Ατομική Διπλωματική Εργασία

Υλοποίηση και Αποτίμηση του Πλαισίου Αναζήτησης σε Δίκτυα Έξυπνων Κινητών Συσκευών

Χρίστος Απλητσιώτης

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2012

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Υλοποίηση και Αποτίμηση του Πλαισίου Αναζήτησης σε Δίκτυα Έξυπνων Κινητών Συσκευών

Χρίστος Απλητσιώτης

Επιβλέπων Καθηγητής

Δημήτρης Ζεϊναλιπούρ

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

Μάιος 2012

Ευχαριστίες

Θα ήθελα αρχικά να ευχαριστήσω τον κύριο Δημήτρη Ζεϊναλιπούρ, ο οποίος μου προσέφερε αυτή την πολύ ενδιαφέρουσα και χρήσιμη εργασία καθώς και για την πλήρη κατανόηση και υποστήριξη που είχα από αυτόν, κατά τη διάρκεια της διπλωματικής μου εργασίας.

Θα ήθελα επίσης να ευχαριστήσω τον Ανδρέα Κωνσταντινίδη για την πολύτιμη βοήθεια που μου έχει προσφέρει σε όλη τη διάρκεια της διπλωματικής μου 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 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

Επίδειξη της παρούσας διπλωματικής εργασίας θα παρουσιαστεί στο ακόλουθο συνέδριο:

- *"SmartP2P: A Multiobjective Framework for Finding Social Content in P2P Smartphone Networks", A. Konstantinidis, C. Aplitsiotis, D. Zeinalipour-Yazti, Demo at the* 13th IEEE International Conference on Mobile Data Management *(MDM'12), IEEE Computer Society, Bangalore, India, July 23-26, 2012.*

Περίληψη

Η παρούσα διπλωματική εργασία εκπονήθηκε με σκοπό την υλοποίηση και την αξιολόγηση του πλαισίου αναζήτησης δεδομένων σε δίκτυα έξυπνων κινητών συσκευών, με τίτλο SmartOpt. Για την υλοποίηση του πλαισίου SmartOpt υλοποιήθηκε ένα σύστημα αναζήτησης δεδομένων σε ένα ομότιμο δίκτυο (*Peer–to–Pee*r) για έξυπνες κινητές συσκευές με λογισμικό Android. Το σύστημα αυτό επιτρέπει στους χρήστες να έχουν μια αποδοτική αναζήτηση γύρω από τα δεδομένα των υπόλοιπων χρηστών. Παράλληλα όμως, κάθε χρήστης κρατά τα δεδομένα του στην τοπική του μνήμη διασφαλίζοντας έτσι το απόρρητο του χρήστη αφού δεν χρειάζεται τα δεδομένα κάθε χρήστη να σταλούν σε κάποιο κεντρικό σημείο. Για την αποδοτική αυτή αναζήτηση υλοποιήθηκαν κάποιες τεχνικές που έχουν σαν στόχο την βελτίωση κάποιων παραμέτρων ταυτόχρονα, όπως είναι η κατανάλωσης της ενέργειας κατά τη διάρκεια της αναζήτησης σε κάθε συσκευή, ο χρόνος απόκρισης και ο ρυθμός ανάκλησης.

Συνολικά, το σύστημα αξιολογήθηκε με βάση δύο απλές τεχνικές και δύο τεχνικές πολυκριτηρίων που έχουν υλοποιηθεί. Η αξιολόγηση έγινε με δύο πραγματικά σύνολα δεδομένων το DBLP [8] το οποίο αφορά στοιχεία σημοσίευσης άρθρων σε συνδυασμό με τις τροχιές GeoLife [49] και το Pics-n-Trails [16] [17] που αφορά φωτογραφίες που έβγαζαν κάποιοι άνθρωποι κατά τη διάρκεια της κίνησης τους.

Η όλη υλοποίηση που έχει γίνει για το σύστημα αφορά μία εφαρμογή Android (SmartP2P), ένα εξυπηρετητή και μια Βάση Δεδομένων. Για την υλοποίηση της εφαρμογής Android και του εξυπηρετητή έχει χρησιμοποιηθεί η αντικειμενοστραφής γλώσσα προγραμματισμού JAVA και για τη Βάση Δεδομένων η MS SQL R2 2008. Η πειραματική υποδομή που χρησιμοποιήθηκε περιλαμβάνει το SmartLab το οποίο είναι ένα νέφος από 40 περίπου Android κινητές συσκευές.

Σύμφωνα με τα πειράματα που εκτελέσαμε οι δύο τεχνικές πολυκριτηρίων που υλοποιήθηκαν παρουσιάζονται να έχουν καλύτερη απόδοση στην κατανάλωση ενέργειας και στον χρόνο απόκρισης των αποτελεσμάτων σε σχέση με τις δύο πιο απλές τεχνικές.

Περιεχόμενα

Ευχαριστίες ..... **I**

Αναγνώριση ..... **II**

Περίληψη ..... **III**

Περιεχόμενα ..... **IV**

Κεφάλαιο 1: Εισαγωγή **1**

1.1 Σκοπός Εργασίας 1

1.2 Βασικό Υπόβαθρο 2

1.3 Αρχιτεκτονική Android 6

1.4 Περίγραμμα Εργασίας 8

Κεφάλαιο 2: Μοντέλο Συστήματος SmartOpt **9**

2.1 Μοντέλο Πλαισίου SmartP2P 9

2.2 Λεπτομέρειες Διασύνδεσης 10

2.3 Τεχνικές Αναζήτησης 11

2.4 Διατύπωση Προβλήματος 12

Κεφάλαιο 3: Πλαίσιο SmartOpt **14**

3.1 Εισαγωγή 14

3.2 Φάση Βελτιστοποίησης 15

3.3 Φάση Απόφασης 19

3.4 Φάση Αναζήτησης 20

Κεφάλαιο 4: Υλοποίηση Εφαρμογής SmartP2P **23**

4.1 Αρχιτεκτονική Συστήματος 23

4.1.1 Βάση Δεδομένων 24

4.1.2 Εξυπηρετητής SmartOpt 29

4.2 GUI Εφαρμογής SmartP2P 38

Κεφάλαιο 5: Πειραματική Μεθοδολογία **44**

5.1 Πειραματική Υποδομή 44

5.1.1 SmartLab 44

5.1.2 PowerTutor 45

5.1.3 Πειραματικός Εξυπηρετητής 46

5.1.4 Πειραματική Εφαρμογή 47

5.1.5 Περιγραφή Συνόλου Δεδομένων 48
5.1.5.1 Σύνολο Δεδομένων DBLP 49
5.1.5.2 Σύνολο Δεδομένων GeoLife 49
5.1.5.3 Σύνολο Δεδομένων Pics-n-Trails 49
5.1.6 Συσκευές που Χρησιμοποιήθηκαν 50
5.2 Πειραματική Διαδικασία 51
Κεφάλαιο 6: Πειραματικά Αποτελέσματα 53
6.1 Εισαγωγή 53
6.2 Πειραματικά Αποτελέσματα για χρόνο απόκρισης 54
6.3 Πειραματικά Αποτελέσματα για ενέργεια 56
6.4 Πειραματικά Αποτελέσματα για ρυθμό ανάκλησης 58
Κεφάλαιο 7: Συμπεράσματα και Μελλοντικές Επεκτάσεις 61
7.1 Συμπεράσματα 61
7.2 Μελλοντικές Επεκτάσεις 62
Βιβλιογραφία 63
Παράρτημα Α: Παρουσίαση Κώδικα MonkeyRunner για την Εφαρμογή SmartP2P A-1
Εισαγωγή A-1
Κώδικας MonkeyRunner A-1
Παράρτημα Β: Πηγαίος Κώδικας B-7
Βάση Δεδομένων B-7
Εξυπηρετητής SmartOpt B-28
Εφαρμογή SmartP2P B-72

Κεφάλαιο 1

Εισαγωγή

1.1 Σκοπός Εργασίας
1.2 Βασικό Υπόβαθρο
1.3 Αρχιτεκτονική Android
1.4 Περίγραμμα Εργασίας

1.1 Σκοπός Εργασίας

Στην εποχή που ζούμε έχουν αναπτυχθεί σε πολύ μεγάλο βαθμό οι τεχνολογίες των έξυπνων κινητών συσκευών. Ο συνδυασμός της ανάπτυξης αυτής μαζί με την εμφάνιση και ανάπτυξη των κοινωνικών δικτύων έχουν φέρει την επανάσταση στις εφαρμογές και υπηρεσίες των έξυπνων κινητών συσκευών. Η ήδη παρουσία και η ραγδαία ανάπτυξη καινοτόμων εφαρμογών [1] [5] έχει ως βασικό υπόβαθρο την έννοια του δικτύου έξυπνων κινητών συσκευών. Ένα παράδειγμα είναι η κοινωνική σελίδα Google Latitude [15] η οποία ενεργοποιεί χρήστες για να εντοπίζει αυτούς και τα κοινωνικά δίκτυα που έχουν επισκεφτεί. Η υπηρεσία αυτή ανέφερε πάνω από 3 εκατομμύρια εγγεγραμμένους χρήστες και πάνω από 1 εκατομμύριο ενεργούς χρήστες, ανεξάρτητα της αμφιλεγόμενης ανησυχίας για τα ιδιωτικά δεδομένα. Ομοίως υπάρχουν πολλές εφαρμογές όπως το Foursquare, Gowalla και Loopt οι οποίες απολαμβάνουν τεράστια επιτυχία στην κοινότητα των έξυπνων κινητών συσκευών. Επιπλέων, βρίσκεται σε εξέλιξη η ακαδημαϊκή έρευνα σε αυτόν τον τομέα.

Το μεγαλύτερο μέρος των υπηρεσιών κοινωνικής δικτύωσης οι οποίες είναι σχεδιασμένες για κοινότητες έξυπνων κινητών συσκευών βασίζονται σε αρχιτεκτονικές

νέφους ή σε αρχιτεκτονικές οι οποίες έχουν κεντρικές οντότητες. Για να μπορέσει να πραγματοποιηθεί αναζήτηση και ανταλλαγή δεδομένων μεταξύ των χρηστών των έξυπνων κινητών συσκευών θα πρέπει οι χρήστες να ανεβάζουν τα δεδομένα τους σε μια κεντρική οντότητα η οποία διαχειρίζεται τα δεδομένα και αναλαμβάνει την διάδοση τους. Τα προβλήματα που παρουσιάζονται είναι ότι γίνεται συγκέντρωση όλων των δεδομένων σε ένα κεντρικό σημείο το οποίο έχει ως αποτέλεσμα να θέτει σε κίνδυνο το απόρρητο του χρήστη. Επίσης, οι κινητές συσκευές καταναλώνουν πολύ ενέργεια προκειμένου να ανεβάσουν τα δεδομένα στην κεντρική οντότητα μέσω σύνδεσης WIFI/3G/4G.

Σκοπός της εργασίας αυτής είναι η ανάπτυξη ενός συστήματος το οποίο θα προσφέρει αποδοτική αναζήτηση γύρω από τα δεδομένα των υπόλοιπων χρηστών ενός δικτύου ενώ παράλληλα θα προστατεύει τα δεδομένα του κάθε χρήστη.

1.2 Βασικό Υπόβαθρο

Το σύστημα το οποίο υλοποιήθηκε στα πλαίσια της παρούσας διπλωματικής εργασίας έχει ως βασικό υπόβαθρο το SmartOpt πλαίσιο το οποίο αναφέρεται σε χρήστες δικτύων σχεδιασμένα για έξυπνες κινητές συσκευές. Το SmartOpt πλαίσιο παρουσιάζει κάποιες τεχνικές για αποδοτική αναζήτηση στα δεδομένα των υπόλοιπων χρηστών του δικτύου. Οι τεχνικές αυτές επιτρέπουν στους χρήστες των έξυπνων κινητών συσκευών να κρατούν τα δεδομένα τους τοπικά, για σκοπούς προφύλαξης των δεδομένων τους και καλύτερης διαχείρισης της ενέργειας των κινητών συσκευών.

Ένα δίκτυο έξυπνων κινητών συσκευών μπορούμε να το ορίσουμε ως ένα σύνολο από έξυπνες κινητές συσκευές που επικοινωνούν με ένα διακριτικό τρόπο, χωρίς την αποκλειστική αλληλεπίδραση του χρήστη, έτσι ώστε να πραγματοποιηθεί μια συνεργασία ή κοινωνική εργασία.

Σε ένα ομότιμο (Peer – to – Peer) δίκτυο έξυπνων κινητών συσκευών μπορεί η αναζήτηση στα δεδομένα των χρηστών του δικτύου να κατηγοριοποιηθεί σε Μη-ενημερωμένη [26] [44] αναζήτηση όπως χρησιμοποιεί το Gnutella [12] δίκτυο και σε

Ενημερωμένη αναζήτηση [40] [20] [35] [29] [21] [6] . Στην Μη-ενημερωμένη αναζήτηση όλοι οι κινητοί κόμβοι (έξυπνες κινητές συσκευές) μεταδίδουν το επερώτημα του χρήστη που ξεκινά τη διαδικασία της αναζήτησης με ένα αυθαίρετο τρόπο σε όσους περισσότερους κόμβους είναι δυνατόν σε σχέση με την Ενημερωμένη αναζήτηση η οποία προωθεί τις ερωτήσεις βασισμένη σε σημασιολογικές πληροφορίες ή πληροφορίες που αφορούν τοποθεσίες. Η αναζήτηση στην παρούσα εργασία αφορά την Ενημερωμένη αναζήτηση με τη διαφορά ότι οι κινητοί κόμβοι εγγράφονται σε μια κεντρική οντότητα. Χρησιμοποιούμε ένα μηχανισμό ο οποίος περιγράφει περιληπτικά τα δεδομένα των χρηστών του δικτύου [35] έτσι ώστε να μπορέσει να τα ανακαλύψει η κεντρική οντότητα με μια ερώτηση. Στην εφαρμογή SmartP2P κάθε χρήστης κατά την εγγραφή του στέλνει στην κεντρική οντότητα (εξυπηρετητή) τα δεδομένα του όπως περιγράφονται περιληπτικά όπου και αποθηκεύονται. Έτσι μπορούν ταυτόχρονα πολλοί χρήστες να χρησιμοποιούν τις συγκεκριμένες πληροφορίες για μια νέα αναζήτηση χωρίς να χρειάζεται η αποστολή τους ξανά.

Σύμφωνα με το πλαίσιο SmartOpt όταν ένας χρήστης Χ διεξάγει αναζήτηση για ένα αντικείμενο μέσα στο δίκτυο πρέπει ως πρώτο στάδιο να κατεβάσει από το εξυπηρετητή SmartOpt το δέντρο δρομολόγησης ερωτήματος στο οποίο θα γίνει η αναζήτηση. Κάθε κόμβος του δέντρου αναπαριστά και κάποιο χρήστη του δικτύου. Το δέντρο σε αυτή τη φάση υπόκειται σε μια βελτιστοποίηση πολλαπλών στόχων με σκοπό να βελτιστοποιεί κάποιες παραμέτρους ταυτόχρονα την στιγμή της αναζήτησης όπως την κατανάλωση της ενέργειας από τις κινητές συσκευές, το χρόνο άφιξης των αποτελεσμάτων στο Χ και το ρυθμό ανάκλησης.

Η βελτιστοποίηση πολλαπλών στόχων περιγράφεται ως η διαδικασία κατά την οποία δύο ή περισσότεροι αντικρουόμενοι στόχοι υπόκεινται ορισμένους περιορισμούς ταυτόχρονα. Στην παρούσα εργασία οι βελτιστοποιήσεις πολλαπλών στόχων χρησιμοποιούν ιδέες από τους εξελικτικούς αλγόριθμους πολλαπλών στόχων οι οποίοι παρουσιάζονται αποτελεσματικοί ως προς τη δημιουργία ενός συνόλου μη-κυριαρχούμενων λύσεων σε μια εκτέλεση τους. Το σύνολο αυτών των μη-κυριαρχούμενων λύσεων ονομάζουμε Pareto Front (PF).

Παρόμοια λειτουργία με τους εξελικτικούς αλγορίθμους έχουν και οι χειριστές Skyline οι οποίοι χρησιμοποιούνται κυρίως από τις βάσεις δεδομένων [30] για να ανακτήσουν ένα γενικό σύνολο μη-κυρίαρχων λύσεων ενός ερωτήματος Skyline με ένα συγκεντρωτικό ή κατανεμημένο τρόπο. Η έρευνα, η οποία επικεντρώνεται σε συγκεντρωτικές βάσεις δεδομένων, σκοπεύει στο μάζεμα όλων των πληροφοριών από όλες τις πηγές σε ένα συγκεντρωτικό κόμβο, ο οποίος με τη σειρά του ανακτά το γενικό skyline χρησιμοποιώντας συστηματικές μεθόδους. Για παράδειγμα, ο Tan [39] χρησιμοποίησε μια προσέγγιση που βασίζεται σε Bitmap για να ανακτήσει το Skyline χρησιμοποιώντας δεκαδικές πράξεις με αναπαράσταση από δυφία, όπως επίσης και έναν αλγόριθμο βασισμένο σε B-δέντρα το οποίο βελτιώνει περεταίρω την προηγούμενη ταχύτητα για απάντηση. Οι προσεγγίσεις Block Nested Loop και Sort-Filter Skyline [7] ψάχνουν εξαντλητικά τα σημεία μέσα στο σύνολο δεδομένων και ανακτούν τα σημεία του Skyline που είναι βασισμένα στη δική τους κυρίαρχη βαθμολογία, έχοντας ως κύρια διαφορά ότι, το δεύτερο αρχικά ταξινομεί τα σημεία του συνόλου δεδομένων σε αύξουσα σειρά πριν να εφαρμοστεί η μέθοδος Block Nested Loop. Παρομοίως, ο Godfrey [13] επέκτεινε τη δουλειά με το να προτείνει τον αλγόριθμο Linear-estimation-sort (LESS) ο οποίος μειώνει το κόστος του Sort-Filter Skyline αποκλείοντας μια μερίδα της βάσης δεδομένων κατά την ταξινόμηση. Ο Papadias [30] και ο Kossman [25] αντιμετώπισαν την πρόκληση της ανάκτησης του συγκεντρωτικού Skyline λαμβάνοντας υπόψη τους, τους κόμβους του R-δέντρου, χρησιμοποιώντας τις προσεγγίσεις branch-and-bound και NN προόδου, διαδοχικά.

Επιπρόσθετα, έγιναν πολλές προσπάθειες για την ανάκτηση του κατανεμημένου Skyline, χρησιμοποιώντας συχνά προσεγγίσεις που χώριζαν τοπικά το σύνολο δεδομένων είτε κάθετα είτε οριζόντια και μετά ανακτούσαν το γενικό Skyline μαζεύοντας όλα τα τοπικά skylines σε ένα κεντρικό κόμβο. Για παράδειγμα, ο Balke [3] πρότεινε ένα κάθετα διαχωρισμένο skyline αλγόριθμο, ο οποίος πραγματοποιεί ταξινόμηση βασισμένη σε round-robin προσέγγιση, μέχρι να βρει όλα τα μη κυρίαρχα σημεία για κάθε ξεχωριστή βάση δεδομένων. Συγκεκριμένα, ο Wu [43] χώρισε το πεδίο της βάσης δεδομένων σε τετραγωνισμένους χώρους και αντιστοίχησε τον κάθε εξυπηρετητή σε ένα πεδίο. Έτσι, κάθε εξυπηρετητής είναι υπεύθυνος για να βρίσκει το Skyline του δικού του πεδίου δεδομένων και μετά λαμβάνοντας υπόψη του κάποιες σχέσεις προτεραιότητας, το γενικό Skyline ανακτάται. Παρομοίως, ο Wang [42] και ο

Vlachou [41] πρότειναν ένα αλγόριθμο ανάκτησης του Skyline σε δίκτυα ομότιμων (Peer-to-Peer) οργανώνοντας τους κόμβους σε ένα επικαλυπτόμενο δίκτυο και σε υποδιαστήματα, διαδοχικά. Ο Zhu [46] πρόσφατα πρότεινε ένα κατανεμημένο αλγόριθμο skyline, βασισμένο σε ανάδραση (FDS), ο οποίος υποστηρίζει οριζόντιους διαχωρισμούς των συνόλων δεδομένων σε απομακρυσμένους εξυπηρετητές. Ωστόσο, όλες οι προαναφερθείς μελέτες, επεξεργάζονται τα τοπικά skyline, χωρίς να τους απασχολεί η κατανάλωση μνήμης και ενέργειας, ένα μείζον ζήτημα για τις έξυπνες κινητές συσκευές, που λαμβάνεται υπόψη στην δική μας περίπτωση.

Η περίπτωση που πλησιάζει πιο πολύ στην δική μας είναι αυτή του Huang [18] , στην οποία εξετάζονται οι ανακτήσεις του Skyline σε κινητές συσκευές του ασύρματων κινητών δικτύων. Προτείνεται ένας αλγόριθμος ο οποίος επαναλαμβανόμενα ανιχνεύει τις κινητές συσκευές και δημιουργεί ένα τοπικό skyline υπόδεντρο, για παράδειγμα, τα σημεία που βρίσκονται στο skyline έχουν τις ρίζες τους σε κάθε ανεξάρτητη συσκευή. Το γενικό skyline ανακτάται μετά που ανιχνεύονται όλες οι κινητές συσκευές. Γενικότερα, η διάσταση των προβλημάτων που αντιμετωπίζονται σε όλες τις προαναφερθέντες μεθόδους είναι χαμηλή. Μια λύση σε αυτού του είδους τα προβλήματα είναι στο σύνολο αποφάσεων όπου υπάρχει μία προς μία αντιστοιχία με το σύνολο των στόχων. Σε αυτές τις περιπτώσεις κάποιος θα μπορούσε να εφαρμόσει εύκολα συστηματικές μεθόδους, να ψάξει για όλες τις πιθανές λύσεις και να βρει το πραγματικό skyline. Ωστόσο, στην περίπτωσή μας, η ανάκτηση ενός δέντρου δρομολόγησης ερωτήματος επιλέγοντας μερικά ή όλα τα ενεργά έξυπνα κινητά τηλέφωνα, αυξάνει την διάσταση του χώρου αναζήτησης για την ανάκτηση ενός μοναδικού δέντρου δρομολόγησης ερωτήματος, και αυτό έχει ως συνακόλουθο μια λύση στην εμβέλεια των στόχων (κατανάλωση ενέργειας, χρόνος απόκρισης, ρυθμός ανάκλησης). Αυτό αυξάνει την πολυπλοκότητα αυτού του είδους των προβλημάτων, συμπεριλαμβανομένου και του γεγονότος ότι στις περισσότερες περιπτώσεις δεν υπάρχει καμία γνώση για το Pareto Front που πρέπει να ανακτηθεί. Επομένως, είναι σχεδόν αδύνατο να χρησιμοποιηθεί μια συστηματική προσέγγιση για να ψάξει όλα τα δέντρα δρομολόγησης ερωτήματος για να βρει λύση στο πρόβλημα. Αυτός είναι και ο σημαντικότερος λόγος που μια στοχαστική προσέγγιση, όπως είναι ο εξελικτικός υπολογισμός, μπορεί να είναι πιο κατάλληλη.

Κάθε μη-κυριαρχούμενη λύση του Pareto Front αναπαριστά και ένα δέντρο δρομολόγησης ερωτήματος το οποίο είναι βασισμένο στις παραμέτρους που αφορούν την κατανάλωση της ενέργειας από τις κινητές συσκευές, το χρόνο άφιξης των αποτελεσμάτων στο Χ και το ρυθμό ανάκλησης. Αφού δημιουργηθεί το Pareto Front από το SmartOpt εξυπηρετητή, ο Χ έχει την δυνατότητα να διαλέξει πιο δέντρο από το PF ανταποκρίνεται πιο κοντά στις δικές του απαιτήσεις. Έτσι καθορίζει τιμές στις παραμέτρους energy (κατανάλωση ενέργειας), time (το χρόνο άφιξης των αποτελεσμάτων στο Χ) και recall (ρυθμός ανάκλησης) και τις στέλνει στο SmartOpt εξυπηρετητή. Ο SmartOpt εξυπηρετητής ελέγχει πιο δέντρο από το PF ανταποκρίνεται περισσότερο στις επιλογές του χρήστη και του το στέλνει.
Τέλος, ο Χ έχει το δέντρο αναζήτησης και ξεκινά τη διαδικασία αναζήτησης μέσα στο δίκτυο.

1.3 Αρχιτεκτονική Android

Η αρχιτεκτονική Android [1] είναι μια στοίβα λογισμικού για κινητές συσκευές και αποτελείται από τρία βασικά στρώματα όπως φαίνεται στην Εικόνα 1.3.1 το λειτουργικό σύστημα, ένα ενδιάμεσο λογισμικό και το στρώμα των βασικών εφαρμογών. Το λειτουργικό Android για κινητές συσκευές είναι βασισμένο στο λειτουργικό Linux το οποίο αναπτύχθηκε από την Google [14] . Πιο κάτω παρουσιάζονται πιο αναλυτικά τα χαρακτηριστικά του κάθε στρώματος σύμφωνα με την Εικόνα 1.3.1:

- **Applications**: Όλες οι εφαρμογές γράφονται στη γλώσσα προγραμματισμού Java όπως ο email client, το πρόγραμμα SMS, το calendar, το maps, ο browser, οι contacts κ.ο.κ. Επίσης όλες οι εφαρμογές είναι ίσες, αφού οι εφαρμογές που βρίσκονται εγκατεστημένες ήδη στην Android συσκευή δεν διαφέρουν στην δομή υλοποίησης από καμιά άλλη εφαρμογή η οποία μπορεί να γραφτεί από οποιονδήποτε.

- **Application Framework**: Το πλαίσιο εφαρμογών είναι ένα API πλαίσιο που χρησιμοποιείται από όλες τις εφαρμογές του πυρήνα και είναι διαθέσιμο σε όλους τους προγραμματιστές για τη δημιουργία προγραμμάτων.

- **Libraries**: Ένα σετ από C/C++ βιβλιοθήκες που χρησιμοποιούνται από τα διάφορα συστατικά του λειτουργικού και εκτίθενται στους προγραμματιστές μέσα από το Applications Framework.

- **Android Runtime (Dalvik Virtual Machine)**: Το Android περιλαμβάνει ένα σετ από core βιβλιοθήκες που περιέχουν τις περισσότερες λειτουργίες των core βιβλιοθηκών της java. Κάθε Android εφαρμογή τρέχει τη δική του διεργασία, με το δικό της στιγμιότυπο του Dalvik virtual machine. Το Dalvik virtual machine έχει γραφτεί έτσι ώστε κάθε συσκευή να μπορεί να τρέξει πολλαπλά virtual machines αποδοτικά.

- **Linux Kernel**: Το Android λειτουργικό είναι βασισμένο στο λειτουργικό Linux 2.6 και χρησιμοποιείται για υπηρεσίες συστήματος πυρήνα όπως ασφάλεια, διαχείριση μνήμης, διαχείριση διαδικασιών, στοίβα δικτύου και μοντέλα οδηγών. Επιπλέων είναι κάτι σαν ένα αφηρημένο στρώμα ανάμεσα στο υλικό και το λογισμικό.

Εικόνα 1.3.1: Αρχιτεκτονική Android

1.4 Περίγραμμα Εργασίας

Στο πρώτο κεφάλαιο παρουσιάζουμε το βασικό υπόβαθρο της εν λόγω διπλωματικής εργασίας καθώς επίσης και την αρχιτεκτονική Android. Στο δεύτερο κεφάλαιο θα παρουσιάσουμε τα χαρακτηριστικά του συστήματος SmartP2P καθώς επίσης και όλους τους τεχνικούς ορισμούς που θα χρησιμοποιήσουμε στην συνέχεια. Στο τρίτο κεφάλαιο θα αναφερθούμε στο πλαίσιο SmartP2P και στις περιγραφές των τριών φάσεων που το αποτελούν. Στο τέταρτο κεφάλαιο θα παρουσιάσουμε την υλοποίηση του πλαισίου SmartP2P. Θα αναφερθούμε στην αρχιτεκτονική του πλαισίου καθώς και στην περιγραφή της βάσης δεδομένων, του εξυπηρετητή και τέλος στην Android εφαρμογή. Στο πέμπτο κεφάλαιο θα γίνει παρουσίαση της πειραματικής διαδικασίας που ακολουθήσαμε και αναφορά στην πειραματική υποδομή που χρειάστηκε. Στο έκτο κεφάλαιο θα γίνει παρουσίαση και ανάλυση των αποτελεσμάτων που πήραμε. Στο έβδομο και τελευταίο κεφάλαιο θα παραθέσουμε τα τελικά συμπεράσματα και τις επιπλέων βελτιστοποιήσεις που μπορούν να γίνουν. Στο παράρτημα Α θα παρουσιάσουμε την υλοποίηση ενός Monkey Runner για την εκτέλεση της εφαρμογής. Στο παράρτημα Β θα δοθεί ο κώδικας που αναπτύχθηκε.

Κεφάλαιο 2

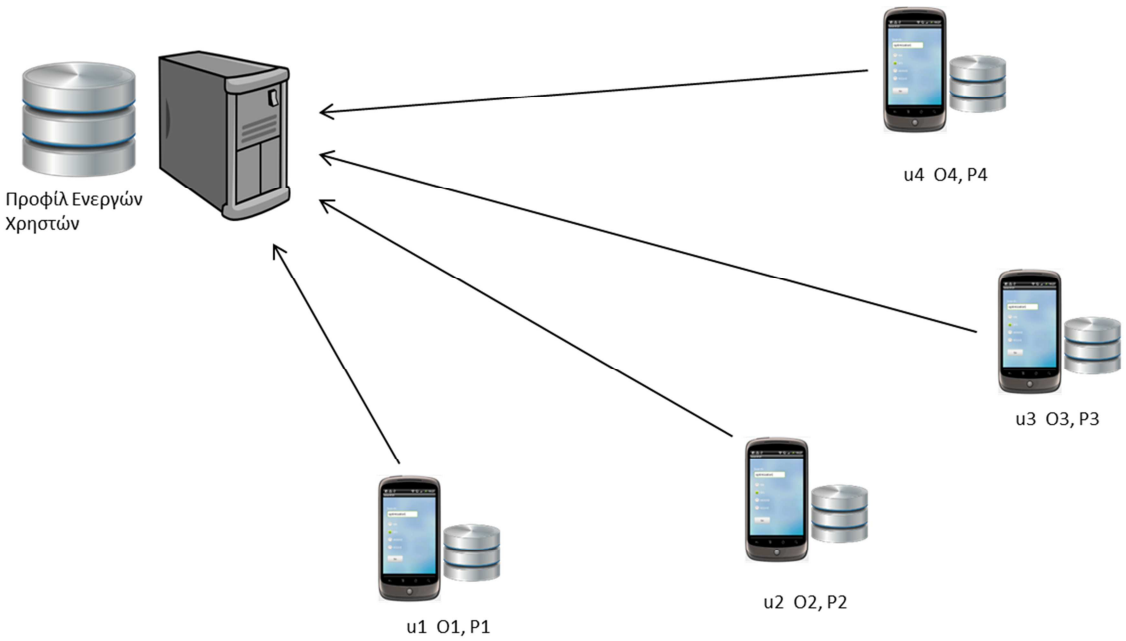
Μοντέλο Συστήματος SmartOpt

2.1 Μοντέλο Πλαισίου SmartOpt
2.2 Λεπτομέρειες Διασύνδεσης (Συνδεσιμότητα)
2.3 Τεχνικές Αναζήτησης
2.4 Διατύπωση Προβλήματος

2.1 Μοντέλο Πλαισίου SmartP2P

Ας υποθέσουμε ότι το C υποδουλώνει μια υπηρεσία κοινωνικής δικτύωσης η οποία διατηρεί κεντρική οντότητα προφίλ P (p_1,p_2, ... , p_M) όπου M είναι οι εγγεγραμμένοι χρήστες U (u_1, u_2….u_M).

Τα προφίλ κρατούν βασικές πληροφορίες του χρήστη όπως διαπιστευτήρια ταυτότητας, τα ενδιαφέροντα του χρήστη και σχέσεις φιλίας που ορίζουν το εννοιολογικό κοινωνικό δίκτυο. Για τις δικές μας ρυθμίσεις ένας χρήστης u_i ($i \leq M$) χρησιμοποιεί μία έξυπνη κινητή συσκευή ή μία ταμπλέτα για να εκτελέσει τις καθημερινές του δραστηριότητες καθώς επίσης και να καταγράψει τα ενδιαφέροντα του σε αυθαίρετες χρονικές στιγμές όπως για παράδειγμα “να βγάλει μια φωτογραφία με το Άγαλμα της Ελευθερίας”. Κάθε αντικείμενο ενδιαφέροντος o_{ik} μπορούμε να του δώσουμε μια ονομασία όπως π.χ. με τις GPS συντεταγμένες του (latitude: 74.03259687, longitude 40.669201355 και tag (liberty statue)). Η εικόνα 2.1.1 παρουσιάζει το μοντέλο συστήματος SmartOpt



Oi: το σύνολο των αντικειμένων που έχει κάθε χρήστης
Pi: οι περιγραφές που χαρακτηρίζουν κάθε αντικείμενο Oi του χρήστη ui

Εικόνα 2.1.1: Μοντέλο Συστήματος SmartOpt

2.2 Λεπτομέρειες Διασύνδεσης

Κάθε $u_ι$ χαρακτηρίζεται από διαφορετικούς τρόπους σύνδεσης στο διαδίκτυο οι οποίοι παρέχουν διακοπτόμενη σύνδεση στο C όπως (π.χ. WIFI/2G/3G/4G). Κάθε $u_ι$ επίσης χαρακτηρίζεται από peer-to-peer σύνδεση η οποία παρέχει σύνδεση στους κόμβους σε χωρική εγγύτητα όπως για παράδειγμα το Bluetooth, το Wi-Fi σε Android. Στην περίπτωση της παρούσας διπλωματικής υποθέτουμε ότι για κάθε $u_ι$ ο οποίος είναι ενωμένος με τον C, ο C γνωρίζει την απόλυτη τοποθεσία του $u_ι$ (π.χ. GPS) ή τη σχετική τοποθεσία όπως είναι οι RSSI δείκτες εντός του πεδίου του $u_ι$ ή κάποιο άλλο μέσο που να χρησιμοποιεί τη γεωγραφική τοποθεσία. Κάθε είδος συνδεσιμότητας έχει διαφορετικά χαρακτηριστικά ως προς την ενέργεια που καταναλώνει και το ρυθμό μετάδοσης δεδομένων.

2.3 Τεχνικές Αναζήτησης

Έστω ένας αυθαίρετος χρήστης u_j ($j \leq M$) ο οποίος ενδιαφέρεται για απάντηση στην ερώτηση Q στους δικούς του γείτονες μέσα στο κοινωνικό του δίκτυο G΄ (G΄ ⊆ G). Έστω ότι το Q είναι ερώτηση οριοθετημένη σε βάθος για και τη κατά πλάτος αναζήτηση στους γείτονες τους u_j στο γράφο G. Αυτό το είδος ερώτησης μπορούμε να το αντιληφθούμε με τους εξής τρόπους: Κεντρική Αναζήτηση και Τυχαία Κατανεμημένη Αναζήτηση.

Ο αλγόριθμος της Κεντρικής Αναζήτησης υποθέτει ότι τα αντικείμενα καθώς και οι ετικέτες που αντιστοιχούν στον κάθε χρήστη ανεβάζονται όλα στο C. Όταν υποβληθεί μια ερώτηση Q, τότε ο C είναι σε θέση από μόνος του να δώσει απάντηση στο Q χρησιμοποιώντας τη βάση δεδομένων του που είναι βασισμένη στις ετικέτες.

Από την άλλη πλευρά, ο αλγόριθμος της Τυχαίας Κατανεμημένης αναζήτησης υποθέτει ότι τα αντικείμενα και οι ετικέτες είναι αποθηκευμένα τοπικά στον κάθε χρήστη (έξυπνες κινητές συσκευές). Έτσι η διαδικασία αναζήτησης έχει ως εξής: Ο χρήστης u_j που υποβάλλει την ερώτηση κατεβάζει από το C τις διευθύνσεις των γειτονικών χρηστών. Στη συνέχεια ο u_j επικοινωνεί με αυτούς τους χρήστες με σκοπό να πραγματοποιήσει μια αναζήτηση σε αυτούς όπως είναι η Peer-to-Peer αναζήτηση. Όταν κάποιος χρήστης u_x παραλάβει την ερώτηση Q κοιτάζει τις δικές του ετικέτες στην δική του βάση δεδομένων για εντοπισμό απάντησης και προωθεί την ερώτηση πιο πέρα στο δίκτυο. Με αυτό τον τρόπο η τυχαία κατανεμημένη αναζήτηση βελτιώνει το μειονέκτημα της γνωστοποίησης δεδομένων σε σχέση με την κεντρική αναζήτηση.

Συγκεκριμένα το ερώτημα Q μπορεί να περάσει από τυχαίους γείτονες παρά από γείτονες οι οποίοι συνδέονται αποκλειστικά με την ερώτηση. Για παράδειγμα στην ερώτηση για το "Άγαλμα της Ελευθερίας" θα προτιμηθεί να ρωτηθεί ένα άτομο το οποίο ζει στο Μανχάταν σε σχέση με ένα άτομο το οποίο ζει στην Καλιφόρνια αφού ο χρήστης που είναι στο Μανχάταν θα έχει περισσότερες πιθανότητες να έχει απάντηση. Επιπλέον αν ο χρήστης u_j έχει δύο φίλους τον u_x να είναι μέσα στα όρια του χώρου που ορίσαμε με τον u_j την ώρα της ερώτησης (εντός κάποιων μέτρων), ενώ ο u_j είναι πολύ πιο μακριά θα προτιμηθεί ο u_x για απάντηση στην ερώτηση.

2.4 Διατύπωση Προβλήματος

Η δομή του δέντρου δρομολόγησης ερωτήματος πολλαπλών στόχων βελτιώνει την διαδικασία αναζήτησης για τον αλγόριθμο τυχαίας κατανεμημένης αναζήτησης βελτιστοποιώντας την διαδικασία επιλογής γειτόνων. Συγκεκριμένα, κάποιος χρήστης κατεβάζει από το C το δέντρο δρομολόγησης ερωτήματος X το οποίο βελτιστοποιείται με βάση την ακόλουθη διατύπωση:

Δεδομένου ενός κοινωνικού δικτύου που αποτελείται από U χρήστες, μια λίστα από ενεργούς χρήστες U' και τις γεωγραφικές συντεταγμένες τους, τα προφίλ των ενεργών P και μια ερώτηση Q η οποία υποβάλλεται από το χρήστη u_j, ο C προσπαθεί να βελτιστοποιήσει τη δομή του χρησιμοποιώντας του ακόλουθους στόχους:

Στόχος 1: Ελαχιστοποίηση της συνολικής κατανάλωσης Ενέργειας του X

$$\text{Ενέργεια (X)} = \min \sum_{\forall(ua,ub)\in X\ (X \subseteq U')} \varepsilon\ (ua, ub)$$

Όπου $\varepsilon(u_a , u_b)$ υποδηλώνει την κατανάλωση ενέργειας για την μετάδοση ενός bit δεδομένου στην αντίστοιχη άκρη.

Στόχος 2: Ελαχιστοποίηση του χρόνου του X

$$\text{Time (X)} = \min (\max (u_a , u_b) \in X\ t\ (u_a , u_b))$$

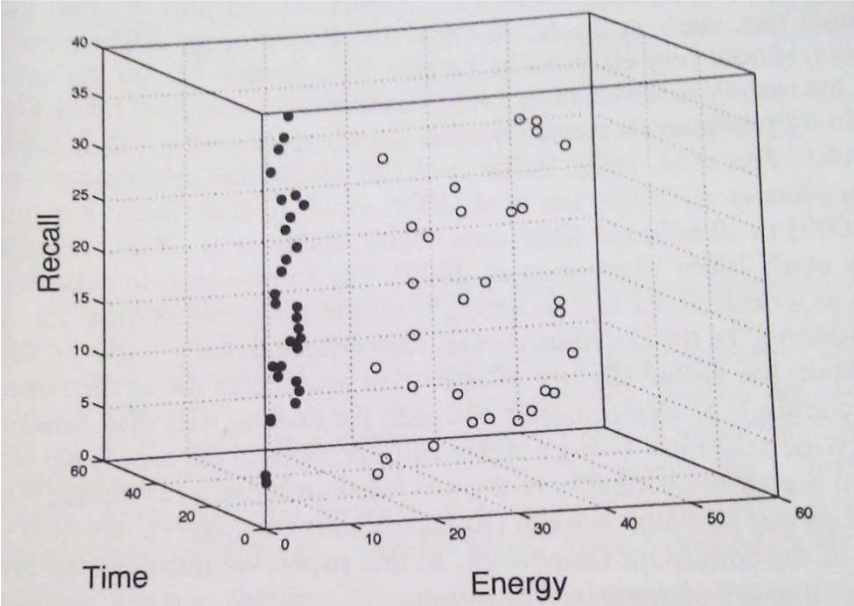
Όπου(u_a , u_b) υποδουλώνει την καθυστέρηση μετάδοσης ενός δυφίου δεδομένου στην αντίστοιχη άκρη.

Στόχος 3: Μεγιστοποίηση του ρυθμού ανάκλησης του X

$$\text{Recall (X,Q)} = \max \frac{Relevant\ (Q) \cap \text{Retrieved (X,Q)}}{Relevant\ (Q)}$$

Όπου *Relevant* υποδουλώνει το σύνολο όλων των αντικειμένων του U’ τα οποία είναι σχετικά με το Q, *Retrieved* (X,Q) υποδουλώνει το σύνολο των αντικειμένων που έχουν ανακτηθεί για απάντηση του Q στη δομή το X.

Σε ένα πρόβλημα πολυκριτηρίων δεν υπάρχει μια μοναδική λύση X η οποία βελτιώνει όλους τους στόχους ταυτόχρονα, αλλά ένα σύνολο από υποψήφιες λύσεις οι οποίες βελτιώνουν ένα μέρος των στόχων ταυτόχρονα. Το σύνολο των αυτών των λύσεων ονομάζεται Pareto Front (PF) (εικόνα 4.2.1).



Εικόνα 4.2.1: Αναπαράσταση του Pareto Front

Κεφάλαιο 3

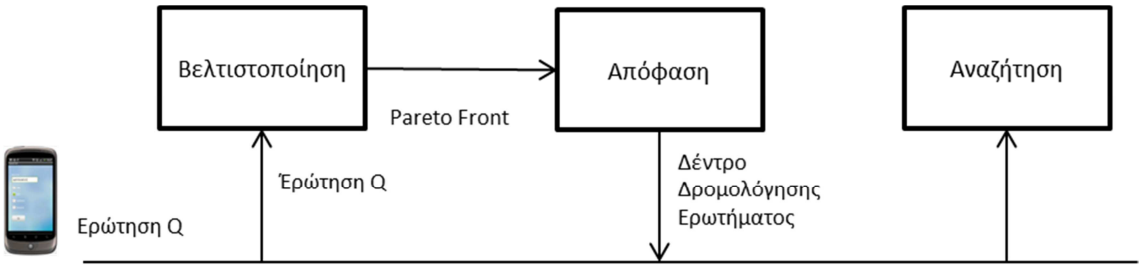
Πλαίσιο SmartOpt

3.1 Εισαγωγή
3.2 Φάση Βελτιστοποίησης
3.3 Φάση Απόφασης
3.4 Φάση Αναζήτησης

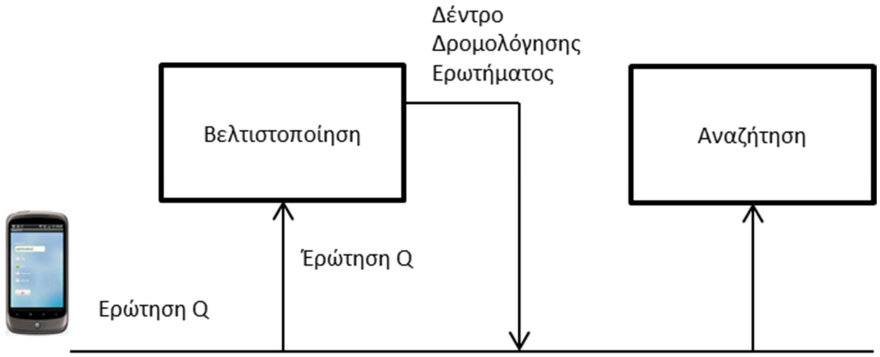
3.1 Εισαγωγή

Σε αυτό το κεφάλαιο παρουσιάζεται το πλαίσιο SmartpOpt το οποίο αποτελείται από τρεις φάσεις: τη φάση της βελτιστοποίησης για την οποία έχουν υλοποιηθεί δύο απλές τεχνικές (εικόνα 3.1.1) και δύο τεχνικές πολυκριτηρίων (εικόνα 3.1.2). Στην περίπτωση των απλών τεχνικών η φάση αυτή έχει σαν έξοδο ένα δέντρο δρομολόγησης ερωτήματος. Αμέσως επόμενη φάση για την περίπτωση επιλογής κάποιας απλής τεχνικής είναι η φάση αναζήτησης. Στην αντίθετη περίπτωση των τεχνικών πολυκριτηρίων δημιουργείται στην έξοδο ένα σετ από μη-κυριαρχούμενες λύσεις όπου κάθε λύση αντιπροσωπεύει και ένα δέντρο δρομολόγησης ερωτήματος. Την φάση αυτή διαδέχεται η φάση απόφασης όπου ο χρήστης καθορίζει τις τιμές του χρόνου απόκρισης του ρυθμού ανάκλησης και της κατανάλωσης της ενέργειας για την επιλογή ενός δέντρου δρομολόγησης ερωτήματος από την προηγούμενη φάση. Αφού έχει καθοριστεί το δέντρο δρομολόγησης ερωτήματος περνούμε στην τελευταία φάση όπου είναι η φάση της αναζήτησης όπου ο χρήστης που υποβάλει το ερώτημα στα παιδιά του και

αυτοί με τη σειρά τους στα δικά τους παιδιά με σκοπό ο κάθε χρήστης να δώσει τις δικές του απαντήσεις στο χρήστη που υπόβαλε το ερώτημα.



Εικόνα 3.1.1: Ροή των φάσεων για Multi-Objective τεχνικές



Εικόνα 3.2.2: Ροή των φάσεων για Single-Objective τεχνικές

3.2 Φάση Βελτιστοποίησης

Η φάση αυτή έχει διαφορετική σημασία και λειτουργία η οποία εξαρτάται από την τεχνική η που θα χρησιμοποιηθεί. Η μία τεχνική αφορά την περιοχή των απλών αλγορίθμων και η άλλη αφορά την περιοχή των αλγορίθμων πολυκριτηρίων. Η φάση αυτή έχει σαν είσοδο την ερώτηση που υποβάλλει ο χρήστης και έχει σαν έξοδο ένα δέντρο δρομολόγησης ερωτήματος στην περίπτωση που αναφερόμαστε στην χρήση μίας απλής τεχνικής ενώ στην περίπτωση που αναφερόμαστε σε χρήση τεχνικών πολυκριτηρίων η φάση αυτή έχει σαν έξοδο το Pareto Front.

Στην περίπτωση χρήσης απλών τεχνικών η φάση βελτιστοποίησης υποστηρίζει δύο απλούς αλγόριθμους το RW και το BFS. Η βασική ιδέα της λειτουργίας του RW είναι

ότι δημιουργεί το δέντρο δρομολόγησης ερωτήματος με μια τυχαία κατανομή ως προς την τοποθέτηση όλων των διαθέσιμων κόμβων μέσα στο δέντρο. Ο BFS αλγόριθμος χρησιμοποιεί και αυτός μια τυχαία κατανομή των κόμβων μέσα στο δέντρο με την διαφορά ότι τοποθετεί όλους τους διαθέσιμους κόμβους μέσα σε αυτό έτσι ώστε το δέντρο που θα προκύψει να είναι πλήρες ή να τείνει να γίνει.

Στην περίπτωση χρήσης τεχνικών πολυκριτηρίων η φάση βελτιστοποίησης υποστηρίζει δύο πιο σύνθετους αλγόριθμους σε σχέση με τους απλούς, τον MOEA/D [45] και τον NSGA-II [9] οι οποίοι έχουν σαν έξοδο τους ένα πιο σύνολο από μη κυριαρχούμενες λύσεις το Pareto Front. Στην συνέχεια ακολουθεί μια πιο λεπτομερής περιγραφή και ανάλυση αυτών των αλγορίθμων.

Ο MOEA/D είναι ένας Multiobjective Evolutionary αλγόριθμος ο οποίος βασίζεται στην αποσύνθεση. Ο όρος αποσύνθεση υποδηλώνει ότι διασπά το πρόβλημα σε ένα αριθμό υποπροβλημάτων το οποίο θα συνεισφέρει στην δημιουργία των μη κυριαρχούμενων λύσεων. Ο αλγόριθμος αυτός έχει τρεις φάσεις εκτέλεσης κατά την λειτουργία του την φάση 1 η οποία περιλαμβάνει την αρχικοποίησης, την φάση 2 η οποία περιλαμβάνει τρις επιμέρους φάσεις την φάση των γενετικών χειριστών την φάση αποκατάστασης και την φάση ενημέρωσης και τέλος η φάση 3 που ελέγχεται αν ο αλγόριθμος πρέπει να τερματίσει. Ο αλγόριθμος με τις φάσεις του MOEA/D παρουσιάζεται πιο κάτω.

Φάση 1: Αρχικοποίηση
Φάση 2:
    Φάση 2.1: Δημιουργία Γενετικών Χειριστών
    Φάση 2.2: Αποκατάσταση
    Φάση 2.3: Ενημέρωση
Φάση 3: Έλεγχος Τερματισμού

Τρόπος Λειτουργίας MOEA/D:

Φάση 1: Αρχικοποίηση

Στην φάση 1 δημιουργείται ένας πληθυσμός (IP_0) μεγέθους M τον οποίο μπορούμε να φανταστούμε σαν ένα μονοδιάστατο πίνακα M θέσεων. Κάθε θέση αυτού του μονοδιάστατου πίνακα αρχικοποιείται με ένα τυχαίο δέντρο δρομολόγησης ερωτήματος. Ταυτόχρονα, στην περίπτωση του προβλήματος μας αρχικοποιούνται σε κάθε κελί του πίνακα και τρείς μεταβλητές energy, time και recall.
Φάση 2: Στην φάση αυτή θεωρούμε τον πληθυσμό IP_0 ως γονέα και θα προσπαθήσουμε να δημιουργήσουμε τον πληθυσμό IP_1 ο οποίος θα έχει καλύτερα χαρακτηριστικά από τον πληθυσμό IP_0.

Φάση 2.1: Δημιουργία Γενετικών Χειριστών

Στην φάση αυτή θα δημιουργήσουμε για κάθε i θέση του πίνακα όπου ι <= M μια νέα λύση (καινούριο δέντρο δρομολόγησης ερωτήματος) η οποία θα στηρίζεται σε κάποιες λύσεις του πατέρα – πληθυσμού. Γίνεται δηλαδή συνδυασμός κάποιων δέντρων δρομολόγησης ερωτήματος από τον πατέρα – πληθυσμό.

Φάση 2.2: Αποκατάστασης

Στη συνέχεια αφού έχει παραχθεί το καινούριο δέντρο δρομολόγησης ερωτήματος υπάρχει περίπτωση το δέντρο να έχει τους κινδύνους κάποιοι κόμβοι να μην είναι συνδεδεμένοι με τον πατέρα τους, κάποιοι κόμβοι να έχουν το ίδιο id και κάποιοι κόμβοι να είναι συνδεδεμένοι έτσι ώστε να παράγουν ατέρμονους βρόχους. Η φάση αυτή είναι υπεύθυνη να ανακαλύπτει αυτά τα προβλήματα και να τα διορθώνει.

Φάση 2.3: Ενημέρωσης

Στην φάση αυτή θα παρθεί η τελική απόφαση αν τελικά το δέντρο που προέκυψε θα τοποθετηθεί στον νέο πληθυσμό. Το κριτήριο είναι να έχει καλύτερες επιδόσεις από τον

πατέρα του στην αντίστοιχη θέση. Αν το ικανοποιεί τότε τοποθετείται, αλλιώς την θέση του παιδιού – δέντρου παίρνει ο πατέρας. Επιπλέων, το νέο δέντρο που προέκυψε θα ελέγχεται με τους κ-γείτονες (κ καθορίζεται από την αρχή πριν ξεκινήσει ο αλγόριθμος) του πατέρα και αν έχει καλύτερες επιδώσεις από τους κ-πατέρες των γειτόνων τοποθετείται στην θέση των κ-παιδιών τους. Τέλος, για την φάση αυτή υπάρχει ένας πίνακας εξωτερικός όπου τα δέντρα με τις καλύτερες επιδόσεις καταχωρούνται εκεί. Αυτός ο πίνακας στο τέλος όταν θα τερματίσει ο αλγόριθμος είναι το Pareto Front.

Η διαδικασία της φάσης 2 επαναλαμβάνεται μέχρι ο πληθυσμός IP_i να γίνει ίσος με το IP_{max} όπου max είναι η τιμή που καθορίστηκε από την αρχή για το πόσους πληθυσμούς θα τρέχει ο αλγόριθμος. Εδώ είναι που η φάση 3 θα εκπληρωθεί αφού ο αριθμός των πληθυσμών θα γίνει ίσος με το max.

Ο NSGA-II λειτουργεί με παρόμοιο τρόπο με τον MOEA/D. Η διαφορά του NSGA-II είναι στην φάση ενημέρωσης. Κάθε νέα λύση-δέντρο που προκύπτει την τοποθετεί σε ένα νέο ενδιάμεσο πληθυσμό IP_0'. Στη συνέχεια, συγχωνεύει τους δύο πληθυσμούς φτιάχνοντας έναν μεγέθους 2M. Ακολούθως χωρίζει τα δέντρα που βρίσκονται τοποθετημένα στο νέο πληθυσμό σε Fronts. Η δημιουργία των Fronts έχει ως εξής: περνά συνεχώς πάνω από τον πληθυσμό και σε κάθε πέρασμα εντοπίζονται οι top μη κυριαρχούμενες λύσεις – δέντρα. Σε κάθε πέρασμα το σύνολο των δέντρων που θα ξεχωρίσει αποτελεί και ένα Front. Στο τέλος, όταν όλα τα δέντρα έχουν τοποθετηθεί σε ένα Front, τα Fronts είναι ταξινομημένα ανάλογα με την σειρά που προέκυψαν. Συνεπώς, το Front F1 έχει καλύτερες επιδόσεις από το Front F2. Ακολούθως, κάθε Front οφείλει να ταξινομήσει τα δέντρα που περιέχονται σε αυτό. Έτσι ακολουθείται η μέθοδος του Crowding Distance όπου τα δέντρα που έχουν τη μεγαλύτερη απόσταση από τα υπόλοιπα μέσα στο Front έχουν και την καλύτερη επίδοση. Με την μέθοδο αυτή ταξινομούνται τα δέντρα μέσα στο Front ανάλογα με την ενεργειακή τους απόδοση. Τέλος, μετά την εσωτερική ταξινόμηση των Fronts ο νέος πληθυσμός που προκύπτει είναι ο μισός αυτού που δημιουργήσαμε και ανακατατάξαμε τα Fronts.

Αφού ικανοποιηθεί η φάση 3 για τον αλγόριθμο NSGA-II τότε το Pareto Front που δημιουργείται είναι το Front 1 που έχει ο τελευταίος πληθυσμός που δημιουργήθηκε.

3.3 Φάση Απόφασης

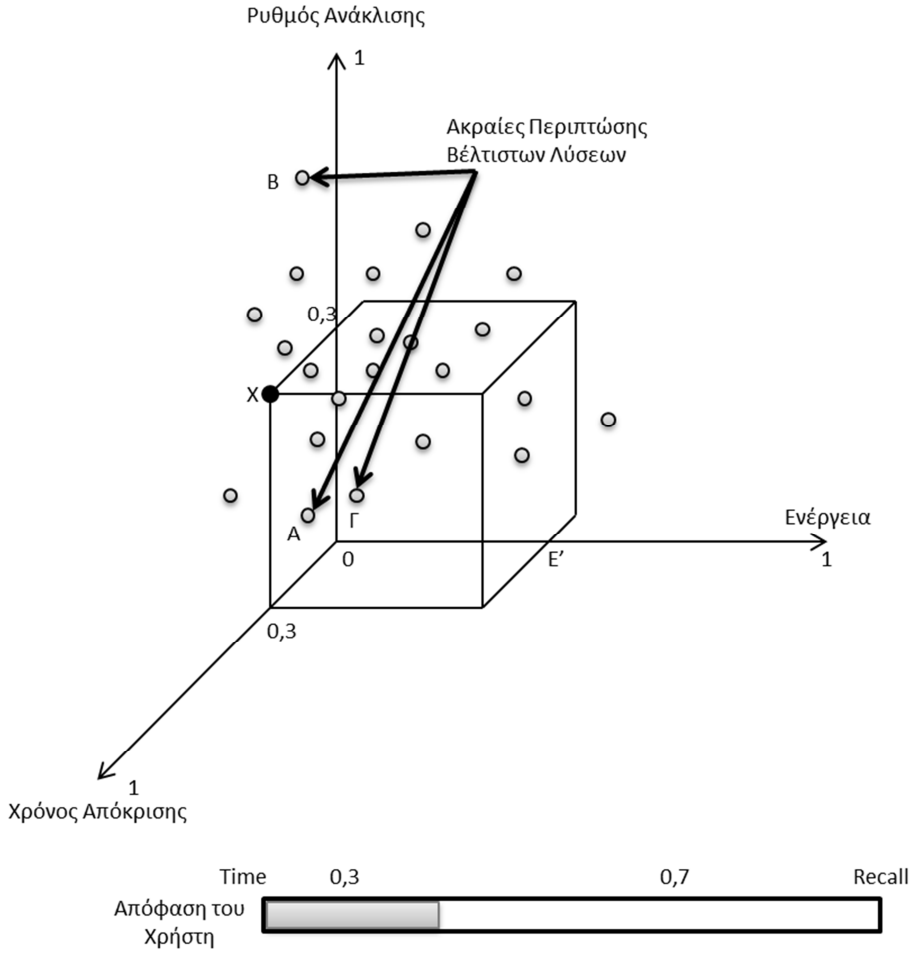
Η συγκεκριμένη φάση αφορά την επιλογή τεχνικών πολυκριτηρίων από την προηγούμενη φάση (φάση βελτιστοποίησης).

Στη φάση αυτή αφού έχει δημιουργηθεί το Pareto Front ο χρήστης u_j καλείται να αποφασίσει τις προτιμήσεις του όσον αφορά το χρόνο (στόχος 2 τέθηκε στο προηγούμενο κεφάλαιο) και το ποσοστό της πληροφορίας (στόχος 3 που τέθηκε στο προηγούμενο κεφάλαιο) για την απάντηση που θα λάβει στο ερώτημα που έθεσε στο δίκτυο έξυπνων κινητών συσκευών. Έτσι γίνεται αναζήτηση στα δέντρα του Pareto Front για να επιλεχθεί εκείνο το δέντρο το οποίο ικανοποιεί τις απαιτήσεις του χρήστη που υπόβαλε. Επιπλέον το δέντρο αυτό θα είναι και το πιο αποδοτικό ως προς την κατανάλωση της ενέργειας (στόχος 1 που τέθηκε στο προηγούμενο κεφάλαιο). Σε αυτό το σημείο δηλαδή ο χρήστης u_j είναι υπεύθυνος να αποφασίσει τις τιμές που είναι προσανατολισμένες στους στόχους του χρήστη οι οποίες είναι ο χρόνος απόκρισης και ο ρυθμός ανάκλασης. Η λειτουργία της απόφασης από την άλλη πλευρά είναι εκείνη που θα αποφασίσει για την τιμή που είναι προσανατολισμένη στο στόχο του συστήματος η οποία είναι η κατανάλωση της ενέργειας.

Για παράδειγμα, το Pareto Front στην εικόνα 3.3.1 έχει δημιουργηθεί από τη φάση βελτιστοποίησης. Η επιλογή του χρήστη παρουσιάζεται στη ράβδο ολίσθησης στην εικόνα όπου η τιμή για το χρόνο είναι 0,3 και για το ρυθμό ανάκλησης 0,3. Έτσι, η λειτουργία της απόφασης υπολογίζει ένα δέντρο X το οποίο είναι κοντινό στην απόφαση του χρήστη μέσω της ευκλείδειας απόστασης και παρέχει ταυτόχρονα τη βέλτιστη κατανάλωση σε θέματα ενέργειας (E΄). Σε περιπτώσεις όπου υπάρχουν περισσότερα από ένα δέντρο που ικανοποιούν τα κριτήρια του χρήστη τότε επιλέγεται το δέντρο που έχει την καλύτερη ενεργειακή απόδοση.

Η εικόνα 3.3.1 επιπλέον δείχνει τις λύσεις – δέντρα A, B, και C οι οποίες είναι ακραίες περιπτώσεις λύσεων. Η λύση A αναπαριστά την καλύτερη λύση ως προς το χρόνο απόκρισης, δηλαδή τα δεδομένα να φτάσουν στο χρήστη που έκανε την ερώτηση όσο το δυνατό πιο γρήγορα χρόνος=0 αγνοώντας εντελώς το ρυθμό ανάκλησης (ποσοστό

δεδομένων) το οποίο έχει την τιμή μηδέν επίσης. Η λύση Β αναπαριστά την καλύτερη λύση ως προς το ρυθμό ανάκλησης (ποσοστό δεδομένων) αγνοώντας εντελώς το χρόνο απόκρισης χρόνος = μέγιστος, ρυθμό ανάκλησης = μέγιστος. Τέλος η λύση C αναπαριστά την καλύτερη λύση ως προς την κατανάλωση ενέργειας την οποία διαλέγει η λειτουργία της απόφασης στην περίπτωση που ο χρήστης δεν έχει κάποια απόφαση για την τιμή των υπόλοιπων 2 στόχων.



Εικόνα 3.3.1: Επιλογή Δέντρου Δρομολόγησης από το Pareto Front

3.4 Φάση Αναζήτησης

Αυτή είναι η τελική φάση όπου ο χρήστης u_j παραλαμβάνει το δέντρο δρομολόγησης ερωτήματος από τη φάση βελτιστοποίησης για την περίπτωση επιλογής ενός Single-

Objective αλγόριθμου. Στην περίπτωση επιλογής ενός Multi-Objective αλγόριθμου το δέντρο δρομολόγησης ερωτήματος παραλαμβάνεται από τη φάση απόφασης.

Το δέντρο X είναι ένα διάνυσμα του οποίου κάθε δείκτης ι έχει την ακόλουθη μορφή (εικόνα 3.4.1):

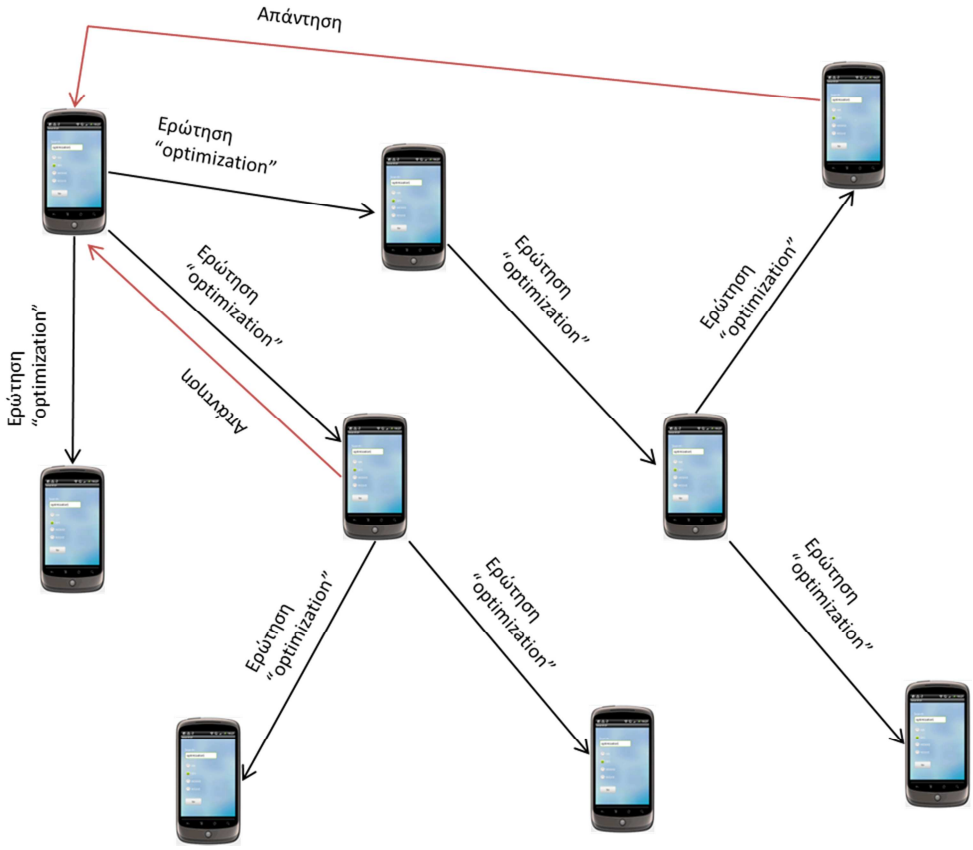
V_i	$V_{i,1}$	$V_{i,2}$	$V_{i,3}$	$V_{i,4}$
V_1	2	-1	10.16.5.100	45000
V_2	1	2	10.16.5.101	45000
V_3	3	2	10.16.5.102	45000
V_4	4	3	10.16.5.103	45000
V_5	5	1	10.16.5.104	45000

Εικόνα 3.4.1: Αναπαράσταση δέντρου δρομολόγησης ερωτήματος

όπου $V_{i,1}$ είναι το id κάποιου χρήστη του δικτύου, $V_{i,2}$ το id του πατέρα του χρήστη $V_{i,1}$, $V_{i,3}$ η ip διεύθυνση του χρήστη $V_{i,1}$ και $V_{i,4}$ η θύρα στην οποία ακούει ο χρήστης $V_{i,1}$.
Το πιο πάνω παράδειγμα δηλώνει ότι ο χρήστης με id=2 είναι ο χρήστης που υποβάλλει το ερώτημα και έχει παιδιά του τους χρήστες 1 και 3. Ο χρήστης 1 με τη σειρά του έχει παιδί του το χρήστη 5 και ο χρήστης 3 έχει παιδί του το χρήστη 4.

Όταν η έξυπνη κινητή συσκευή η οποία ανήκει στο χρήστη που υποβάλει το ερώτημα παραλάβει το δέντρο δρομολόγησης ερωτήματος προωθεί το ερώτημα του στα παιδιά του σύμφωνα με δέντρο και στη συνέχεια το δέντρο.

Κάθε έξυπνη κινητή συσκευή η οποία παραλαμβάνει ένα ερώτημα δημιουργεί ένα σύνολο από αντικείμενα τα οποία απαντούν στο ερώτημα και τα προωθεί στο χρήστη ο οποίος έχει υποβάλει το ερώτημα. Ακολούθως προωθεί το ερώτημα στα παιδιά του σύμφωνα με το δέντρο δρομολόγησης ερωτήματος. Η ροή της φάσης αυτής παρουσιάζεται στην εικόνα 3.4.2.



Εικόνα 3.4.2: Διαδικασία Φάσης Αναζήτησης

Κεφάλαιο 4

Υλοποίηση Εφαρμογής SmartP2P

4.1 Αρχιτεκτονική Συστήματος

4.1.1 Βάση Δεδομένων

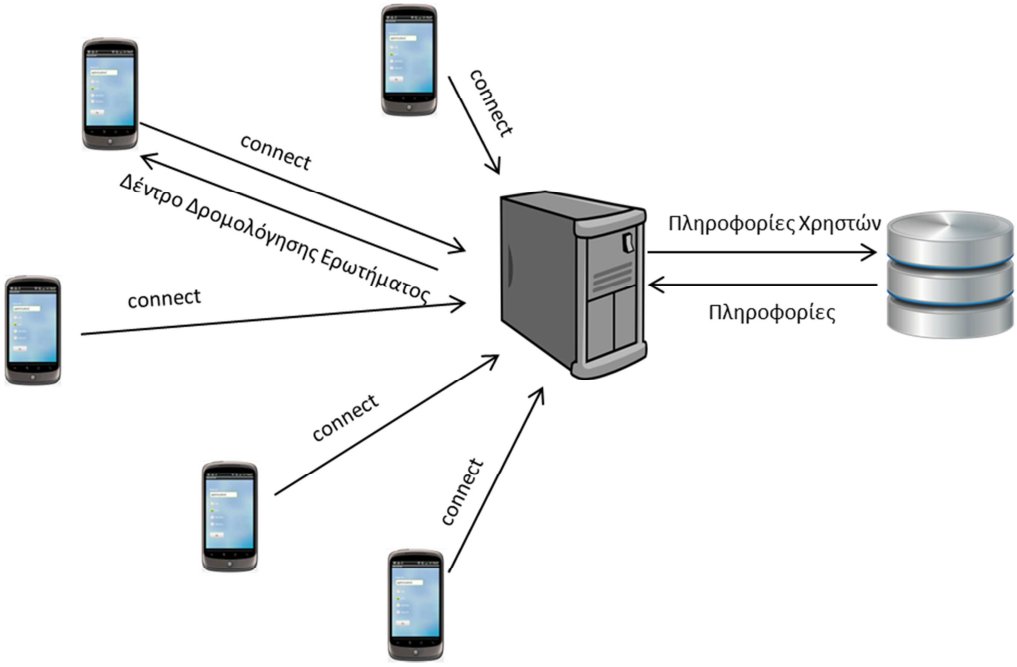
4.1.2 Εξυπηρετητής SmartOpt

4.2 GUI Εφαρμογής SmartP2P

4.1 Αρχιτεκτονική Συστήματος

Στο κεφάλαιο αυτό θα παρουσιάσουμε την υλοποίηση ενός συστήματος για αναζήτηση δεδομένων σε έξυπνες κινητές συσκευές. Οι έξυπνες κινητές συσκευές βρίσκονται ενωμένες σε ένα δίκτυο ομότιμων (peer-to-peer). Η όλη υλοποίηση του συστήματος είναι αποτίμηση του πλαισίου SmartOpt το οποίο προϋποθέτει βάση κάποιων τεχνικών αποδοτική αναζήτηση στις υπόλοιπες συσκευές καθώς επίσης και ασφάλεια των δεδομένων των χρηστών του δικτύου.

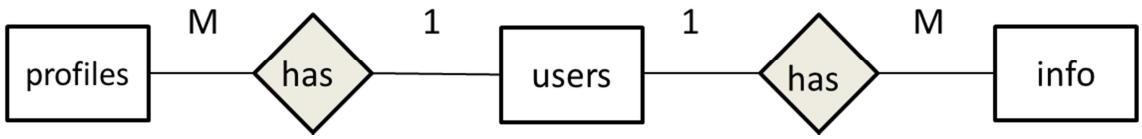
Το σύστημα αποτελείται από τρεις οντότητες τη Βάση Δεδομένων, τον εξυπηρετητή SmartOpt και τον Πελάτη (εφαρμογή SmartP2P) οι οποίες αλληλεπιδρούν μεταξύ τους έτσι ώστε να λειτουργεί το σύστημα αποδοτικά και να παρουσιάζει στους χρήστες του αποτελέσματα με τη καλύτερη δυνατή επίδοση και απόδοση. Η εικόνα 4.1.1 παρουσιάζει την αρχιτεκτονική του συστήματος.



Εικόνα 4.1.1: Αρχιτεκτονική Συστήματος

4.1.1 Βάση Δεδομένων

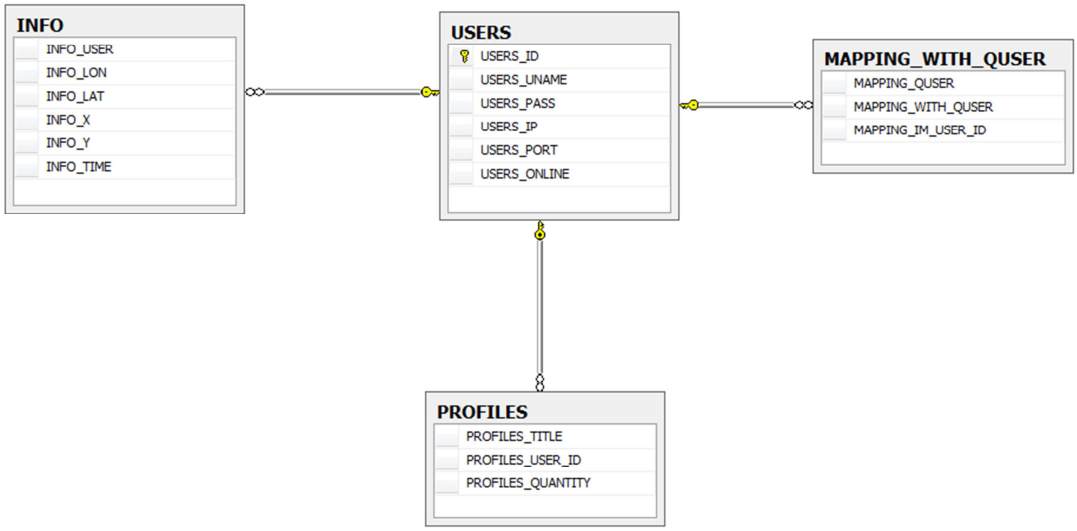
Η Βάση Δεδομένων είναι υλοποιημένη σε MS SQL 2008 R2 [36] . Είναι σχεδιασμένη έτσι ώστε να μπορεί να κρατά πληροφορίες για τους πελάτες του συστήματος όπως είναι το όνομα και το μοναδικό τους id, τις GPS συντεταγμένες που βρίσκονται σε μια χρονική στιγμή και το προφίλ του κάθε χρήστη.



Εικόνα 4.1.1.1: ER Διάγραμμα Βάσης Δεδομένων

Στην εικόνα 4.1.1.1 παρουσιάζεται το ER διάγραμμα της βάσης δεδομένων που δημιουργήθηκε. Όπως βλέπουμε υπάρχει η οντότητα users που περιγράφει τους χρήστες, η οντότητα info που κρατά πληροφορίες για τις γεωγραφικές συντεταγμένες των χρηστών του δικτύου και η οντότητα profiles που κρατά πληροφορίες για τις περιγραφές των δεδομένων των χρηστών του δικτύου. Όπως σχεδιάστηκε η βάση δεδομένων κάποιος χρήστης μπορεί να έχει πολλές γεωγραφικές συντεταγμένες και σε κάθε στιγμή που θα χρειαστεί να ανακτηθούν οι συντεταγμένες του χρήστη,

ανακτώνται εκείνες που καταχωρήθηκαν τελευταίες αφού για κάθε καταχώρηση συντεταγμένων αποθηκεύεται και η χρονική στιγμή που καταχωρήθηκαν. Αυτός ο τρόπος σχεδιασμού προτιμήθηκε για την περίπτωση που σε μελλοντική επέκταση της εφαρμογής, η εφαρμογή θα στέλνει σε σύντομα χρονικά διαστήματα μηνύματα ότι είναι ενωμένη στο δίκτυο. Έτσι ανά πάσα στιγμή η βάση δεδομένων θα γνωρίζει την τοποθεσία των χρηστών της εφαρμογής. Επίσης, κάποιος χρήστης θα μπορεί να έχει πολλές καταχωρήσεις από δεδομένα. Το σχεσιακό σχήμα της βάσης δεδομένων παρουσιάζεται στην εικόνα 4.1.1.2.



Εικόνα 4.1.1.2: Σχεσιακό Σχήμα Βάσης Δεδομένων

Για να μπορέσουν τα δεδομένα να οργανωθούν πιο σωστά χρησιμοποιήθηκε και ένας επιπλέων πίνακας. Επίσης, για το σωστό συντονισμό της Βάσης Δεδομένων με τον εξυπηρετητή χρειάστηκε η δημιουργία Stored Procedures και Functions. Αναλυτικότερα παρουσιάζονται πιο κάτω η λειτουργίες των πινάκων καθώς και των Stored Procedures και Functions:

Πίνακες:

- USERS: Σε αυτό το πίνακα αποθηκεύονται τα στοιχεία του κάθε χρήστη που αφορούν το μοναδικό id του, το μοναδικό του όνομα, το κωδικό του το οποίο χρειάζεται για σκοπούς εισαγωγής κάθε πελάτη στο σύστημα, την κατάσταση του κάθε πελάτη αν είναι ενωμένος στο δίκτυο η όχι, την ip διεύθυνση του η οποία κάθε

φορά που ο πελάτης εισέρχεται στο σύστημα μπορεί να είναι διαφορετική, την θύρα στην οποία ακούει κάθε πελάτης αφού σε ένα Peer-to-Peer δίκτυο οι πελάτες λειτουργούν και σαν εξυπηρετητές ταυτόχρονα.

- PROFILES: Σε αυτό το πίνακα αποθηκεύονται τα στοιχεία που έχουν σχέση με το προφίλ για τον κάθε χρήστη. Συγκεκριμένα, για το σύνολο δεδομένων DBLP όπου αναφερόταν σε άρθρα που αφορούσαν κάποιο πελάτη το προφίλ για το κάθε πελάτη διαμορφωνόταν έτσι ώστε να περιέχει μόνο τους τίτλους των άρθρων για το κάθε πελάτη.

- INFO: Σε αυτό το πίνακα αποθηκεύονται τα στοιχεία που έχουν σχέση με τις GPS συντεταγμένες για το κάθε πελάτη καθώς επίσης και την χρονική στιγμή που έχει καταχωρηθεί το κάθε σετ συντεταγμένων GPS (Longitude, Latitude).

- MAPPING_WITH_QUSER: Αυτός είναι ο επιπλέον πίνακας ο οποίος κρατά τις αντιστοιχίες μεταξύ των πραγματικών id των πελατών και κάποιων υποθετικών id κατά τη διάρκεια που κάποιος πελάτης αναζητά δεδομένα από τους υπόλοιπους ενεργούς πελάτες του δικτύου. Συγκεκριμένα, θέτει το id του πελάτη που κάνει την αναζήτηση ίσο με το μηδέν και ταυτίζει ξανά τον κάθε πελάτη με ένα υποθετικό id. Η διαδικασία αυτή χρειάζεται για να μπορέσουν να τρέξουν οι Multiobjective Evolutionary αλγόριθμοι στον εξυπηρετητή αφού θεωρούν πάντα ότι ο πελάτης που κάνει πάντα την αναζήτηση είναι εκείνος που έχει id ίσο με το μηδέν.

Stored Procedures:

- REGISTER_USER: Η συγκεκριμένη διαδικασία επιτρέπει την καταχώρηση νέων χρηστών στη Βάση Δεδομένων εφόσον το όνομα του δίνει ο νέος χρήστης δεν υπάρχει ξανά καταχωρημένο σε άλλο χρήστη.

- CONNECT_USER: Η συγκεκριμένη διαδικασία επιτρέπει την αλλαγή της κατάστασης ενός καταχωρημένου χρήστη σε ενεργό χρήστη του συστήματος του

οποίου ενημερώνονται το ip και η θύρα του εφόσον διαφέρουν από τα ήδη καταχωρημένα.

- TERMINATE_CONNECTION_USER: Η συγκεκριμένη διαδικασία επιτρέπει την αλλαγή της κατάστασης ενός ενεργού χρήστη σε ανενεργό. Επίσης, διαγράφει από τη Βάση Δεδομένων το προφίλ του συγκεκριμένου χρήστη καθώς και τις καταχωρήσεις που έχουν σχέση με τις GPS συντεταγμένες του.

- INSERT_PROFILES: Η συγκεκριμένη διαδικασία επιτρέπει την εισαγωγή μιας νέας καταχώρησης στο πίνακα PROFILES που αφορά το προφίλ ενός συγκεκριμένου χρήστη.

- INSERT_INFO: Η συγκεκριμένη διαδικασία επιτρέπει την εισαγωγή μιας νέας καταχώρησης στο πίνακα INFO που αφορά τις GPS συντεταγμένες ενός συγκεκριμένου χρήστη.

- REMOVE_USER: Η συγκεκριμένη διαδικασία επιτρέπει την διαγραφή ενός συγκεκριμένου χρήστη από το σύστημα.

- REMOVE_USER_PROFILE: Η συγκεκριμένη διαδικασία επιτρέπει την διαγραφή των καταχωρήσεων που αφορούν το προφίλ ενός συγκεκριμένου χρήστη.

- REMOVE_USER_INFO: Η συγκεκριμένη διαδικασία επιτρέπει την διαγραφή των καταχωρήσεων που αφορούν τις GPS συντεταγμένες ενός συγκεκριμένου χρήστη.
- CREATE_QUERY: Η συγκεκριμένη διαδικασία θέτει το υποθετικό id του χρήστη που κάνει την αναζήτηση ίσο με το μηδέν και ταυτίζει ξανά τον κάθε πελάτη με ένα υποθετικό id μέσα στο πίνακα MAPPING_WITH_QUSER.

Functions:

- GET_ACTIVE_USERS: Η συγκεκριμένη συνάρτηση επιστρέφει τον αριθμό των ενεργών χρηστών του συστήματος.

- GET_NUM_USERS: Η συγκεκριμένη συνάρτηση επιστρέφει τον αριθμό των χρηστών του συστήματος.

- GET_USER_PROFILE: Η συγκεκριμένη συνάρτηση επιστρέφει το σύνολο των καταχωρήσεων που έχει στο προφίλ του ένας συγκεκριμένος χρήστης.

- GET_NUMBER_OF_USER_PAPER: Η συγκεκριμένη συνάρτηση επιστρέφει τον αριθμό του συνόλου των καταχωρήσεων που έχει στο προφίλ του ένας συγκεκριμένος χρήστης.

- GET_NUMBER_OF_USER_PAPER_WITH_SPESIFIC_WORD: Η συγκεκριμένη συνάρτηση επιστρέφει τον αριθμό του συνόλου των καταχωρήσεων που έχει στο προφίλ του ένας συγκεκριμένος χρήστης και τα οποία περιέχουν μια συγκεκριμένη λέξη η έκφραση.

- RESULTS_WITH_COUNT: Η συγκεκριμένη συνάρτηση επιστρέφει για κάποιους χρήστες του συστήματος πληροφορίες που αφορούν τον αριθμό των καταχωρήσεων που έχουν στο προφίλ τους καθώς και τον αριθμό των καταχωρήσεων που έχουν στο προφίλ τους τα οποία περιέχουν μια συγκεκριμένη λέξη ή έκφραση.

- GET_USER_ID_BY_NAME: Η συγκεκριμένη συνάρτηση επιστρέφει το id κάποιου χρήστη δοθέντος του ονόματος του.

- CALCULATEGPSDISTANCE: Η συγκεκριμένη συνάρτηση επιστρέφει την απόσταση σε μέτρα δοθέντος δύο GPS συντεταγμένων.

- DISTANCES_BETWEEN_A_USERS: Η συγκεκριμένη συνάρτηση επιστρέφει τις αποστάσεις μεταξύ των ενεργών χρηστών.

- RETURNMAPPEDUSER: Η συγκεκριμένη συνάρτηση επιστρέφει το πραγματικό id ενός χρήστη δοθέντος του υποθετικού του.

- RETURNLASTLONUSER: Η συγκεκριμένη συνάρτηση επιστρέφει το τελευταίο latitude που έχει καταχωρηθεί για ένα χρήστη στο πίνακα INFO σύμφωνα με την χρονική στιγμή που καταχωρήθηκε.

- RETURNLASTLATUSER: Η συγκεκριμένη συνάρτηση επιστρέφει το τελευταίο longitude που έχει καταχωρηθεί για ένα χρήστη στο πίνακα INFO σύμφωνα με την χρονική στιγμή που καταχωρήθηκε.

- GET_PORT: Η συγκεκριμένη συνάρτηση επιστρέφει τη θύρα στην οποία ακούει ένας χρήστης.

- GET_IP: Η συγκεκριμένη συνάρτηση επιστρέφει την ip διεύθυνση ενός χρήστη.

- ISACTIVE: Η συγκεκριμένη συνάρτηση επιστρέφει ένα αν ο ένας συγκεκριμένος χρήστης είναι ενεργός και μηδέν εάν δεν είναι.

4.1.2 Εξυπηρετητής SmartOpt

Ο εξυπηρετητής είναι υλοποιημένος με την αντικειμενοστραφή γλώσσα προγραμματισμού JAVA. Χρησιμοποιώντας τη JAVA υπάρχει η δυνατότητα η εφαρμογή που θα αναπτυχθεί να τρέχει σε διαφορετικές πλατφόρμες και άρα κάποιος ο οποίος θέλει να χρησιμοποιήσει το συγκεκριμένο σύστημα δεν θα χρειαστεί να ξαναγράψει τον κώδικα σε άλλη γλώσσα προγραμματισμού. Επίσης, η JAVA παρέχει την δυνατότητα αυτόματης αποδέσμευσης τμημάτων μνήμης τα οποία δεν χρησιμοποιούνται άλλο και δεν χρειάζονται. Επιπλέων, στην περίπτωση που μια εφαρμογή γραφτεί λάθος δεν θα επηρεάσει το υπόλοιπο σύστημα του υπολογιστή αφού η εφαρμογή αυτή τρέχει σε μια εικονική μηχανή.

Ο εξυπηρετητής είναι πολυνηματικός έτσι ώστε να μπορεί να εξυπηρετεί πολλούς πελάτες ταυτόχρονα. Επίσης έχει την δυνατότητα να επικοινωνεί με τους πελάτες και με την Βάση Δεδομένων. Ο τρόπος επικοινωνίας μεταξύ πελάτη και εξυπηρετητή

γίνεται μέσω TCP σύνδεσης και η ανταλλαγή πληροφοριών και δεδομένων μέσω sockets ενώ μεταξύ της Βάσης Δεδομένων και εξυπηρετητή η επικοινωνία επιτυγχάνεται μέσω του MICROSOFT SQL Server JDBC DRIVER 3.0.

Στην υλοποίηση του εξυπηρετητή περιλαμβάνονται βασικές λειτουργίες τις οποίες χρησιμοποιεί ο πελάτης όπως την καταχώρηση καινούριου πελάτη, την σύνδεση κάποιου πελάτη μέσα στο σύστημα, την αποσύνδεση κάποιου πελάτη από το σύστημα, την δημιουργία του δέντρου δρομολόγησης ερωτήματος στην οποία θα βασιστεί η αναζήτηση του πελάτη όπως και την αποστολή του δέντρου δρομολόγησης ερωτήματος στο πελάτη που θα πραγματοποιήσει την αναζήτηση.

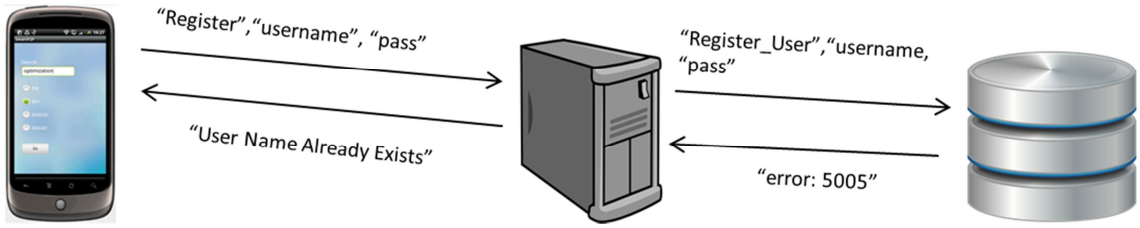
Επιπρόσθετα, υλοποιήθηκαν δύο απλοί αλγόριθμοι τυχαίας κατανεμημένης αναζήτησης και δύο πιο σύνθετοι αλγόριθμοι. Οι απλοί αλγόριθμοι είναι οι Random Walk (RW) και Breadth First Search (BFS), ενώ οι σύνθετοι είναι εκείνοι που έχουν την δυνατότητα να προσφέρουν μια Multi-Objective βελτιστοποίηση ο Multi-Objective Evolutionary Algorithm (MOEA/D) και ο Non-Dominated Sorting Genetic Algorithm II (NSGA-II). Στην περίπτωση που ο πελάτης επιθυμεί να χρησιμοποιήσει ένα από τους πιο πάνω Multi-Objective αλγόριθμους ο εξυπηρετητής δημιουργεί μία τρισδιάστατη γραφική παράσταση που αναπαριστά το Pareto Front (PF) και την αποστέλλει στο πελάτη.

Όλες οι πιο πάνω λειτουργίες χρησιμοποιούν λειτουργίες της Βάσης Δεδομένων για να μπορέσουν να παράγουν το επιθυμητό αποτέλεσμα. Πιο κάτω παρουσιάζεται πως ο εξυπηρετητής διαχειρίζεται κάθε αίτημα που δέχεται από τον πελάτη:

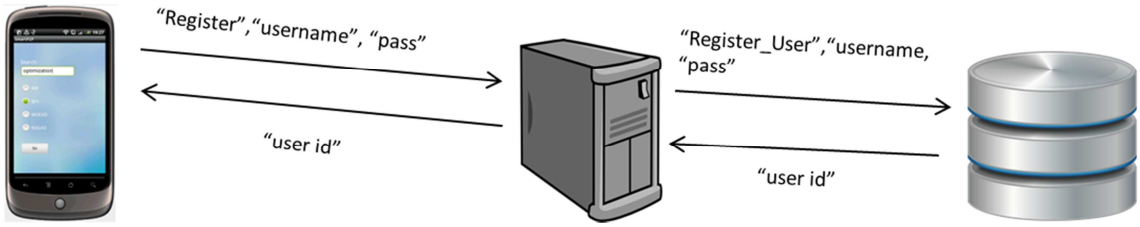
Αίτημα: Καταχώρηση Νέου Χρήστη

Σε αυτή την περίπτωση ο πελάτης στέλνει στον εξυπηρετητή μήνυμα ‘‘Register” ακολούθως του στέλνει το όνομα και το μυστικό κωδικό. Ο εξυπηρετητής επικοινωνεί με τη Βάση Δεδομένων στην οποία καλεί τη διαδικασία REGISTER_USER. Αν η καταχώρηση γίνει επιτυχώς τότε η Βάση Δεδομένων θα επιστρέψει το id που αναθέτει στο νέο πελάτη όπως φαίνεται στην εικόνα 4.1.2.2. Στην αντίθετη περίπτωση θα

εγείρει το error 5005 το οποίο θα διαχειριστεί ο εξυπηρετητής στέλνοντας στον πελάτη το μήνυμα ‘‘User Name Already Exists’’ όπως φαίνεται στην εικόνα 4.1.2.1.



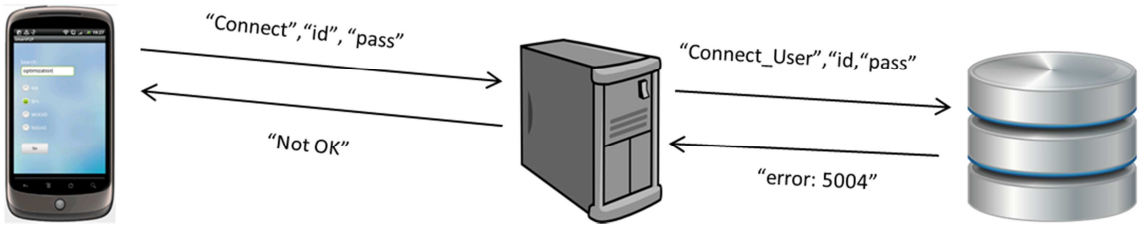
Εικόνα 4.1.2.1: Σφάλμα κατά την καταχώρηση του χρήστη στο σύστημα



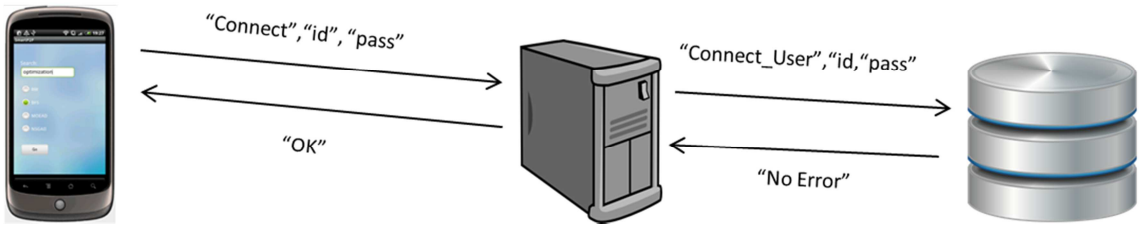
Εικόνα 4.1.2.2: Διαδικασία καταχώρησης ενός χρήστη στο σύστημα

Αίτημα: Αλλαγή κατάστασης του Πελάτη από ανενεργό σε ενεργό

Σε αυτή την περίπτωση ο πελάτης στέλνει στον εξυπηρετητή το μήνυμα ‘‘Connect” ακολούθως του στέλνει το id του, το μυστικό κωδικό του και τη θύρα που ακούει ο εξυπηρετητής στο πελάτη. Ο εξυπηρετητής επικοινωνεί με τη Βάση Δεδομένων στην οποία καλεί τη διαδικασία CONNECT_USER. Αν η αλλαγή της κατάστασης του πελάτη δεν γίνει επιτυχώς θα εγείρει το error 5004 το οποίο θα διαχειριστεί ο εξυπηρετητής στέλνοντας στον πελάτη το μήνυμα ‘‘NOT OK’’ όπως φαίνεται στην εικόνα 4.1.2.3. Στην αντίθετη περίπτωση θα σταλεί στον πελάτη το μήνυμα ‘‘OK’’ όπως φαίνεται στ ην εικόνα 4.1.2.4.



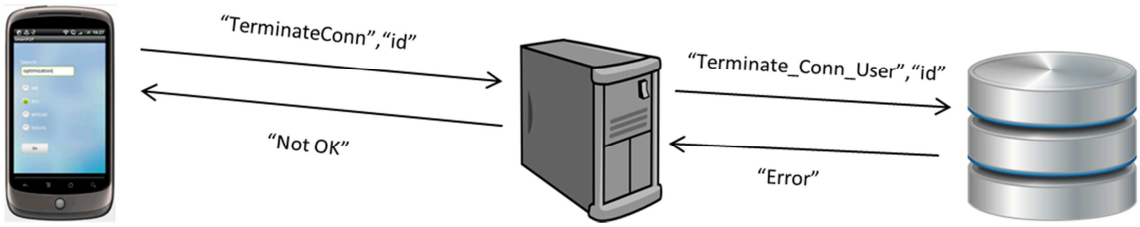
Εικόνα 4.1.2.3: Σφάλμα κατά την αλλαγή της κατάστασης του Πελάτη από ανενεργό σε ενεργό



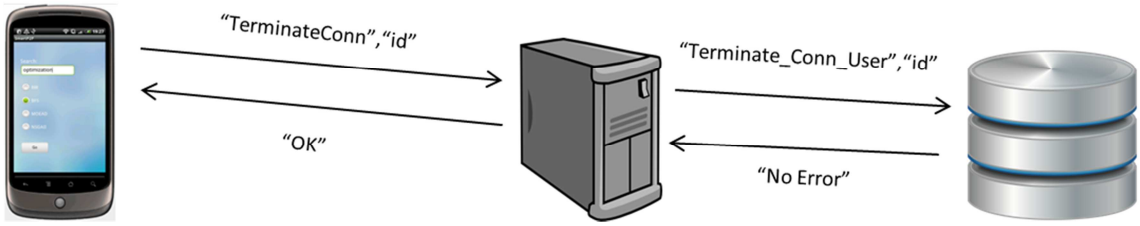
Εικόνα 4.1.2.4: Αλλαγή κατάστασης του Πελάτη από ανενεργό σε ενεργό

Αίτημα: Αλλαγή κατάστασης του Πελάτη από ενεργό σε ανενεργό

Σε αυτή την περίπτωση ο πελάτης στέλνει στον εξυπηρετητή το μήνυμα ‘‘Terminate_Conn” και ακολούθως το id του. Ο εξυπηρετητής επικοινωνεί με τη Βάση Δεδομένων στην οποία καλεί τη διαδικασία TERMINATE_CONNECTION_USER. Αν η αλλαγή της κατάστασης του πελάτη γίνει επιτυχώς ο εξυπηρετητής θα στείλει στο πελάτη το μήνυμα ‘‘OK’’ όπως παρουσιάζεται στην εικόνα 4.1.2.6 αλλιώς θα στείλει το μήνυμα ‘‘NOT OK’’ όπως παρουσιάζεται στην εικόνα 4.1.2.5.



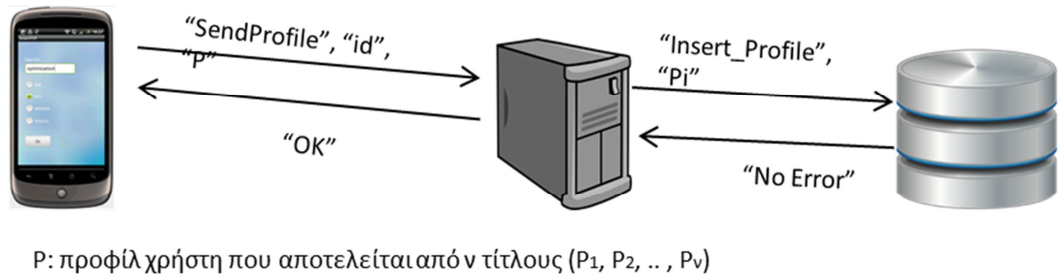
Εικόνα 4.1.2.5: Σφάλμα κατά την αλλαγή της κατάστασης του πελάτη από ενεργό σε ανενεργό



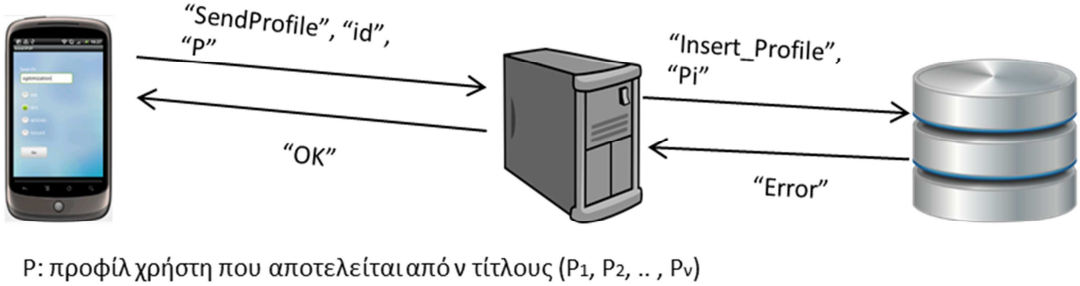
Εικόνα 4.1.2.6: Αλλαγή κατάστασης του Πελάτη από ενεργό σε ανενεργό

Αίτημα: Αποστολή του προφίλ

Σε αυτή την περίπτωση ο πελάτης αφού έχει ήδη γίνει ενεργός χρήστης μέσα στο δίκτυο πρέπει να στείλει το προφίλ του στον εξυπηρετητή έτσι ώστε να καταχωρηθεί στη Βάση Δεδομένων. Αρχικά, ο πελάτης στέλνει στον εξυπηρετητή το μήνυμα ‘‘SendProfile’’ και το id του. Στη συνέχεια, ο πελάτης ξεκινά να στέλνει το περιεχόμενο του προφίλ του και αυτό καταχωρείται στη Βάση Δεδομένων μέσω της διαδικασίας INSERT_PROFILES. Αν η καταχώρηση του προφίλ του πελάτη γίνει επιτυχώς ο εξυπηρετητής θα στείλει στο πελάτη το μήνυμα ‘‘OK’’ όπως παρουσιάζεται στην εικόνα 4.1.2.8 αλλιώς θα στείλει το μήνυμα ‘‘NOT OK’’ όπως παρουσιάζεται στην εικόνα 4.1.2.7.



Εικόνα 4.1.2.7: Σφάλμα στην διαδικασία αποστολής του προφίλ

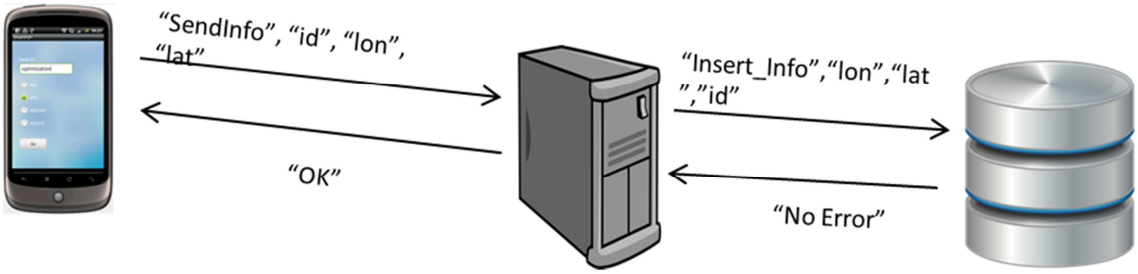


Εικόνα 4.1.2.8: Διαδικασία αποστολής του Προφίλ στον εξυπηρετητή και καταχώρηση του στην Βάση Δεδομένων

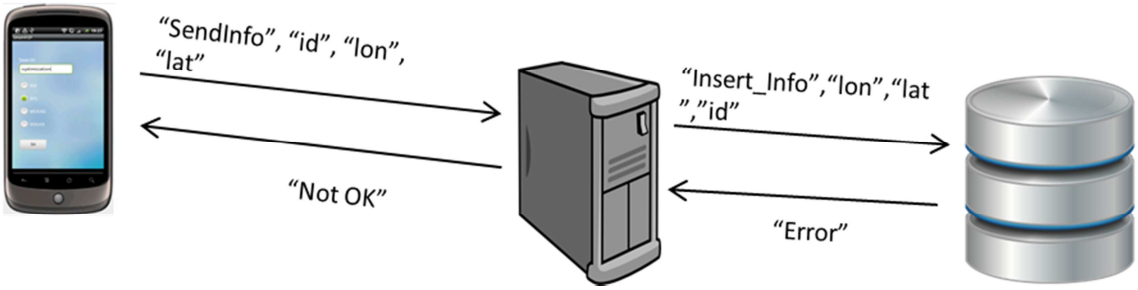
Αίτημα: Αποστολή των GPS συντεταγμένων του πελάτη

Σε αυτή την περίπτωση, όπως και στην πιο πάνω, ο πελάτης αφού έχει ήδη γίνει ενεργός χρήστης μέσα στο δίκτυο πρέπει να στείλει τις GPS συντεταγμένες του στον εξυπηρετητή έτσι ώστε να καταχωρηθεί στη Βάση Δεδομένων. Αρχικά, ο πελάτης

στέλνει στον εξυπηρετητή το μήνυμα ‘‘SendInfo’’ και το id του. Στη συνέχεια, ο πελάτης στέλνει τις συντεταγμένες του στον εξυπηρετητή ο οποίος τις καταχωρεί στη Βάση Δεδομένων μέσω της διαδικασίας INSERT_INFO. Αν η καταχώρηση γίνει επιτυχώς ο εξυπηρετητής θα στείλει στο πελάτη το μήνυμα ‘‘OK’’ όπως παρουσιάζεται στην εικόνα 4.1.2.9 αλλιώς θα στείλει το μήνυμα ‘‘NOT OK’’ όπως παρουσιάζεται στην εικόνα 4.1.2.10.



Εικόνα 4.1.2.9: Σφάλμα αποστολής γεωγραφικών συντεταγμένων

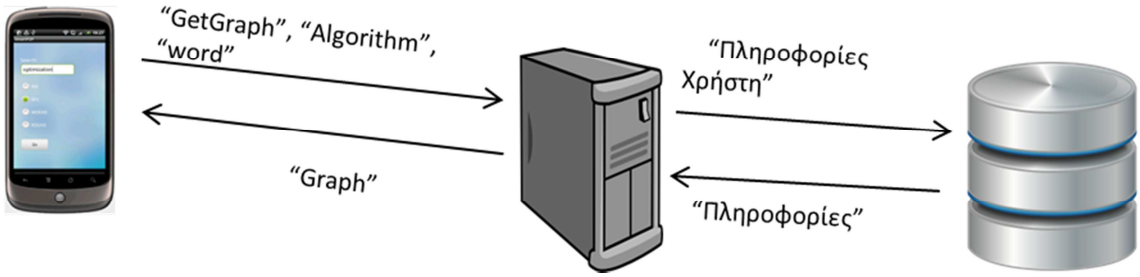


Εικόνα 4.1.2.10: Διαδικασία αποστολής του γεωγραφικών συντεταγμένων στον εξυπηρετητή και καταχώρηση του στην Βάση Δεδομένων

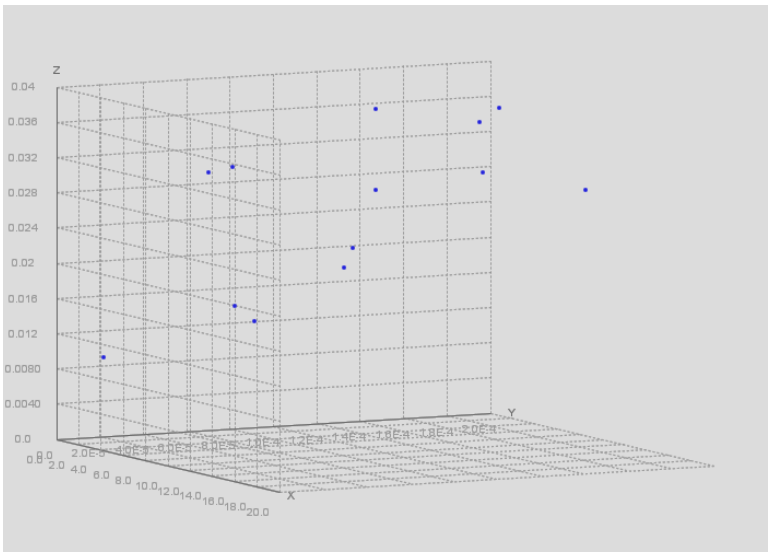
Αίτημα: Αποστολής Γραφικής Παράστασης στο Πελάτη

Σε αυτή την περίπτωση ο πελάτης ζητά από τον εξυπηρετητή να του στείλει τη γραφική παράσταση όπου αναπαρίσταται το Pareto Front. Ο πελάτης χρησιμοποιεί το συγκεκριμένο αίτημα μόνο όταν θα αναζητήσει κάποια δεδομένα με κάποια Multi-Objective τεχνική. Εδώ είναι η στιγμή όπου τρέχουν οι Multi-Objective τεχνικές και παράγεται το Pareto Front. Αρχικά ο πελάτης στέλνει το αίτημα ‘‘GetGraph’’ και ακολούθως στέλνει την πληροφορία που περιγράφει τον αλγόριθμο που θέλει να γίνει η

αναζήτηση και την έκφραση της αναζήτησης. Ο αλγόριθμος με τη σειρά του ζητά πληροφορίες σχετικά με τους χρήστες του δικτύου και των δεδομένων που έχει κάθε χρήστης από τη Βάση Δεδομένων και εκτελείται. Η διαδικασία αυτή ισχύει και για τις δύο Multi-Objective τεχνικές και παρουσιάζεται από την εικόνα 4.1.2.11. Ακολούθως, αφού έχει δημιουργηθεί το Pareto Front, ο εξυπηρετητής δημιουργεί μία τρισδιάστατη γραφική παράσταση (εικόνα 4.1.2.12) όπου κάθε σημείο της αναπαριστά και ένα δέντρο στο Pareto Front. Κάθε άξονας περιγράφει μία από τις πληροφορίες για κατανάλωση ενέργειας, ρυθμό ανάκλησης και χρόνο άφιξης των αποτελεσμάτων. Τέλος, η γραφική παράσταση αποστέλλεται στον πελάτη.



Εικόνα 4.1.2.11: Διαδικασία αποστολής της γραφικής παράστασης που αναπαριστά το Pareto Front στο πελάτη

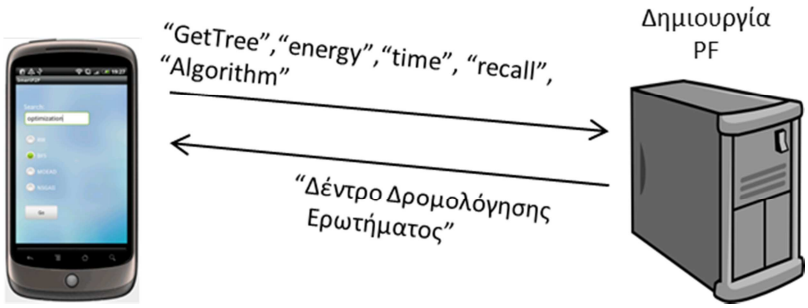


Εικόνα 4.1.2.12: Γραφική παράσταση που αναπαριστά το Pareto Front

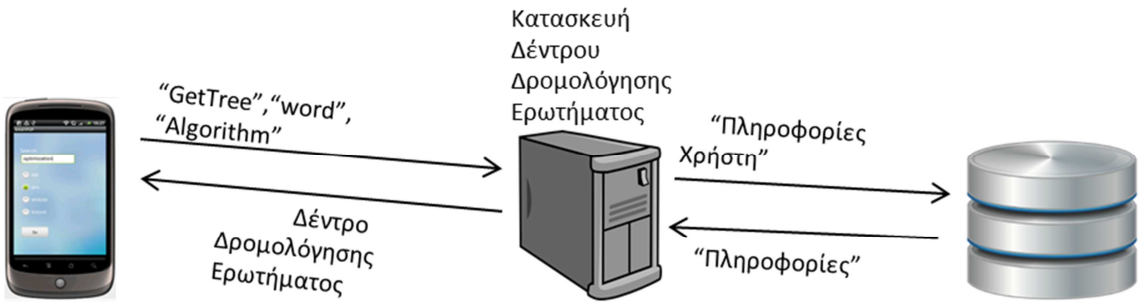
Αίτημα: Αποστολή δέντρου αναζήτησης στον πελάτη

Σε αυτή την περίπτωση ο πελάτης, αφού έχει ήδη στείλει τις GPS συντεταγμένες του και το προφίλ του στον εξυπηρετητή, έχει τη δυνατότητα να διεξάγει κάποια αναζήτηση γύρω από τα προφίλ των υπόλοιπων ενεργών χρηστών του δικτύου. Έτσι, ο πελάτης στέλνει στον εξυπηρετητή το μήνυμα ‘‘GetTree’’ με σκοπό να παραλάβει ένα δέντρο αναζήτησης από τον εξυπηρετητή του οποίου κάθε κόμβος του αναπαριστά και ένα ενεργό χρήστη του δικτύου. Η δημιουργία του δέντρου καθορίζεται από την επιλογή του πελάτη σχετικά με ποιος αλγόριθμος θα δημιουργήσει το δέντρο. Ο εξυπηρετητής υποστηρίζει τέσσερεις αλγόριθμους δύο Single-Objective τεχνικές (RW, BFS) και δύο Multi-Objective τεχνικές (MOEAD, NSGAII). Στην περίπτωση που επιλεχθεί κάποια Single-Objective τεχνική η δημιουργεί του δέντρου θα γίνει σε αυτή τη φάση όπως παρουσιάζεται στην εικόνα 4.1.2.14. Συγκεκριμένα αν επιλεχθεί η τεχνική RW η δημιουργεί του δέντρου γίνεται με τυχαία επιλογή ενεργών χρηστών του δικτύου και τυχαία τοποθέτηση τους μέσα στο δέντρο και ακολούθως αποστέλλεται στο πελάτη. Αν επιλεχθεί η τεχνική BFS η δημιουργία του δέντρου γίνεται με την επιλογή όλων των ενεργών χρηστών του δικτύου και την τυχαία τοποθέτηση τους μέσα στο δέντρο με τρόπο όπου κάθε κόμβος του δέντρου να έχει log(n) τυχαία παιδιά (όπου n είναι ο αριθμός των ενεργών χρηστών) για να μπορέσει να σχηματιστεί ένα πλήρες δέντρο αν είναι δυνατό. Ακολούθως αποστέλλεται στο πελάτη το δέντρο. Στην περίπτωση που θα επιλεχθεί μία Multi-Objective τεχνική θεωρείται ότι η τεχνική αυτή έχει εκτελεστεί και έχει δημιουργηθεί το Pareto Front. Έτσι ο πελάτης θα πρέπει να στείλει κάποιες επιπλέων πληροφορίες στον εξυπηρετητή (εικόνα 4.1.2.13)που σχετίζονται με την κατανάλωση της ενέργειας από τις κινητές συσκευές, το χρόνο άφιξης των αποτελεσμάτων στο πελάτη και το ρυθμό ανάκλησης. Ακολούθως, αν η πληροφορία για την κατανάλωση της ενέργειας έχει την τιμή 0 θα επιλεγεί το δέντρο το οποίο θα καταναλώνει την ελάχιστη ενέργεια. Στην αντίθετη περίπτωση όπου η πληροφορία για την κατανάλωση της ενέργειας έχει την τιμή 1 τότε θα λαμβάνονται υπόψη οι υπόλοιπες δύο πληροφορίες. Για την επιλογή του δέντρου, χρησιμοποιείται η μέθοδος της ευκλείδειας απόστασης όπου ελέγχεται πιο από τα δέντρα που υπάρχουν στο Pareto Front είναι πιο κοντά στις απαιτήσεις που πελάτη με βάση τις πληροφορίες για το ρυθμό ανάκλησης και το χρόνο άφιξης των αποτελεσμάτων. Στην περίπτωση που

υπάρχουν περισσότερα από ένα δέντρο που πληρεί αυτές τις απαιτήσεις θα σταλεί εκείνο που καταναλώνει την λιγότερη ενέργεια.



Εικόνα 4.1.2.13: Διαδικασία αποστολής δέντρου δρομολόγησης ερωτήματος στην περίπτωση που χρησιμοποιούμε Multi – Objective τεχνικές



Εικόνα 4.1.2.14: Διαδικασία αποστολής δέντρου δρομολόγησης ερωτήματος στην περίπτωση που χρησιμοποιούμε Single – Objective τεχνικές

4.2 GUI Εφαρμογής SmartP2P

Η εφαρμογή SmartP2P υλοποιεί το μοντέλο ενός πελάτη ο οποίος επικοινωνεί με τον εξυπηρετητή SmartOpt. Ξεκινώντας την εφαρμογή η εικόνα που παρουσιάζεται στη έξυπνη κινητή συσκευή φαίνεται πιο κάτω στην εικόνα 4.2.1 (α). Ακολούθως πατώντας το κουμπί menu εμφανίζονται στο χρήστη έχει τρεις επιλογές Register, Connect και Exit όπως παρουσιάζεται στην εικόνα 4.2.1 (β). Με την επιλογή Exit η εφαρμογή τερματίζει.

(α) **(β)**

Εικόνα 4.2.1: Εφαρμογή SmartP2P

Επιλέγοντας το Register (εικόνα (β)) ο χρήστης έχει την δυνατότητα να εγγραφεί στη Βάση Δεδομένων του SmartP2P μέσω της διπροσωπίας που παρουσιάζεται στην εικόνα 4.2.2 (α). Θα πρέπει να τοποθετήσει το ip του εξυπηρετητή στο κουτί κάτω από την ετικέτα IP, τη θύρα στην οποία ακούει ο εξυπηρετητής στο κουτί κάτω από την ετικέτα Server Port, ένα μοναδικό όνομα στο κουτί κάτω από την ετικέτα Username και το κωδικό του χρήστη στο κουτί κάτω από την ετικέτα Password. Αν τα πιο πάνω στοιχεία είναι ορθά, πατώντας το κουμπί Register θα εμφανιστεί στην οθόνη η κύρια εικόνα της εφαρμογής και στην μέση θα εμφανιστεί ένα μήνυμα με το μοναδικό id του χρήστη (εικόνα 4.2.2 (β)).

Στην περίπτωση που το όνομα του χρήστη υπάρχει ήδη θα εμφανιστεί στην οθόνη το μήνυμα ‘‘User Name Already Exists’’

Στην περίπτωση που το ip του εξυπηρετητή είναι λάθος ή η θύρα στην οποία ακούει θα εμφανιστεί στην οθόνη το μήνυμα ‘‘No Connection With Server’’

Πιο κάτω παρουσιάζεται η διαδικασία εγγραφής στο σύστημα από το γραφικό περιβάλλον της εφαρμογής:

(α) **(β)**

Εικόνα 4.2.2: Διαδικασία εγγραφής κάποιου χρήστη στο σύστημα

Επιλέγοντας το Connect (εικόνα (β)) ο χρήστης έχει τη δυνατότητα να γίνει ενεργός χρήστης μέσα στο δίκτυο όπως παρουσιάζεται στην εικόνα 4.2.3 (α). Θα πρέπει να τοποθετήσει το ip του εξυπηρετητή στο κουτί κάτω από την ετικέτα IP, τη θύρα στην οποία ακούει ο εξυπηρετητής στο κουτί κάτω από την ετικέτα Server Port, το μοναδικό του id στο κουτί κάτω από την ετικέτα User Id και το κωδικό του χρήστη στο κουτί κάτω από την ετικέτα Password (εικόνα 4.2.3 (β)). Αν τα πιο πάνω στοιχεία είναι ορθά, πατώντας το κουμπί Connect η εφαρμογή στέλνει στον εξυπηρετητή το περιεχόμενο του προφίλ του χρήστη καθώς και τις GPS συντεταγμένες του (latitude και longitude) και στην οθόνη του χρήστη παρουσιάζεται το περιεχόμενο της εικόνας που επιτρέπει στο χρήστη να ξεκινήσει μία αναζήτηση. Το περιεχόμενο του προφίλ του κάθε χρήστη βρίσκεται στο φάκελο “mnt/sdcard/SmartP2P”. Παράλληλα ξεκινά ένα εξυπηρετητής να τρέχει στη συσκευή στη θύρα 45000.

Στην περίπτωση που το ip του εξυπηρετητή είναι λάθος ή η θύρα στην οποία ακούει θα εμφανιστεί στην οθόνη το μήνυμα ‘‘No Connection With Server’’

Αν ο χρήστης επιθυμεί να αποσυνδεθεί από το δίκτυο και να είναι ανενεργός αρκεί να πατήσει το κουμπί “Back”.

Πιο κάτω στην εικόνα παρουσιάζεται η διαδικασία για να γίνει ενεργός ο χρήστης στο σύστημα από το γραφικό περιβάλλον της εφαρμογής:

(α) (β) (γ)

Εικόνα 4.2.3: Διαδικασία αλλαγής κατάστασης του χρήστη σε ενεργό

Αφού γίνει ενεργός ο χρήστης μπορεί να ξεκινήσει μια αναζήτηση. Ας υποθέσουμε ότι ο χρήστης ενδιαφέρεται να αναζητήσει δεδομένα που έχουν σχέση με τη λέξη “optimization”. Θα πρέπει να τοποθετήσει τη λέξη “optimization” στο κουτί κάτω από την ετικέτα “Search” και να επιλέξει μία από τις επιλογές αλγορίθμων που ακολουθούν (εικόνα 4.2.3 (γ)).

Στην περίπτωση που διαλέξει ένα από τους αλγόριθμους RW (Random Walk) ή BFS (Breadth First Search) η εφαρμογή ζητά από τον εξυπηρετητή να της στείλει το δέντρο δρομολόγησης αναζήτησης ανάλογα με τον αλγόριθμο που έχει επιλεχθεί. Μόλις λάβει το δέντρο, η εφαρμογή ξεκινά τη διαδικασία αναζήτησης.

Στην περίπτωση που διαλέξει ένα από τους αλγόριθμους πολυκριτηρίων (MOEAD ή NSGA-II) η εφαρμογή ζητά από τον εξυπηρετητή να τρέξει τον αλγόριθμο που έχει επιλεχθεί για να δημιουργηθεί το σύνολο των μη κυριαρχούμενων λύσεων όπου κάθε λύση αναπαριστά και κάποιο βελτιστοποιημένο δέντρο για την διαδικασία αναζήτησης.

Στη συνέχεια η εφαρμογή ζητά από τον εξυπηρετητή να του στείλει την γραφική αναπαράσταση των μη κυριαρχούμενων λύσεων ως προς τις βελτιώσεις που έγιναν στην κατανάλωση ενέργειας, στο ρυθμό ανάκλησης και στο χρόνο που χρειάζονται τα αποτελέσματα να φτάσουν στο χρήστη που υπόβαλε το ερώτημα της αναζήτησης. Όταν ο χρήστης λάβει την γραφική αναπαράσταση εμφανίζεται στην οθόνη του χρήστη η γραφική παράσταση μαζί με κάποιες επιλογές από κάτω όπως φαίνεται στην εικόνα 4.2.4 (β). Ο χρήστης βλέπει πόσο βελτιστοποιεί τις παραμέτρους κάθε δέντρο στην γραφική παράσταση και έχει την δυνατότητα να διαλέξει ένα από αυτά βάση των επιλογών κάτω από την γραφική όπως φαίνεται στο κάτω μέρος της εικόνας 4.2.4 (β). Όταν δεν είναι επιλεγμένη η επιλογή “Decision Making” πατώντας το κουμπί “Go” ο χρήστης ζητά από τον εξυπηρετητή να του στείλει το δέντρο που θα έχει την μικρότερη κατανάλωση. Στην αντίθετη περίπτωση ζητά από τον εξυπηρετητή να του στείλει ένα δέντρο που βασίζεται στις τιμές που θα αναθέσει ο χρήστης για το “Time” και “Recall”. Το Time αναπαριστά το χρόνο που χρειάζονται τα αποτελέσματα για να φτάσουν στο χρήστη που υπόβαλε το ερώτημα της αναζήτησης και το Recall το ρυθμό ανάκλησης. Αν για παράδειγμα ο χρήστης επιλέξει μέγιστη τιμή για το Time τότε η τιμή που αντιστοιχεί για το Recall είναι μέγιστη επίσης. Αφού γίνουν οι πιο πάνω ρυθμίσεις πατώντας το κουμπί Go η εφαρμογή ζητά από τον εξυπηρετητή να της στείλει το δέντρο βάση των ρυθμίσεων που έθεσε ο χρήστης και αφού λάβει το δέντρο η διαδικασία της αναζήτησης είναι έτοιμη να ξεκινήσει.

(α) **(β)**

Εικόνα 4.2.4: Απόφαση του Πελάτη

Η διαδικασία της αναζήτησης ξεκινά από τη συσκευή του χρήστη που υποβάλλει το ερώτημα της αναζήτησης. Η εφαρμογή στέλνει το ερώτημα στους άμεσους γείτονες – παιδία όπως υποδεικνύονται από το δέντρο αναζήτησης. Κάθε γείτονας που λαμβάνει την ερώτηση λαμβάνει και το δέντρο αναζήτησης από τον πατέρα του. Ως πρώτο στάδιο προωθεί την ερώτηση στους δικούς ρου γείτονες παιδιά μαζί με το δέντρο δρομολόγησης και ακολούθως ξεκινά την αναζήτηση στη δική του βάση δεδομένων για το ερώτημα που του τέθηκε. Αν έχει κάποια απάντηση ξεκινά μια νέα σύνδεση με τον εξυπηρετητή του χρήστη που υπόβαλε την ερώτηση και του στέλνει την απάντηση. Η διαδικασία τερματίζει όταν όλοι οι χρήστες που βρίσκονται μέσα στο δέντρο λάβουν την ερώτηση του αρχικού χρήστη και τελειώσουν την αναζήτηση στη βάση δεδομένων τους.

Η εφαρμογή του χρήστη που ξεκίνησε την αναζήτηση είναι όπως φαίνεται στην εικόνα 4.2.5(α) και για να εμφανιστούν τα αποτελέσματα πρέπει ο χρήστης να πατήσει το κουμπί “Results” εικόνα 4.2.5 (β).

(α) **(β)** **(γ)**

Εικόνα 4.2.5: Παρουσίαση Αποτελεσμάτων και Αναπαράσταση των τοποθεσιών των χρηστών σε Google Map

Η εφαρμογή επιτρέπει στους χρήστες της μετά το τέλος της αναζήτησης να δουν τις τοποθεσίες των χρηστών που συνεισέφεραν στην αναζήτηση στο χάρτη βασισμένη των γεωγραφικών τοποθεσιών του κάθε χρήστη. Η αναπαράσταση αυτή γίνετε σε χάρτες

της Google όπως φαίνεται στην εικόνα 4.2.5 (γ). Αγγίζοντας σε κάποια πινέζα στο χάρτη θα εμφανιστεί το id του χρήστη που αντιστοιχείτε στη συγκεκριμένη τοποθεσία. Οι χρήστες είναι ενωμένοι με γραμμές για να δειχθεί η αναπαράσταση του δέντρου που το αποτελούν.

Κεφάλαιο 5

Πειραματική Μεθοδολογία

5.1 Πειραματική Υποδομή
5.1.1 SmartLab
5.1.2 PowerTuror
5.1.3 Πειραματικός Εξυπηρετητής
5.1.4 Πειραματική Εφαρμογή
5.1.5 Περιγραφή Datasets
5.1.5.1 Dataset DBLP
5.1.5.2 Dataset Pics n Trails
5.1.6 Συσκευές που Χρησιμοποιήθηκαν
5.2 Πειραματική Διαδικασία

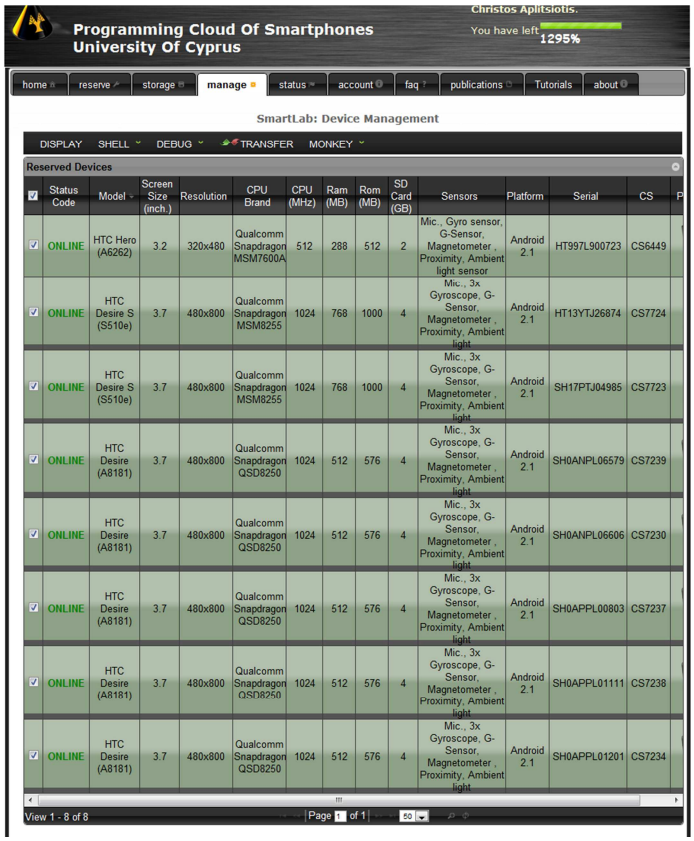
5.1 Πειραματική Υποδομή

Σε αυτή την ενότητα θα περιγραφεί όλη η πειραματική υποδομή που χρησιμοποιήθηκε για να τρέξουμε τα πειράματα. Στόχος μας είναι να συγκρίνουμε τις τεχνικές που αναφερθήκαμε σε πιο πάνω ενότητες και να εξάγουμε τα συμπεράσματα μας.

5.1.1 SmartLab

Το SmartLab [37] είναι μια πλατφόρμα Android έξυπνων κινητών συσκευών για πειραματικές δοκιμές η οποία αναπτύχθηκε από το Τμήμα Πληροφορικής του

Πανεπιστημίου Κύπρου. Μέσω μιας διορατικής Web-based διαπροσωπίας το SmartLab παρέχει δημόσια μια μόνιμή πλατφόρμα δοκιμών για ανάπτυξη δικτυακών εφαρμογών για έξυπνες κινητές συσκευές. Οι χρήστες της πλατφόρμας αυτής έχουν την δυνατότητα να ανεβάζουν τα APK αρχεία τους στις κινητές συσκευές και να τα εγκαθιστούν, να τρέχουν την εφαρμογή τους, να ανακτούν και να ανεβάζουν αρχεία από και προς τις συσκευές. Το SmartLab αποτελείται από περίπου 40 έξυπνες συσκευές Android και παρουσιάζεται στην εικόνα 5.1.1.1.



Status Code	Model	Screen Size (inch.)	Resolution	CPU Brand	CPU (MHz)	Ram (MB)	Rom (MB)	SD Card (GB)	Sensors	Platform	Serial	CS
ONLINE	HTC Hero (A6262)	3.2	320x480	Qualcomm Snapdragon MSM7600A	512	288	512	2	Mic., Gyro sensor, G-Sensor, Magnetometer , Proximity, Ambient light sensor	Android 2.1	HT997L900723	CS6449
ONLINE	HTC Desire S (S510e)	3.7	480x800	Qualcomm Snapdragon MSM8255	1024	768	1000	4	Mic., 3x Gyroscope, G-Sensor, Magnetometer , Proximity, Ambient light	Android 2.1	HT13YTJ26874	CS7724
ONLINE	HTC Desire S (S510e)	3.7	480x800	Qualcomm Snapdragon MSM8255	1024	768	1000	4	Mic., 3x Gyroscope, G-Sensor, Magnetometer , Proximity, Ambient light	Android 2.1	SH17PTJ04985	CS7723
ONLINE	HTC Desire (A8181)	3.7	480x800	Qualcomm Snapdragon QSD8250	1024	512	576	4	Mic., 3x Gyroscope, G-Sensor, Magnetometer , Proximity, Ambient light	Android 2.1	SH0ANPL06579	CS7239
ONLINE	HTC Desire (A8181)	3.7	480x800	Qualcomm Snapdragon QSD8250	1024	512	576	4	Mic., 3x Gyroscope, G-Sensor, Magnetometer , Proximity, Ambient light	Android 2.1	SH0ANPL06606	CS7230
ONLINE	HTC Desire (A8181)	3.7	480x800	Qualcomm Snapdragon QSD8250	1024	512	576	4	Mic., 3x Gyroscope, G-Sensor, Magnetometer , Proximity, Ambient light	Android 2.1	SH0APPL00803	CS7237
ONLINE	HTC Desire (A8181)	3.7	480x800	Qualcomm Snapdragon QSD8250	1024	512	576	4	Mic., 3x Gyroscope, G-Sensor, Magnetometer , Proximity, Ambient light	Android 2.1	SH0APPL01111	CS7238
ONLINE	HTC Desire (A8181)	3.7	480x800	Qualcomm Snapdragon QSD8250	1024	512	576	4	Mic., 3x Gyroscope, G-Sensor, Magnetometer , Proximity, Ambient light	Android 2.1	SH0APPL01201	CS7234

Εικόνα 5.1.1.1: SmartLab

5.1.2 PowerTutor

Για να μπορέσουμε να πάρουμε τις αναγκαίες μετρήσεις σχετικά με την κατανάλωση της ενέργεια από τις κινητές συσκευές χρησιμοποιήσαμε την εφαρμογή PowerTutor [31] . Το PowerTutor χρησιμοποιεί ένα μοντέλο για την κατανάλωση της ενέργειας που δημιουργείται από άμεσες μετρήσεις κατά τη διάρκεια ενός προσεκτικού ελέγχου των καταστάσεων της διαχείρισης της ενέργειας από τις κινητές συσκευές. Αυτό το μοντέλο

παρέχει γενικά την εκτίμηση της κατανάλωσης της ενέργειας στο 5% των πραγματικών τιμών. Το PowerTutor χρησιμοποιείται γενικά για οποιαδήποτε εφαρμογή τρέχει στην κινητή συσκευή. Η εφαρμογή PowerTutor παρουσιάζεται στην εικόνα 5.1.2.1.

Εικόνα 5.1.2.1: PowerTutor

5.1.3 Πειραματικός Εξυπηρετητής

Για την πειραματική διαδικασία δημιουργήσαμε ένα πειραματικό εξυπηρετητή έτσι ώστε να διευκολύνεται η διαδικασία εκτέλεσης των πειραμάτων. Ο πειραματικός εξυπηρετητής είναι σχεδιασμένος βάση των απαιτήσεων της πειραματικής εφαρμογής. Συγκεκριμένα, ο πειραματικός εξυπηρετητής εκτελεί κάποιες λειτουργίες από μόνος του χωρίς να του δώσει ο πελάτης συγκεκριμένες εντολές όπως για παράδειγμα την διαδικασία απόφασης για τις Multi-Objective τεχνικές όπου γίνετε η απόφαση με βάση κάποιες προεπιλεγμένες παραμέτρους χωρίς τη μεσολάβηση της γραφικής παράστασης. Σε καμία περίπτωση όμως δεν αλλάζει ο τρόπος εκτέλεσης του συστήματος από την κανονική εφαρμογή.

5.1.4 Πειραματική Εφαρμογή

Για την πειραματική διαδικασία δημιουργήσαμε μία πειραματική εφαρμογή όπου οι επιλογές όλες έχουν συμπτυχτεί σε τρία πλαίσια σε σχέση με την κανονική εφαρμογή που έχει οκτώ εικόνα 5.1.4.1.

Στο πρώτο πλαίσιο (εικόνα 5.1.4.1 (α)) υπάρχουν τρεις επιλογές που πρέπει να συμπληρώσει ο χρήστης. Το κουτί κάτω από την ετικέτα Number of Users αναφέρεται στον αριθμό των χρηστών (N) που θα προσομοιωθούν στην συσκευή. Το κουτί κάτω από την ετικέτα Start User αναφέρεται στο Id του πρώτου χρήστη από του N χρήστες που θα προσημειωθούν στην κινητή συσκευή. Οι υπόλοιποι N-1 χρήστες θα έχουν τους αμέσως διαδοχικούς αριθμούς του Id για Id τους αντίστοιχα. Πατώντας το κουμπί Go η εφαρμογή ξεκινά να στέλνει αίτημα στον εξυπηρετητή για να αλλάξει τις καταστάσεις των N χρηστών σε ενεργούς. Παράλληλα ξεκινούν να τρέχουν N εξυπηρετητές στην έξυπνη κινητή συσκευή.

Στο δεύτερο πλαίσιο (εικόνα 5.1.4.1 (β)) ο χρήστης πρέπει να διαλέξει ένα αλγόριθμο κάτω από την ετικέτα Algorithm. Ακολούθως, μπορεί να επιλέξει το κουτί δίπλα από την ετικέτα Decision Making στην περίπτωση που έχει επιλέξει ένα από τους δύο Multi-Objective αλγόριθμους και να καθορίσει την τιμή του χρόνου και του ρυθμού ανάκλησης ενημερώνοντας το κουτί δίπλα από την ετικέτα Time. Στην συνέχεια, μπορεί να τοποθετήσει την λέξη για την οποία πραγματοποιεί την αναζήτηση κάτω από την ετικέτα Word. Πιο κάτω από την ετικέτα User ID μπορεί να συμπληρώσει το id του χρήστη που πραγματοποιεί την αναζήτηση μιας και η συσκευή έχει την δυνατότητα να προσομοιώσει πάρα πολλούς χρήστες. Σε τελευταίο στάδιο ο χρήστης μπορεί να συμπληρώσει το όνομα του αρχείου στο οποίο επιθυμεί να αποθηκευτούν τα αποτελέσματα του πειράματος κάτω από την ετικέτα File Name. Το κουμπί Time αποθηκεύει στο αρχείο filename.time.txt την χρονική στιγμή που παραλαμβάνει ο χρήστης που υπόβαλε το ερώτημα κάθε απάντηση. Το κουμπί Calculate Power ελέγχει το τελευταίο log αρχείο που δημιουργήθηκε από το PowerTutor το οποίο περιέχει την κατανάλωση της ενέργειας για κάθε εφαρμογή που τρέχει στην συσκευή. Συγκεκριμένα, ελέγχει κάθε καταχώρηση του log αρχείου και όσες αναφέρονται στην συγκεκριμένη εφαρμογή ξαναγράφονται στο αρχείο filename.power. Επειδή το log

αρχείο κρατά καταχωρήσεις για κατανάλωση της ενέργειας από το CPU και για κατανάλωση της ενέργειας από το WiFi η διαδικασία εντοπισμού των καταχωρήσεων για κατανάλωση της ενέργεια διεξάγετε από μόνη της δύο φορές έτσι ώστε να εντοπιστεί πρώτα η κατανάλωση της εφαρμογής από το CPU και στη συνέχεια η κατανάλωση από το WIFI. (filename: το όνομα του αρχείου που δόθηκε κάτω από την ετικέτα File Name)

Στο τρίτο πλαίσιο (εικόνα 5.1.4.1 (γ)) παρουσιάζονται τα αποτελέσματα της εκτέλεσης της διαδικασίας αναζήτησης όπως παρουσιάζονται και στην κανονική εφαρμογή. Η διαφορά είναι ότι δεν υπάρχει επιλογή εμφάνισης των χρηστών στο χάρτη αφού η εφαρμογή είναι πειραματική και δεν υπάρχει περίπτωση επιρροής των αποτελεσμάτων.

(α) **(β)** **(γ)**

Εικόνα 5.1.4.1: Πειραματική εφαρμογή

5.1.5 Περιγραφή Συνόλου Δεδομένων

Για την πειραματική διαδικασία χρησιμοποιήθηκαν τρία διαφορετικά πραγματικά σύνολα δεδομένων. Το πρώτο ονομάζεται DBLP[8] , το δεύτερο GeoLife [49] και το τρίτο Pics-n-Trails[16] [17] . Τα σύνολα δεδομένων αναλύονται στην συνέχεια καθώς επίσης και οι παραδοχές που έχουν γίνει προκειμένου να προσαρμοστούν στις ρυθμίσεις του συστήματος.

5.1.5.1 Σύνολο Δεδομένων DBLP

Το DBLP [8] είναι ένα πραγματικό σύνολο δεδομένων από την DBLP Computer Science Bibliography ιστοσελίδα η οποία αφορά περισσότερο από 1.4 εκατομμύρια δημοσιεύσεις σε XML μορφή. Συγκεκριμένα, οι καταχωρήσεις αφορούν τίτλους εγγράφων, αναφορά σε διαδικτυακούς συνδέσμους, συνδημιουργούς, συνδέσμους μεταξύ εγγράφων και δημιουργών και άλλες χρήσιμες πληροφορίες.

Για να μπορέσουμε να αντιστοιχίσουμε το πρόβλημα μας με το συγκεκριμένο σύνολο δεομένων υποθέσαμε ότι κάθε καταχώρηση $O_{i,k}$ είναι ένα έγγραφο ενός δημιουργού και επιπλέων κάθε καταχώρηση έχει ως ετικέτες λέξεις οι οποίες βρίσκονται μέσα στο τίτλο που αντιστοιχεί στην καταχώρηση. Ο κοινωνικός γράφος είναι κατασκευασμένος από τις σχέσεις μεταξύ σων συνδημιουργών που είναι μέρος του συνόλου δεδομένων.

5.1.5.2 Σύνολο Δεδομένων GeoLife

Το GeoLife [49] είναι ένα πραγματικό σύνολο δεδομένων από 1100 GPS τροχιές που συλλέγονται από το Microsoft Research Asia στα πλαίσια του σχεδίου GeoLife με 165 χρήστες σε μια περίοδο άνω των δύο ετών από το Απρίλιος 2007 - Αύγουστος 2009. Οι χρήστες κινούνται στην πόλη Beijing. Το 95% του συνόλου δεδομένων αναφέρεται σε διακριτότητα του ενός δείγματος κάθε 2-5 δευτερολέπτων η 5-10 δευτερολέπτων.

5.1.5.3 Σύνολο Δεδομένων Pics-n-Trails

Το Pics-n-Trails [16] [17] είναι ένα πραγματικό σύνολο δεδομένων που αποτελείται από GPS τροχιές από χρήστες οι οποίοι κινούνται στο Τόκυο της Ιαπωνίας κατά την διάρκεια των ετών 2007 – 2008 και από μια συλλογή από φωτογραφίες οι οποίες χαρακτηρίζονται από σύντομες περιγραφές. Συγκεκριμένα, το σύνολο δεδομένων αποτελείται από 4179 φωτογραφίες από αξιοθέατα και εκδηλώσεις στην Ιαπωνία καθώς επίσης και τροχιές με διακριτότητα του ενός δείγματος κάθε 5 δευτερόλεπτα ή κάθε 10

δευτερόλεπτα. Η παραδοχή που κάναμε στο συγκεκριμένο σύνολο δεδομένων είναι οτι έχουμε εικόνες από 75 διαφορετικές ημέρες τις οποίες θεωρήσαμε ότι αναπαριστούν 75 χρήστες οι οποίοι κινούνται στην Ιαπωνία την ίδια ημέρα. Κάθε χρήστης έχει ένα τυχαίο αριθμό φακέλων όπου σε κάθε φάκελο βρίσκετε και ένα σύνολο από φωτογραφίες.

5.1.6 Συσκευές που Χρησιμοποιήθηκαν

Για την εκτέλεση των πειραμάτων έχουν χρησιμοποιηθεί 10 συσκευές HTC Desire (εικόνα 5.1.6.1). Κάθε έξυπνη κινητή συσκευή μπορεί να υποστηρίξει περίπου 100 παράλληλα νήματα. Έτσι μπορούμε να χρησιμοποιήσουμε 100 ιδεατούς χρήστες στο δίκτυο μας.

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

Εικόνα 5.1.6.1: Android συσκευή

5.2 Πειραματική Διαδικασία

Καταρχάς έχει γραφτεί ο κώδικας για το πειραματικό εξυπηρετητή και την πειραματική εφαρμογή σε Android. Στη συνέχεια ελέγξαμε τον κώδικα αρκετές φορές για να βεβαιωθούμε για την ορθότητα του και εκτέλεση του στις συσκευές.

Στη συνέχεια προσαρμόσαμε τα σύνολα δεδομένων βάση των παραδοχών που κάναμε πιο πάνω και τα καταχωρήσαμε σε μια βάση δεδομένων. Συγκεκριμένα με τα σύνολα δεδομένων DBLP και GeoLife δημιουργήσαμε ένα καινούριο σύνολο δεδομένων το οποίο είναι συνδυασμός αυτών των δύο. Στο σύνολο δεδομένων GeoLife υπάρχουν 1100 τροχιές τις οποίες αντιστοιχίσαμε με τους 1100 πρώτους χρήστες του DBLP οι οποίοι έχουν τα πιο πολλά άρθρα. Ακολούθως, αφού τοποθετήσαμε το καινούριο σύνολο δεδομένων στη βάση δεδομένων εκτελούσαμε ερωτήματα στη βάση δεδομένων για να δημιουργήσουμε σενάρια για διαφορετικές ώρες της μέρας όπου το δίκτυο θα είχε διαφορετικό αριθμό ενεργών χρηστών. Επίσης, σε κάθε χρονική στιγμή οι γεωγραφικές συντεταγμένες κάθε χρήστη αλλάζουν αφού κάθε χρήστης κινείτε σε μία τροχιά. Έτσι, δημιουργήσαμε πέντε σενάρια όπως παρουσιάζονται στον πιο κάτω πίνακα (εικόνα 5.2.1) για το ερώτημα της λέξης από κάποιο χρήστη “optimization”:

Σενάριο	Χρονική στιγμή	Μέγεθος δικτύου	Δεδομένα Χρηστών	Δεδομένων που Απαντούν “optimization”
Σ1	Πρωί	20	2015	19
Σ2	Πρωί	50	3245	36
Σ3	Πρωί	80	5325	53
Σ4	Πρωί	95	6439	101
Σ5	Πρωί	138	9583	113

Εικόνα 5.2.1: Πίνακας σεναρίων για σύνολο δεδομένων DBLP και GeoLife

Το κάθε σενάριο έχει τα δικά του δεδομένα τα οποία εισάγαμε στη κινητή συσκευή για να προσομοιώσουμε τα πειράματα.

Με παρόμοιο τρόπο εισάγαμε τα δεδομένα του συνόλου Pics-n-Trails σε μια βάση δεδομένων σε συνδυασμό με την παραδοχή που κάναμε στην ενότητα 5.1.5.3. Ακολούθως, εκτελούσαμε ερωτήματα στη βάση δεδομένων για να δημιουργήσουμε σενάρια για διαφορετικές ώρες της μέρας όπου το δίκτυο θα είχε διαφορετικό αριθμό

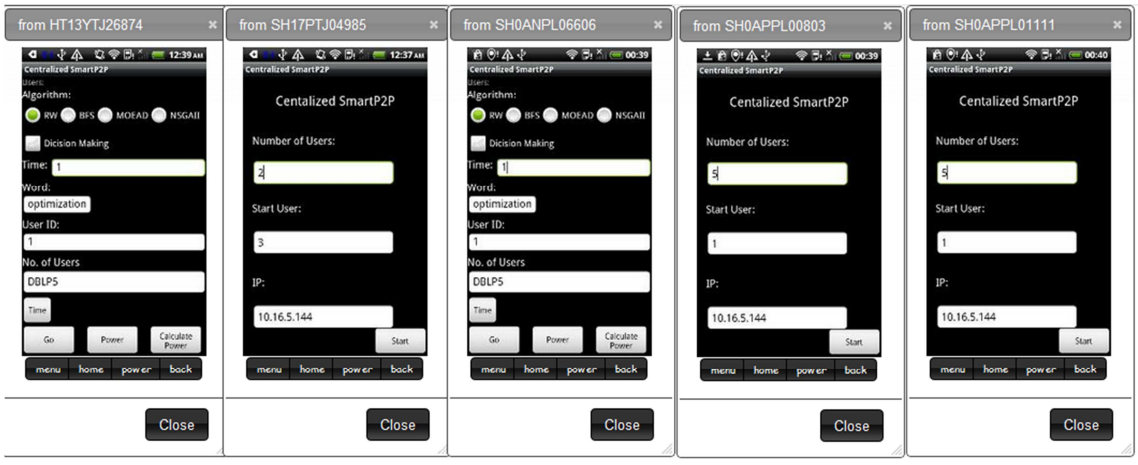
ενεργών χρηστών (εικόνα 5.2.2). Επίσης, σε κάθε χρονική στιγμή οι γεωγραφικές συντεταγμένες κάθε χρήστη αλλάζουν αφού κάθε χρήστης κινείτε σε μία τροχιά. Έτσι, δημιουργήσαμε τρία σενάρια όπως παρουσιάζονται στον πιο κάτω πίνακα για το ερώτημα της λέξης από κάποιο χρήστη “kyoto”:

Σενάριο	Χρονική στιγμή	Μέγεθος δικτύου	Δεδομένα Χρηστών	Δεδομένων που Απαντούν “kyoto”
Σ1	Πρωί	20	1954	30
Σ2	Πρωί	35	3784	60
Σ3	Πρωί	50	5049	81
Σ4	Πρωί	61	7085	99
Σ5	Πρωί	75	8167	128

Εικόνα 5.2.2: Πίνακας σεναρίων για σύνολο δεδομένων Pics-n-Trails

Εκτελώντας όλα τα πιο πάνω σενάρια στις κινητές συσκευές όπως φαίνεται στην εικόνα 5.2.3 στο SmartLab θα αποθηκευτεί σε κάθε συσκευή στο αρχείο “filename.time.txt” ο χρόνος απόκρισης των αποτελεσμάτων όπως αναφέραμε στην ενότητα 5.1.4 και στο αρχείο “filename.power.txt” η κατανάλωση της ενέργειας της κινητής συσκευής.

Στην συνέχεια θα οργανώσουμε τα όλα τα πιο πάνω αποτελέσματα για την δημιουργεί των γραφικών παραστάσεων όπου θα μας βοηθήσουν να εξάγουμε τα δικά μας συμπεράσματα.



Εικόνα 5.2.3: Εκτέλεση πειραμάτων στο SmartLab

Κεφάλαιο 6

Πειραματικά Αποτελέσματα

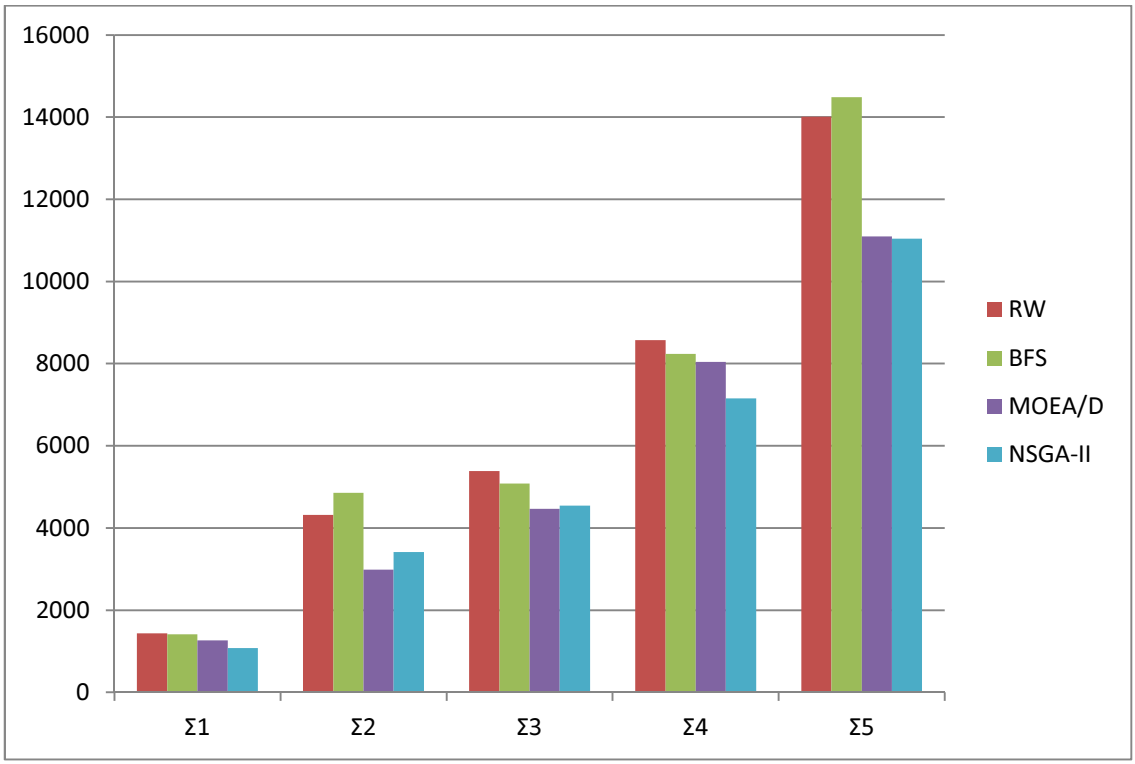
6.1 Εισαγωγή
6.2 Πειραματικά Αποτελέσματα για χρόνο απόκρισης
6.3 Πειραματικά Αποτελέσματα για ενέργεια
6.4 Πειραματικά Αποτελέσματα για χρόνο απόκρισης
6.4.1 Σύνολο Δεδομένων DBLP και GeoLife
6.4.2 Σύνολο Δεδομένων Pics-n-Trails

6.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα παρουσιάσουμε τα πειραματικά αποτελέσματα τα οποία προέκυψαν από τα σενάρια που τρέξαμε. Στόχος μας είναι να αξιολογήσουμε στο σύστημα τις τεχνικές που αναπτύχθηκαν αν είναι αποδοτικές ως προς την κατανάλωση της ενέργειας, το χρόνο απόκρισης των αποτελεσμάτων που φτάνουν στο χρήστη που υποβάλλει το ερώτημα καθώς επίσης και ως προς το ρυθμό ανάκλησης (αριθμός δεδομένων που φτάνουν στο χρήστη που υποβάλλει το ερώτημα σε σχέση με το συνολικό αριθμό αποτελεσμάτων που έχει όλο το δίκτυο για το συγκεκριμένο ερώτημα). Η πειραματική μελέτη έγινε πάνω σε 10 έξυπνες κινητές συσκευές HTC Desire με πολλαπλά νήματα. Για να έχουμε πιο ακριβή αποτελέσματα έχουμε επαναλάβει τα πειράματα τέσσερεις (4) φορές.

6.2 Πειραματικά Αποτελέσματα για χρόνο απόκρισης

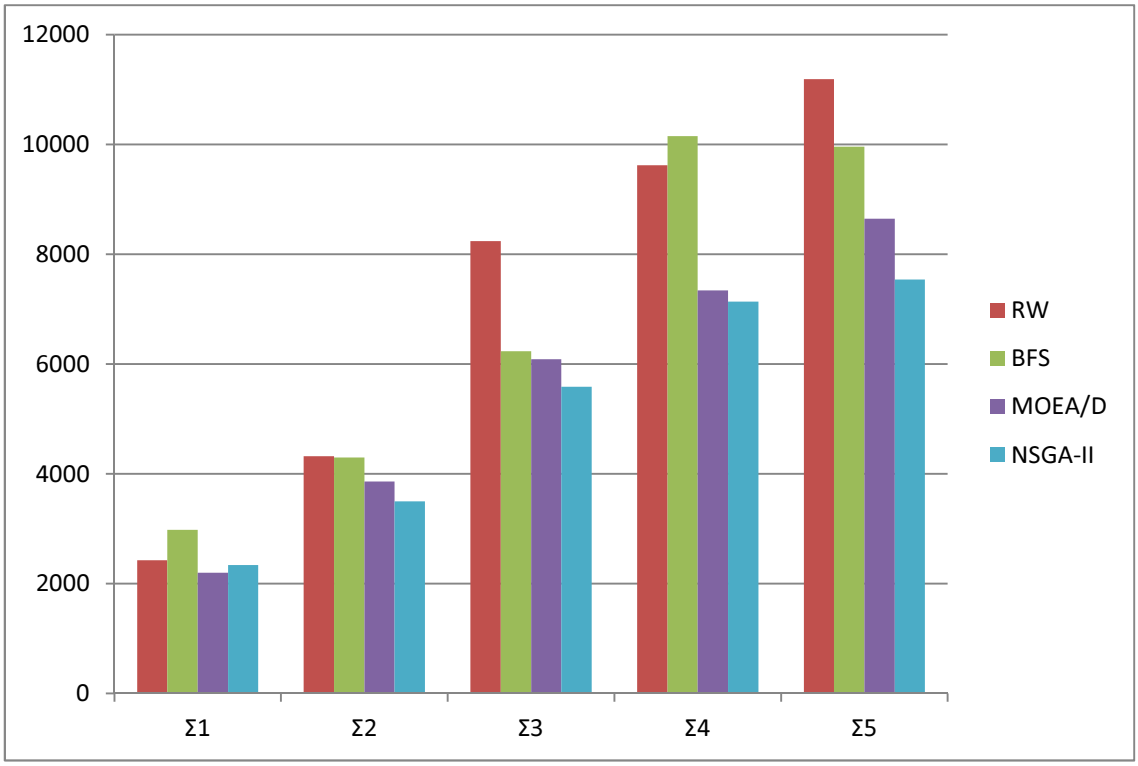
- **Σύνολο Δεδομένων DBLP και GeoLife**



Εικόνα 6.2.1: Γραφική παράσταση του χρόνου απόκρισης για κάθε σενάριο για σύνολο δεδομένων DBLP και GeoLife

Η πιο πάνω γραφική παράσταση (εικόνα 6.2.1) παρουσιάζει το χρόνο που χρειάζονται να φτάσουν τα αποτελέσματα στο χρήστη ο οποίος υποβάλλει το ερώτημα στο δίκτυο για το σύνολο δεδομένων DBLP και GeoLife. Για κάθε σενάριο τρέξαμε το πείραμα και για τις τέσσερις (4) τεχνικές (MOEAD, NSGA-II, RW, BFS).

- **Σύνολο Δεδομένων Pics-n-Trails**



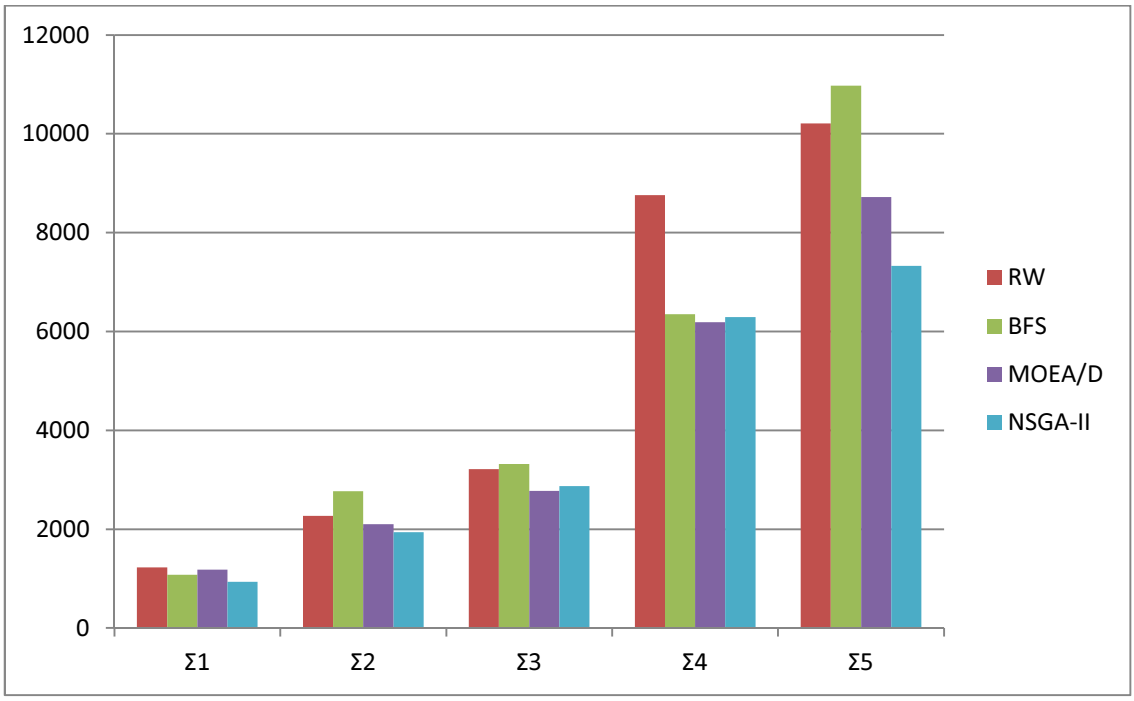
Εικόνα 6.2.2: Γραφική παράσταση του χρόνου απόκρισης για κάθε σενάριο για σύνολο δεδομένων Pics-n-trails

Η πιο πάνω γραφική παράσταση (εικόνα 6.2.2) παρουσιάζει το χρόνο που χρειάζονται να φτάσουν τα αποτελέσματα στο χρήστη ο οποίος υποβάλλει το ερώτημα στο δίκτυο για το σύνολο δεδομένων Pics-n-trails. Για κάθε σενάριο τρέξαμε το πείραμα και για τις τέσσερις (4) τεχνικές (MOEAD, NSGA-II, RW, BFS).

Από τις πιο πάνω γραφικές παραστάσεις (εικόνα 6.2.1 και εικόνα 6.2.2) μπορούμε να παρατηρήσουμε ότι οι τεχνικές πολυκριτηρίων (MOEA/D, NSGA-II) έχουν καλύτερες επιδόσεις στο χρόνο απόκρισης σε σχέση με τις απλές τεχνικές (BFS, RW). Αυτό οφείλεται στο γεγονός ότι στις τεχνικές πολυκριτιρίων δεν συμπεριλαμβάνονται όλοι οι κόμβοι μέσα στο δέντρο δρομολόγησης αν δεν χρειάζονται και έτσι το ερώτημα θα περάσει από πιο λίγους κόμβους μέσα στο δίκτυο. Στην αντίθετη περίπτωση το ερώτημα θα περάσει από όλους τους κόμβους αφού και οι δύο τεχνικές (RW, BFS) συμπεριλαμβάνουν όλους τους ενεργούς χρήστες του δικτύου μέσα στο δέντρο δρομολόγησης.

6.3 Πειραματικά Αποτελέσματα για ενέργεια

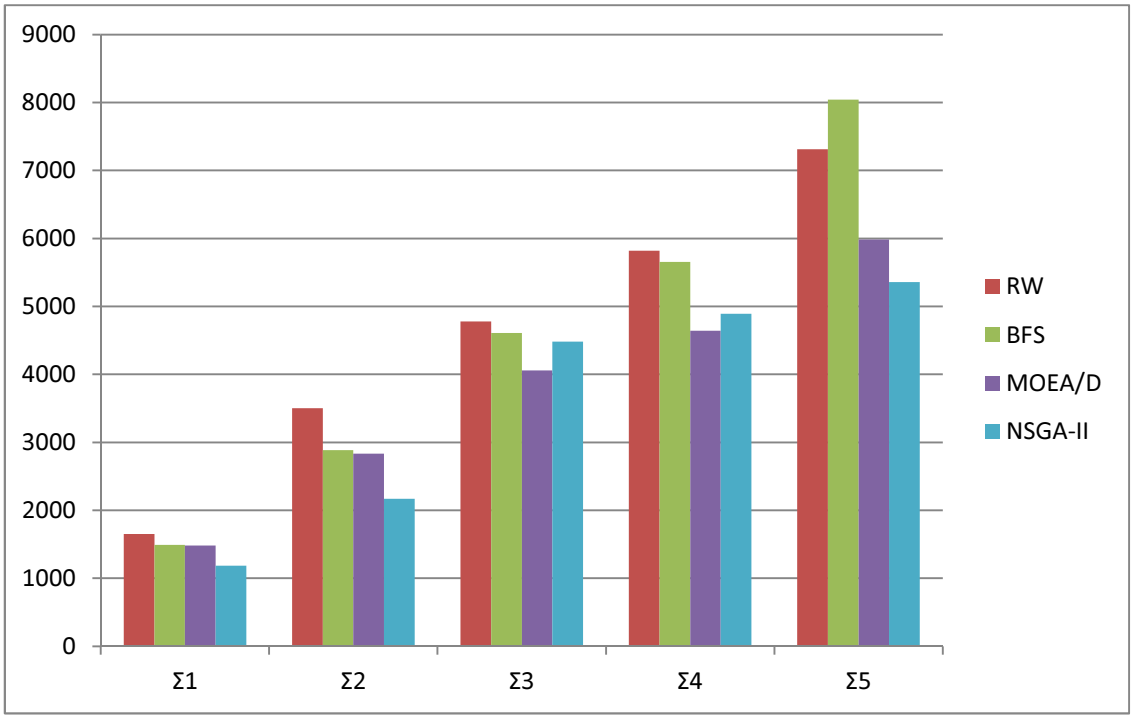
- **Σύνολο Δεδομένων DBLP και GeoLife**



Εικόνα 6.3.1: Γραφική παράσταση της κατανάλωσης ενέργειας για κάθε σενάριο για σύνολο δεδομένων DBLP και GeoLife

Η πιο πάνω γραφική παράσταση (εικόνα 6.3.1) παρουσιάζει την ενέργεια που καταναλώνουν όλοι οι ενεργοί χρήστες σε κάθε σενάριο για το σύνολο δεδομένων DBLP και GeoLife. Για κάθε σενάριο τρέξαμε το πείραμα και για τις τέσσερις (4) τεχνικές (MOEAD, NSGA-II, RW, BFS).

- **Σύνολο Δεδομένων Pics-n-Trails**



Εικόνα 6.3.2: Γραφική παράσταση της κατανάλωσης ενέργειας για κάθε σενάριο για σύνολο δεδομένων Pics-n-trails

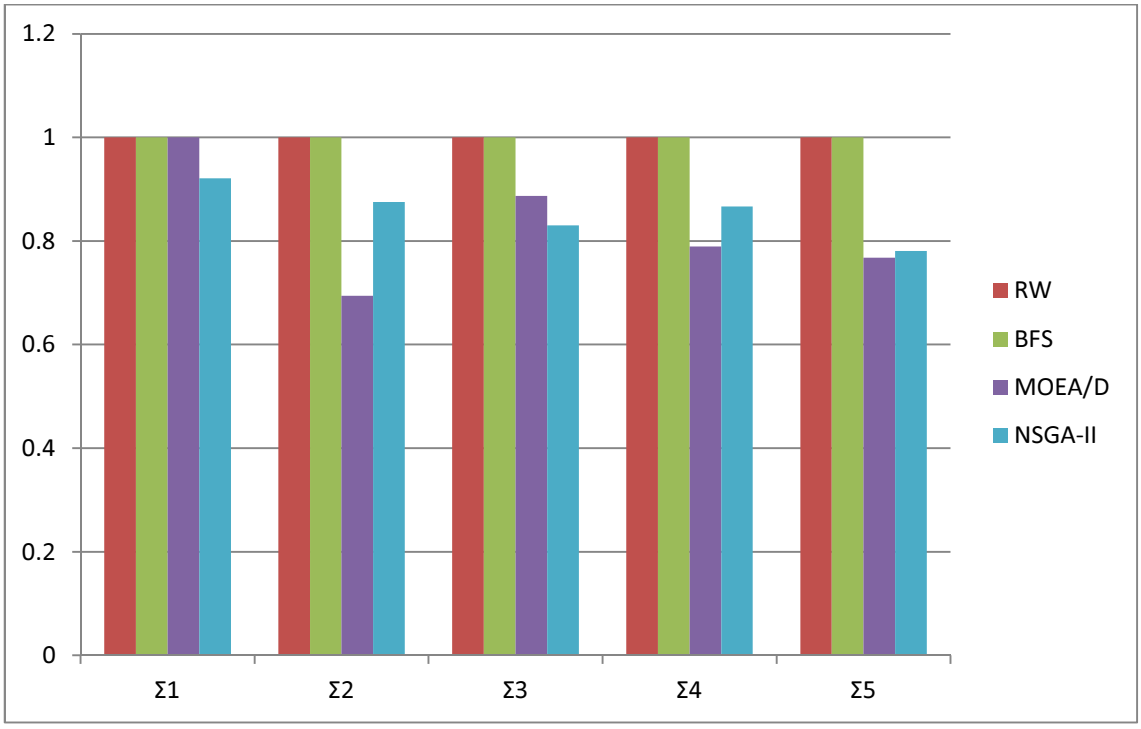
Η πιο πάνω γραφική παράσταση (εικόνα 6.3.2) παρουσιάζει την ενέργεια που καταναλώνουν όλοι οι ενεργοί χρήστες σε κάθε σενάριο για το σύνολο δεδομένων Pics-n-trails. Για κάθε σενάριο τρέξαμε το πείραμα και για τις τέσσερις (4) τεχνικές (MOEAD, NSGA-II, RW, BFS).

Από τις πιο πάνω γραφικές παραστάσεις (εικόνα 6.3.1 και εικόνα 6.3.2) παρατηρούμε οτι οι τεχνικές πολυκριτηρίων (MOEA/D, NSGA-II) έχουν καλύτερες επιδόσεις στην συνολική κατανάλωση της ενέργειας από όλους τους ενεργούς χρήστες του δικτύου σε σχέση με τις απλές τεχνικές (BFS, RW). Αυτό οφείλεται στο γεγονός ότι στις τεχνικές πολυκριτιρίων δεν συμπεριλαμβάνονται όλοι οι ενεργοί χρήστες μέσα στο δέντρο δρομολόγησης αν δεν χρειάζονται και έτσι το ερώτημα θα περάσει από πιο λίγους κόμβους μέσα στο δίκτυο. Στην αντίθετη περίπτωση το ερώτημα θα περάσει από όλους τους κόμβους αφού και οι δύο τεχνικές (RW, BFS) συμπεριλαμβάνουν όλους τους ενεργούς χρήστες του δικτύου μέσα στο δέντρο δρομολόγησης. Επίσης παρατηρούμε

ότι με την αύξηση των χρηστών του δικτύου πετυχαίνεται μεγαλύτερη μείωση στην συνολική κατανάλωση της ενέργειας.

6.4 Πειραματικά Αποτελέσματα για ρυθμό ανάκλησης

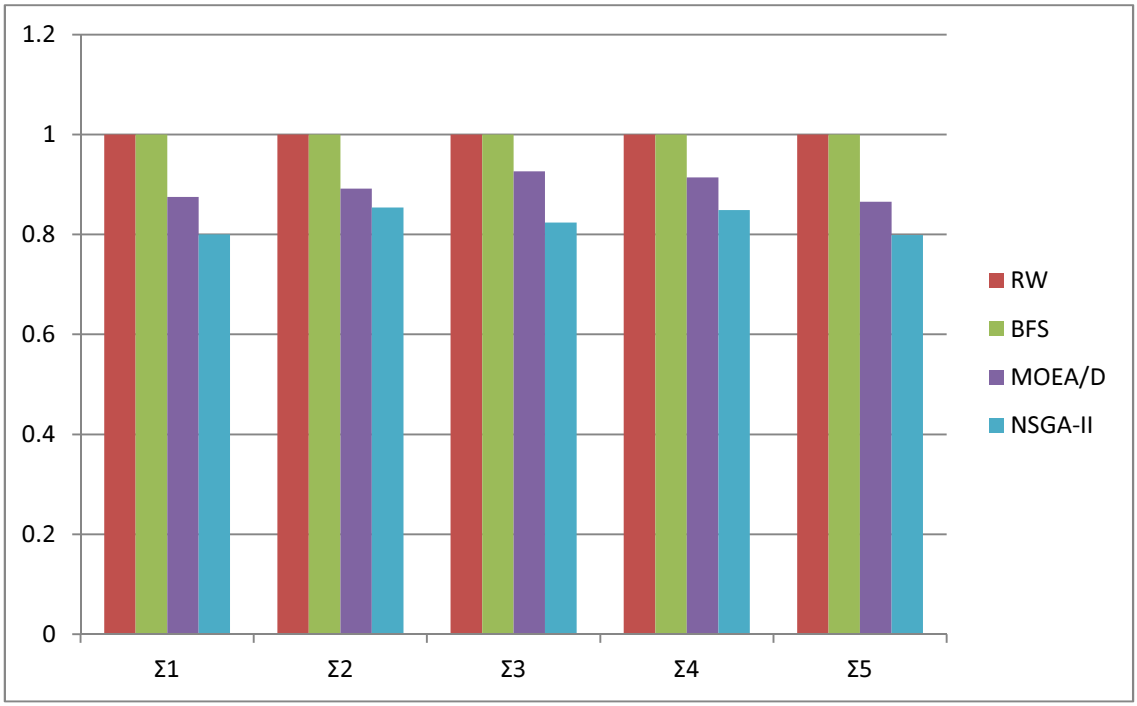
- **Σύνολο Δεδομένων DBLP και GeoLife**



Εικόνα 6.4.1: Γραφική παράσταση του ρυθμού ανάκλησης για κάθε σενάριο για σύνολο δεδομένων DBLP και GeoLife

Η πιο πάνω γραφική παράσταση (εικόνα 6.4.1) παρουσιάζει το ποσοστό των απαντήσεων που φτάνουν στο χρήστη ο οποίος υπόβαλε το ερώτημα στους υπόλοιπους ενεργούς χρήστες του δικτύου για το σύνολο δεδομένων DBLP και GeoLife. Για κάθε σενάριο τρέξαμε το πείραμα και για τις τέσσερις (4) τεχνικές (MOEAD, NSGA-II, RW, BFS).

- **Σύνολο Δεδομένων Pics-n-Trails**



Εικόνα 6.4.2: Γραφική παράσταση του ρυθμού ανάκλησης για κάθε σενάριο για σύνολο δεδομένων Pics-n-trails

Η πιο πάνω γραφική παράσταση (εικόνα 6.4.2) παρουσιάζει το ποσοστό των απαντήσεων που φτάνουν στο χρήστη ο οποίος υπόβαλε το ερώτημα στους υπόλοιπους ενεργούς χρήστες του δικτύου για το σύνολο δεδομένων Pics-n-trails. Για κάθε σενάριο τρέξαμε το πείραμα και για τις τέσσερις (4) τεχνικές (MOEAD, NSGA-II, RW, BFS).

Από τις πιο πάνω γραφικές παραστάσεις (εικόνα 6.4.1 και εικόνα 6.4.2) μπορούμε να παρατηρήσουμε ότι περίπου το 80% των συνολικών απαντήσεων που διαθέτουν όλοι οι ενεργοί χρήστες φτάνει στον χρήστη που υπόβαλε το ερώτημα για τις τεχνικές πολυκριτηρίων (MOEAD, NSGA-II). Αυτό είναι λογικό αφού στις τεχνικές αυτές μπορεί να μην συμπεριλαμβάνουν κάποιους χρήστες που διαθέτουν απαντήσεις στο ερώτημα για λόγους καλύτερης επίδοσης ως προς την ενέργεια και χρόνο. Στην αντίθετη περίπτωση στις υπόλοιπες τεχνικές (RW, BFS) παρατηρείται να φτάνουν όλες οι απαντήσεις στο χρήστη που υπόβαλε το ερώτημα λόγω του ότι το ερώτημα θα περάσει από όλους τους ενεργούς χρήστες του δικτύου. Επίσης, για τις εκτελέσεις των πειραμάτων για το σύνολο δεδομένων Pics-n-trails όπου πίσω από κάθε περιγραφή

υπάρχει ένα σύνολο φωτογραφιών παρατηρούμε ότι για την τεχνική MOEA/D φτάνουν περισσότερα δεδομένα στον χρήστη που υπόβαλε το ερώτημα σε σχέση με την τεχνική NSGA-II. Για το σύνολο δεδομένων DBLP και GeoLife όπου έχουμε την ίδια ποσότητα δεδομένων κάτω από κάθε τίτλο δεν μπορούμε να κάνουμε την ίδια διαπίστωση αφού σε κάποια σενάρια έχει καλύτερες επιδώσεις η τεχνική MOEA/D και σε άλλα η τεχνική NSGA-II.

Κεφάλαιο 7

Συμπεράσματα και Μελλοντικές Επεκτάσεις

7.1 Συμπεράσματα
7.2 Μελλοντικές Επεκτάσεις

7.1 Συμπεράσματα

Στόχος της παρούσας διπλωματικής εργασίας ήταν η δημιουργία ενός συστήματος αναζήτησης δεδομένων σε ένα δίκτυο ομότιμων (p2p) για έξυπνες κινητές συσκευές το οποίο υλοποιεί κάποιες τεχνικές για αποδοτική αναζήτηση και ασφάλεια δεδομένων. Η ανάγκη αυτή προήλθε από την ραγδαία ανάπτυξη των έξυπνων κινητών συσκευών τα οποία παρατηρήθηκαν να υστερούν σε θέματα ενέργειας και από την ραγδαία ανάπτυξη των κοινωνικών δικτύων του οποίου οι χρήστες χρειάζονται ένα ασφαλή τρόπο για την ανταλλαγή δεδομένων αλλά και αναζήτηση αυτών στους υπόλοιπους χρήστες.

Έτσι υλοποιήθηκε η εφαρμογή SmartP2P η οποία υλοποιούσε δύο απλές τεχνικές αναζήτησης και δύο τεχνικές πολυκριτηρίων. Ο τρόπος λειτουργίας του συστήματος διασφάλιζε την προστασία των δεδομένων των χρηστών αφού οι χρήστες δεν ήταν υποχρεωμένοι να στέλνουν τα δεδομένα τους στον εξυπηρετητή του συστήματος αλλά μόνο μια σύντομη περιγραφή που τα χαρακτήριζε. Με αυτό τον τρόπο καλύφθηκε το κομμάτι της ασφάλειας των δεδομένων των χρηστών.

Στη συνέχεια η αποδοτική αναζήτηση διαπιστώθηκε μέσα από μια σειρά πειραμάτων που πραγματοποιήθηκαν για την αξιολόγηση των τεχνικών πολυκριτηρίων. Σύμφωνα

με τα αποτελέσματα που πήραμε από την πειραματική διαδικασία καταλήξαμε στο συμπέρασμα ότι με τις τεχνικές πολυκριτηρίων μπορούμε να πετύχουμε μια αποδοτική αναζήτηση αφού οι τεχνικές αυτές προσφέρουν καλύτερες επιδόσεις τόσο στο χρόνο απόκρισης των αποτελεσμάτων όσο και στην ενεργειακή απόδοση των κινητών συσκευών.

Συνεπώς, οι τεχνικές πολυκριτηρίων συνέβαλαν στην αδυναμία της κατανάλωσης της ενέργειας από τις έξυπνες κινητές συσκευές που είχε παρουσιαστεί κατά την αναζήτηση δεδομένων στους υπόλοιπους χρήστες του δικτύου. Επίσης ένα άλλο συμπέρασμα που διαπιστώθηκε κατά την πειραματική διαδικασία ήταν ότι με την αύξηση των χρηστών του δικτύου το ποσοστό εξοικονόμησης ενέργειας από τις κινητές συσκευές μεγάλωνε.

7.2 Μελλοντικές Επεκτάσεις

Το σύστημα που υλοποιήθηκε θα μπορούσε να επεκταθεί έτσι ώστε να είναι εντελώς κατανεμημένη. Δηλαδή, να απομακρυνθεί ο εξυπηρετητής ο οποίος εκτελεί τους αλγορίθμους (MOEA/D, NSGA-II, BFS και RW) και να δημιουργηθεί ένα αποκεντρικοποιημένο σύστημα ομότιμων δικτύων (p2p). Έτσι θα μπορέσουμε να συγκρίνουμε την απόδοση των δύο συστημάτων για τις τεχνικές που υλοποιήθηκαν.

Βιβλιογραφία

[1] Android, http://developer.android.com/index.html.

[2] Azizyan M.,Constandache I., Choudhury R. –R., "SurroundSense: mobile phone localization via ambience fingerprinting," In MobiCom'09

[3] Balke, W, -T, Gntzer, U., and Zheng, J. X. 2004, Efficient distributed skylining for web information systems. In IN EDBT. 256-273.

[4] CAMBELL,A.,EISENMAN, S., LANE, N.,MILUZZO, E., PETERSON. R.,H. X.Z., MUSOLESI, M., FODOR, K., AND AHN, G.2008. The rise of people –centric sensing. In IEEE Internet Computing 12,4,12-21.

[5] Campbell A., Eisenman S., Lane N., Miluzzo E., and Peterson R., "People-centric urban sensing," In *WICON*, 2006.

[6] CHEN, S.-K.AND WANG, P.-C, 2011. Design and implementation of an anycast services discovery in mobile ad hoc networks. ACM Transactions on Autonomouw and Adaptive Systems (TAAS) 6,1,2.

[7] Chomicki, J., Godfrey, P., Gryz, J., and Liang, D. 2005. Skyline with presorting: Theory and optimization. In In Int. Inf. Sys. Conference IIS. Springer, 593-602

[8] DBLP.2010.DBLP Computer Science Bibliography, http://dblp.uni-trier.de/xml/.

[9] Deb K., Pratap a., Agarwal S., Meyarivan T., "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA II," *IEEE TEvC*, 2002

[10] EISENMAN, S., MILUZZO, E., LANE, N., PETERSON, R., SEOP-AHN.G., AND CAMPBELL, A. 2009. Bikenet: A mobile sensing system

for cyclist experience mapping. ACM Transactions on Sensor Networks (TOSN'09) 6,1.

[11] GAHNG-SEOP, A., MUSOLESI, M., LU, H., OLFATI-SABER, R., AND CAMPBELL, A. 2010. Metrotrack: Predictive tracking of mobile events using mobile phones. In DCOSS, 230-243.

[12] GNUTELLA, 2000. Gnutella peer-to-peer network, http://gnutella.wego.com.

[13] Godfrey, P., Shipley, R., and Gryz, J. 2005, Maximal vector computation in large data sets. In proceedings of the 31st international conference on very large data bases. VLDB '05. VLDB Endowment, 229-240.

[14] Google, http://www.google.com.cy/about/.

[15] Google Latitude, 11/2010, http://www.google.com/latitude

[16] G. C. de Silva, T. Yamasaki, K. Aizawa, "Sketch-Based Spatial Queries for Retrieving Human Locomotion Patterns From Continuously Archived GPS Data," IEEE Trans. on Multimedia, vol.11, no.7, 2009.

[17] G. C. de Silva, K. Aizawa, "Retrieving Multimedia Travel Stories Using Location Data and Spatial Queries", In Proc. The 17th ACM International Conference on Multimedia, pp. 785-788, October 2009.

[18] Huang, Z., Jensen, C. S., Lu, H., and Ooi, B. C. 2006. Skyline queries against mobile lightweight devices in manets. In In Proc. of ICDE.

[19] JIA, J.,CHEN, J., CHANG, G., WEN, Y., AND SONG, J.2009. Multi-objective optimization for coverage control in wireless sensor network with adjustable sensing radius. Computers and Mathematics with Applications 57,11-12, 1767-1775.

[20] KALOGERAKI, V., GUNOPULOS, D., AND ZEINALIPOUR-YAZTI, D. 2002. A local search mechanism of peer-to-peer networks. In 11th International Conference on Information and Knowledge Management (CIKM'02). McLean, Virginia, USA, 300-307.

[21] Ko, Y,-B AND VAIDYA, N.H. 2000. Location –aided routing (lar) in mobile ad hoc networks. Wirel. Netw. 6,4,307-321.

[22] KONSTANTINIDIS, A. AND YANG, K.2011. Multi-objective energy-efficient dense deployment in wireless sensor networks using a hybrid problem-specific MOEA/D. Applied Soft Computing In Press.

[23] KONSTANTINIDIS, A., YANG, K.,ZHANG,Q., AND ZEINALIPOUR-YATZTI, D. 2010. A multi-objective evolutionary algorithm for the deployment and power assignment problem in wireless sensor networks. New Network Paradigms, Elsevier Computer Networkw 54, 960-976.

[24] KONSTANTINIDIS, A., ZEINALIPOUR-YAZTI, D., ANDREOU, P., AND SAMARAS, G. 2011. Multi-objective query optimization in smartphone social networks. In 12th International Conference in Mobile Data Management.

[25] Kossmann, D., Ramsak, F., and Rost, S. 2002. Shooting stars in the sky: an online algorithm for skyline queries. In In VLDB. 275-286.

[26] LV, Q., CAO, P., COHEN, E., LI, K., AND SHENKER, S. 2002A. Search and replication in unstructed peer-to-peer networks. In 16th international conference on Supercomputing (ICS'02. New York, USA, 84-95.

[27] LV, Q., CAO, P., COHEN, E., LI, K., AND SHENKER, S. 2002b. Search and replication in unstructured peer-to-peer networks. In ICS (2002-12-17). 84-95.

[28] MILLER, B, L., MILLER, B.L., GOLDBERG, D.E., AND GOLDBERG, D.E. 1995. Genetic algorithms, tournament selection, and the effects of noise. Complex Systems 9, 193-212.

[29] NG, W,S., OOI, B. C., TAN, K,-L., AND ZHOU, A. 2003. Peerdb: A. P2p-based system for distributed data sharing. Data Engineering, Internnational Conference on 0, 633.

[30] Papadias, D., Tao, Y., Fu, G., and Seeger, B. 2003. An optimal and progressive algorithm for skyline queries. In Proceedings of the 2003 ACM SIGMOD international conference of Management of data. SIGMOD'03. ACM, New York, NY, USA, 467-478.

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

[32] RA, M.-R.,PAEK, J., SHARMA, A., GOVINDAN, R., KRIEGER, M,H., AND NEELY, M.J., AND NEELY, M.J. 2010. Energy-delay tradeoffs in smartphone applications. In MobiSys. 255-270.

[33] RAJAGOPALAN, R., MOHAN, C., MEHROTRA, K. G., AND VARSHNEY, P. K. 2006. Emoca: An evolutionary multi-objective crowding algorithm. Journal of Intelligent Systems.

[34] RAJAGOPALAN, R., MOHAN, C. K., VARSHNEY, P.K., AND MEHROTRA, K. 2005. Multi-objective mobile agent routing in wireless senson networks. In Poc. IEEE CEC'05. Edinburgh, Scotland.

[35] REPANTIS, T. AND KALOGERAKI, V. 2005. Data dissemination in mobile peer-to peer networks. In 6th International Conference on Mobile Data Management (MDM'05). Ayia Napa, Cyprus, 211-219.

[36] Sql Server 2008 R2 express, http://www.microsoft.com/sqlserver/en/us/editions/2012-editions/express.aspx

[37] SmartLab, University of Cyprus, Department of Computer Science, http://smartlab.cs.ucy.ac.cy/

[38] TOMIYASU, H., MAEKAWA, T., HARA, T., AND NISHIO, S. 2006. Profile-based query routing in a mobile social network. In Mobile Data Management, 2006. MDM 2006, 7th International Conference on. 105-105.

[39] Tan, K. –L., Eng, P. –K, and Ooi, B. C. 2001. Efficient progressive skyline computation. In Proceedings of the 27th International Conference on Very Large Data Bases. VLDB '01. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 301-310

[40] TSOUMAKOS, D. AND ROUSSOPOULOS, N. 2003. Adaptive probabilistic search for peer-to-peer networks. In Third International Conference on Peer-to-Peer Computing (P2P'03). 102-109.

[41] VLACHOU, A., DOULKERIDIS, C., KOTIDIS, Y., AND VAZIRGIANNIS, M. 2007. Skypeer: Efficient subspace skyline computation over distributed data. International Conference on Data Engineering. 416-425.

[42] WANG, S., OOI, B.C., AND TUNG, A. K. H. 2007. Efficient skyline query processing on peer-to-peer networks. In In IEEE International Conference on Data Engineering (ICDE) (2007. 1126-1135).

[43] WU.P, ZHANG, C., FENG, Y., ZHAO, B.Y., AGRAWAL, D., AND ABBADI, A. E. 2006. Parallelizing skyline queries for scalable distribution. In In EDBT06. 112-130.

[44] XU, B., WOLFSON, O., AND NAIMAN, C. 2009. Machine learning in disruption-tolerant manets. ACM Transactions on Autonomous and Adaptive Systems (TAAS) 4,4.

[45] Zhang Q., Li H., "MOEA/D: A Multi-objective Evolutionary Algorithm Based on Decomposition," IEEE TEvC, 2007.

[46] Zhu, L., Tao, Y., and Zhou, S. 2009. Distributed skyline retrieval with low bandwidth consumption. IEEE Trans. On Knowl. And Data Eng.21, 384-400

[47] ZEINALIPOUR-YAZTI, D., KALOGERAKI, V., AND GUNOPULOS, D. 2005. Exploiting locality for scalable information retrieval in peer-to-peer systems. Information Systems (InfoSys). Elsevier 30,4, 277-298.

[48] ZEINALIPOUR-YAZTI, D., KALOGERAKI, V., AND GUNOPULOS, D. 2007. Pfusion: An architecture for internet-scale content-based search and retrieval. IEEE TPDS 18, 6, 804-807.

[49] Zheng Y., Liu L., Wang L., Xie X., "Learning transportation mode from raw gps data for geographic applications on the web," In *WWW'08*, 2008

[50] ZITZLER, E. AND THIELE, L. 1998. Multiobjective optimization using evolutionary algorithms – a comparative case study. Springer, 292-301.

[51] ZITZLER, E, AND THIELE, L. 1999. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Trans. Evolutionary Computation 257-271.

Παράρτημα Α

Παρουσίαση Κώδικα MonkeyRunner για την Εφαρμογή SmartP2P

Εισαγωγή
Κώδικας MonkeyRunner

Εισαγωγή

Το εργαλείο monkeyrunner παρέχει ένα API για τη συγγραφή προγραμμάτων που ελέγχουν μια συσκευή Android ή εξομοιωτή (emulator) εκτός του Android κώδικα. Για τη δημιουργία ενός monkeyrunner μπορεί να γραφτεί ένα πρόγραμμα σε Python που εγκαθιστά μια εφαρμογή Android ή ελέγχει πακέτα τα οποία μπορεί να τρέξει, στέλνει πληκτρολογήσεις στην κινητή συσκευή, παίρνει στιγμιότυπα της διεπαφής, και αποθηκεύει εικόνες στο σταθμό εργασίας. Το εργαλείο monkeyrunner έχει σχεδιαστεί κυρίως για να δοκιμάζονται εφαρμογές και συσκευές στο λειτουργικό τους κομμάτι αλλά υπάρχει και η δυνατότητα να χρησιμοποιηθούν και για άλλους σκοπούς.

Κώδικας MonkeyRunner

```
#! /usr/bin/env monkeyrunner
from com.android.monkeyrunner import MonkeyRunner, MonkeyDevice
import time
import os
import sys
def select_tree (device, recall):
    device.touch(30, 690, device.DOWN_AND_UP)
```



```
if recall == '100':
        device.drag((60, 550), (409, 550), 1.0, 120)
    if int (recall) < 100:
        s=61+35*int(recall)/10
        device.touch(s, 550, device.DOWN_AND_UP)

def select_alg (device, alg, word):

    MonkeyRunner.sleep(0.85)
    #write search word
    device.touch(250, 240, device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.type(word)

    device.touch(20, 790, device.DOWN_AND_UP)
    MonkeyRunner.sleep(0.55)

        #select algorithm
    if alg == 'RW':
        device.touch(80, 330, device.DOWN_AND_UP) # (y) RW: 330 BFS: 417
MOEAD: 502, NSGAII: 570
    if alg == 'BFS':
```

```
        device.touch(80,   417,   device.DOWN_AND_UP)   #   (y)   RW:   330   BFS:   417
MOEAD: 502, NSGAII: 570
    if alg == 'MOEAD':
        device.touch(80,   502,   device.DOWN_AND_UP)   #   (y)   RW:   330   BFS:   417
MOEAD: 502, NSGAII: 570
    if alg == 'NSGAII':
        device.touch(80,   570,   device.DOWN_AND_UP)   #   (y)   RW:   330   BFS:   417
MOEAD: 502, NSGAII: 570
            #press go button
    device.touch(170, 667, device.DOWN_AND_UP)



def write (device, ip, port, id, pass1):
    #write ip
    device.touch(274, 95, device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.type(ip)

    #write port
    device.touch(274, 225, device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
```



```
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.type(port)

    #write user id
    device.touch(274, 350, device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.type(id)

    device.touch(20, 790, device.DOWN_AND_UP)
    MonkeyRunner.sleep(0.55)

    #write password
    device.touch(274, 450, device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.press('KEYCODE_DEL', device.DOWN_AND_UP)
    device.type(pass1)

    device.touch(20, 790, device.DOWN_AND_UP)
    MonkeyRunner.sleep(0.55)
    device.touch(20, 790, device.DOWN_AND_UP)
    MonkeyRunner.sleep(0.95)

    device.touch(420, 790, device.DOWN_AND_UP)
    MonkeyRunner.sleep(0.55)
    #select_alg (device)
```

```
def connect (device, ip, port, id, pass1):
    #open menu
    MonkeyRunner.sleep(0.55)
    device.press('KEYCODE_MENU', device.DOWN_AND_UP)
    MonkeyRunner.sleep(0.55)

    #push connect button
    device.touch(388, 658, device.DOWN_AND_UP)
    MonkeyRunner.sleep(0.55)

    write(device, ip, port, id, pass1)

def main():
    device = MonkeyRunner.waitForConnection()

device.startActivity(component='dmsl.Smartp2p.app/dmsl.Smartp2p.app.SMARTP2PF
ULLActivity')
    if len(sys.argv) == 6:
        ip = sys.argv[2]
        port= sys.argv[3]
        id = sys.argv[4]
        pass1 = sys.argv[5]
        connect(device, ip, port, id, pass1)
    if len(sys.argv) == 8:
        ip = sys.argv[2]
        port= sys.argv[3]
        id = sys.argv[4]
        pass1 = sys.argv[5]
        alg = sys.argv[6]
        word = sys.argv[7]
        connect(device, ip, port, id, pass1)
        select_alg (device, alg, word)
```



```
    if len(sys.argv) == 10:
        ip = sys.argv[2]
        port= sys.argv[3]
        id = sys.argv[4]
        pass1 = sys.argv[5]
        alg = sys.argv[6]
        word = sys.argv[7]
        recall = sys.argv[8]
        decision = sys.argv[9]
        connect(device, ip, port, id, pass1)
        select_alg (device, alg, word)
        MonkeyRunner.sleep(130.0)
        select_tree (device, recall)
        device.touch(50, 761, device.DOWN_AND_UP)

if __name__ == '__main__':
    main()
```

Παράρτημα Β

Βάση Δεδομένων

Εξυπηρετητής SmartOpt

SmaerP2P Εφαρμογή

Βάση Δεδομένων

Δημιουργία Πινάκων

```
CREATE TABLE USERS
(
	USERS_ID INT IDENTITY (1,1),
	USERS_UNAME VARCHAR (30) NOT NULL UNIQUE,
	USERS_PASS VARCHAR (10) NOT NULL,
	USERS_IP VARCHAR (16),
	USERS_PORT VARCHAR (5),
	USERS_ONLINE INT

	CONSTRAINT [PK_USERS]
		PRIMARY KEY ([USERS_ID])
)

GO
-------------------------------------------------------------------------------
CREATE TABLE PROFILES
(
	PROFILES_TITLE VARCHAR (300),
	PROFILES_USER_ID INT,
	PROFILES_QUANTITY INT

	CONSTRAINT [FK__PROFILE_1]
```



```
		FOREIGN KEY ([PROFILES_USER_ID]) REFERENCES [USERS]
([USERS_ID])
		ON DELETE CASCADE
)

GO
-------------------------------------------------------------------------------
CREATE TABLE INFO
(
	INFO_USER INT,
	INFO_LON FLOAT,
	INFO_LAT FLOAT,
	INFO_X FLOAT,
	INFO_Y FLOAT,
	INFO_TIME datetime

	CONSTRAINT [FK__INFO_1]
		FOREIGN KEY ([INFO_USER]) REFERENCES [USERS]
([USERS_ID])
		ON DELETE CASCADE
)
-------------------------------------------------------------------------------

CREATE TABLE MAPPING_WITH_QUSER
(
	MAPPING_QUSER INT,
	MAPPING_WITH_QUSER INT,
	MAPPING_IM_USER_ID INT

	CONSTRAINT [FK_MAPPING_WITH_QUSER_1]
		FOREIGN KEY ([MAPPING_QUSER]) REFERENCES [USERS]
([USERS_ID])
		ON DELETE CASCADE)
```

Συναρτήσεις - Διαδικασίες

```
USE SMARTP2P
	GO
	create PROCEDURE REGISTER_USER
		@IP VARCHAR (16),
			@PORT VARCHAR (5),
			@NAME VARCHAR (30),
			@PASS VARCHAR (10)
	AS

		if ((SELECT USERS_UNAME FROM USERS WHERE
USERS_UNAME = @NAME) = @NAME)
		BEGIN
			Raiserror(50005, 11, 1)
			RETURN
		END

		INSERT INTO [USERS] (USERS_IP, USERS_PORT,
USERS_ONLINE, USERS_UNAME, USERS_PASS)
		VALUES (@IP, @PORT, 0, @NAME, @PASS);

		DECLARE @ID INT

		SET @ID = DBO.GET_USER_ID_BY_NAME (@NAME)

		declare @i int
		declare @max int
		set @i=1;
		set @max = dbo.GET_NUM_USERS()
		while (@i<=@max)
		begin
```



```
			insert into [MAPPING_WITH_QUSER] (MAPPING_QUSER,
MAPPING_WITH_QUSER)
			values (@ID, @i);
			set @i=@i+1
		end
		set @i=1;
		while (@i<=@max-1)
		begin
			insert into [MAPPING_WITH_QUSER] (MAPPING_QUSER,
MAPPING_WITH_QUSER)
			values (@i, @ID);
			set @i=@i+1
		end

	GO

	--------------------------------------------------------------------------------

	USE SMARTP2P
	GO
	create PROCEDURE CONNECT_USER
		@ID INT,
			@PASS VARCHAR (10),
			@IP VARCHAR (16),
			@PORT VARCHAR (5)
	AS

		if ((SELECT USERS_ID FROM USERS WHERE USERS_PASS =
@PASS AND USERS_ID = @ID) = @ID)
		BEGIN
			UPDATE USERS
			SET USERS_IP=@IP, USERS_PORT=@PORT,
USERS_ONLINE = 0
```

```
		WHERE USERS_ID = @ID AND USERS_PASS = @PASS

		EXECUTE REMOVE_USER_PROFILE @ID
		EXECUTE REMOVE_USER_INFO @ID

		UPDATE USERS
		SET	USERS_IP=@IP,	USERS_PORT=@PORT,
USERS_ONLINE = 1
		WHERE USERS_ID = @ID AND USERS_PASS = @PASS

	END
	else
	BEGIN
		Raiserror(50004, 11, 1)
		RETURN
	END
GO

-------------------------------------------------------------------------------

USE SMARTP2P
GO
create PROCEDURE TERMINATE_CONNECTION_USER
	@ID INT
AS

	UPDATE USERS
	SET USERS_ONLINE = 0
	WHERE USERS_ID = @ID

	EXECUTE REMOVE_USER_PROFILE @ID
	EXECUTE REMOVE_USER_INFO @ID
GO
```



```
-------------------------------------------------------------------------------

USE SMARTP2P
GO
ALTER PROCEDURE INSERT_PROFILES
	@ID INT,
		@TITLE VARCHAR (300),
		@QNUM INT
AS

	if ((SELECT USERS.USERS_ONLINE FROM USERS WHERE
USERS.USERS_ID = @ID) = 1)
	BEGIN
		INSERT INTO [PROFILES] (PROFILES_TITLE,
PROFILES_USER_ID, PROFILES_QUANTITY)
		VALUES (@TITLE, @ID, @QNUM);

	END
	else
	BEGIN
		Raiserror(50002, 11, 1)
		RETURN
	END

GO

-------------------------------------------------------------------------------

USE SMARTP2P
GO
CREATE PROCEDURE INSERT_INFO
	@ID INT,
```

```
        @LATIT FLOAT,
        @LONGT FLOAT
AS

    if ((SELECT USERS.USERS_ONLINE FROM USERS WHERE
USERS.USERS_ID = @ID) = 1)
    BEGIN
        INSERT INTO [INFO] (INFO_USER, INFO_LAT, INFO_LON,
INFO_TIME)
        VALUES (@ID, @LATIT, @LONGT, GETDATE ());

    END
    else
    BEGIN
        Raiserror(50003, 11, 1)
        RETURN
    END
GO

-------------------------------------------------------------------------

USE SMARTP2P
GO
create FUNCTION GET_ACTIVE_USERS ()
RETURNS INT

AS
BEGIN
    RETURN (SELECT COUNT(*) FROM USERS WHERE
dbo.USERS.USERS_ONLINE=1)
END
GO
```



```
-------------------------------------------------------------------------

USE SMARTP2P
GO
create FUNCTION GET_NUM_USERS ()
RETURNS INT

AS
BEGIN
    RETURN (SELECT COUNT(*) FROM USERS)
END
GO

-------------------------------------------------------------------------

USE SMARTP2P
GO
CREATE FUNCTION GET_USER_PROFILE (@ID INT)
RETURNS @TITLE TABLE
(
    STRING VARCHAR (300)
)

AS
BEGIN
    INSERT INTO @TITLE
        SELECT PROFILES.PROFILES_TITLE
        FROM PROFILES
        WHERE PROFILES.PROFILES_USER_ID = @ID

    RETURN;
END
```

```
GO

-------------------------------------------------------------------------------

USE SMARTP2P
GO
CREATE FUNCTION GET_NUMBER_OF_USER_PAPER (@ID INT)
RETURNS INT

AS
BEGIN
	RETURN (SELECT COUNT (*) FROM DBO.GET_USER_PROFILE (@ID))
END
GO

-------------------------------------------------------------------------------

USE SMARTP2P
GO
CREATE FUNCTION GET_NUMBER_OF_USER_PAPER_WITH_SPESIFIC_WORD (@ID INT, @WORD VARCHAR (30))
RETURNS INT

AS
BEGIN
	RETURN (SELECT COUNT (*) FROM DBO.GET_USER_PROFILE (@ID) WHERE STRING LIKE '%' + @WORD + '%')
END
GO

-------------------------------------------------------------------------------
```



```
-- id =  id ekeinou pou einai o querry peer
USE SMARTP2P
GO
aLTER FUNCTION RESULTS_WITH_COUNT (@ID INT, @WORD VARCHAR (30))
RETURNS @ResultTable Table
(
	UsersId int,
	COUNT_PAPERS int,
	COUNT_WITH_FILTER int,
	TOTAL_TITLE_BYTES int
)

AS
BEGIN
	declare @i int
	declare @max int
	declare @real_id int
	set @i = 0;
	set @max = dbo.GET_ACTIVE_USERS()
	while (@i<@max)
	begin
		set @real_id = (select dbo.ReturnMappedUser (@ID, @i))
		insert into @ResultTable (UsersId, COUNT_PAPERS, COUNT_WITH_FILTER, TOTAL_TITLE_BYTES)
		values (@i,
				(select dbo.GET_NUMBER_OF_USER_PAPER(@real_id)),
				(select dbo.GET_NUMBER_OF_USER_PAPER_WITH_SPESIFIC_WORD(@real_id, @WORD)),
```

```
                (select (sum (LEN
(PROFILES.PROFILES_TITLE)*PROFILES.PROFILES_QUANTITY))
                from PROFILES
                where PROFILES.PROFILES_USER_ID =
@real_id and PROFILES.PROFILES_TITLE LIKE '%' + @WORD + '%'))
            set @i = @i +1;
        end

        return

    END
    GO


    USE SMARTP2P
    GO
    CREATE PROCEDURE REMOVE_USER
        @ID INT
    AS
        DELETE FROM USERS
        WHERE USERS.USERS_ID = @ID
    GO

    --execute REMOVE_USER 8

    ---------------------------------------------------------------------------

    USE SMARTP2P
    GO
    CREATE PROCEDURE REMOVE_USER_PROFILE
        @ID INT
    AS
        DELETE FROM PROFILES
```



```
        WHERE PROFILES.PROFILES_USER_ID = @ID
    GO


    USE SMARTP2P
    GO
    CREATE PROCEDURE REMOVE_USER_INFO
        @ID INT
    AS
        DELETE FROM INFO
        WHERE INFO.INFO_USER = @ID
    GO


    USE SMARTP2P
    GO
    CREATE FUNCTION GET_USER_ID_BY_NAME (@NAME VARCHAR
(30))
    RETURNS INT

    AS
    BEGIN
        RETURN (Select USERS_ID FROM USERS WHERE
USERS_UNAME =@NAME)
    END
    GO

    USE SMARTP2P
    GO
    create function CalculateGPSDistance
    (
        @lat1 float,
        @lon1 float,
```

```
        @lat2 float,
        @lon2 float
)
returns float
as
begin

        /*
        a = Math.PI / 180;
        lat1 = eval(form.latadd.value * a);//convert to radian
        lat2 = eval(form.latbdd.value * a);
        lat1dir = form.latadirdd.value;
        lat2dir = form.latbdirdd.value;
        lon1 = eval(form.lonadd.value * a);
        lon2 = eval(form.lonbdd.value * a);
        lon1dir = form.lonadirdd.value;
        lon2dir = form.lonbdirdd.value;
        */
        DECLARE @a FLOAT
        SET @a = PI()/180.0

        SET @lat1 = @lat1 * @a
        SET @lat2 = @lat2 * @a
        SET @lon1 = @lon1 * @a
        SET @lon2 = @lon2 * @a

        /*
        calculate readian distance, assume lat1, lon1, lat2, lon2 are in degrees
        t1 = Math.sin(lat1) * Math.sin(lat2);
        t2 = Math.cos(lat1) * Math.cos(lat2);
        t3 = Math.cos(lon1 - lon2);
        t4 = t2 * t3;
        t5 = t1 + t4;
```



```
        rad_dist = Math.atan(-t5/Math.sqrt(-t5 * t5 +1)) + 2 * Math.atan(1);
        */
        DECLARE @t1 FLOAT
        DECLARE @t2 FLOAT
        DECLARE @t3 FLOAT
        DECLARE @t4 FLOAT
        DECLARE @t5 FLOAT


        SET @t1 = SIN(@lat1) * SIN(@lat2)
        SET @t2 = COS(@lat1) * COS(@lat2)
        SET @t3 = COS(@lon1 - @lon2)
        SET @t4 = @t2 * @t3
        SET @t5 = @t1 + @t4

        DECLARE @RESULT FLOAT
        IF @t5<>1
        BEGIN
                SET @RESULT = ATAN((-1*@t5)/SQRT(-1*@t5*@t5 + 1)) +
2 * ATAN(1)
        END
        ELSE
        BEGIN
                SET @RESULT = 2 * ATAN(1)
        END
        --CALCULATE MILES
        SET @RESULT = 3437.74677 * 1.1508 * @RESULT
        --CALCULATE METERS FROM MILES
        SET @RESULT = 1.6093470878864446*1000.0*@RESULT
        RETURN @RESULT

end
```

```
--------------------------------------------------------------------------------

GO

USE SMARTP2P
GO
create function Distances_between_A_Users (@ID int)
returns @DISTANCE_TABLE TABLE
(
        USER_A INT,
        USER_B INT,
        DISTANCE FLOAT

)
as
begin

        DECLARE @i int
        DECLARE @j int
        DECLARE @max int
        DECLARE @user1 int
        DECLARE @user2 int
        DECLARE @long1 float
        DECLARE @long2 float
        DECLARE @lat1 float
        DECLARE @lat2 float
        set @i = 0;
        set @max = dbo.GET_ACTIVE_USERS()
        while (@i<@max)
        begin
                set @j = @i+1;
                while (@j<@max)
                begin
```



```
                        set @user1 = (select dbo.ReturnMappedUser(@ID, @i));
                        set @user2 = (select dbo.ReturnMappedUser(@ID, @j));

                        --a) ipologismos gia to kathe userid to longtitude kai
latitude gia tin teleftea xroniki stigmi
                        Set @long1 = (select dbo.ReturnLastLonUser(@user1));
                        Set @long2 = (select dbo.ReturnLastLonUser(@user2));
                        Set @lat1 = (select dbo.ReturnLastLatUser(@user1));
                        Set @lat2 = (select dbo.ReturnLastLatUser(@user2));
                        --b)na ta perasw san parametro stin pio panw sinartisi
CalculateGPSDistance

                        insert INTO @DISTANCE_TABLE (USER_A,
USER_B, DISTANCE)
                        VALUES(@i, @j,
dbo.CalculateGPSDistance(@lat1,@long1, @lat2, @long2));
                        set @j=@j+1
                end
                set @i=@i+1
        end


        RETURN ;
end

GO

--------------------------------------------------------------------------------

USE SMARTP2P
GO
CREATE Procedure create_Query
```

```
	@ID INT
AS
	UPDATE MAPPING_WITH_QUSER
	SET MAPPING_IM_USER_ID = -1
	WHERE MAPPING_QUSER = @ID

	UPDATE MAPPING_WITH_QUSER
	SET MAPPING_IM_USER_ID = 0
	WHERE MAPPING_QUSER = @ID and MAPPING_WITH_QUSER =
@ID


	declare @i int
	declare @p int
	declare @max int
	declare @count int

	set @i=1;
	set @count=1;
	set @max = dbo.GET_ACTIVE_USERS()
	set @p = (select dbo.isActive(@i))
	while (@count<@max)
	begin
		if (@i<>@ID and @p = 1)
		begin
			UPDATE MAPPING_WITH_QUSER
			SET MAPPING_IM_USER_ID = @count
			WHERE MAPPING_QUSER = @ID and
MAPPING_WITH_QUSER = @i

			set @count = @count+1;
		end

		set @i=@i+1
```



```
		set @p = (select dbo.isActive(@i))
	end

GO

USE SMARTP2P

GO

create function ReturnMappedUser (@Qid int, @userId int)
returns int
as
begin
	Return (select MAPPING_WITH_QUSER from
MAPPING_WITH_QUSER where MAPPING_QUSER = @Qid and
MAPPING_IM_USER_ID = @userId)
end

GO
--------------------------------------------------------------------------------
USE SMARTP2P

GO

create function ReturnLastLonUser (@userId int)
returns float
as
begin
	Return (Select INFO.INFO_LON
		from INFO
		Where info.INFO_USER=@userId
			and INFO.INFO_TIME = (select
MAX(INFO.INFO_TIME)
```

```
from INFO

where info.INFO_USER=@userId))
end

go

-------------------------------------------------------------------------------
USE SMARTP2P

GO
create function ReturnLastLatUser (@userId int)
returns float
as
begin
	Return (Select INFO.INFO_LAT
			from INFO
			Where info.INFO_USER=@userId
				and INFO.INFO_TIME = (select
MAX(INFO.INFO_TIME)

from INFO

where info.INFO_USER=@userId))
end

go
-------------------------------------------------------------------------------

USE SMARTP2P
GO
CREATE FUNCTION GET_PORT (@ID INT)
```



```
RETURNS INT

AS
BEGIN
	RETURN (SELECT USERS.USERS_PORT FROM users where
users.USERS_ID = @ID)
END
GO

-------------------------------------------------------------------------------

USE SMARTP2P
GO
CREATE FUNCTION GET_IP (@ID INT)
RETURNS VARCHAR(15)

AS
BEGIN
	RETURN (SELECT USERS.USERS_IP FROM users where
users.USERS_ID = @ID)
END
GO

-------------------------------------------------------------------------------

USE SMARTP2P
GO
CREATE FUNCTION isActive (@ID INT)
RETURNS INT

AS
BEGIN
```

```
        if ((select USERs_ID from USERS where USERS_ONLINE = 1 and
USERS_ID = @ID) = @ID)
                return 1
        return 0
    END
    GO
```

Δημιουργία Σφαλμάτων

```
Exec sp_addmessage
@msgnum=50002,
@severity=11,
@msgtext='Not possible to add profile, Not connected',
@with_log='true'
/*,
@replace= 'replace'*/

Exec sp_addmessage
@msgnum=50003,
@severity=11,
@msgtext='Not possible to add info, Not connected',
@with_log='true'

Exec sp_addmessage
@msgnum=50004,
@severity=11,
@msgtext='Invalid User Id or Password',
@with_log='true'

Exec sp_addmessage
@msgnum=50005,
@severity=11,
@msgtext='Username Already Exists',
```



```
@with_log='true'
```

Εξυπηρετητής SmartOpt

Server.java

```
package ServerApi;

import java.io.*;

import java.net.*;

public class Server {

    private ServerSocket serverSocket;

    public Server(int port) {

        try
        {
            serverSocket = new ServerSocket(port);

            while (true)
            {

                Socket socket = serverSocket.accept();
                JoinedPeer t = new JoinedPeer(socket);
                t.start();
            }
        }
        catch (IOException e) {
            System.out.println("Exception on new ServerSocket: " + e);
        }
```

```
	}

	public static void main(String[] arg) {
		new Server(4500);


	}

}
```

JoinedPeer.java

```
package ServerApi;

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;
import java.net.UnknownHostException;
import java.sql.CallableStatement;
import java.sql.SQLException;
import java.util.ArrayList;

import org.math.plot.Draw.Draw3DPlots;

import BFS.BFSAlg;
```



```
import matlabcontrol.MatlabConnectionException;
import matlabcontrol.MatlabInvocationException;
import matlabcontrol.MatlabProxy;
import matlabcontrol.MatlabProxyFactory;
import matlabcontrol.MatlabProxyFactoryOptions;

class JoinedPeer extends Thread {

	byte[] data;
	Socket socket;
	ObjectInputStream Sinput;
	ObjectOutputStream Soutput;
	DBConnector con;

	public void sendTree(ArrayList<String[]> tree) throws IOException {

		String str= "";
		for (int i = 0; i < tree.size(); ++i) {
			str = "";
			for (int j = 0; j<tree.get(i).length; ++j){
				if (j!=0)
					str = str + " ";
				str = str + tree.get(i)[j];
			}
			Soutput.writeObject(str);
			Soutput.flush();
		}
		Soutput.writeObject("NULL");
		Soutput.flush();
	}

	public ArrayList<String[]> convertTree (ArrayList<String[]> tree, String Qid)
throws SQLException
```

```
{
        ArrayList<String[]> tree2 = null;
        String str = null;
        int id1 = 0, id2 = 0, port = 0;
        String ip =null;
        String lon=null, lat=null;

        CallableStatement cstmt = null;
        String SQL;

        tree2 = new ArrayList<String[]> ();
        for (int i=0; i<tree.size(); ++i)
        {
                SQL = "Select dbo.ReturnMappedUser('" + Qid + "','" + i + "') as
id";
                con.getReadOnlyRecordSet(SQL);
                while (con.GetResult().next()) {
                        id1 = con.GetResult().getInt ("id");
                }
                /**/
                SQL = "Select dbo.GET_PORT('" + id1 + "') as id";
                con.getReadOnlyRecordSet(SQL);
                while (con.GetResult().next()) {
                        port = con.GetResult().getInt ("id");
                }
                SQL = "Select dbo.GET_IP('" + id1 + "') as id";
                con.getReadOnlyRecordSet(SQL);
                while (con.GetResult().next()) {
                        ip = con.GetResult().getString ("id");
                }

                SQL = "Select dbo.ReturnLastLonUser('" + id1 + "') as id";
                con.getReadOnlyRecordSet(SQL);
```



```
                while (con.GetResult().next()) {
                        lon = con.GetResult().getString ("id");
                }
                SQL = "Select dbo.ReturnLastLatUser('" + id1 + "') as id";
                con.getReadOnlyRecordSet(SQL);
                while (con.GetResult().next()) {
                        lat = con.GetResult().getString ("id");
                }


                if (tree.get(i)[0].equals("-1"))
                        id2 = -1;
                else
                {
                        System.out.println ("sdfsdfsdfsdip: " + Qid + "dfsdfs: " +
tree.get(i)[0]);
                        SQL = "Select dbo.ReturnMappedUser('" + Qid + "','" +
tree.get(i)[0] + "') as id2";
                        con.getReadOnlyRecordSet(SQL);
                        while (con.GetResult().next()) {
                                id2 = con.GetResult().getInt("id2");
                        }
                }
                if (id2==0)
                        id2=-1;
                if (id1==id2)
                        str = new String (id1 + " " + "-1" + " " + ip + " " + port + "
" + lon + " " + lat);
                else
                        str = new String (id1 + " " + id2 + " " + ip + " " + port + "
" + lon + " " + lat);
                tree2.add(str.split(" "));
        }
```

```
        return tree2;
    }

    public void printTree(ArrayList<String[]> tree) {
        for (int i = 0; i < tree.size(); ++i) {
            System.out.println(tree.get(i)[0] + " " + tree.get(i)[1] + " "
                    + tree.get(i)[2] + " " + tree.get(i)[3] + " "
                    + tree.get(i)[4]);
        }

    }

    public void sendTree() throws IOException, ClassNotFoundException,
SQLException {
        String id = (String) Sinput.readObject();
        System.out.println("id: " + id);
        String pass = (String) Sinput.readObject();
        System.out.println("pass: " + pass);
        String decision = (String) Sinput.readObject();
        System.out.println("dec: " + decision);
        String time = (String) Sinput.readObject();
        System.out.println("time: " + time);
        String recall = (String) Sinput.readObject();
        System.out.println("recall: " + recall);
        String Alg = (String) Sinput.readObject();
        System.out.println("Algorithm: " + Alg);

        //id="4";
        if (Alg.equals("MOEAD"))
        {
            ArrayList<String[]> outputAlgorithm = new
ArrayList<String[]>();
```



```
            outputAlgorithm =
IO.getFileContent("C:\\Users\\Christos\\Documents\\SmartP2P workspace\\SmartP2P
Server\\src\\MOEAD\\Exp" + id + "\\BestPF\\outBests.log", " ");

            int minPos=0;


            if (decision.equals("0"))
            {
                Double minVal =
Double.valueOf(outputAlgorithm.get(0)[1]);
                for (int i=1; i<outputAlgorithm.size(); ++i)
                    if (Double.valueOf(outputAlgorithm.get(i)[1]) <
minVal)
                    {
                        minPos = i;
                        minVal =
Double.valueOf(outputAlgorithm.get(i)[1]);
                    }
            }
            else
            {
                Double minVal, x1, x2, y1, y2, temp;

                x1 = Double.valueOf (time);
                System.out.println("Time: " + time + " Recall: " +
recall);
                y1 = Double.valueOf (time);
                x2 = Double.valueOf(outputAlgorithm.get(0)[2]);
                y2 = Double.valueOf(outputAlgorithm.get(0)[3]);

                minVal = distance (x1, x2, y1, y2);
                System.out.println("first min: " + minVal);
```

```
                for (int i=1; i<outputAlgorithm.size(); ++i)
                {
                        x2 = Double.valueOf(outputAlgorithm.get(i)[2]);
                        y2 = Double.valueOf(outputAlgorithm.get(i)[3]);
                        temp = distance (x1, x2, y1, y2);
                        System.out.println("temp: " + temp);
                        if (temp<minVal)
                        {
                                minPos = i;
                                minVal = temp;
                        }
                }
                System.out.println("minpos: " + minPos);
        }

        ArrayList<String[]> tree = new ArrayList<String[]>();
        tree = IO.getFileContent("C:\\Users\\Christos\\Documents\\SmartP2P workspace\\SmartP2P Server\\src\\MOEAD\\Exp" + id + "\\PFtrees\\PFtree" + minPos + ".log", " ");

        tree = this.convertTree(tree, id);
        for (int i=0; i<tree.size(); ++i)
                System.out.println(tree.get(i)[0] + "  " + tree.get(i)[1] + " " + tree.get(i)[2] + "  " + tree.get(i)[3]);


        sendTree(tree);
    }

    if (Alg.equals("NSGAII"))
    {
        ArrayList<String[]> outputAlgorithm = new ArrayList<String[]>();
```



```
        outputAlgorithm = IO.getFileContent("C:\\Users\\Christos\\Documents\\SmartP2P workspace\\SmartP2P Server\\src\\NSGAII\\Exp" + id + "\\BestPF\\outBests.log", " ");

        int minPos=0;


        if (decision.equals("0"))
        {
                Double minVal = Double.valueOf(outputAlgorithm.get(0)[1]);
                for (int i=1; i<outputAlgorithm.size(); ++i)
                        if (Double.valueOf(outputAlgorithm.get(i)[1]) < minVal)
                        {
                                minPos = i;
                                minVal = Double.valueOf(outputAlgorithm.get(i)[1]);
                        }
        }
        else
        {
                Double minVal, x1, x2, y1, y2, temp;

                x1 = Double.valueOf (time);
                System.out.println("Time:  " + time + "  Recall: " + recall);
                y1 = Double.valueOf (time);
                x2 = Double.valueOf(outputAlgorithm.get(0)[2]);
                y2 = Double.valueOf(outputAlgorithm.get(0)[3]);

                minVal = distance (x1, x2, y1, y2);
                System.out.println("first min: " + minVal);
```

```
                for (int i=1; i<outputAlgorithm.size(); ++i)
                {
                        x2 = Double.valueOf(outputAlgorithm.get(i)[2]);
                        y2 = Double.valueOf(outputAlgorithm.get(i)[3]);
                        temp = distance (x1, x2, y1, y2);
                        System.out.println("temp: " + temp);
                        if (temp<minVal)
                        {
                                minPos = i;
                                minVal = temp;
                        }
                }
                System.out.println("minpos: " + minPos);
        }

        ArrayList<String[]> tree = new ArrayList<String[]>();
        tree =
IO.getFileContent("C:\\Users\\Christos\\Documents\\SmartP2P workspace\\SmartP2P
Server\\src\\NSGAII\\Exp" + id + "\\PFtrees\\PFtree" + minPos + ".log", " ");

        tree = this.convertTree(tree, id);
        for (int i=0; i<tree.size(); ++i)
                System.out.println(tree.get(i)[0] + "  " + tree.get(i)[1] + "
" + tree.get(i)[2] + "  " + tree.get(i)[3]);


        sendTree(tree);
    }

    if (Alg.equals("RW"))
    {
        String word = (String) Sinput.readObject();
        System.out.println("word: " + word);
```



```
        CallableStatement cstmt = null;
        String SQL;

        SQL = "{call dbo.create_Query (?)}";

        try {
                cstmt = con.getConnection().prepareCall(SQL);
                cstmt.setString(1, id);
                cstmt.execute();
        } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }

        IO.deleteDir(new File ("./" + "src/RW/Exp" + id));
        RW.src.RW.RWAlg C = new RW.src.RW.RWAlg
(Integer.parseInt(id), word);
        ArrayList<String[]> tree = new ArrayList<String[]>();
        tree =
IO.getFileContent("C:\\Users\\Christos\\Documents\\SmartP2P workspace\\SmartP2P
Server\\src\\RW\\Exp" + id + "\\tree" + ".log", " ");

        tree = this.convertTree(tree, id);
        for (int i=0; i<tree.size(); ++i)
                System.out.println(tree.get(i)[0] + "  " + tree.get(i)[1] + "
" + tree.get(i)[2] + "  " + tree.get(i)[3]);

        sendTree(tree);

    }

    if (Alg.equals("BFS"))
    {
```

```
                String word = (String) Sinput.readObject();
                System.out.println("word: " + word);
                CallableStatement cstmt = null;
                String SQL;

                SQL = "{call dbo.create_Query (?)}";

                try {
                        cstmt = con.getConnection().prepareCall(SQL);
                        cstmt.setString(1, id);
                        cstmt.execute();
                } catch (SQLException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }

                IO.deleteDir(new File ("./" + "src/BFS/Exp" + id));
                BFSAlg C = new BFS.BFSAlg (Integer.parseInt(id), word);
                ArrayList<String[]> tree = new ArrayList<String[]>();
                tree =
IO.getFileContent("C:\\Users\\Christos\\Documents\\SmartP2P workspace\\SmartP2P
Server\\src\\BFS\\Exp" + id + "\\tree" + ".log", " ");

                tree = this.convertTree(tree, id);
                for (int i=0; i<tree.size(); ++i)
                        System.out.println(tree.get(i)[0] + "  " + tree.get(i)[1] + "
" + tree.get(i)[2] + "  " + tree.get(i)[3]);

                sendTree(tree);

        }

}
```



```
public Double distance (Double x1, Double x2, Double y1, Double y2)
{
        Double res = null;

        res = Math.sqrt(((x2-x1) * (x2-x1)) + ((y2-y1) * (y2-y1)));

        return res;
}

public void register() throws IOException, ClassNotFoundException,
SQLException {
        String name = (String) Sinput.readObject();
        System.out.println("id: " + name);
        String pass = (String) Sinput.readObject();
        System.out.println("pass: " + pass);

        CallableStatement cstmt = null;
        String SQL;

        SQL = "{call dbo.REGISTER_USER (?,?,?,?)}";

        try {
                cstmt = con.getConnection().prepareCall(SQL);
                cstmt.setString(1, socket.getInetAddress().getHostAddress());
                cstmt.setInt(2, socket.getPort());
                cstmt.setString(3, name);
                cstmt.setString(4, pass);
                cstmt.executeQuery();
        } catch (SQLException e) {
                String uId = null;
                if (e.getSQLState() == null && e.getErrorCode() != 50005) {
```

```
                        SQL = "Select dbo.GET_USER_ID_BY_NAME('" +
name + "') as num";

                        con.getReadOnlyRecordSet(SQL);
                        while (con.GetResult().next()) {
                                uId = con.GetResult().getString ("num");
                        }
                        Soutput.writeObject(uId);
                        Soutput.flush();
                        return;
                }else if (e.getErrorCode() == 50005)
                {
                        Soutput.writeObject("User Name Already Exists");
                        Soutput.flush();
                        return;
                }
                Soutput.writeObject("NOT OK");
                Soutput.flush();
                return;
        }
    }

    public void connect() throws IOException, ClassNotFoundException {
        String id = (String) Sinput.readObject();
        System.out.println("id: " + id);
        String pass = (String) Sinput.readObject();
        System.out.println("pass: " + pass);
        String port = (String) Sinput.readObject();
        System.out.println("port: " + port);
        CallableStatement cstmt = null;
        String SQL;

        SQL = "{call dbo.CONNECT_USER (?,?,?,?)}";
```



```
        try {
                cstmt = con.getConnection().prepareCall(SQL);
                cstmt.setString(3, socket.getInetAddress().getHostAddress());
                cstmt.setString(4, port);
                cstmt.setString(1, id);
                cstmt.setString(2, pass);
                cstmt.executeQuery();
        } catch (SQLException e) {
                if (e.getSQLState() == null && e.getErrorCode() != 50004) {
                        Soutput.writeObject("OK");
                        Soutput.flush();
                        return;
                }
                Soutput.writeObject("NOT OK");
                Soutput.flush();
                return;
        }
}

public void terminateConnection() throws IOException,
                ClassNotFoundException {
        String id = (String) Sinput.readObject();
        System.out.println("id: " + id);

        CallableStatement cstmt = null;
        String SQL;

        SQL = "{call dbo.TERMINATE_CONNECTION_USER (?)}";

        try {
                cstmt = con.getConnection().prepareCall(SQL);
                cstmt.setString(1, id);
```

```
                cstmt.executeQuery();
        } catch (SQLException e) {
                if (e.getSQLState() == null) {
                        Soutput.writeObject("OK");
                        Soutput.flush();
                        return;
                }
                Soutput.writeObject("NOT OK");
                Soutput.flush();
                return;
        }
}

public ArrayList<String[]> receiveTree() throws IOException,
                ClassNotFoundException {
        ArrayList<String[]> tree = new ArrayList<String[]>();
        String str;

        while (true) {
                str = (String) Sinput.readObject();
                if (str.equals("NULL"))
                        break;
                tree.add(str.split(" "));
        }

        printTree(tree);
        return tree;

}

public void receiveANDaddProfile() throws IOException,
                ClassNotFoundException {
        CallableStatement cstmt = null;
```



```
        String SQL;

        String id = (String) Sinput.readObject();
        System.out.println("id: " + id);
        ArrayList<String> aList = new ArrayList<String>();
        String str;

        while (true) {
                str = (String) Sinput.readObject();

                if (str.equals("NULL"))
                        break;
                aList.add(str);
        }

        int flag = 1;
        for (int i = 0; i < aList.size(); i++) {

                SQL = "{call dbo.INSERT_PROFILES (?,?)}";

                try {
                        cstmt = con.getConnection().prepareCall(SQL);
                        cstmt.setString(2, aList.get(i));
                        cstmt.setString(1, "" + id);
                        cstmt.executeQuery();
                } catch (SQLException e) {
                        if (e.getSQLState() == null && e.getErrorCode() !=
50002)
                                flag = flag * 1;
                        else
                                flag = flag * 0;

                }
```

```
            }
            if (flag == 1) {
                Soutput.writeObject("OK");
                Soutput.flush();
                return;
            }
            Soutput.writeObject("NOT OK");
            Soutput.flush();
            return;

        }

        public void receiveANDaddInfo() throws IOException,
ClassNotFoundException {
            CallableStatement cstmt = null;
            String SQL;

            String id = (String) Sinput.readObject();
            System.out.println("id: " + id);

            String str = (String) Sinput.readObject();
            String[] str2 = str.split(" ");

            SQL = "{call dbo.INSERT_INFO (?,?,?)}";

            try {
                cstmt = con.getConnection().prepareCall(SQL);
                cstmt.setString(3, str2[0]);
                cstmt.setString(2, str2[1]);
                cstmt.setString(1, "" + id);
                cstmt.executeQuery();
            } catch (SQLException e) {
                if (e.getSQLState() == null && e.getErrorCode() != 50003) {
```



```
                    Soutput.writeObject("OK");
                    Soutput.flush();
                    return;
                }
                Soutput.writeObject("NOT OK");
                Soutput.flush();
                return;
            }

        }

        JoinedPeer(Socket socket) throws IOException {

            con = new DBConnector("127.0.0.1", 1433, "SMARTP2P");
            this.socket = socket;
            Soutput = new ObjectOutputStream(socket.getOutputStream());
            Soutput.flush();
            Sinput = new ObjectInputStream(socket.getInputStream());

        }

        public void Search(String req, String id) throws NumberFormatException,
                IOException, ClassNotFoundException {

            Client pReq = new Client(Integer.parseInt(id));
            pReq.setTree(this.receiveTree());
            pReq.Request("Search " + req);

        }

        public boolean GetGraph() throws IOException, ClassNotFoundException,
MatlabConnectionException, MatlabInvocationException, InterruptedException
        {
```

```
CallableStatement cstmt = null;
String SQL;

String id = (String) Sinput.readObject();
System.out.println("id: " + id);
String word = (String) Sinput.readObject();
System.out.println("word: " + word);
String Alg = (String) Sinput.readObject();
System.out.println("Algorithm: " + Alg);

SQL = "{call dbo.create_Query (?)}";

try {
	cstmt = con.getConnection().prepareCall(SQL);
	cstmt.setString(1, id);
	cstmt.execute();
} catch (SQLException e) {
	// TODO Auto-generated catch block
	e.printStackTrace();
}
if (Alg.equals("MOEAD"))
{
	IO.deleteDir(new File ("./" + "src/MOEAD/Exp" + id));
	IO.deleteDir(new File
("C:\\Users\\Christos\\Documents\\SmartP2P
workspace\\Figures\\figure3Duser"+id+".png"));
	MOEAD.Algorithm C = new
MOEAD.Algorithm(Integer.parseInt(id), word);

	Draw3DPlots draw = new Draw3DPlots ();
	draw.Draw("C:\\Users\\Christos\\Documents\\SmartP2P
workspace\\SmartP2P Server\\src\\MOEAD\\Exp"+id+"\\BestPF\\outBests.log",
```



```
"C:\\Users\\Christos\\Documents\\SmartP2P
workspace\\Figures\\figure3Duser"+id+".png");
	draw.Draw("C:\\Users\\Christos\\Documents\\SmartP2P
workspace\\SmartP2P Server\\src\\MOEAD\\Exp"+id+"\\BestPF\\outBests.log",
"C:\\Users\\Christos\\Documents\\SmartP2P
workspace\\Figures\\figure3Duser"+id+".png");
	Thread.sleep(1000);

	this.sendFile("C:\\Users\\Christos\\Documents\\SmartP2P
workspace\\Figures\\figure3Duser"+id+".png");
}

if (Alg.equals("NSGAII"))
{
	IO.deleteDir(new File ("./" + "src/NSGAII/Exp" + id));
	IO.deleteDir(new File
("C:\\Users\\Christos\\Documents\\SmartP2P
workspace\\Figures\\figure3Duser"+id+".png"));
	NSGAII.Algorithm C = new
NSGAII.Algorithm(Integer.parseInt(id), word);

	Draw3DPlots draw = new Draw3DPlots ();
	draw.Draw("C:\\Users\\Christos\\Documents\\SmartP2P
workspace\\SmartP2P Server\\src\\MOEAD\\Exp"+id+"\\BestPF\\outBests.log",
"C:\\Users\\Christos\\Documents\\SmartP2P
workspace\\Figures\\figure3Duser"+id+".png");
	draw.Draw("C:\\Users\\Christos\\Documents\\SmartP2P
workspace\\SmartP2P Server\\src\\MOEAD\\Exp"+id+"\\BestPF\\outBests.log",
"C:\\Users\\Christos\\Documents\\SmartP2P
workspace\\Figures\\figure3Duser"+id+".png");
	Thread.sleep(1000);
```

```
                this.sendFile("C:\\Users\\Christos\\Documents\\SmartP2P
workspace\\Figures\\figure3Duser"+id+".png");
            }

            return true;
        }

        public void sendFile(String file) throws IOException {



            FileInputStream fis = new FileInputStream(file);
            byte[] buffer = new byte[fis.available()];
            fis.read(buffer);
            ObjectOutputStream oos = new
ObjectOutputStream(socket.getOutputStream()) ;
            oos.writeObject(buffer);

        }

        public void run() {

            while (true) {
                try {

                    String request = (String) Sinput.readObject();
                    System.out.println(request + ": ");

                    if (request.equals("SendProfile")) {
                        receiveANDaddProfile();
                    }
                    if (request.equals("SendInfo")) {
                        receiveANDaddInfo();
                    }
```



```
                    if (request.equals("GetTree")) {
                        sendTree();
                    }
                    if (request.equals("SendTree")) {
                        printTree(receiveTree());
                    }
                    if (request.equals("Register")) {
                        register();
                    }
                    if (request.equals("Terminate_Conn")) {
                        terminateConnection();
                    }
                    if (request.equals("Connect")) {
                        connect();
                    }
                    if (request.equals("GetGraph")) {
                        GetGraph();
                    }
                    if (request.equals("Search")) {
                        String req = (String) Sinput.readObject();
                        String[] SplittedReq = req.split(" ");
                        System.out.println("word:" + SplittedReq[0]);
                        System.out.println("id:" + SplittedReq[1]);
                    }

                } catch (IOException e) {
                    return;
                } catch (ClassNotFoundException o) {

                } catch (SQLException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (MatlabConnectionException e) {
```

```
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (MatlabInvocationException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }

        }

}
```

IO.java

```
package ServerApi;


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

public class IO {


   public static boolean deleteDir(File dir) {
```



```
      if (dir.isDirectory()) {
         String[] children = dir.list();
         for (int i=0; i<children.length; i++) {
            boolean success = deleteDir(new File(dir, children[i]));
            if (!success) {
               return false;
            }
         }
      }

      // The directory is now empty so delete it
      return dir.delete();
   }


      public static ArrayList<String[]> getFileContent(String file, String del) {

            String strLine;

            ArrayList<String[]> str = new ArrayList<String[]>();

            try {

                  FileInputStream fstream = new FileInputStream(file);

                  DataInputStream in = new DataInputStream(fstream);

                  BufferedReader br = new BufferedReader(new
InputStreamReader(in));

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

                        str.add(strLine.split(del));
```

```
            }

            in.close();

        } catch (Exception e) {

            System.err.println("Error: " + e.getMessage());

        }

        return str;

}

public static void displayFileContent(ArrayList<String[]> str) {

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

            //for (int j = 0; j < str.get(i).length; ++j)
            {

                System.out.print(str.get(i)[4] + " ");

            }

            System.out.println();

        }

}

static int find(String path, String word) {
```



```
        FileInputStream fstream;
        int count = 0;
        try {
            fstream = new FileInputStream(path);

            DataInputStream in = new DataInputStream(fstream);

            BufferedReader br = new BufferedReader(new
InputStreamReader(in));

            String strLine;
            String [] temp;
            while ((strLine = br.readLine()) != null)
            {

                temp = strLine.split(" ");
                for (int i=0; i<temp.length; ++i)
                {
                    String a;
                    a = temp[i].toLowerCase();
                    if (a.contains(word))
                    {
                        count++;
                        break;
                    }
                }

            }

            in.close();

        } catch (FileNotFoundException e) {
```

```
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

            return count;
        }
}
```

DBConnector.java

```
package ServerApi;

import java.sql.*;

import com.microsoft.sqlserver.jdbc.*;

/*
 * @author Christos
 */
public class DBConnector {
        private Connection con;
        private ResultSet rs;
        private Statement stmt; // for typical SQL statements
```



```
        private PreparedStatement pstmt; // for stored procedures
        private String connectionUrl;
        public DBConnector(String ip, int port, String DBName) {
            try {
                connectionUrl = "jdbc:sqlserver://"+ ip +":"+ port
+";databaseName=" + DBName +";integratedSecurity=true;";
        Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
                con = DriverManager.getConnection(connectionUrl);
            } catch (ClassNotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
        public ResultSet getReadOnlyRecordSet(String query) throws SQLException {
            rs = null;
```

```
        stmt = con.createStatement();

        rs = stmt.executeQuery(query);
        return rs;
}

public ResultSet GetResult() {
        return rs;
}

public void setResultSet(Object a) {
        rs = (ResultSet) a;
}

public void MakeStatement() throws SQLException {
        stmt = con.createStatement();

}

public Statement getStatement() {
        return stmt;
}

public void setPreparedStatement(PreparedStatement a) {
        pstmt = a;
        // pstmt = con.c

}

public Connection getConnection() {
        return con;
}
```



```
        public PreparedStatement getPreparedStatement() {
                return pstmt;
        }

        public void test() {

                System.out.println("Testing...");

                String name = "Koullis";
                try {

                        String SQL = "Select dbo.GET_USER_ID_BY_NAME('" +
name + "') AS NUM";

                        stmt = con.createStatement();

                        rs = stmt.executeQuery(SQL);

                        while (rs.next()) {

                                System.out.println(rs.getString("NUM") + " ");

                        }

                } catch (Exception e) {

                        e.printStackTrace();

                }
                System.out.println("OK");
        }
}
```

Client.java

```
package ServerApi;

import java.net.*;
import java.util.ArrayList;

import java.io.*;

public class Client {

	private ObjectInputStream Sinput;
	private ObjectOutputStream Soutput;
	private Socket socket;
	private ArrayList<String[]> tree;
	private int id;

	Client(int Id) {
		id = Id;
	}

	Client(Socket sock, int Id) throws IOException {

		id = Id;
		socket = sock;
		Sinput = new ObjectInputStream(socket.getInputStream());
		Soutput = new ObjectOutputStream(socket.getOutputStream());

	}

	Client(String ip, int port, int Id) {

		id = Id;
```



```
		connect(ip, port);
	}

	public void setTree (ArrayList<String[]> t)
	{
		this.tree = t;
	}

	public void RenewConnection(String ip, int port) throws IOException {

		socket = new Socket(ip, port);
		Sinput = new ObjectInputStream(socket.getInputStream());
		Soutput = new ObjectOutputStream(socket.getOutputStream());

	}

	public boolean connect(String ip, int port) {

		try {
			socket = new Socket(ip, port);
			Sinput = new ObjectInputStream(socket.getInputStream());
			Soutput = new ObjectOutputStream(socket.getOutputStream());
		} catch (UnknownHostException e) {
			return false;
		} catch (IOException e) {
			return false;
		}
		return true;
	}

	public void terminateConnection() throws IOException {
		Soutput.writeObject("Terminate");
		Soutput.flush();
```

```
        if (socket != null) {
                socket.close();
                socket = null;
        }
}

public void register() throws IOException {
        Soutput.writeObject("Register");
        Soutput.writeObject(id + "");
        Soutput.flush();
}

public ArrayList<String[]> peerTree() {
        return this.tree;
}

public boolean Request(String request) throws IOException,
                ClassNotFoundException {
        String[] req = request.split(" ");
        if (req[0].equals("GET")) {
                return receiveFile("a.txt");
        }
        if (req[0].equals("SEND")) {
                return sendFile("a.txt");
        }
        if (req[0].equals("GetTree")) {
                tree = getTree();
                //printTree (tree);
                if (tree != null)
                        return true;
                return false;
                // printTree (tree);
```



```
        }
        if (req[0].equals("sendTree")) {
                return sendTree(tree);
        }
        if (req[0].equals("Terminate")) {
                this.terminateConnection();
                return true;
        }
        if (req[0].equals("Register")) {
                this.register();
                return true;
        }
        if (req[0].equals("Search")) {
                this.Search(req[1], this.tree);
                return true;
        }

        return false;

}

/* Send tree to server */
public boolean sendTree(ArrayList<String[]> tree)
                throws UnknownHostException, IOException {

        if (this.socket != null && !this.socket.isClosed()
                        && this.socket.isConnected()) {

                Soutput.writeObject("SendTree");
                Soutput.flush();

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

```
                        Soutput.writeObject(tree.get(i)[0] + " " + tree.get(i)[1] + "
"
                                + tree.get(i)[2] + " " + tree.get(i)[3] + " "
                                + tree.get(i)[4]);
                        Soutput.flush();
                }
                Soutput.writeObject("NULL");
                Soutput.flush();
                return true;
        }
        return false;
    }

    /* Send tree to server */
    public boolean sendTree(ArrayList<String[]> tree, String ip, int port)
                throws UnknownHostException, IOException {

        Socket sock = new Socket(ip, port);

        if (sock == null || sock.isClosed() || !sock.isConnected())
                return false;

        ObjectOutputStream Soutp = new ObjectOutputStream(
                        sock.getOutputStream());
        Soutp.flush();
        ObjectInputStream Sinp = new
ObjectInputStream(sock.getInputStream());

        Soutp.writeObject("SendTree");
        Soutp.flush();

        for (int i = 0; i < tree.size(); ++i) {
                Soutp.writeObject(tree.get(i)[0] + " " + tree.get(i)[1] + " "
```



```
                        + tree.get(i)[2] + " " + tree.get(i)[3] + " "
                        + tree.get(i)[4]);
                Soutp.flush();
        }
        Soutp.writeObject("NULL");
        Soutp.flush();
        Soutp.writeObject("Terminate");
        Soutp.flush();
        sock.close();
        return true;
    }

    public void printTree(ArrayList<String[]> tree) {
        for (int i = 0; i < tree.size(); ++i) {
                System.out.println(tree.get(i)[0] + " " + tree.get(i)[1] + " "
                        + tree.get(i)[2] + " " + tree.get(i)[3] + " "
                        + tree.get(i)[4]);
        }

    }

    /* get tree from Server* */
    public ArrayList<String[]> getTree() throws UnknownHostException,
                IOException, ClassNotFoundException {
        if (this.socket != null && !this.socket.isClosed()
                        && this.socket.isConnected()) {

                Soutput.writeObject("GetTree");
                Soutput.flush();

                String str;
                ArrayList<String[]> tree = new ArrayList<String[]>();
                while (true) {
```

```
                str = (String) Sinput.readObject();
                if (str.equals("NULL"))
                    break;
                tree.add(str.split(" "));
            }
            return tree;
        }
        return null;
    }

    /* get tree from Server* */
    public ArrayList<String[]> getTree(String ip, int port)
            throws UnknownHostException, IOException,
ClassNotFoundException {
        Socket sock = new Socket(ip, port);

        if (sock == null || sock.isClosed() || !sock.isConnected())
            return null;

        ObjectOutputStream Soutp = new ObjectOutputStream(
                sock.getOutputStream());
        Soutp.flush();
        ObjectInputStream Sinp = new
ObjectInputStream(sock.getInputStream());

        Soutp.writeObject("GetTree");
        Soutp.flush();

        String str;
        ArrayList<String[]> tree = new ArrayList<String[]>();
        while (true) {
            str = (String) Sinp.readObject();
            if (str.equals("NULL"))
```



```
                break;
            tree.add(str.split(" "));
        }
        Soutp.writeObject("Terminate");
        Soutp.flush();
        sock.close();
        return tree;
    }

    /* Send file to server */
    public boolean sendFile(String file) throws UnknownHostException,
            IOException {
        if (this.socket != null && !this.socket.isClosed()
                && this.socket.isConnected()) {
            Socket sock = new
Socket(socket.getInetAddress().getHostAddress(),
                    socket.getPort());

            if (sock == null || sock.isClosed() || !sock.isConnected())
                return false;

            ObjectOutputStream Soutp = new ObjectOutputStream(
                    sock.getOutputStream());
            Soutp.flush();
            ObjectInputStream Sinp = new ObjectInputStream(
                    sock.getInputStream());

            Soutp.writeObject("Send");
            Soutp.flush();

            BufferedInputStream bis;
            BufferedOutputStream bos;
            byte[] byteArray;
```

```
                int in;
                try {
                        bis = new BufferedInputStream(new
FileInputStream(file));
                        bos = new
BufferedOutputStream(sock.getOutputStream());
                        byteArray = new byte[8192];
                        while ((in = bis.read(byteArray)) != -1) {
                                bos.write(byteArray, 0, in);
                        }
                        bis.close();
                        bos.flush();
                } catch (IOException e) {
                        e.printStackTrace();
                }
                Soutp.writeObject("Terminate");
                Soutp.flush();
                sock.close();
                return true;
        }
        return false;
}

/* Send file to server */
public boolean sendFile(String file, String ip, int port)
                throws UnknownHostException, IOException {

        Socket sock = new Socket(ip, port);

        if (sock == null || sock.isClosed() || !sock.isConnected())
                return false;

        ObjectOutputStream Soutp = new ObjectOutputStream(
```



```
                        sock.getOutputStream());
                Soutp.flush();
                ObjectInputStream Sinp = new
ObjectInputStream(sock.getInputStream());

                Soutp.writeObject("Send");
                Soutp.flush();

                BufferedInputStream bis;
                BufferedOutputStream bos;
                byte[] byteArray;
                int in;
                try {
                        bis = new BufferedInputStream(new FileInputStream(file));
                        bos = new BufferedOutputStream(sock.getOutputStream());
                        byteArray = new byte[8192];
                        while ((in = bis.read(byteArray)) != -1) {
                                bos.write(byteArray, 0, in);
                        }
                        bis.close();
                        bos.flush();
                } catch (IOException e) {
                        e.printStackTrace();
                }
                Soutp.writeObject("Terminate");
                Soutp.flush();
                sock.close();
                return true;
        }

        /* request file from server */
        public boolean receiveFile(String file) throws UnknownHostException,
                        IOException {
```

```
            if (this.socket != null && !this.socket.isClosed()
                    && this.socket.isConnected()) {
                Socket sock = new Socket(socket.getInetAddress().getHostAddress(),
                        socket.getPort());

                if (sock == null || sock.isClosed() || !sock.isConnected())
                    return false;

                ObjectOutputStream Soutp = new ObjectOutputStream(
                        sock.getOutputStream());
                Soutp.flush();
                ObjectInputStream Sinp = new ObjectInputStream(
                        sock.getInputStream());
                Soutp.writeObject("Get");
                Soutp.flush();

                BufferedInputStream bis;
                BufferedOutputStream bos;
                byte[] data;
                int in;

                try {
                    byte[] receivedData = new byte[8192];
                    bis = new BufferedInputStream(sock.getInputStream());
                    bos = new BufferedOutputStream(new FileOutputStream(file));
                    while ((in = bis.read(receivedData)) != -1) {
                        bos.write(receivedData, 0, in);
                    }
                    bos.close();
                } catch (IOException e) {
```



```
                    e.printStackTrace();
                }

                Soutp.writeObject("Terminate");
                Soutp.flush();
                sock.close();
                return true;
            }
            return false;
        }

        /* request file from server */
        public boolean receiveFile(String file, String ip, int port)
                throws UnknownHostException, IOException {

            Socket sock = new Socket(ip, port);

            if (sock == null || sock.isClosed() || !sock.isConnected())
                return false;

            ObjectOutputStream Soutp = new ObjectOutputStream(
                    sock.getOutputStream());
            Soutp.flush();
            ObjectInputStream Sinp = new ObjectInputStream(sock.getInputStream());
            Soutp.writeObject("Get");
            Soutp.flush();

            BufferedInputStream bis;
            BufferedOutputStream bos;
            byte[] data;
            int in;
```

```
        try {
                byte[] receivedData = new byte[8192];
                bis = new BufferedInputStream(sock.getInputStream());
                bos = new BufferedOutputStream(new FileOutputStream(file));
                while ((in = bis.read(receivedData)) != -1) {
                        bos.write(receivedData, 0, in);
                }
                bos.close();
        } catch (IOException e) {
                e.printStackTrace();
        }

        Soutp.writeObject("Terminate");
        Soutp.flush();
        sock.close();
        return true;
}

public void Search(String word, ArrayList<String[]> tree)
                throws NumberFormatException, IOException {

                        try {
                                Soutput.writeObject("Search");
                                Soutput.writeObject(word + " " + tree.get(0)[0]);
                                Soutput.flush();
                        } catch (IOException e) {
                        }
}

//thewreite oti o QP einai o 0
public void AnswerToQP(String ans) throws NumberFormatException,
                IOException {
        this.connect(tree.get(0)[3], Integer.parseInt(tree.get(0)[4]));
```



```
                try {
                        Soutput.writeObject("Answer");
                        Soutput.writeObject(ans);
                        Soutput.flush();
                } catch (IOException e) {
                }
}
}
```

Εφαρμογή SmartP2P

Server.java

```
package dmsl.Smartp2p.core;

import java.io.*;

import java.net.*;

public class Server extends Thread {

        public ServerSocket peerSocket;
        private int id;

        public Server (int port, int id1)
        {
                id=id1;
                try {
                        peerSocket = new ServerSocket(port);
                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
```

```
            start();
        }

        public void run() {

            try     {
                while (true){

                    Socket socket = peerSocket.accept();
                    JoinedPeer t = new JoinedPeer(socket, id);
                    t.start();
                }
            }
            catch (IOException e) {
                System.out.println("Exception on new PeerSocket: " + e);
            }
        }

        public boolean isBound()
        {
            return this.peerSocket.isBound();
        }
}
```

JoinedPeer.java

```
package dmsl.Smartp2p.core;

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
```



```
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.OptionalDataException;
import java.net.Socket;
import java.util.ArrayList;

import dmsl.Smartp2p.app.Message;


class JoinedPeer extends Thread {

        byte[] data;
        Socket socket;
        ObjectInputStream Sinput;
        ObjectOutputStream Soutput;
        String id;
        ArrayList<String[]> tree;

        public void sendTree(ArrayList<String[]> tree) throws IOException {
            for (int i = 0; i < tree.size(); ++i) {
                Soutput.writeObject(tree.get(i)[0] + " " + tree.get(i)[1] + " "
                                    + tree.get(i)[2] + " " + tree.get(i)[3]/* + " "
                                    + tree.get(i)[4]*/);
                Soutput.flush();
            }
            Soutput.writeObject("NULL");
            Soutput.flush();
        }

        public void printTree(ArrayList<String[]> tree) {
```

```
        for (int i = 0; i < tree.size(); ++i) {
                System.out.println(tree.get(i)[0] + " " + tree.get(i)[1] + " "
                                + tree.get(i)[2] + " " + tree.get(i)[3] + " "
                                + tree.get(i)[4]);
        }

}

public ArrayList<String[]> receiveTree() throws IOException,
                ClassNotFoundException {
        ArrayList<String[]> tree = new ArrayList<String[]>();
        String str;

        while (true) {
                str = (String) Sinput.readObject();
                if (str.equals("NULL"))
                        break;
                tree.add(str.split(" "));
        }

        //printTree(tree);
        return tree;

}

JoinedPeer (Socket socket, int id1) throws IOException {

        //con = new DBConnector("127.0.0.1", 1433, "SMARTP2P");
        this.socket = socket;
        Soutput = new ObjectOutputStream(socket.getOutputStream());
        Soutput.flush();
        Sinput = new ObjectInputStream(socket.getInputStream());
        id = String.valueOf(id1);
```



```
        }

        public void Search(String word){

                for (int i=0; i<tree.size(); ++i)
                {
                        if (tree.get(i)[1].equals(id))
                        {
                                Client c = new Client (tree.get(i)[2],
Integer.parseInt(tree.get(i)[3]), Integer.parseInt(id), "");
                                c.sendTree(tree, word);
                        }
                }
                /*if (id.equals(tree.get(0)[1]))
                return;*/
                ArrayList <String> result =
IO.find("/mnt/sdcard/SmartP2P/profile"+id+".txt", word);
                //int res = IO.find("/mnt/sdcard/SmartP2P/profile.txt", word);
                if (result!=null && result.size()>0)
                {
                        Client c2 = new Client (tree.get(0)[2],
Integer.parseInt(tree.get(0)[3]), Integer.parseInt(id), "");
                        c2.answer(result);
                }
                //Client pReq = new Client(Integer.parseInt(id));
                //pReq.setTree(this.receiveTree());
                //pReq.Request("Search " + req);
                /*
                 * int count; count = IO.find("/mnt/sdcard/SmartP2PV2/"+ t +"/DBLP/"+
t
                 *      +       "."     +       socket.getLocalPort()   %       Message.netSize
+".dblp.keyword.txt",
                 * req); if (count > 0){ pReq.SendRequest(String.valueOf(count) + " " +
```

```
            * socket.getLocalPort() % Message.netSize, "ANSWER"); }
            */

    }

    public void answer () throws OptionalDataException, ClassNotFoundException,
IOException
    {
        while (true)
        {
            String res = (String) Sinput.readObject();
            if (res.equals("NULL"))
                break;
            Message.change(res);
            //String[] num = res.split(" ");
            Message.Sum(/*num[1]*/);
        }
    }

    public void run() {

        while (true) {
            try {

                String request = (String) Sinput.readObject();
                System.out.println(request + ": ");

                if (request.equals("Search")) {
                    String req = (String) Sinput.readObject();



                    tree = this.receiveTree();
                    Search(req);
```



```
                }

                if (request.equals("Answer"))
                {
                    answer ();
                }

            } catch (IOException e) {
                return;
            } catch (ClassNotFoundException o) {

            }
        }
    }
}
```

Client.java

```
package dmsl.Smartp2p.core;

import java.net.*;
import java.util.ArrayList;

import java.io.*;

import android.os.Environment;

public class Client {

    private ObjectInputStream Sinput;
    private ObjectOutputStream Soutput;
    private Socket socket;
    private ArrayList<String[]> tree;
```

```
private int id;
private String pass;

public Client(int Id, String pas) {
	id = Id;
	pass = pas;
}

public Client(Socket sock, int Id, String pas)  {

	id = Id;
	pass = pas;
	socket = sock;
	try {
		Sinput = new ObjectInputStream(socket.getInputStream());
		Soutput = new ObjectOutputStream(socket.getOutputStream());
		Soutput.flush();
	} catch (StreamCorruptedException exception) {
		return;
	} catch (IOException exception) {
		return;
	}


}

public Client(String ip, int port, int Id, String pas) {

	id = Id;
	pass = pas;
	connect(ip, port);
}
```



```
	public void setTree(ArrayList<String[]> t) {
		this.tree = t;
	}

	public boolean RenewConnection(String ip, int port) {

		try {
			SocketAddress sockaddr = new InetSocketAddress(ip, port);
			socket = new Socket();
			socket.connect(sockaddr);
			if (socket.isConnected())
			{
				Sinput = new
ObjectInputStream(socket.getInputStream());
				Soutput = new
ObjectOutputStream(socket.getOutputStream());
				Soutput.flush();
				return true;
			}
		} catch (UnknownHostException exception) {
			return false;
		} catch (IOException exception) {
			return false;
		}
		return false;
	}

	@SuppressWarnings("finally")
	public boolean connect(String ip, int port) {

		socket = null;
		try {
			SocketAddress sockaddr = new InetSocketAddress(ip, port);
```

```
                socket = new Socket();
                socket.connect(sockaddr);
                if (socket.isConnected())
                {
                        Sinput = new ObjectInputStream(socket.getInputStream());
                        Soutput = new ObjectOutputStream(socket.getOutputStream());
                        Soutput.flush();
                        return true;
                }

        } catch (UnknownHostException e) {
                return false;
        } catch (IOException e) {
                return false;
        }
        return false;

}

static public String register(String Name, String pass, String ip, int port)  {
        Socket sock;
        ObjectInputStream Sinp = null;
        ObjectOutputStream Soutp = null;
        try {
                SocketAddress sockaddr = new InetSocketAddress(ip, port);
                sock = new Socket();
                sock.connect(sockaddr);
                if (sock.isConnected())
                {
                        Sinp = new ObjectInputStream(sock.getInputStream());
```



```
                        Soutp = new ObjectOutputStream(sock.getOutputStream());
                        Soutp.flush();

                }

                Soutp.writeObject("Register");
                Soutp.writeObject(Name);
                Soutp.writeObject(pass);
                Soutp.flush();

                String reply = (String) Sinp.readObject();
                if (reply.equals("NOT OK")){
                        return "Not Complited1";
                }
                if (reply.equals("User Name Already Exists")){
                        return reply;
                }
                return reply;
        } catch (UnknownHostException exception) {
                return "No Connection With Server";
        } catch (IOException exception) {
                return "No Connection With Server";
        } catch (ClassNotFoundException exception) {
                return "Not Complited4";
        }

}

public ArrayList<String[]> peerTree() {
        return this.tree;
}
```

```
public boolean goOnline(String port) {
	try {
		Soutput.writeObject("Connect");
		Soutput.writeObject(Integer.toString(id));
		Soutput.writeObject(pass);
		Soutput.writeObject(port);
		Soutput.flush();

		String reply = (String) Sinput.readObject();
		if (reply.equals("OK")){
			this.pass=pass;
			return true;
		}

	} catch (IOException exception) {
		return false;
	} catch (ClassNotFoundException exception) {
		return false;
	}

	return false;
}

public boolean terminateConnection(int id) {
	try {
		Soutput.writeObject("Terminate_Conn");
		Soutput.writeObject(Integer.toString(id));
		Soutput.flush();

		String reply = (String) Sinput.readObject();
		if (reply.equals("OK"))
			return true;
	} catch (IOException exception) {
```



```
			return false;
		} catch (ClassNotFoundException exception) {
			return false;
		}

		return false;
	}

	public String receiveFile(String file) throws ClassNotFoundException,
StreamCorruptedException, IOException {


		ObjectInputStream ois;
		try {
			ois = new ObjectInputStream(socket.getInputStream());
		} catch (StreamCorruptedException e) {
			return "1";
		} catch (IOException e) {
			return "1";
		}

		byte[] buffer;
		try {
			buffer = (byte[])ois.readObject();
		} catch (OptionalDataException e) {
			return "2";
		} catch (IOException e) {
			return "2";
		}

		FileWriter fstream;
		try {
			fstream = new FileWriter(file);
```

```
            /*BufferedWriter out = new BufferedWriter(fstream);
            out.write(buffer.toString());*/
        } catch (IOException e2) {
        }


        FileOutputStream fos;
        try {
            fos = new FileOutputStream(new File (file));
        } catch (FileNotFoundException e) {
            return "3";
        }
        try {
            fos.write(buffer);
            fos.close();
        } catch (IOException e) {
            return "4";
        }

        return "0";

    }

    public String getGraph(int id, String Alg, String word) throws IOException,
ClassNotFoundException
    {
        Soutput.writeObject("GetGraph");
        Soutput.writeObject(Integer.toString(id));
        Soutput.writeObject(word);
        Soutput.writeObject(Alg);

        Soutput.flush();
```



```
            return
this.receiveFile(Environment.getExternalStorageDirectory().toString()+"/SmartP2P/pic"
+ id +".png");
            //System.out.print("ok");



    }

    public boolean Request(String request) throws IOException,
ClassNotFoundException{
        String[] req = request.split(" ");

        if (req[0].equals("CONNECT")) {
            return goOnline(req[1]);
        }
        if (req[0].equals("TERMINATE_CONN")) {
            return terminateConnection(id);
        }
        if (req[0].equals("GetTree")) {
            tree = getTree(Integer.parseInt(req[1]), req[2], req[3], req[4],
req[5], req[6], req[6]);
            // printTree (tree);
            if (tree != null)
                return true;
            return false;
            // printTree (tree);
        }
        if (req[0].equals("sendTree")) {
            return sendTree(tree, req[1]);
        }
        if (req[0].equals("GetGraph")) {
            getGraph(Integer.parseInt(req[1]), req[2], req[3]);
```

```
        }
        /*if (req[0].equals("Register")) {
                System.out.println(this.register(req[1]));
                return true;
        }*/
        if (req[0].equals("Search")) {
                try {
                        this.Search(req[1], this.tree);
                } catch (NumberFormatException exception) {
                        return false;
                }
                return true;
        }
        if (req[0].equals("Send_Profile")) {
                return this.sendProfile("profile.txt", socket.getInetAddress()
                                .getHostAddress(), socket.getPort());

        }
        if (req[0].equals("Send_Info")) {
                return this.sendInfo(socket.getInetAddress().getHostAddress(),
                                socket.getPort(), req[1], req[2]);

        }

        return false;

}

public boolean sendInfo(String ip, int port, String Lon, String Lat) {

        Socket sock;
        try {
                /*sock = new Socket(ip, port);
```



```
                ObjectOutputStream Soutp = new ObjectOutputStream(
                                sock.getOutputStream());
                Soutp.flush();
                ObjectInputStream Sinp = new
ObjectInputStream(sock.getInputStream());*/

                Soutput.writeObject("SendInfo");
                Soutput.writeObject(id + "");
                Soutput.flush();



                Soutput.writeObject(Lon + " " + Lat);
                Soutput.flush();

                String reply = (String) Sinput.readObject();
                if (reply.equals("OK"))
                        return true;
        } catch (UnknownHostException exception) {
                return false;
        } catch (IOException exception) {
                return false;
        } catch (ClassNotFoundException exception) {
                return false;
        }


        return false;

}

public boolean sendProfile(String file, String ip, int port){

        ArrayList<String[]> aList;
```

```
Socket sock;
try {
	sock = new Socket(ip, port);

	ObjectOutputStream Soutp = new ObjectOutputStream(
			sock.getOutputStream());
	Soutp.flush();
	ObjectInputStream Sinp = new
ObjectInputStream(sock.getInputStream());

	Soutp.writeObject("SendProfile");
	Soutp.writeObject(id + "");
	Soutp.flush();

	aList = IO.getFileContent(file, "\n");

	for (int i = 0; i < aList.size(); i++) {
		Soutp.writeObject(aList.get(i)[0]);
		Soutp.flush();
	}
	Soutp.writeObject("NULL");
	Soutp.flush();

	String reply = (String) Sinp.readObject();
	if (reply.equals("OK"))
		return true;
} catch (UnknownHostException exception) {
	return false;
} catch (IOException exception) {
	return false;
} catch (ClassNotFoundException exception) {
	return false;
}
```



```
	return false;

}

/* Send tree to server */
public boolean sendTree(ArrayList<String[]> tree, String word){

	if (this.socket != null && !this.socket.isClosed()
			&& this.socket.isConnected()) {

		try {
			Soutput.writeObject("Search");
			Soutput.writeObject(word);
			Soutput.flush();

			for (int i = 0; i < tree.size(); ++i) {
				Soutput.writeObject(tree.get(i)[0] + " " +
tree.get(i)[1] + " "
						+ tree.get(i)[2] + " " +
tree.get(i)[3]);
				Soutput.flush();
			}
			Soutput.writeObject("NULL");
			Soutput.flush();
			return true;
		} catch (IOException exception) {
			// TODO Auto-generated catch-block stub.
			exception.printStackTrace();
		}

	}
	return false;
```

```
	}

	/* Send tree to server */
	public boolean sendTree(ArrayList<String[]> tree, String ip, int port){

		Socket sock;
		try {
			sock = new Socket(ip, port);
			if (sock == null || sock.isClosed() || !sock.isConnected())
				return false;

			ObjectOutputStream Soutp = new ObjectOutputStream(
					sock.getOutputStream());
			Soutp.flush();
			ObjectInputStream Sinp = new
ObjectInputStream(sock.getInputStream());

			Soutp.writeObject("SendTree");
			Soutp.flush();

			for (int i = 0; i < tree.size(); ++i) {
				Soutp.writeObject(tree.get(i)[0] + " " + tree.get(i)[1] + " "
						+ tree.get(i)[2] + " " + tree.get(i)[3] + " "
						+ tree.get(i)[4]);
				Soutp.flush();
			}
			Soutp.writeObject("NULL");
			Soutp.flush();
			/*Soutp.writeObject("Terminate");
			Soutp.flush();*/
			sock.close();
		} catch (UnknownHostException exception) {
			return false;
```



```
		} catch (IOException exception) {
			return false;
		}

		return true;
	}

	public void printTree(ArrayList<String[]> tree) {
		for (int i = 0; i < tree.size(); ++i) {
			System.out.println(tree.get(i)[0] + " " + tree.get(i)[1] + " "
					+ tree.get(i)[2] + " " + tree.get(i)[3] + " "
					+ tree.get(i)[4]);
		}

	}

	/* get tree from Server* */
	public ArrayList<String[]> getTree(int id, String pass, String decisionMaking,
String time, String recall, String Alg, String word) {
		if (this.socket != null && !this.socket.isClosed()
				&& this.socket.isConnected()) {

			try {
				Soutput.writeObject("GetTree");
				Soutput.flush();

				Soutput.writeObject(id+"");
				Soutput.writeObject(pass);
				Soutput.writeObject(decisionMaking);
				Soutput.writeObject(time);
				Soutput.writeObject(recall);
				Soutput.writeObject(Alg);
				if (Alg.equals("RW") || Alg.equals("BFS"))
```

```
                Soutput.writeObject(word);
            Soutput.flush();

            String str;
            ArrayList<String[]> tree = new ArrayList<String[]>();
            while (true) {
                str = (String) Sinput.readObject();
                if (str.equals("NULL"))
                    break;
                tree.add(str.split(" "));
            }
            return tree;
        } catch (IOException exception) {
            return null;
        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
    return null;
}

/* get tree from Server* */
public ArrayList<String[]> getTree(String ip, int port){
    Socket sock;
    try {
        sock = new Socket(ip, port);
        if (sock == null || sock.isClosed() || !sock.isConnected())
            return null;

        ObjectOutputStream Soutp = new ObjectOutputStream(
                sock.getOutputStream());
        Soutp.flush();
```



```
        ObjectInputStream Sinp = new
ObjectInputStream(sock.getInputStream());

        Soutp.writeObject("GetTree");
        Soutp.flush();

        String str;
        ArrayList<String[]> tree = new ArrayList<String[]>();
        while (true) {
            str = (String) Sinp.readObject();
            if (str.equals("NULL"))
                break;
            tree.add(str.split(" "));
        }
        sock.close();
    } catch (UnknownHostException exception) {
        return null;
    } catch (IOException exception) {
        return null;
    } catch (ClassNotFoundException exception) {
        return null;
    }
    return tree;
}


public void Search(String word, ArrayList<String[]> tree){
    try {
        Soutput.writeObject("Search");
        Soutput.writeObject(word + " " + tree.get(0)[0]);
        Soutput.flush();
    } catch (IOException e) {
    }
```

```
	}

	public void answer (int res)
	{
		try {
			Soutput.writeObject("Answer");
			Soutput.writeObject(id + " " + res);
			Soutput.flush();
		} catch (IOException e) {
		}
	}

	public void answer (ArrayList<String> res)
	{
		try {
			Soutput.writeObject("Answer");
			Soutput.flush();

			for (int i=0; i<res.size(); ++i)
			{
				Soutput.writeObject(res.get(i));
				Soutput.flush();
			}
			Soutput.writeObject("NULL");
			Soutput.flush();
		} catch (IOException e) {
		}
	}
}
```


SmartP2PActivity.java

```
package dmsl.Smartp2p.app;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.OutputStream;
import java.io.OutputStreamWriter;

import dmsl.Smartp2p.core.Client;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.os.Environment;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnTouchListener;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.Toast;


public class SmartP2PActivity extends Activity {
   /** Called when the activity is first created. */
```

```
        Context context;
        int duration;
    @Override
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.intro);

                context = getApplicationContext();
                duration = Toast.LENGTH_LONG;
    }

        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
          MenuInflater inflater = getMenuInflater();
          inflater.inflate(R.menu.intro_menu, menu);
          return true;
        }

        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
          // Handle item selection
          switch (item.getItemId()) {
          case R.id.register:

                        Intent startReg = new Intent(SmartP2PActivity.this,
RegisterActivity.class);
                        startActivity(startReg);

              return true;
          case R.id.connect:

                        Intent startCon = new Intent(SmartP2PActivity.this,
ConnectActivity.class);
```



```
                        startActivity(startCon);

              return true;
            case R.id.exit:
                        finish();
              return true;
            default:
              return super.onOptionsItemSelected(item);
            }
          }
}
```

ConnectActivity.java

```
package dmsl.Smartp2p.app;

import java.io.IOException;
import java.net.Socket;
import java.net.UnknownHostException;

import dmsl.Smartp2p.core.Client;
import dmsl.Smartp2p.core.KeepAlivePeer;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
```

```
import android.widget.Toast;

public class ConnectActivity extends Activity {

	private TextView txIp;
	private TextView txPort;
	private TextView txId;
	private TextView txPass;
	private TextView txMyPort;
	private Button connectButton;
	KeepAlivePeer k;
	Client c;
	String id;

	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.connect);
		connectBut();
	}

	void connectBut() {

		connectButton = (Button) findViewById(R.id.button1);
		connectButton.setOnClickListener(new OnClickListener() {
			public void onClick(View v) {
				txIp = (TextView) findViewById(R.id.editText1);
				String ip = txIp.getText().toString();
				txPort = (TextView) findViewById(R.id.editText2);
				String port = txPort.getText().toString();
				txId = (TextView) findViewById(R.id.editText3);
				id = txId.getText().toString();
				txPass = (TextView) findViewById(R.id.editText4);
```



```
				String pass = txPass.getText().toString();
				/*txMyPort = (TextView) findViewById(R.id.editText5);
				String myPort = txMyPort.getText().toString();*/

				Intent startSearch = new Intent(ConnectActivity.this,
SearchActivity.class);
				Bundle b = new Bundle();
				b.putString("ip", ip);
				b.putString("port", port);
				b.putString("id", id);
				b.putString("pass", pass);
				b.putString("myPort", "45001"/*myPort*/);
				startSearch.putExtras(b);
				startActivity(startSearch);
			}
		});
	}
}
```

SearchActivity.java

```
package dmsl.Smartp2p.app;

import java.io.IOException;
import java.net.ServerSocket;
import java.util.ArrayList;

import dmsl.Smartp2p.core.Client;
import dmsl.Smartp2p.core.IO;
import dmsl.Smartp2p.core.Server;
import android.app.Activity;
import android.content.Context;
```

```
import android.content.Intent;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.RadioButton;
import android.widget.TextView;
import android.widget.Toast;

public class SearchActivity extends Activity {

	String ip;
	int port;
	int id;
	int myPort;
	String pass;
	Client c;

	Context context;
	int duration;

	private Button GoButton;
	private RadioButton RButton1;
	private RadioButton RButton2;
	private RadioButton RButton3;
	private RadioButton RButton4;

	Server myServer = null;
```



```
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.search);

		Bundle extra = getIntent().getExtras();
		if (extra != null) {
			ip = extra.getString("ip");
			port = Integer.parseInt(extra.getString("port"));
			id = Integer.parseInt(extra.getString("id"));
			pass = extra.getString("pass");
			myPort  =  Integer.parseInt(extra.getString("myPort"));//port  pou
akouw gia na 3erei o server

			context = getApplicationContext();
			duration = Toast.LENGTH_LONG;
			if (ConnectToServer()){
				myServer = new Server (45001, id);
				if (!myServer.isBound())
					finish();
				this.myPort = 45001;

				goBut();
			}
			else
				finish();


		}

	}
```

```
	void terminate() {

		if (id != 0)
			c.terminateConnection(id);

		this.stopService(this.getIntent());
		this.finish();

	}

	boolean ConnectToServer() {
		c = new Client(id, pass);
		if (!c.connect(ip, port)) {
			Toast toast = Toast.makeText(context,
					"No Connection Available On Server", duration);
			toast.show();
			return false;
		}

		if (c.goOnline(myPort + "")) {
			if
(!c.sendProfile("/mnt/sdcard/SmartP2P/profile"+id+".txt",ip,port))
			{
				Toast toast = Toast.makeText(context,
						"Cannot Upload Profile", duration);
				toast.show();
				terminate();
			}



			ArrayList <String []> gps =
IO.getFileContent("/mnt/sdcard/SmartP2P/gps.txt", " ");
```



```
			if (!c.sendInfo(ip, port, gps.get(id-1)[1], gps.get(id-1)[0]))
			{
				Toast toast = Toast.makeText(context,
						"Cannot Send GPS Info", duration);
				toast.show();
				terminate();
			}

		} else {
			Toast toast = Toast.makeText(context,
					"No Connection Available On Database",
duration);
			toast.show();
			return false;
		}
		return true;

	}

	@Override
	public boolean onCreateOptionsMenu(Menu menu) {
		MenuInflater inflater = getMenuInflater();
		inflater.inflate(R.menu.search_menu, menu);
		return true;
	}

	@Override
	public boolean onOptionsItemSelected(MenuItem item) {
		// Handle item selection
		switch (item.getItemId()) {
		case R.id.terminate:
			try {
```

```
                this.myServer.peerSocket.close();
                this.myServer.stop();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            terminate();
            return true;
        default:
            return super.onOptionsItemSelected(item);
        }
}

@Override
public void onBackPressed() {
        // TODO Auto-generated method stub
        super.onBackPressed();
        try {
            this.myServer.peerSocket.close();
            this.myServer.stop();
        } catch (IOException e) {

        }
        terminate();
}

void goBut() {

        GoButton = (Button) findViewById(R.id.button1);
        GoButton.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {

                Message.change("NULL");
```



```
                TextView txWord = (TextView)
findViewById(R.id.editText1);
                String word = txWord.getText().toString();
                RButton1 = (RadioButton) findViewById(R.id.radio0);
                RButton2 = (RadioButton) findViewById(R.id.radio1);
                RButton3 = (RadioButton) findViewById(R.id.radio2);
                RButton4 = (RadioButton) findViewById(R.id.radio3);

                if (RButton3.isChecked())
                {
                    Intent startSearch1 = new
Intent(SearchActivity.this, Search1Activity.class);
                    Bundle b = new Bundle();
                    b.putString("ip", ip);
                    b.putString("port", port+"");
                    b.putString("id", id+"");
                    b.putString("pass", pass);
                    b.putString("word", word);
                    b.putString("Alg", "MOEAD");
                    startSearch1.putExtras(b);
                    startActivity(startSearch1);
                }
                if (RButton1.isChecked())
                {
                    Intent startSearch1 = new
Intent(SearchActivity.this, ResultList.class);
                    Bundle b = new Bundle();
                    b.putString("ip", ip);
                    b.putString("port", port+"");
                    b.putString("id", id+"");
                    b.putString("pass", pass);
                    b.putString("word", word);
                    b.putString("Alg", "RW");
```

```
                startSearch1.putExtras(b);
                startActivity(startSearch1);
            }
            if (RButton2.isChecked())
            {
                Intent startSearch1 = new
Intent(SearchActivity.this, ResultList.class);
                Bundle b = new Bundle();
                b.putString("ip", ip);
                b.putString("port", port+"");
                b.putString("id", id+"");
                b.putString("pass", pass);
                b.putString("word", word);
                b.putString("Alg", "BFS");
                startSearch1.putExtras(b);
                startActivity(startSearch1);
            }

            if (RButton4.isChecked())
            {
                Intent startSearch1 = new
Intent(SearchActivity.this, Search1Activity.class);
                Bundle b = new Bundle();
                b.putString("ip", ip);
                b.putString("port", port+"");
                b.putString("id", id+"");
                b.putString("pass", pass);
                b.putString("word", word);
                b.putString("Alg", "NSGAII");
                startSearch1.putExtras(b);
                startActivity(startSearch1);
            }
```



```
            }
        });
    }

}
```

Search1Activity.java

```
package dmsl.Smartp2p.app;

import java.io.File;
import java.io.IOException;

import dmsl.Smartp2p.core.Client;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.os.Environment;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.ImageView;
import android.widget.SeekBar;
import android.widget.TextView;
import android.widget.Toast;
```

```
public class Search1Activity extends Activity implements
SeekBar.OnSeekBarChangeListener {

    ImageView im;
  SeekBar mSeekBar;
  TextView mProgressText;
  Double time, recall;
  int decisionMaking = 0;
  private Button GoButton;
  String word;

    String ip;
    int port;
    int id;
    String pass;
    String Algorithm;

    Client c;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.search1);

        Bundle extra = getIntent().getExtras();
        if (extra != null) {
            ip = extra.getString("ip");
            port = Integer.parseInt(extra.getString("port"));
            id = Integer.parseInt(extra.getString("id"));
            pass = extra.getString("pass");
            word = extra.getString("word");
            Algorithm = extra.getString("Alg");
```



```
        }

        c = new Client(id, pass);
        if (!c.connect(ip, port)) {
            finish();
        }
        try {
            c.getGraph(id, Algorithm, word);
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        im = (ImageView) findViewById(R.id.imageView1);


        File imgFile = new
File(Environment.getExternalStorageDirectory().toString()+"/SmartP2P/pic" + id
+".png");
        if(imgFile.exists()){

            Bitmap myBitmap =
BitmapFactory.decodeFile(imgFile.getAbsolutePath());

            im.setImageBitmap(myBitmap);

        }
    mSeekBar = (SeekBar)findViewById(R.id.seekBar1);

    mSeekBar.setOnSeekBarChangeListener(this);
```

```
mProgressText = (TextView)findViewById(R.id.progress);
this.mProgressText.setText(null);
time=0.5;
recall=0.5;


CheckBox repeatChkBx = ( CheckBox ) findViewById( R.id.checkBox1);
repeatChkBx.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener()
{
   public void onCheckedChanged(CompoundButton buttonView, boolean
isChecked)
   {
      if ( isChecked )
      {
         decisionMaking=1;
      }
      else
           decisionMaking=0;

   }
});

GoBut();

   }
   @Override
   public void onProgressChanged(SeekBar seekBar, int progress,
                  boolean fromUser) {
   recall=Double.parseDouble(progress+"")/100;
   time=Double.parseDouble((100-progress)+"")/100;
mProgressText.setText(null);
```



```
   }
   @Override
   public void onStartTrackingTouch(SeekBar seekBar) {
         // TODO Auto-generated method stub

   }
   @Override
   public void onStopTrackingTouch(SeekBar seekBar) {
         // TODO Auto-generated method stub

   }


   void GoBut() {

         GoButton = (Button) findViewById(R.id.button1);
         GoButton.setOnClickListener(new OnClickListener() {
               public void onClick(View v) {
                     Message.change("NULL");

                     Intent getTree = new Intent(Search1Activity.this,
ResultList.class);
                     Bundle b = new Bundle();
                     b.putString("ip", ip);
                     b.putString("port", port+"");
                     b.putString("id", id+"");
                     b.putString("pass", pass);
                     b.putString("dec", String.valueOf(decisionMaking));
                     b.putString("time", String.valueOf(time));
                     b.putString("recall", String.valueOf(recall));
                     b.putString("word", word);
                     b.putString("Alg", Algorithm);
                     getTree.putExtras(b);
```

```
                startActivity(getTree);

            }
        });
    }

}
```

ResultList.java

```
package dmsl.Smartp2p.app;

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

import dmsl.Smartp2p.core.Client;
import dmsl.Smartp2p.core.IO;
import android.app.Activity;
import android.app.ListActivity;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListAdapter;
import android.widget.TextView;
import android.widget.Toast;
```



```
public class ResultList extends ListActivity{

    int id;
    String pass;
    String ip;
    int port;
    String Algorithm;

    String time, recall, decisionMaking, word;
    Client c;


    Context context;
    int duration;


    Button printButton;

    ArrayList <String []> tree;
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.results);

    Bundle extra = getIntent().getExtras();
            if (extra != null) {
                ip = extra.getString("ip");
                port = Integer.parseInt(extra.getString("port"));
                id = Integer.parseInt(extra.getString("id"));
                pass = extra.getString("pass");

                time = extra.getString("time");
                recall = extra.getString("recall");
                decisionMaking = extra.getString("dec");
```

```
                word = extra.getString("word");
                Algorithm = extra.getString("Alg");

                context = getApplicationContext();
                duration = Toast.LENGTH_LONG;
        }

        printBut();

        connect();

        requestForTree ();

}

void connect()
{
    c = new Client(id, pass);
        if (!c.connect(ip, port)) {
                Toast toast = Toast.makeText(context,
                                "No Connection Available On Server", duration);
                toast.show();
                finish();
        }
}

void requestForTree ()
{
    tree = c.getTree(id, pass, decisionMaking, time, recall, Algorithm, word);
    for (int i=1; i<tree.size(); ++i)
        {
                if (tree.get(i)[1].equals(id+""))
                {
```



```
                        Client c = new Client (tree.get(i)[2],
Integer.parseInt(tree.get(i)[3]), id, "");
                        c.sendTree(tree, word);
                        /*Toast toast = Toast.makeText(context,
                                tree.get(i)[0] + "  " + tree.get(i)[2] + "  " +
tree.get(i)[3], duration);
                        toast.show();*/
                }
        }
    String s;
    ArrayList<String[]> map = new ArrayList<String[]> ();

    for (int j = 0; j < tree.size(); j++)
    {
        s = tree.get(j)[0]/*parentId*/+ " " + tree.get(j)[1]/*childId*/ + " " +
tree.get(j)[4]/*lon*/ + " " + tree.get(j)[5]/*lat*/;
        map.add(s.split(" "));
    }

    try {
            IO.writeNewFile(map, "/mnt/sdcard/SmartP2P/map.txt");
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

  }

  void print ()
  {
    String m = null;
    m = tree.get(0)[0] + " " + tree.get(0)[1] + " " + tree.get(0)[2] + " " +
tree.get(0)[3] + "\n";
```

```
        for (int i=1; i<tree.size(); ++i)
                m= m + tree.get(i)[0] + " " + tree.get(i)[1] + " " + tree.get(i)[2] + " " +
tree.get(i)[3] + "\n";
        this.setListAdapter(new                    ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, Message.msg.split("\n")));
    }

    void printBut()
    {
        printButton = (Button) findViewById(R.id.button1);
                printButton.setOnClickListener(new OnClickListener() {
                        public void onClick(View v) {
                                TextView          resTxtView          =          (TextView)
findViewById(R.id.textView1);
                                resTxtView.setText(Message.Res + "  Results: ");
                                print ();

                        }
                });
    }

        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
            MenuInflater inflater = getMenuInflater();
            inflater.inflate(R.menu.result_menu, menu);
            return true;
        }

        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
            // Handle item selection
            switch (item.getItemId()) {
            case R.id.map:
```



```
                Intent map = new Intent(ResultList.this, Maps.class);
                startActivity(map);
                return true;

            default:
                return super.onOptionsItemSelected(item);
            }
        }
}
```

Message.java

```
package dmsl.Smartp2p.app;

import java.util.ArrayList;

public class Message {

        public static String msg;
        public static int Res;
        public static String serverPort;
        public static ArrayList<String[]> gps = new ArrayList<String []> ();

        public static synchronized void change(String ca) {
                if (ca.equals("NULL")) {

                        msg="";
                        Res=0;

                } else if (ca.equals("print file")) {

                } else{
```

```
                msg = ca + "\n" + msg;
            }
        }

        public static synchronized void Sum(/*String ca*/) {

            //Res = Res + Integer.parseInt(ca);
            Res++;
        }

        public static synchronized void port (String port)
        {
            serverPort = port;
        }

        public static synchronized void GPS (String ca) {
            if (ca.equals("NULL")) {

                gps.clear();

            } else {

                gps.add(ca.split(" "));

            }
        }
}
```

Maps.java

```
package dmsl.Smartp2p.app;
```



```
import java.util.ArrayList;
import java.util.List;

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;
import com.google.android.maps.MapView.LayoutParams;

import dmsl.Smartp2p.core.IO;
import dmsl.Smartp2p.map.*;

import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Point;
import android.graphics.RectF;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.view.Gravity;
import android.view.KeyEvent;
import android.view.LayoutInflater;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.View.OnTouchListener;
import android.widget.LinearLayout;
import android.widget.Toast;
import android.widget.TabHost.OnTabChangeListener;

public class Maps extends MapActivity {
```

```
MapView mapView;
MapController mc;
GeoPoint p;

ArrayList<String[]> geoFile;
ArrayList<String[]> treeFile;

@Override
public void onCreate(Bundle savedInstanceState) {
	super.onCreate(savedInstanceState);
	setContentView(R.layout.map);
	geoFile = IO.getFileContent("/mnt/sdcard/SmartP2P/map.txt", " ");

	mapView = (MapView) findViewById(R.id.mapView);
	mapView.setBuiltInZoomControls(true);
	mc = mapView.getController();

	MapOverlay mapOverlay = new MapOverlay();
	List<Overlay> listOfOverlays = mapView.getOverlays();
	listOfOverlays.clear();
	listOfOverlays.add(mapOverlay);

	double lat = Double.parseDouble(geoFile.get(0)[3]);
	double lng = Double.parseDouble(geoFile.get(0)[2]);

	p = new GeoPoint((int) (lat * 1E6), (int) (lng * 1E6));

	mc.animateTo(p);
	mc.setZoom(10);
	DrawLinesFromFile.drawLinesInAllTree(geoFile, treeFile, mapView);
	Drawable defaultIcon =
this.getResources().getDrawable(R.drawable.pin);
```



```
	MyItemizedOverlay mO = new MyItemizedOverlay(defaultIcon, this);

	OverlayItem item = new OverlayItem(p, "Query Peer: ",
geoFile.get(0)[0]);
	item.setMarker(defaultIcon);
	mO.addOverlay(item);
	listOfOverlays.add(mO);

	for (int i = 1; i < geoFile.size(); ++i) {

		if (Integer.parseInt(geoFile.get(i)[1]) != -1)
		{
			lat = Double.parseDouble(geoFile.get(i)[3]);
			lng = Double.parseDouble(geoFile.get(i)[2]);
			p = new GeoPoint((int) (lat * 1E6), (int) (lng * 1E6));

			item = new OverlayItem(p, "Peer: ", geoFile.get(i)[0]);
			mO.addOverlay(item);
			listOfOverlays.add(mO);
		}
	}

	mapView.invalidate();

}

@Override
protected boolean isRouteDisplayed() {
	return false;
}

public class MapOverlay extends Overlay implements OnTabChangeListener,
```

```
        OnTouchListener {
    public boolean draw(Canvas canvas, MapView mapView, boolean
shadow,
            long when) {
        super.draw(canvas, mapView, shadow);

        // ---translate the GeoPoint to screen pixels---
        Point screenPts = new Point();
        mapView.getProjection().toPixels(p, screenPts);

        // canvas.drawBitmap(bmp, screenPts.x, screenPts.y, null);
        return true;
    }

    @Override
    public boolean onTouchEvent(MotionEvent event, MapView mapView)
{
        // ---when user lifts his finger---
        if (event.getAction() == 1) {
            GeoPoint p = mapView.getProjection().fromPixels(
                    (int) event.getX(), (int) event.getY());

        }
        return false;
    }

    @Override
    public void onTabChanged(String tabId) {
        // TODO Auto-generated method stub

    }

    @Override
```



```
        public boolean onTouch(View v, MotionEvent event) {
            // TODO Auto-generated method stub
            return false;
        }

    }

    public boolean onKeyDown(int keyCode, KeyEvent event) {
        MapController mc = mapView.getController();
        switch (keyCode) {
        case KeyEvent.KEYCODE_3:
            mc.zoomIn();
            break;
        case KeyEvent.KEYCODE_1:
            mc.zoomOut();
            break;
        }
        return super.onKeyDown(keyCode, event);
    }
}
```

RegisterActivity.java

```
package dmsl.Smartp2p.app;


import java.io.IOException;
import java.util.Calendar;

import dmsl.Smartp2p.core.Client;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
```

```
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;


public class RegisterActivity extends Activity {

        private TextView txIp;
        private TextView txPort;
        private TextView txUname;
        private TextView txPass;
        private Button regButton;


   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.register);
      regBut();
   }

        void regBut() {

                regButton = (Button) findViewById(R.id.button1);
                regButton.setOnClickListener(new OnClickListener() {
                        public void onClick(View v) {
```



```
                txIp = (TextView) findViewById(R.id.editText1);
                String ip = txIp.getText().toString();
                txPort = (TextView) findViewById(R.id.editText2);
                String port = txPort.getText().toString();
                txUname = (TextView) findViewById(R.id.editText3);
                String uname = txUname.getText().toString();
                txPass = (TextView) findViewById(R.id.editText4);
                String pass = txPass.getText().toString();


                try {
                        String reply = Client.register(uname,pass,ip,
Integer.parseInt(port));
                        if (reply.equals("NOT OK"))
                        {
                                Context context =
getApplicationContext();
                                CharSequence text = reply;
                                int duration = Toast.LENGTH_LONG;

                                Toast toast = Toast.makeText(context, text,
duration);
                                toast.show();
                        }
                        if (reply.equals("User Name Already Exists"))
                        {
                                Context context =
getApplicationContext();
                                CharSequence text = reply;
                                int duration = Toast.LENGTH_LONG;

                                Toast toast = Toast.makeText(context, text,
duration);
```

```
                toast.show();
            }
            if (reply.equals("No Connection With Server"))
            {
                Context context =
getApplicationContext();
                CharSequence text = reply;
                int duration = Toast.LENGTH_LONG;

                Toast toast = Toast.makeText(context, text,
duration);
                toast.show();
            }
            else
            {
                Context context =
getApplicationContext();
                CharSequence text = "Your User Id: " +
reply;
                int duration = Toast.LENGTH_LONG;

                Toast toast = Toast.makeText(context, text,
duration);
                toast.show();
                finish();
            }

        } catch (NumberFormatException exception) {

        }

    }
});
```



```
    }
}
```