
191
192             private boolean isIntNumber(String num) {
193                 try {
194                     Integer.parseInt(num);
195                 } catch (NumberFormatException nfe) {
196                     return false;
197                 }
198                 return true;
199             }
200         }
201
202         graphicDisplay serverDisplay = new graphicDisplay();
203     } // end Server_Basic constructor
204
205     private void updateGUI(final Color c, final String s) {
206         if (!Server.graphics) {
207             if (c.equals(Color.RED))
208                 System.err.print(s);
209             else
210                 System.out.print(s);
211             return;
212         }
213         EventQueue.invokeLater(new Runnable() {
214
215             public void run() {
216
217                 serverDisplay.append(c, s);
```



```
218
219             }
220
221         });
222
223     }
224
225     public void runServer() {
226
227         try {
228
229             updateGUI(Color.BLACK, "Server started ");
230             try {
231                 InetAddress here = InetAddress.getLocalHost();
232                 String host = here.getHostName();
233                 // get Server's ip address
234                 IPaddress ip = new IPaddress();
235                 updateGUI(Color.BLACK, "on ");
236                 updateGUI(Color.BLUE, host);
237                 updateGUI(Color.BLACK, " with ip:" + ip.getInterfaces() + "\nport :" + port + "\n");
238             } catch (UnknownHostException e) {
239                 updateGUI(Color.RED, "Problem with local host\n");
240             }
241             //TODO SKIP THE AKSING PART
242             /*
243             updateGUI(Color.BLACK,
244                     "Please give the minimum require number of clients at bottom of the window\n");
245             if (!graphics) {
246                 BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
247                 try {
248                     String sclients = in.readLine();
249                     clients = Integer.parseInt(sclients);
250                 } catch (Exception e) {
251                     // TODO: handle exception
252                     updateGUI(Color.RED,
253                             "You gave wrong clients's number.\nThe default number (3) will be used\n");
254
255                 }
256             } else {
257                 // Synchronized the server with the key event below
258                 synClient.pause("synClient.pause();");
259             }
260             */
261             // Create the TCP Server socket
262             listenSocket = new ServerSocket(port);
263
264             // new runnable to RESPONSIBLE FOR accept OR NOT ACCEPT CLIENTS
265             Runnable r = new Runnable() {
266                 public void run() {
267                     try {
268                         while (true) {
269                             try {
270
271                                 connection = listenSocket.accept();
272                                 //
273                             } catch (Exception e) {
274                                 // TODO: handle exception
275                                 break;
276                             }
277                             updateGUI(new java.awt.Color(0, 202, 202),
278                                     "Connection request received\n");
279                             ServeClient thread = new ServeClient(connection, serverDisplay);
280
281                             thread.start();
282                         }
283                     } catch (Exception x) {
284                         // in case ANY exception slips through
285                         x.printStackTrace();
286                     }
287                 }
288             };
289             // new thread to RESPONSIBLE FOR accept OR NOT ACCEPT CLIENTS
290             Thread internalThread = new Thread(r);
```

```
291             internalThread.start();
292             updateGUI(Color.BLACK,
293                     "\n--------------------------------------------------------------------------------\n");
294             Set<ServeClient> set;
295             // Barrier for the server.The clients should be more than the
296             // minimum clients
297             long    time=0;
298             while (true) {
299
300                 while (ClientList.size() < clients && !Server.Trajset) {
301                     try {
302                         if (DEBUG)
303                             System.out.println("(countClients < clients)Barrier:"
304                                     + ClientList.size() + "<" + clients);
305                         Thread.sleep(interval);
306
307                     } catch (InterruptedException e) {
308                         // TODO Auto-generated catch block
309                         e.printStackTrace();
310                     }
311                 }
312                 time = System.currentTimeMillis();
313                 synchronized (ClientList) {
314                     set = ClientList.keySet();
315                 }
316
317                 // wait until all threads are started
318                 // System.out.println("countUBClients in server=" +
319                 // countUBClients);
320                 // System.out.println("Is trajectory set = " + Trajset);
321                 // System.out.println("UBtable.size() = " + UBList.size());
322                 // This is a barrier if client searcher is the first one
323
324
325                 // the first and the second of elements is the element with the
326                 // biggest UB
327                 // for (ThreadUB tub : UBList) {
328                 // System.out.println(tub.thread.getId() + "." + tub.thread.usr
329                 // + ".UB = "
330                 // + tub.UBLCSS);
331                 // }
332
333                 // wait for all threads
334                 while (LCSSList.size() < clients - 1 && ClientList.size() > clients - 1) {
335                     try {
336                         if (DEBUG)
337                             System.out
338                                     .println("(LCSSList.size() < 2 && ClientList.size() > 2)Barrier:"
339                                             + LCSSList.size()
340                                             + "<2 && "
341                                             + ClientList.size()
342                                             + "> 2");
343
344                         Thread.sleep(interval);
345                     } catch (InterruptedException e) {
346                         // TODO Auto-generated catch block
347                         e.printStackTrace();
348                     }
349                 }
350                 // the second element
351                 synchronized (LCSSList) {
352                     Collections.sort(LCSSList, new myComparator());
353                 }
354
355
356                 // System.out.println("LCSSthread.size(" + LCSSthread.size() +
357                 // ") ");
358                 // wait until all the LCSS calls are answered
359                 while (LCSSList.size() < ClientList.size()-1) {
360                     try {
361                         if (DEBUG)
362                             System.out.println("(LCSSthread.size() < countLCSSCalls) Barrier:"
363                                     + LCSSList.size() + "<" + countLCSSCalls);
```



```
364                         Thread.sleep(interval);
365                     } catch (InterruptedException e) {
366                         // TODO Auto-generated catch block
367                         e.printStackTrace();
368                     }
369                 }
370
371                 synchronized (LCSSList) {
372
373                     Collections.sort(LCSSList, new myComparator());
374                 }
375                 // send the trajectory of the most common
376                 // by calling the responsible thread
377                 if (threadSearch != null) {
378                     synchronized (threadSearch) {
379                         threadSearch.synSearch.unpause("threadSearch.synSearch.unpause();");
380
381                     }
382                 }
383                 // Print the winner
384                 if (LCSSList.size() > 0) {
385                     updateGUI(Color.BLUE, LCSSList.get(0).thread.usr);
386                     updateGUI(Color.BLACK, " has the most similar trajectory with you with ip "
387                             + LCSSList.get(0).thread.ip + "\n");
388                      time = System.currentTimeMillis() - time;
389                         System.out.println(" The test took " + time + " milliseconds");
390                     // wait until it's job is finished
391                     synSearch.pause("synSearch.pause();");
392                 } else
393                     updateGUI(Color.RED, "The minimum client left!!!");
394                 updateGUI(Color.BLACK,
395                         "\n--------------------------------------------------------------------------------\n");
396
397                 // Initialize all the variables
398                 // END
399                 if (threadSearch != null)
400                     synchronized (threadSearch) {
401                         threadSearch = null;
402                     }
403                 Server.Trajset = false;
404
405                 synchronized (QueryTrajectory) {
406                     QueryTrajectory = "";
407                 }
408                 synchronized (countUBCalls) {
409                     countUBCalls = 0;
410                 }
411                 synchronized (countLCSSCalls) {
412                     countLCSSCalls = 0;
413                 }
414                 synchronized (ClientList) {
415                     for (ServeClient thread : set) {
416                         thread.syn.unpause("thread.syn.unpause();");
417                     }
418                 }
419                 // initialize the counter for the clients and the tables
420                 synchronized (LCSSList) {
421                     LCSSList.clear();
422                 }
423                 synchronized (UBList) {
424                     UBList.clear();
425                 }
426
427
428             }// end of while(true)
429         } // end try
430         catch (IOException ex) {
431             updateGUI(Color.RED, "\n"+ex.getMessage()+"\n");
432             try {
433                 Thread.sleep(1000);
434             } catch (InterruptedException e) {
435                 // TODO Auto-generated catch block
436                 e.printStackTrace();
```

```
1  package cs.ucy.ac.cy;
2
3  import java.util.Random;
4
5  import android.app.Activity;
6  import android.os.Bundle;
7  import android.view.View;
8  import android.widget.ArrayAdapter;
9  import android.widget.Button;
10 import android.widget.EditText;
11 import android.widget.Spinner;
12 import android.widget.TextView;
13 import android.widget.Toast;
14
15 public class SmartTraceThreaded extends Activity {
16     static int parK;
17     EditText edtLogFile;
18     EditText edtInFile;
19     EditText edtThreads;
20     Button btnStart;
21     Button btnSearch;
22     Spinner mode;
23     EditText edtparK;
24     EditText edtserverAdd;
25     EditText edtserverPort;
26    // public static TextView txtMsg;
27     public static String terminal = new String();
28     private Random r = new Random();
29     private EditText edtSearch;
30     public static int delta = 100;
31     static String arrName[] = { "Michael", "Christopher", "Matthew", "Joshua",
32         "Andrew", "David", "Justin", "Daniel", "James", "Robert", "John",
33         "Joseph", "Ryan", "Nicholas", "Jonathan", "William", "Brandon",
34         "Anthony", "Kevin", "Eric", "Jessica", "Ashley", "Amanda", "Sarah",
35         "Jennifer", "Brittany", "Stephanie", "Samantha", "Nicole",
36         "Elizabeth", "Lauren", "Megan", "Tiffany", "Heather", "Amber",
37         "Melissa", "Danielle", "Emily", "Rachel", "Kayla" };
38
39     /** Called when the activity is first created. */
40     @Override
41     public void onCreate(Bundle savedInstanceState) {
42     super.onCreate(savedInstanceState);
43     setContentView(R.layout.main);
44     init();
45     }
46
47     void init() {
48     edtLogFile = (EditText) findViewById(R.id.LogFile);
49     edtInFile = (EditText) findViewById(R.id.InputFile);
50     edtSearch = (EditText) findViewById(R.id.SearchFile);
51     edtThreads = (EditText) findViewById(R.id.Threads);
52     btnStart = (Button) findViewById(R.id.btnStart);
53     btnSearch = (Button) findViewById(R.id.btnSearch);
54     mode = (Spinner) findViewById(R.id.mode);
55
56        ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(
57                this, R.array.algo_array, android.R.layout.simple_spinner_item);
58        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
59        mode.setAdapter(adapter);
60
61     edtparK = (EditText) findViewById(R.id.parK);
62     edtserverAdd = (EditText) findViewById(R.id.serverAdd);
63     edtserverPort = (EditText) findViewById(R.id.serverPort);
64 //  txtMsg = (TextView) findViewById(R.id.msg);
65     btnStart.setOnClickListener(new View.OnClickListener() {
66
67         @Override
68         public void onClick(View v) {
69         // TODO Auto-generated method stub
70         //txtMsg.setText("Starting\n");
71         int threads = 1;
72         try {
73             threads = Integer.parseInt(edtThreads.getText().toString());
```



```
74         } catch (Exception e) {
75             // TODO: handle exception
76             Toast.makeText(getBaseContext(),
77                 "Wrong fomat of threads' field", Toast.LENGTH_SHORT)
78                 .show();
79             threads = 1;
80         }
81
82         String mm="";
83         switch (mode.getSelectedItemPosition()) {
84         case 0:
85             mm="Centralize";
86             CentralizeConnect[] cc = new CentralizeConnect[threads];
87             for (int i = 0; i < cc.length; i++) {
88
89                   cc[i] = new CentralizeConnect(getApplicationContext(), false,
90                    arrName[r.nextInt(arrName.length)], edtserverPort
91                        .getText().toString(), edtserverAdd
92                        .getText().toString(), edtLogFile.getText()
93                        .toString(),
94                    edtInFile.getText().toString(), "0.001");
95                 cc[i].start();
96             }
97
98             break;
99         case 1:
100            mm="Decentralize";
101            DecentralizeConnect[] dc = new DecentralizeConnect[threads];
102            for (int i = 0; i < dc.length; i++) {
103
104                dc[i] = new DecentralizeConnect(getApplicationContext(), false,
105                    arrName[r.nextInt(arrName.length)], edtserverPort
106                        .getText().toString(), edtserverAdd
107                        .getText().toString(), edtLogFile.getText()
108                        .toString(),
109                    edtInFile.getText().toString(), "0.001");
110                dc[i].start();
111            }
112
113
114            break;
115        case 2:
116            mm="SmartTrace";
117            Connect[] c = new Connect[threads];
118            Toast.makeText(getBaseContext(), "SmartTrce threads are  "+threads,Toast.LENGTH_SHORT).show()↙
    ;
119            for (int i = 0; i < c.length; i++) {
120                c[i] = new Connect(getApplicationContext(), false,
121                    arrName[r.nextInt(arrName.length)], edtserverPort
122                        .getText().toString(), edtserverAdd
123                        .getText().toString(), edtLogFile.getText()
124                        .toString(),
125                    edtInFile.getText().toString(), "0.001");
126                c[i].start();
127            }
128            break;
129        default:
130            break;
131        }
132
133        Toast.makeText(getBaseContext(), "You choose "+mm
134        ,Toast.LENGTH_SHORT).show();
135
136
137
138
139        // Disable the button
140 //     btnStart.setEnabled(false);
141        }
142 });
143
144 btnSearch.setOnClickListener(new View.OnClickListener() {
145
```

```
146         @Override
147         public void onClick(View v) {
148 
149         // TODO Auto-generated method stub
150         Search s = new Search(getApplicationContext(), false,edtparK.getText().toString(), arrName[r
151             .nextInt(arrName.length)], edtserverPort.getText()
152             .toString(), edtserverAdd.getText().toString(),
153             edtLogFile.getText().toString(), edtSearch.getText()
154                 .toString(), "0.001");
155 
156         s.start();
157 
158         }
159     });
160 
161     }
162 
163 }
```

```java
/*
 * (C) Copyright University of Cyprus. 2010-2011.
 *
 * SmartTrace Server API
 *
 * @version         : 1.0
 * @author : Costantinos Costa(costa.costantinos@gmail.com)
 * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
 * Computer Science Department , University of Cyprus
 *
 *
 */
package SmartTrace.apk;

import SmartTrace.apk.R;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.view.View;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.Toast;

public class Splash extends Activity {

    private static final int STOPSPLASH = 0;
    // time in milliseconds
    private static final long SPLASHTIME = 3000;

    // handler for splash screen
    private Handler splashHandler = new Handler() {
        /*
         * (non-Javadoc)
         *
         * @see android.os.Handler#handleMessage(android.os.Message)
         */
        @Override
        public void handleMessage(Message msg) {
            switch (msg.what) {
            case STOPSPLASH:
                // remove SplashScreen from view
//              splash.setVisibility(View.GONE);
                startActivity(i);
                info.setText("Welcome...");
                info.show();
                break;
            }
            super.handleMessage(msg);
            finish();
            System.exit(0);
        }
    };
    private Toast info;
    private Intent i;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.splash);
        info = Toast.makeText(getApplicationContext(), "", Toast.LENGTH_LONG);
        View textView = info.getView();
        LinearLayout lay = new LinearLayout(getApplicationContext());
        lay.setOrientation(LinearLayout.HORIZONTAL);
        ImageView view = new ImageView(getApplicationContext());
        view.setImageResource(R.drawable.info);
        lay.addView(view);
        lay.addView(textView);
        info.setView(lay);

        i = new Intent(Splash.this, MainActivity.class);
```



```java

        Message msg = new Message();
        msg.what = STOPSPLASH;
        splashHandler.sendMessageDelayed(msg, SPLASHTIME);
    }
}
```

```
1  /*
2   * (C) Copyright University of Cyprus. 2010-2011.
3   *
4   * Centralize Server API
5   *
6   * @version         : 1.0
7   * @author : Costantinos Costa(costa.costantinos@gmail.com)
8   * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
9   * Computer Science Department , University of Cyprus
10  *
11  *
12  */
13 public class synServer {
14     private volatile boolean threadSuspended;
15     private volatile boolean threadAlready;
16     public synchronized void pause(String string) {
17         if(!threadAlready)
18         threadSuspended = false;
19         if(Server.DEBUG)
20         System.err.println("PAUSE:"+string);
21         while (!threadSuspended)
22             try {
23                 wait();
24             } catch (InterruptedException e) {
25                 System.out.println("InterruptedException caught");
26             }
27             threadAlready=false;
28     }
29
30     public synchronized void unpause(String string) {
31         if(Server.DEBUG)
32         System.err.println("UNPAUSE:"+string);
33         threadSuspended = true;
34         notify();
35         threadAlready=true;
36
37     }
38 }
```

```java
/*
 * (C) Copyright University of Cyprus. 2010-2011.
 *
 * Centralize Server API
 *
 * @version         : 1.0
 * @author : Costantinos Costa(costa.costantinos@gmail.com)
 * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
 * Computer Science Department , University of Cyprus
 *
 *
 */


import java.util.ArrayList;

public class UB {
    static double upperBound(ArrayList<myPoint> serverTraj,
            ArrayList<myPoint> clientTraj, double eUb) {
        //double the size of the envelop that is used for lcss
        double e=eUb*2;
        double UB = 0.0;

        if(serverTraj==null || clientTraj==null)
        return 0;
        int size = Math.min(serverTraj.size(), clientTraj.size());
        for (int i = 0; i < size; i++) {
            // create envelop e
            if (serverTraj.get(i).getX() < (clientTraj.get(i).getX() + e)
                    && serverTraj.get(i).getX() > (clientTraj.get(i).getX() - e))
                if (serverTraj.get(i).getY() < (clientTraj.get(i).getY() + e)
                        && serverTraj.get(i).getY() > (clientTraj.get(i).getY() - e))
                    UB++;
        }
        if (size == 0)
            return 0;
        else
            return (UB / size )* 100;
    }
}
```

```java
/*
 * (C) Copyright University of Cyprus. 2010-2011.
 *
 * SmartTrace Server API
 *
 * @version         : 1.0
 * @author : Costantinos Costa(costa.costantinos@gmail.com)
 * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
 * Computer Science Department , University of Cyprus
 *
 *
 */
package SmartTrace.apk;

import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.Socket;
import java.util.HashMap;
import android.content.Context;
import android.os.Handler;
import android.os.Message;

public class WConnect extends Thread {

    enum state {
        CLOSED, ESTABLISH, CALCULATE, SEARCH, RETRIEVE;
    }

    private DataOutputStream out;
    private BufferedReader in;
    private String serverAddress = new String();
    private String epsilon = new String();
    private String usr = new String();
    private Socket connection = null;
    static String ClientTRAJ;
    private double lcss;
    private String SSID;
    private String QUID;
    String host = new String();
    String buf = new String();
    String[] traj = null;
    InputStream inStream = null;
    OutputStream outStream = null;
    private String psw = "";
    private String LCSS;
    private int port = 443;
    private WifiGraph parent;
    private boolean flag=false;
    private String wserverPort="443";
    static state curState = state.CLOSED;
    static Handler messageHandler;

    private String rmPREFIX(String msg) {
        String[] sArr = msg.split(" ");
        String temp = new String();
        for (int i = 1; i < sArr.length; i++)
            temp += sArr[i] + " ";
        return temp;
    }

    // FUNCTION FOR THE MESSAGES
    static public void Send(String msg, OutputStream out) throws IOException {
        out.write((msg + Messages.CRLF).getBytes());
        out.flush();
    }

    WConnect(Context c, boolean flag, String username, WifiGraph parent) {

        // set username
```



```java
        this.usr = username;
        this.flag = flag;
        this.parent = parent;
        // initialized the message handler for the messages from the thread
        messageHandler = new Handler() {

            public void handleMessage(Message msg) {
                super.handleMessage(msg);
                Message msg1 = new Message();
                switch (msg.arg1) {
                // These means that the client want to use smartTrace search
                // We create new thread so the GUI doesn't block
                case Messages.SEARCH:

                    // // TODO Auto-generated method stub
                    if (!curState.equals(state.ESTABLISH)) {
                        switch (curState) {
                        case CALCULATE:
                            msg1.arg1 = Messages.STATECAL;
                            break;
                        case SEARCH:
                            msg1.arg1 = Messages.STATESER;
                            break;
                        case RETRIEVE:
                            msg1.arg1 = Messages.STATERET;
                            break;

                        default:
                            msg1.arg1 = Messages.SERVER_ERR;
                            break;
                        }

                        WifiGraph.messageHandler.sendMessage(msg1);
                        return;
                    }
                    String temp = WifiGraph.preferences.getString("wparK", "1");
                    try{
                        WifiGraph.K = Integer.parseInt(temp);
                    }catch (Exception e) {
                        // TODO: handle exception
                        MainActivity.K=1;
                    }

                    try {
                        Send(("SEARCH "+WifiGraph.K+" " + SSID + " " + ClientTRAJ), out);
                    } catch (IOException e) {
                        // send a massage to main thread
                        msg1.arg1 = Messages.OTHER;
                        WifiGraph.messageHandler.sendMessage(msg1);
                    }
                    // write everything to the transaction string
                    synchronized (wMore.terminal) {
                        wMore.terminal += "Client send : SEARCH " + SSID + " Client TRAJECTORY \n";

                    }
                    msg1.arg1 = Messages.SEARCH;
                    WifiGraph.messageHandler.sendMessage(msg1);
                    // }
                    // }).start();
                    break;
                case Messages.QUIT:

                    if (connection == null) {
                        // appClose();
                        return;
                    }
                    try {
                        Send("QUIT", out);
                        connection.close();
                    } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }
```

```
147                     synchronized (wMore.terminal) {
148                         wMore.terminal += "Client send : QUIT \n";
149
150                     }
151                     Message msg11 = new Message();
152                     msg11.arg1 = Messages.QUIT;
153                     WifiGraph.messageHandler.sendMessage(msg11);
154                     appClose();
155                     return;
156
157                 default:
158
159                     break;
160                 }
161
162             }
163         };
164         this.setUncaughtExceptionHandler(new UncaughtExceptionHandler() {
165
166             @Override
167             public void uncaughtException(Thread thread, Throwable ex) {
168                 // TODO Auto-generated method stub
169                 Message msg = new Message();
170                 msg.arg1 = Messages.OTHER;
171                 WifiGraph.messageHandler.sendMessage(msg);
172                 wMore.terminal += "Client(" + thread.getId() + ").Connection ERROR :\n"
173                         + ex.getLocalizedMessage();
174                 // toast.show();
175             }
176         });
177     }
178
179     public void run() {
180         // Check for any change in preferences
181         // Check if the user-name is set
182         if (!WifiGraph.isUsernameSet) {
183             Message msg = new Message();
184             msg.arg1 = Messages.SERVER_ERR;
185             msg.arg2 = 0;
186             WifiGraph.messageHandler.sendMessage(msg);
187             return;
188         }
189         serverAddress = WifiGraph.preferences.getString("wserverAdd", "n/a");
190         if (serverAddress.equals("n/a") || serverAddress.equals("")) {
191             Message msg = new Message();
192             msg.arg1 = Messages.INVALID_ADDRESS;
193             WifiGraph.messageHandler.sendMessage(msg);
194             return;
195         }
196
197         //serverPort
198         wserverPort = WifiGraph.preferences.getString("wserverPort", "65001");
199         if (wserverPort.equals("n/a") || wserverPort.equals("")) {
200             Message msg = new Message();
201             msg.arg1 = Messages.INVALID_ADDRESS;
202             MainActivity.messageHandler.sendMessage(msg);
203             return;
204         }
205
206
207         if (flag) {
208             parent.checkTrace();
209         }
210         serverConnect();
211     }
212
213     // function for the ESTABLISH STATE
214     private boolean ESTABLISH() throws IOException {
215         curState = state.ESTABLISH;
216         // GET THE +OK READY FROM SERVER
217         buf = in.readLine();
218         wMore.terminal += "Server Response: " + buf + "\n";
219         // Wait For Proceeding
```



```
220         // Get The Session Id
221         buf = "USER " + usr + " " + psw;
222         // SEND A +OK REPLY and user-name TO SERVER
223         Send(buf, out);
224         wMore.terminal += "Client Send: " + buf + "\n";
225         // GET THE +OK or -ERR FROM SERVER
226         buf = in.readLine();
227         wMore.terminal += "Server Response: " + buf + "\n";
228         if (buf.startsWith("+OK")) {
229             // Set SessionID
230             SSID = rmPREFIX(buf);
231             return true;
232         }
233         return false;
234
235     }
236
237     // Function For the SEARCH STATE
238     private boolean SEARCH() throws IOException {
239         curState = state.SEARCH;
240         // Read The Message To Continue
241         wMore.terminal += "Server Response: " + buf + "\n";
242         // Set SessionID
243         QUID = rmPREFIX(buf);
244         // Read The Message To Continue
245         buf = in.readLine();
246         synchronized (wMore.terminal) {
247             wMore.terminal += "Server Response: " + buf + "\n";
248         }
249         if (buf.contains("Goodbye") || buf.startsWith("-ERR"))
250             return false;
251         LCSS = rmPREFIX(buf);
252         return true;
253     }
254
255     // Function For the CALCULATE STATE
256     private boolean CALCULATE() throws IOException {
257         if (!buf.startsWith("CALCULATE"))
258             return true;
259         //set the current state
260         curState = state.CALCULATE;
261         synchronized (wMore.terminal) {
262             wMore.terminal += "Server Response: " + "CALCULATE queryTrajectory" + "\n";
263         }// toast.show();
264         //Remove the CALCULATE
265         buf=rmPREFIX(buf);
266
267         try {
268             QUID = buf.split(" ")[0];
269         } catch (Exception e) {
270             // TODO: handle exception
271             return false;
272         }
273         String query = rmPREFIX(buf);
274         buf = "+OK CALCULATE";
275         Send(buf, out);
276         synchronized (wMore.terminal) {
277             wMore.terminal += "Client send :" + buf + "\n";
278         }
279         // split the points
280         traj = query.split("#");
281         if (traj == null) {
282             connection.close();
283             synchronized (wMore.terminal) {
284                 wMore.terminal += "ERROR :The query is empty\n";
285             }
286             return false;
287         }
288
289         WifiGraph.ServerTrajectory.clear();
290         String[] tmppnt;
291         int value = 0;
292         String key;
```

```
293         HashMap<String, Integer> tempMap;
294 //      File outFile = new File("/sdcard/debug.txt");//
295 //      BufferedWriter writer = new BufferedWriter(new FileWriter(outFile));//
296         for (int i = 0; i < traj.length; i++) {
297             //Remove any Blank spaces
298             tmppnt = traj[i].replaceAll("\\s+", "").split("&");
299             tempMap = new HashMap<String, Integer>();
300             for (int j = 0; j < tmppnt.length; j++) {
301                 if (tmppnt != null )
302                 if(tmppnt[j].length()>1)
303                 {
304                 try {
305 //                  writer.write(tmppnt[j]);
306                     key = tmppnt[j].split(",")[0];
307                     value = Integer.parseInt(tmppnt[j].split(",")[1]);
308
309                 } catch (Exception e) {
310                     // TODO: handle exception
311
312                     break;
313                 }
314
315                 tempMap.put(key, value);
316                 }
317             }
318             WifiGraph.ServerTrajectory.add(tempMap);
319         }
320 //writer.close();
321         // CALCULATE THE UB
322         Message msg = new Message();
323         msg.arg1 = Messages.UPBOUND;
324         WifiGraph.messageHandler.sendMessage(msg);
325         // SEND THE UB TO SERVER
326         double eps = 0.1;
327         try {
328             eps = Double.parseDouble(epsilon);
329         } catch (Exception e) {
330             // TODO: handle exception
331         }
332         buf = "+OK " + QUID + " "
333                 + WifiUB.upperBound(WifiGraph.ServerTrajectory, WifiGraph.ClientTrajectory, eps);
334         Send(buf, out);
335         synchronized (wMore.terminal) {
336             wMore.terminal += "Client send :" + buf + "\n";
337         }
338         // Wait for the Server to send "LCSS" OR "CLOSE"
339         buf = in.readLine();
340         synchronized (wMore.terminal) {
341             wMore.terminal += "Server Response: " + buf + ".\n";
342         }
343         // IF LCSS SEND THE Coordinates[] ARRAY
344         if (buf.startsWith("LCSS")) {
345             synchronized (wMore.terminal) {
346                 wMore.terminal += "Calculate LCCS and send it to server...\n";
347             }
348             lcss = WifiMobileLCSS.LCSS(WifiGraph.ClientTrajectory,WifiGraph.ServerTrajectory,  eps);
349             buf = "+OK " + QUID + " " + lcss + " " + usr;
350             Send(buf, out);
351             synchronized (wMore.terminal) {
352                 wMore.terminal += "Client send :" + buf + "\n";
353             }
354         }
355         return true;
356
357     }
358
359     // Function For the CALCULATE STATE
360     private boolean RETRIEVE() throws IOException {
361         curState = state.RETRIEVE;
362         // send RETRIEVE message to the server
363         Send("RETRIEVE " + QUID, out);
364         wMore.terminal += "Client send :" + "RETRIEVE" + "\n";
365         buf = in.readLine();
```



```
366         if (!buf.startsWith("+OK"))
367             return true;
368         buf = rmPREFIX(buf);
369
370
371         String[] temp=buf.split("&");
372         WifiGraph.QueryTrajectories = new String[temp.length];
373         int i=0;
374         for (String ss : temp) {
375             WifiGraph.QueryTrajectories[i] = ss;
376                 i++;
377         }
378
379
380
381         // MainActivity.QueryTrajectories[0] = buf.split("\t");
382         wMore.terminal += "Server response :" + buf + "\n";
383         return true;
384
385     }
386
387     private void searchDone() {
388         // TODO Auto-generated method stub
389         // close the connection
390         Message smsg = new Message();
391         smsg.arg1 = Messages.ENDSRCH;
392         smsg.arg2 = (int) (Double.parseDouble(LCSS) * 100);
393         WifiGraph.messageHandler.sendMessage(smsg);
394     }
395
396     // Connect to server to use the upper bound
397     private void serverConnect() {
398
399         // open the coordinates file
400 //      if (MainActivity.ClientTrajectory.size() == 0) {
401 //          Message msg = new Message();
402 //          msg.arg1 = MainActivity.USRINV;
403 //          WifiGraph.messageHandler.sendMessage(msg);
404 //          return;
405 //      }
406         try {
407             // ACCEPT FROM SERVER
408             host = serverAddress;
409             try{
410             port=Integer.parseInt(wserverPort);
411             }catch (Exception e) {
412                 // TODO: handle exception
413                 port=443;
414             }
415
416             connection = new Socket(host, port);
417             synchronized (wMore.terminal) {
418                 wMore.terminal += "Socket created.\nConnecting to server on " + host + "...\n";
419             }
420             // INITIALIZE THE IN AND OUT STREAMS
421             inStream = connection.getInputStream();
422             in = new BufferedReader(new InputStreamReader(inStream));
423             outStream = connection.getOutputStream();
424             out = new DataOutputStream(outStream);
425         } catch (IOException e) {
426             Message msg = new Message();
427             msg.arg1 = Messages.ENDCONN;
428             WifiGraph.messageHandler.sendMessage(msg);
429             return;
430         }
431         // --------------------------------ESTABLISH-STATE----------------------------------------
432         try {
433             // If the ESTABLISH-STATE is wrong
434             if (!ESTABLISH()) {
435                 appClose();
436                 return;
437             }
438         } catch (IOException e) {
```

```
            synchronized (wMore.terminal) {
                wMore.terminal += "Error Connection: " + e.getMessage();
            }
            appClose();
            return;
        }
        // ------------------------------ESTABLISH-STATE---------------------------------------

        while (true) {

            // the thread is send a search query
            // and the server answer with an +OK or -ERR message
            curState = state.ESTABLISH;
            try {
                buf = in.readLine();
                //Reset the epsilon value
                // // Check for any change in preferences
                epsilon = WifiGraph.preferences.getString("wepsilon", "n/a");
                if (epsilon.equals("n/a") || epsilon.equals("")) {
                    Message msg = new Message();
                    msg.arg1 = Messages.DEFAULT_EPSILON;
                    WifiGraph.messageHandler.sendMessage(msg);
                    // toast.show();
                    epsilon = "5";
                }


                wMore.terminal += "";
            } catch (IOException e) {
                synchronized (wMore.terminal) {
                    wMore.terminal += "Error Connection: " + e.getMessage();
                    // toast.show();
                }
                break;
            }

            // ------------------------------SEARCH-STATE---------------------------------------
            if (buf.startsWith("+OK")) {
                if (buf.contains("Goodbye")) {
                    wMore.terminal += "\n+OK Goodbye\n";
                    break;
                }
                try {
                    if (!SEARCH())
                        break;
                } catch (IOException e) {
                    synchronized (wMore.terminal) {
                        wMore.terminal += "Error Connection: " + e.getMessage();
                        // toast.show();
                    }
                    break;
                } // end try
                try {
                    if (!RETRIEVE())
                        break;
                } catch (IOException e) {
                    synchronized (wMore.terminal) {
                        wMore.terminal += "Error Connection: " + e.getMessage();
                        // toast.show();
                    }
                    break;
                } // end try
                searchDone();
                continue;

            } else if (buf.startsWith("-ERR")) {
                Message msg1 = new Message();
                msg1.arg1 = Messages.SERVER_ERR;
                WifiGraph.messageHandler.sendMessage(msg1);
                wMore.terminal += "Server Busy\nPlease try again";
                continue;
            }
            // ------------------------------SEARCH-STATE---------------------------------------
```



```
            // ------------------------------CALCULATE-STATE---------------------------------------
            try {
                if (!CALCULATE())
                    break;
            } catch (IOException e) {
                synchronized (wMore.terminal) {
                    wMore.terminal += "Error Connection: " + e.getMessage();
                    // toast.show();
                }
                break;
            } // end try
            // ------------------------------CALCULATE-STATE---------------------------------------

        }// end while

        // ELSE CLOSE THE PROGRAM
        // ending and closing the application
        try {
            connection.close();
            synchronized (wMore.terminal) {
                wMore.terminal += "Done.\nConnection closed.\n";
                // toast.show();
            }
        } catch (IOException e) {
            synchronized (wMore.terminal) {
                wMore.terminal += "Error Connection: " + e.getMessage();
                // toast.show();
            }
            appClose();
            return;
        } // end try
        Message mmsg = new Message();
        mmsg.arg1 = Messages.ENDCONN;
        WifiGraph.messageHandler.sendMessage(mmsg);
        return;

    }

    /**
     *
     */
    void appClose() {
        // ending and closing the application
        try {
            connection.close();
        } catch (IOException e) {
        }
        Message msg = new Message();
        msg.arg1 = Messages.OTHER;
        WifiGraph.messageHandler.sendMessage(msg);

    }
}
```

```java
/*
 * (C) Copyright University of Cyprus. 2010-2011.
 *
 * SmartTrace Server API
 *
 * @version         : 1.0
 * @author : Costantinos Costa(costa.costantinos@gmail.com)
 * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
 * Computer Science Department , University of Cyprus
 *
 *
 */
package SmartTrace.apk;

import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.text.TextUtils;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

public class wEditTextBrowser extends Activity {

    final public static String wPATH = "wifiPath";
    Button btnOk;
    Button btnCancel;
    Button btnBrowse;
    private SharedPreferences mPrefs;
    public static EditText txb;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.dialog_layout);
        btnOk = (Button) findViewById(R.id.btnOk);
        btnCancel = (Button) findViewById(R.id.btnCancel);
        btnBrowse = (Button) findViewById(R.id.btnBrowse);
        txb = (EditText) findViewById(R.id.txbFile);
        txb.setScrollContainer(true);
        btnCancel.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                Stop();
            }
        });
        btnBrowse.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                Intent i = new Intent(wEditTextBrowser.this,
                        AndroidFileBrowser.class);
                AndroidFileBrowser.flagMainOrwifi=false;
                startActivity(i);
            }
        });

        mPrefs = PreferenceManager.getDefaultSharedPreferences(this);
        WifiGraph.custFilePath = mPrefs.getString(wPATH, "/sdcard/");

        txb.setText(WifiGraph.custFilePath);
        btnOk.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
```



```java
                WifiGraph.custFilePath = txb.getText().toString();
                Stop();
            }
        });
    }

    @Override
    protected void onRestoreInstanceState(Bundle savedInstanceState) {
        WifiGraph.custFilePath = savedInstanceState.getString(wPATH);
        if (!TextUtils.isEmpty(WifiGraph.custFilePath)) {
            // Update lbResult
            wEditTextBrowser.txb.setText(savedInstanceState.getString(wPATH));
        }
        super.onRestoreInstanceState(savedInstanceState);
    }

    @Override
    protected void onSaveInstanceState(Bundle outState) {
        /*************************************************
         * Here we do _temporary_ save all critical data, like our
         * selectedProduct.
         *************************************************/
        outState.putString(wPATH, WifiGraph.custFilePath);
        super.onSaveInstanceState(outState);
    }

    void Stop() {
        SharedPreferences.Editor ed = mPrefs.edit();
        ed.putString(wPATH, WifiGraph.custFilePath);
        ed.commit();
        finish();
    }

}
```

```
/*
 * (C) Copyright University of Cyprus. 2010-2011.
 *
 * SmartTrace Server API
 *
 * @version         : 1.0
 * @author : Costantinos Costa(costa.costantinos@gmail.com)
 * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
 * Computer Science Department , University of Cyprus
 *
 *
 */
/*
 * (C) Copyright University of Cyprus. 2010-2011.
 *
 * SmartTrace Server API
 *
 * @version         : 1.0
 * @author : Costantinos Costa(costa.costantinos@gmail.com)
 * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
 * Computer Science Department , University of Cyprus
 *
 *
 */
package SmartTrace.apk;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Timer;
import java.util.TimerTask;
import android.app.Activity;
import android.app.AlertDialog;
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.DialogInterface.OnClickListener;
import android.content.SharedPreferences.Editor;
import android.net.ConnectivityManager;
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.provider.Settings;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.Toast;

public class WifiGraph extends Activity {
    protected static int K;

    protected volatile static SharedPreferences preferences;
    protected volatile static Handler messageHandler;
    public volatile static String[] QueryTrajectories = null;
    protected volatile static boolean privacy = false;
    WifiManager mainWifi;
```



```
    WifiReceiver receiverWifi;
    List<ScanResult> wifiList;
    StringBuilder sb = new StringBuilder();
    static ArrayList<HashMap<String, Integer>> ClientTrajectory = new ArrayList<HashMap<String, Integer>>
    ();
    static ArrayList<HashMap<String, Integer>> ServerTrajectory = new ArrayList<HashMap<String, Integer>>
    ();

    Intent intent; // Reusable Intent for each intent

    private Toast warn;
    private Toast info;
    private boolean connected = false;

    private int key = 0;
    private WConnect con;
    private String username = new String();
    private LinearLayout graphLayout;
    static String custFilePath = "/sdcard/wifi.txt";;
    public static boolean isUsernameSet = false;
    private boolean useTrace = false;
    private static String outpFileloc;
    private GraphView graphView;
    private BufferedWriter writer = null;
    //private BufferedReader reader = null;
    private File outFile = null;
    private File inFile = null;
    private boolean append;
    private double Density;
    private boolean record = false;
    private Timer timer;

    @Override
    public void onOptionsMenuClosed(Menu menu) {
        // TODO Auto-generated method stub
        super.onOptionsMenuClosed(menu);
        if (con != null) {
            con.start();
            con = null;
        }
    }

    private void initializeComponents() {
        // initialize toast for warnings
        warn = Toast.makeText(WifiGraph.this, "", Toast.LENGTH_LONG);
        View wifitextView = warn.getView();
        LinearLayout wifilay = new LinearLayout(WifiGraph.this);
        wifilay.setOrientation(LinearLayout.HORIZONTAL);
        ImageView wifiview = new ImageView(WifiGraph.this);
        wifiview.setImageResource(R.drawable.warning);
        wifilay.addView(wifiview);
        wifilay.addView(wifitextView);
        warn.setView(wifilay);
        // initialize toast for informations
        info = Toast.makeText(WifiGraph.this, "", Toast.LENGTH_LONG);
        View textView = info.getView();
        LinearLayout lay = new LinearLayout(WifiGraph.this);
        lay.setOrientation(LinearLayout.HORIZONTAL);
        ImageView view = new ImageView(WifiGraph.this);
        view.setImageResource(R.drawable.info);
        lay.addView(view);
        lay.addView(textView);
        info.setView(lay);
        graphLayout = (LinearLayout) findViewById(R.id.wifiLayout);
        messageHandler = new Handler() {
            public void handleMessage(Message msg) {
                super.handleMessage(msg);

                // Connection mycon;
                switch (msg.arg1) {

                case Messages.CONNECT:
                    info.setText("Connecting...");
```

```
                    info.show();
                    break;
                case Messages.INVALID_ADDRESS:
                    warn.setText("Invalid Address...");
                    warn.show();
                    connected = false;
                    // searched = false;
                    break;
                case Messages.SEARCH:
                    info.setText("You are Searching...");
                    info.show();
                    break;
                case Messages.DEFAULT_EPSILON:
                    info.setText("Using default epsilon(5)...");
                    info.show();
                    break;
                case Messages.LCSS:
                    info.setText("Calculating LCSS...!");
                    info.show();
                    // PlotSmrt(msg.arg2);
                    break;
                case Messages.UPBOUND:
                    info.setText("Calculating UB...!");
                    info.show();
                    break;
                case Messages.ENDSRCH:
                    info.setText("Searching done succesfully!");
                    info.show();
                    wifiSmrt(msg.arg2);
                    break;
                case Messages.QUIT:
                case Messages.ENDCONN:
                    info.setText("Connecting done!");
                    info.show();
                    connected = false;
                    break;
                case Messages.SERVER_ERR:
                    warn.setText(Messages.errors.get(msg.arg2));
                    warn.show();
                    break;
                case Messages.STATECAL:
                    warn.setText("You are in calculate state\nPlease try again!");
                    warn.show();
                    break;
                case Messages.STATERET:
                    warn.setText("You are in retrieval state\nPlease try again!");
                    warn.show();
                    break;
                case Messages.STATESER:
                    warn.setText("You are already searching!\nPlease try again!");
                    warn.show();
                    break;
                default:
                    warn.setText("Connection closed.");
                    warn.show();

                    connected = false;
                    // searched = false;
                    break;
                }

            }
        };

        final PopUp popup = (PopUp) findViewById(R.id.popupWifiwindow);
        final PopUp popuptxt = (PopUp) findViewById(R.id.popupWifiText);
        popup.setVisibility(View.GONE);
        popup.setTitle("Do you want to switch to GPS mode?");
        final Button btn = (Button) findViewById(R.id.ppWifibutton);
        final Button btnCancel = (Button) findViewById(R.id.btnWCancel);
        final Button btnOk = (Button) findViewById(R.id.btnWOk);
        final Button btnClose = (Button) findViewById(R.id.btnWifiClose);
        preferences = PreferenceManager.getDefaultSharedPreferences(this);
```



```

        // initialize the input file
        custFilePath = preferences.getString(wEditTextBrowser.wPATH, "/sdcard/");

        btnClose.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                btnClose.setBackgroundResource(R.drawable.disclose);
                AlertDialog.Builder builder = new AlertDialog.Builder(WifiGraph.this);
                builder.setMessage("Are you sure you want to exit?").setCancelable(false)
                        .setPositiveButton("Yes", new OnClickListener() {

                            @Override
                            public void onClick(DialogInterface dialog, int which) {
                                // TODO Auto-generated method stub
                                appClose();

                            }
                        }).setNegativeButton("No", new OnClickListener() {

                            @Override
                            public void onClick(DialogInterface dialog, int which) {
                                // TODO Auto-generated method stub
                                btnClose.setBackgroundResource(R.drawable.close);
                                return;

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

        });

        btnCancel.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                popup.setVisibility(View.GONE);
                btn.setBackgroundResource(R.drawable.leftarrow);
                popuptxt.setVisibility(View.VISIBLE);
                btnClose.setVisibility(View.VISIBLE);
            }
        });

        btnOk.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                Intent i = new Intent(WifiGraph.this, MainActivity.class);
                startActivity(i);
                appClose();
            }
        });

        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View arg0) {
                if (key == 0) {
                    key = 1;
                    popup.setVisibility(View.VISIBLE);
                    btn.setBackgroundResource(R.drawable.selectorrigth);
                    popuptxt.setVisibility(View.GONE);
                    btnClose.setVisibility(View.GONE);
                } else if (key == 1) {
                    key = 0;
                    popup.setVisibility(View.GONE);
                    btn.setBackgroundResource(R.drawable.selectorleft);
                    popuptxt.setVisibility(View.VISIBLE);
                    btnClose.setVisibility(View.VISIBLE);
                }
            }
```

```
291         });
292
293         // show that scanning started
294         info.setText("Scanning...");
295         info.show();
296
297         mainWifi = (WifiManager) getSystemService(Context.WIFI_SERVICE);
298         receiverWifi = new WifiReceiver();
299         registerReceiver(receiverWifi, new IntentFilter(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION));
300         mainWifi.startScan();
301         timer = new Timer();
302         // change the mode of the comparison
303         timer.schedule(new TimerTask() {
304
305             @Override
306             public void run() {
307                 // TODO Auto-generated method stub
308                 mainWifi.startScan();
309
310             }
311         }, 2000, 2000);
312     }
313
314     @Override
315     public void onCreate(Bundle savedInstanceState) {
316         super.onCreate(savedInstanceState);
317         setContentView(R.layout.wifi);
318         initializeComponents();
319         //set The UserName
320         if(!setUsername()){
321
322             final AlertDialog.Builder alert = new AlertDialog.Builder(this);
323             final EditText input = new EditText(this);
324             alert.setView(input);
325             alert.setMessage("Please write your Screen Name");
326             alert.setPositiveButton("Ok", new DialogInterface.OnClickListener() {
327                 public void onClick(DialogInterface dialog, int whichButton) {
328                     username = input.getText().toString().trim();
329                     Editor editor = preferences.edit();
330                     editor.putString("wusrName", username);
331                     editor.commit();
332                     warn.setText(username);
333                     warn.show();
334                 }
335             });
336
337             alert.setNegativeButton("Cancel",
338                     new DialogInterface.OnClickListener() {
339                         public void onClick(DialogInterface dialog, int whichButton) {
340                             dialog.cancel();
341                         }
342                     });
343             alert.show();
344
345
346
347
348
349         }
350     }
351
352     protected void appClose() {
353         // TODO Auto-generated method stub
354         finish();
355         System.exit(0);
356     }
357
358     /* Creates the menu items */
359     /* Creates the menu items */
360     public boolean onCreateOptionsMenu(Menu menu) {
361         menu.add(0, R.id.record, 0, "Record").setIcon(R.drawable.recorded);
362         menu.add(0, R.id.Cons, 1, "Connect").setIcon(R.drawable.conser);
363         menu.add(0, R.id.SmrtTrc, 2, "Smart Trace").setIcon(R.drawable.smart);
```



```
364         menu.add(0, R.id.Share, 3, "Share").setIcon(R.drawable.share);
365         // menu.add(0, R.id.OnOff, 3,
366         // "Online/Offline").setIcon(R.drawable.offgps);
367         // menu.add(0, R.id.mapMode, 0, "Map Mode").setIcon(R.drawable.mode);
368         menu.add(0, R.id.preferences, 4, "Settings").setIcon(R.drawable.prefer);
369         menu.add(0, R.id.more, 5, "More ...").setIcon(R.drawable.more);
370         return true;
371     }
372
373     @Override
374     public boolean onMenuOpened(int featureId, Menu menu) {
375         // TODO Auto-generated method stub
376         MenuItem item = menu.getItem(1);
377         MenuItem itemrecord = menu.getItem(0);
378         if (!connected) {
379             item.setIcon(R.drawable.conser);
380             item.setTitle("Connect");
381         } else {
382             item.setIcon(R.drawable.disser);
383             item.setTitle("Disconnect");
384         }
385         if (!record) {
386             itemrecord.setIcon(R.drawable.recorded);
387             itemrecord.setTitle("Record");
388         } else {
389             itemrecord.setIcon(R.drawable.record);
390             itemrecord.setTitle("Recording...");
391         }
392         return super.onMenuOpened(featureId, menu);
393     }
394
395     // This method is called once the menu is selected
396     public boolean onOptionsItemSelected(MenuItem item) {
397         Intent i = null;
398         Message msg = null;
399         switch (item.getItemId()) {
400         // We have only 5 menu option
401         case R.id.OnOff:
402             // Launch OnOffPreferences activity
403             i = new Intent(WifiGraph.this, WifiOnOffPreferences.class);
404             startActivity(i);
405
406             break;
407         // Smart trace request to Server with new thread
408         case R.id.SmrtTrc:
409
410             if (!connected) {
411                 warn.setText("You should connect first!");
412                 warn.show();
413                 break;
414             }
415             msg = new Message();
416             msg.arg1 = Messages.SEARCH;
417             WConnect.messageHandler.sendMessage(msg);
418
419             break;
420         case R.id.Cons:
421             // Connect to Server with new thread
422             final ConnectivityManager connMgr = (ConnectivityManager) this
423                     .getSystemService(Context.CONNECTIVITY_SERVICE);
424             final android.net.NetworkInfo wifi = connMgr
425                     .getNetworkInfo(ConnectivityManager.TYPE_WIFI);
426
427             if (!connected) {
428                 if (setUsername()) {
429                     if (wifi.isAvailable()) {
430                         // set the trace flag and the trace file' s name
431                         custFilePath = preferences.getString(wEditTextBrowser.wPATH, "/sdcard/");
432                         // useTrace = preferences.getBoolean("OffWifi", false);
433                         if (record) {
434                             warn.setText("Stop recording wifi points");
435                             warn.show();
436                             record = false;
```

```
437                         }
438                         // create a new thread for the connection
439                         con = new WConnect(getApplicationContext(), useTrace, username, this);
440                         wMore.terminal = "";
441                         connected = true;
442                     } else {
443                         gpsWifiAlert("Wifi");
444                         break;
445                     }
446                 } else {
447                     warn.setText("The username is not set yet!!!");
448                     warn.show();
449                     break;
450                 }
451 
452             } else {
453                 msg = new Message();
454                 msg.arg1 = Messages.QUIT;
455                 WConnect.messageHandler.sendMessage(msg);
456                 connected = false;
457             }
458 
459             break;
460 
461         case R.id.record:
462             if (useTrace) {
463                 warn.setText("You are using a trace file.\nTo collect point go in online mode");
464                 warn.show();
465                 break;
466             }
467             // Launch ModeOfMap activity
468             if (!record) {
469                 WConnect.ClientTRAJ = "";
470                 outpFileloc = preferences.getString("wOutpFile", "/sdcard/wifi.txt");
471                 append = preferences.getBoolean("wAppend", false);
472                 try {
473                     outFile = new File(outpFileloc);
474                     writer = new BufferedWriter(new FileWriter(outFile, append));
475                 } catch (Exception e) {
476                     warn.setText(
477 
478                     "Error when opening output file:\n" + e.getMessage());
479                     warn.show();
480                     break;
481                 }
482                 record = true;
483             } else {
484                 record = false;
485                 if (writer != null)
486                     try {
487                         writer.close();
488                     } catch (IOException e) {
489                         // TODO Auto-generated catch block
490                         e.printStackTrace();
491                     }
492 
493             }
494             break;
495         case R.id.more:
496             // Launch more activity
497             i = new Intent(WifiGraph.this, wMore.class);
498             startActivity(i);
499 
500             break;
501         case R.id.Share:
502             // Launch Preferences activity
503             i = new Intent(WifiGraph.this, wPreferences.class);
504             startActivity(i);
505             info.setText("Here are the application's preferences.");
506             info.show();
507             // con = null;
508             break;
509         case R.id.preferences:
```



```
510             // Launch Preferences activity
511             i = new Intent(WifiGraph.this, WifiOnOffPreferences.class);
512             startActivity(i);
513             info.setText("Here are the application's settings.");
514             info.show();
515             break;
516         }
517         return true;
518     }
519 
520     class WifiReceiver extends BroadcastReceiver {
521 
522         public void onReceive(Context c, Intent intent) {
523             wifiList = mainWifi.getScanResults();
524             HashMap<String, Integer> WifiAP = new HashMap<String, Integer>();
525             WifiAP.clear();
526 
527             for (int i = 0; i < wifiList.size(); i++) {
528                 WifiAP.put(wifiList.get(i).BSSID/* +"\n"+wifiList.get(i).SSID */,
529                         wifiList.get(i).level);
530                 if (record) {
531                     try {
532                         WConnect.ClientTRAJ += wifiList.get(i).BSSID + "," + wifiList.get(i).level
533                                 + "&";
534                         writer.append(wifiList.get(i).BSSID + "," + wifiList.get(i).level + "\n");
535 
536                     } catch (IOException e) {
537                         // TODO Auto-generated catch block
538                         warn.setText("Exception in onReceive(): " + e.getMessage());
539                         warn.show();
540                     }
541                 }
542 //              else
543 //                  break;
544             }
545             if (record) {
546                 try {
547 
548                     // add the new line
549                     WConnect.ClientTRAJ += "#";
550                     writer.append("#\n");
551                 } catch (IOException e) {
552                     // TODO Auto-generated catch block
553                     warn.setText("Exception in onReceive(): " + e.getMessage());
554                     warn.show();
555                 }
556                 ClientTrajectory.add(WifiAP);
557             }
558             // toast.setText("Size=" + ClientTrajectory.size());
559             // toast.show();
560             int i = 0;
561             float[] values = new float[WifiAP.size()];
562             String[] verlabels = new String[] { "-100db", "-90db", "-80db", "-70db", "-60db",
563                     "-50db", "-40db", "-30db", "-20db", "-10db", "0db" };
564             String[] horlabels = new String[WifiAP.size()];
565             for (String string : WifiAP.keySet()) {
566                 horlabels[i] = string;
567                 values[i] = (float) WifiAP.get(string);
568                 i++;
569             }
570 
571             graphView = new GraphView(graphLayout.getContext(), values, "wifi", horlabels,
572                     verlabels, GraphView.BAR);
573             graphLayout.removeAllViews();
574             graphLayout.addView(graphView);
575             graphView.invalidate();
576         }
577     }
578 
579     protected void onPause() {
580         // unregisterReceiver(receiverWifi);
581         getPref();
582         super.onPause();
```

```
583     }
584
585     protected void onResume() {
586         // registerReceiver(receiverWifi, new
587         // IntentFilter(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION));
588         getPref();
589         super.onResume();
590     }
591
592     void wifiSmrt(double lcss) {
593
594
595         String folowedUsers = "";
596
597         for (int i=1;i<QueryTrajectories.length;i++) {
598             folowedUsers+=QueryTrajectories[i];
599         }
600
601         String msg;
602         if(folowedUsers.length()>0)
603             msg="folowed by "+folowedUsers;
604         else
605             msg="";
606         AlertDialog.Builder builder = new AlertDialog.Builder(WifiGraph.this);
607
608         DecimalFormat df = new DecimalFormat("##.##");
609         builder.setMessage("LCSS = " + df.format(lcss / 100) + "% with " + QueryTrajectories[0] +msg +".↙
    n")
610                 .setCancelable(false).setPositiveButton("OK", new OnClickListener() {
611                     @Override
612                     public void onClick(DialogInterface dialog, int which) {
613                         // TODO Auto-generated method stub
614                         return;
615
616                     }
617                 });
618         AlertDialog alert = builder.create();
619         alert.show();
620
621     }
622
623     private void readFromFile() {
624
625         String line = "";
626
627         inFile = new File(custFilePath);
628         BufferedReader reader = null;
629         if (!inFile.exists()) {
630             warn.setText("Provide path/filename for input file that exists.\n(menu preferences)\n");
631             warn.show();
632             return;
633         }
634         try {
635             reader = new BufferedReader(new FileReader(inFile));
636         } catch (IOException e1) {
637             // TODO Auto-generated catch block
638             warn.setText("Error:" + e1.getMessage());
639             warn.show();
640         }
641
642         try {
643             WConnect.ClientTRAJ = "";
644             HashMap<String, Integer> tempMap = new HashMap<String, Integer>();
645             while ((line = reader.readLine()) != null) {
646                 if (line.equals("#")) {
647                     WConnect.ClientTRAJ += "#";
648                     ClientTrajectory.add(tempMap);
649                     tempMap = new HashMap<String, Integer>();
650                 } else {
651                     tempMap.put(line.split(",")[0], Integer.parseInt(line.split(",")[1]));
652                     WConnect.ClientTRAJ += line + "&";
653                 }
654             }
```



```
655             reader.close();
656         } catch (Exception e) {
657             // TODO Auto-generated catch block
658             warn.setText("Error:" + e.getMessage());
659             warn.show();
660         }
661     }
662
663     boolean setUsername() {
664         // Check for any change in preferences
665         username = WifiGraph.preferences.getString("wusrName", "");
666         if (username.length() <= 0 || username.contains(" \t\n\b\"\'){}[];")) {
667             isUsernameSet = false;
668             return false;
669         } else {
670             isUsernameSet = true;
671             return true;
672         }
673
674     }
675
676     private void gpsWifiAlert(final String sett) {
677
678         final AlertDialog.Builder builder = new AlertDialog.Builder(this);
679         builder.setMessage("Your " + sett + " seems to be disabled, do you want to enable it?")
680                 .setCancelable(false).setPositiveButton("Yes",
681                         new DialogInterface.OnClickListener() {
682                             @Override
683                             public void onClick(DialogInterface dialog, int which) {
684                                 // TODO Auto-generated method stub
685                                 if (sett.equals("GPS"))
686                                     startActivity(new Intent(
687                                             Settings.ACTION_LOCATION_SOURCE_SETTINGS));
688                                 else
689                                     startActivity(new Intent(Settings.ACTION_WIFI_SETTINGS));
690
691                             }
692                         }).setNegativeButton("No", new DialogInterface.OnClickListener() {
693                     public void onClick(DialogInterface dialog, int which) {
694                         dialog.cancel();
695                     }
696                 });
697         final AlertDialog alert = builder.create();
698         alert.show();
699     }
700
701     public void checkTrace() {
702         WifiGraph.ClientTrajectory.clear();
703         if (!useTrace) {
704             if (WifiGraph.ClientTrajectory.size() != 0) {
705                 appClose();
706             } else {
707                 warn.setText("Press Wifi record button to collect coordinates");
708                 warn.show();
709                 // appClose();
710             }
711             return;
712         }
713         try {
714             inFile = new File(WifiGraph.custFilePath);
715             if (!inFile.exists()) {
716                 warn
717                         .setText("Provide path/filename for input file that exists.\n(menu preferences)\n↙
    ");
718                 warn.show();
719                 // appClose();
720                 return;
721             }
722             if (WifiGraph.custFilePath.equals("n/a") || WifiGraph.custFilePath.equals("")) {
723                 info.setText("Using default trajectory file (Wifi) data).");
724                 info.show();
725                 inFile = new File(WifiGraph.outpFileloc);
726             } else {
```

```
727                 info.setText("Using trajectory file: " + WifiGraph.custFilePath + ".");
728                 info.show();
729                 // appClose();
730             }
731
732 //          reader = new BufferedReader(new FileReader(inFile));
733
734         } catch (Exception e) {
735             warn.setText("Error when opening file:" + e.getMessage());
736             warn.show();
737             // appClose();
738         }
739         // save the file for check if it is already processed
740         readFromFile();
741     }
742
743     void getPref() {
744         useTrace = preferences.getBoolean("OffWifi", false);
745         append = preferences.getBoolean("wAppend", false);
746         outpFileloc = preferences.getString("wOutpFile", "/sdcard/wifi.txt");
747         custFilePath = preferences.getString(WEditTextBrowser.wPATH, "/sdcard/wifi.txt");
748         privacy=preferences.getBoolean("wPrivacyTraj", false);
749         Density = preferences.getInt("wDensity", 1) == 0 ? 0.5 : preferences.getInt("wDensity", 1);
750         timer.cancel();
751         timer.purge();
752         timer = new Timer();
753         timer.schedule(new TimerTask() {
754
755             @Override
756             public void run() {
757                 // TODO Auto-generated method stub
758                 mainWifi.startScan();
759             }
760         }, (int) (Density * 1000), (int)( Density * 1000));
761
762     }
763
764 }
```

```java
/*
 * (C) Copyright University of Cyprus. 2010-2011.
 *
 * SmartTrace Server API
 *
 * @version         : 1.0
 * @author : Costantinos Costa(costa.costantinos@gmail.com)
 * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
 * Computer Science Department , University of Cyprus
 *
 *
 */
package SmartTrace.apk;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;

public class WifiMobileLCSS {
    static double LCSS(ArrayList<HashMap<String, Integer>> client,
            ArrayList<HashMap<String, Integer>> server, double eLCSS) {
        // DecimalFormat myFormatter = new DecimalFormat("000");

        // FileWriter fstream1 = null;
        // try {
        // fstream1 = new FileWriter("traces/"+file1+"-"+file2+".txt");
        // } catch (IOException e) {
        // // TODO Auto-generated catch block
        // e.printStackTrace();
        // }
        // BufferedWriter out1 = new BufferedWriter(fstream1);
        // FileWriter fstream2 = null;
        // try {
        // fstream2 = new FileWriter("traces/"+file2+"-"+file1+".txt");
        // } catch (IOException e) {
        // // TODO Auto-generated catch block
        // e.printStackTrace();
        // }
        // BufferedWriter out2 = new BufferedWriter(fstream2);

        if (server.size() <= 0) {
            return -1;
        }
        int level = Math.max(client.size(), server.size());

        double LccsSum = 0.0;
        // set that will have all the access point that we want to check
        HashSet<String> set;
        HashSet<String> sets = new HashSet<String>();
        HashSet<String> setc = new HashSet<String>();
        HashSet<String> union = new HashSet<String>();

        // fill the access point that we want to check
        int j = 0;
        for (j = 0; j < level; j++) {
            if (j < server.size()) {
                union.addAll(server.get(j).keySet());
                sets.addAll(server.get(j).keySet());
            }
            if (j < client.size()) {
                union.addAll(client.get(j).keySet());
                setc.addAll(client.get(j).keySet());
            }
        }
        HashSet<String> intersection = new HashSet<String>(setc);
        intersection.retainAll(sets);
        //set what you want to use
    //  set=intersection;
        set=union;
```



```java

//      System.out.println("Client 1:The number of AP is " + setc.size());
//      System.out.println("Server 2:The number of AP is " + sets.size());
//      System.out.println("Union :The number of AP is " + union.size());
//      System.out.println("Intersect :The number of AP is " + intersection.size());

//      System.out.println("The total set number is " + j);

        Integer num = 0;
        ArrayList<Integer> serverTemp = new ArrayList<Integer>();
        ArrayList<Integer> clientTemp = new ArrayList<Integer>();
        for (String str : set) {
            serverTemp.clear();
            clientTemp.clear();
            for (int i = 0; i < level; i++) {
                if (i < client.size()) {
                    num = client.get(i).get(str);
                    num = num == null ? new Integer(-110) : num;
                    clientTemp.add(num);
                    // try {
                    // out1.append(str+","+myFormatter.format(num)+"|   \t|");
                    // } catch (IOException e) {
                    // // TODO Auto-generated catch block
                    // e.printStackTrace();
                    // }
                }
                if (i < server.size()) {
                    num = server.get(i).get(str);
                    num = num == null ? new Integer(-110) : num;
                    serverTemp.add(num);
                    // try {
                    // out2.append(str+","+myFormatter.format(num)+"|   \t|");
                    // } catch (IOException e) {
                    // // TODO Auto-generated catch block
                    // e.printStackTrace();
                    // }
                }
            }
            LccsSum += LCSS1D(serverTemp, clientTemp, eLCSS);

            // try {
            // out1.append("\n");
            // out2.append("\n");
            // } catch (IOException e) {
            // // TODO Auto-generated catch block
            // e.printStackTrace();
            // }
        }
        // try {
        // out1.close();
        // out2.close();
        // } catch (IOException e) {
        // // TODO Auto-generated catch block
        // e.printStackTrace();
        // }

        return LccsSum / set.size()*((double)intersection.size()/(double)union.size());
    }

    // LCCSS for one dimension
    static double LCSS1D(ArrayList<Integer> serverTraj, ArrayList<Integer> clientTraj, double eps) {
        // PHASE 1
        // Initialize first column and row to assist the DP Table
        double[][] L = new double[clientTraj.size() + 1][serverTraj.size() + 1];

        for (int i = 1; i < clientTraj.size() + 1; i++) {
            for (int j = 1; j < serverTraj.size() + 1; j++) {
                if (containInFolder(clientTraj.get(i - 1), serverTraj.get(j - 1), eps))
                    L[i][j] = L[i - 1][j - 1] + 1;
                else
                    L[i][j] = Math.max(L[i - 1][j], L[i][j - 1]);
            }
        }
```

```
147
148         double result = L[clientTraj.size()][serverTraj.size()];
149
150         return ((result / (double) (Math.min(clientTraj.size(), serverTraj.size()))) * 100);
151     }
152
153         // envelope for Query
154     public static boolean containInFolder(int valueC, int valueS, double eps) {
155         if ((valueC - eps) <= valueS && (valueC + eps) >= valueS)
156             return true;
157         return false;
158
159     }
160
161     public  static void readFromFile(String custFilePath,ArrayList<HashMap<String, Integer>> Trajectory )
      {
162
163         String line = "";
164
165         File inFile = new File(custFilePath);
166         BufferedReader reader = null;
167         if (!inFile.exists()) {
168             System.out.print("Provide path/filename for input file that exists.\n(menu preferences)\n");
169
170             return;
171         }
172         try {
173             reader = new BufferedReader(new FileReader(inFile));
174         } catch (IOException e1) {
175             // TODO Auto-generated catch block
176         e1.printStackTrace();
177         }
178
179         try {
180             HashMap<String, Integer> tempMap = new HashMap<String, Integer>();
181             while ((line = reader.readLine()) != null) {
182                 if (line.equals("#")) {
183
184                     Trajectory.add(tempMap);
185                     tempMap = new HashMap<String, Integer>();
186                 } else {
187                     tempMap.put(line.split(",")[0], Integer.parseInt(line.split(",")[1]));
188
189                 }
190             }
191             reader.close();
192         } catch (Exception e) {
193             // TODO Auto-generated catch block
194             e.printStackTrace();
195         }
196     }
197
198
199 }
200
```

```
/*
 * (C) Copyright University of Cyprus. 2010-2011.
 *
 * SmartTrace Server API
 *
 * @version         : 1.0
 * @author : Costantinos Costa(costa.costantinos@gmail.com)
 * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
 * Computer Science Department , University of Cyprus
 *
 *
 */
package SmartTrace.apk;

import SmartTrace.apk.R;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.preference.CheckBoxPreference;
import android.preference.EditTextPreference;
import android.preference.Preference;
import android.preference.PreferenceActivity;
import android.preference.PreferenceManager;
import android.preference.Preference.OnPreferenceClickListener;

public class WifiOnOffPreferences extends PreferenceActivity {
    /** Called when the activity is first created. */
    String ListPreference;
    String editTextPreference;
    String ringtonePreference;
    String secondEditTextPreference;
    String customPref;
    SharedPreferences prefs;
    CheckBoxPreference onPref;
    CheckBoxPreference offPref;
    EditTextPreference etpUser;
    Preference OffCategory;
    Preference OnCategory;
    Preference InputFile;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.wifionoffprefers);
        // android:key="OffCategory"
        prefs = PreferenceManager.getDefaultSharedPreferences(getBaseContext());
        OffCategory = (Preference) findPreference("wOffCategory");
        OnCategory = (Preference) findPreference("wOnCategory");
        InputFile = (Preference) findPreference("wCustFile");
        onPref = (CheckBoxPreference) findPreference("OnWifi");

//      // when a click event is creating
        onPref.setOnPreferenceClickListener(new OnPreferenceClickListener() {
            public boolean onPreferenceClick(Preference preference) {
                checkOn();
                return true;
            }

        });
//
        offPref = (CheckBoxPreference) findPreference("OffWifi");
        offPref.setOnPreferenceClickListener(new OnPreferenceClickListener() {
            public boolean onPreferenceClick(Preference preference) {
                checkOff();
                return true;
            }

        });
        InputFile.setOnPreferenceClickListener(new OnPreferenceClickListener() {

            @Override
            public boolean onPreferenceClick(Preference preference) {
                // TODO Auto-generated method stub
                // Launch OnOffPreferences activity
```



```
                Intent i = new Intent(WifiOnOffPreferences.this, wEditTextBrowser.class);
                startActivity(i);
                return false;
            }
        });
    }
//
    @Override
    protected void onStart() {
        // TODO Auto-generated method stub
        super.onStart();
        checkOff();
        checkOn();
    }
//
    public void checkOff() {
        if (offPref.isChecked()) {
            offPref.setChecked(true);
            OffCategory.setEnabled(true);
            onPref.setChecked(false);
            OnCategory.setEnabled(false);
            onPref.setEnabled(true);
        } else {
            onPref.setChecked(true);
            OnCategory.setEnabled(true);
            offPref.setChecked(false);
            OffCategory.setEnabled(false);
            offPref.setEnabled(true);
        }
    }
//
    public void checkOn() {
        if (onPref.isChecked()) {
            onPref.setChecked(true);
            OnCategory.setEnabled(true);
            offPref.setChecked(false);
            OffCategory.setEnabled(false);
            offPref.setEnabled(true);
        } else {
            offPref.setChecked(true);
            OffCategory.setEnabled(true);
            onPref.setChecked(false);
            OnCategory.setEnabled(false);
            onPref.setEnabled(true);
        }
    }
}
```

```java
/*
 * (C) Copyright University of Cyprus. 2010-2011.
 *
 * SmartTrace Server API
 *
 * @version         : 1.0
 * @author : Costantinos Costa(costa.costantinos@gmail.com)
 * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
 * Computer Science Department , University of Cyprus
 *
 *
 */
package SmartTrace.apk;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;

public class WifiUB {
        static double upperBound(ArrayList<HashMap<String, Integer>> client,
            ArrayList<HashMap<String, Integer>> server, double eUb) {
        // double the size of the envelop that is used for lcss
//      double UB = 0.0;

        int level = Math.max(server.size(), client.size());

        // set that will have all the access point that we want to check
//      HashSet<String> set;
        HashSet<String> sets = new HashSet<String>();
        HashSet<String> setc = new HashSet<String>();
        HashSet<String> union = new HashSet<String>();

        // fill the access point that we want to check
        int j = 0;
        for (j = 0; j < level; j++) {
            if (j < server.size()) {
                union.addAll(server.get(j).keySet());
                sets.addAll(server.get(j).keySet());
            }
            if (j < client.size()) {
                union.addAll(client.get(j).keySet());
                setc.addAll(client.get(j).keySet());
            }
        }
        HashSet<String> intersection = new HashSet<String>(setc);
        intersection.retainAll(sets);
        // set what you want to use
         //set=intersection;
    //  set = union;

        //Integer num = 0;
//      ArrayList<Integer> serverTemp = new ArrayList<Integer>();
//      ArrayList<Integer> clientTemp = new ArrayList<Integer>();
//      for (String str : set) {
//          serverTemp.clear();
//          clientTemp.clear();
//          for (int i = 0; i < level; i++) {
//              if (i < client.size()) {
//                  num = client.get(i).get(str);
//                  num = num == null ? new Integer(-110) : num;
//                  clientTemp.add(num);
//              }
//              if (i < server.size()) {
//                  num = server.get(i).get(str);
//                  num = num == null ? new Integer(-110) : num;
//                  serverTemp.add(num);
//              }
//          }
//          UB += uBound(clientTemp, serverTemp, eUb);
//      }

        if (level <= 0)
            return 0;
```



```java
    //  return (UB / set.size()*((double)intersection.size()/((double)union.size()-(double)intersection.
    size())));
    return ((double)intersection.size()/(double)union.size())*100;
    }

//  private static double uBound(ArrayList<Integer> serverTemp, ArrayList<Integer> clientTemp,
//          double e) {
//      // TODO Auto-generated method stub
//      // double the size of the envelop that is used for lcss
//      double UB = 0.0;
//      int size = Math.min(serverTemp.size(), clientTemp.size());
//      for (int i = 0; i < size; i++) {
//          // create envelop e
//          if (WifiMobileLCSS.containInFolder(serverTemp.get(i), clientTemp.get(i), e))
//          UB++;
//      }
//      if (size == 0)
//          return 0;
//      else
//          return ((UB / size)) * 100;
//  }
}
```

```
/*
 * (C) Copyright University of Cyprus. 2010-2011.
 *
 * SmartTrace Server API
 *
 * @version         : 1.0
 * @author : Costantinos Costa(costa.costantinos@gmail.com)
 * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
 * Computer Science Department , University of Cyprus
 *
 *
 */
package SmartTrace.apk;

import SmartTrace.apk.R;
import android.app.Activity;
import android.graphics.Color;
import android.os.Bundle;
import android.text.method.ScrollingMovementMethod;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class wMore extends Activity {
    public static String terminal="";
    private TextView textView;
    private Button btnClear;
    private Button btnAbout;
    private Button btnBack;
    private Button btncon;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.wmore);
        textView = (TextView) findViewById(R.id.textView);
        textView.setMovementMethod(new ScrollingMovementMethod());
        textView.setText("");
        btnClear = (Button) findViewById(R.id.clear);
        btnClear.setBackgroundResource(R.drawable.selector_clear);
        btnAbout = (Button) findViewById(R.id.about);
        btnAbout.setBackgroundResource(R.drawable.selector_about);
        btnBack = (Button) findViewById(R.id.back);
        btnBack.setBackgroundResource(R.drawable.selector_back);
        btncon = (Button) findViewById(R.id.btncon);
        btncon.setBackgroundResource(R.drawable.selector_terminal);
        initComp();
    }

    private void initComp() {
        textView.setScrollContainer(true);
        btnClear.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                textView.setTextColor(Color.WHITE);
                textView.setText("Clearing...\n\n");
            }
        });
        btnAbout.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                textView.setTextColor(0xFF80EE76);
                textView.setText("\n\nThe application created by...\n\n" );
                textView.append("\tCosta Costantinos\n");
                textView.append("\tZeinalipour Demetrios\n");
                textView.append("\tLaoudias Christos\n");
                textView.append("\n\t\tThank you\n");


            }
        });
        btnBack.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                finish();
            }
        });
```



```
        btncon.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                textView.setTextColor(Color.WHITE);
                if(terminal.equals(""))
                    textView.setText("None transaction yet..");
                    else
                textView.setText(terminal);
            }
        });

    }

}
```

```
/*
 * (C) Copyright University of Cyprus. 2010-2011.
 *
 * SmartTrace Server API
 *
 * @version         : 1.0
 * @author : Costantinos Costa(costa.costantinos@gmail.com)
 * Project Supervision : Demetris Zeinalipour (dzeina@cs.ucy.ac.cy)
 * Computer Science Department , University of Cyprus
 *
 *
 */
package SmartTrace.apk;

import android.content.SharedPreferences;
import android.os.Bundle;
import android.preference.EditTextPreference;
import android.preference.Preference;
import android.preference.PreferenceActivity;
import android.preference.Preference.OnPreferenceChangeListener;
import android.view.View;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.Toast;

public class wPreferences extends PreferenceActivity {
    private Toast toast;
    private EditTextPreference etpUser;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.wprefers);
        etpUser= (EditTextPreference) findPreference("wusrName");
        toast = Toast.makeText(this, "", Toast.LENGTH_LONG);
        View textView = toast.getView();
        LinearLayout lay = new LinearLayout(this);
        lay.setOrientation(LinearLayout.HORIZONTAL);
        ImageView view = new ImageView(this);
        view.setImageResource(R.drawable.warning);
        lay.addView(view);
        lay.addView(textView);
        toast.setView(lay);
        etpUser.setOnPreferenceChangeListener(new OnPreferenceChangeListener() {

            @Override
            public boolean onPreferenceChange(Preference preference, Object newValue) {
                // TODO Auto-generated method stub
                //
                // Check for any change in preferences

                String username = newValue.toString();
                        if (username.length() <= 0 || username.contains(" \t\n\b\"\'){}[];")) {
                            toast.setText("Invalid Username " + username);
                            toast.show();
                            SharedPreferences.Editor ed = preference.getEditor();
                            ed.putString("wusrName", "");
                            ed.commit();
                            toast.setText("Invalid Username");
                            toast.show();
                        } else {
                            MainActivity.isUsernameSet = true;
                            toast.setText("Thank you " + username);
                            toast.show();
                        }
                return true;
            }
        });
    }

}

```

Παράρτημα C