ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Σύστημα Καταγραφής της Εμπειρίας Κινητών Χρηστών σε Δίκτυα Ασύρματων Επικοινωνιών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αλέξανδρος Σ. Ντάκας

Επιβλέπων : Ευστάθιος Δ. Συκάς
Καθηγητής

Αθήνα, Φεβρουάριος 2011

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Σύστημα Καταγραφής της Εμπειρίας Κινητών Χρηστών σε Δίκτυα Ασύρματων Επικοινωνιών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αλέξανδρος Σ. Ντάκας

Επιβλέπων : Ευστάθιος Δ. Συκάς
Καθηγητής

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 24η Φεβρουαρίου 2011.

.....	.....	.....
Ευστάθιος Συκάς	Μιχαήλ Θεολόγου	Μιλτιάδης Αναγνώστου
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2011

.....

Αλέξανδρος Σ. Ντάκας

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

“Τρέχουμε ξένοιαστοι προς το γκρεμό, αφού πριν, βάλουμε κάποιο εμπόδιο μπροστά μας, που να μας εμποδίζει να βλέπουμε το βάραθρο.”, Blaise Pascal – Les Pensées

Περιεχόμενα

Πρόλογος

Η νέα πραγματικότητα στο χώρο της κινητής τηλεφωνίας δομείται γύρω από τη μεταφορά δεδομένων με υψηλές ταχύτητες. Οι χρήστες έχουν πλέον τη δυνατότητα να πλοηγούνται στον παγκόσμιο ιστό, να βλέπουν βίντεο και να ακούν μουσική, να διαβάζουν ηλεκτρονικές εφημερίδες και πολλά άλλα, μέσω των κινητών τους τηλεφώνων. Στο πλαίσιο της νέας τεχνολογικής πραγματικότητας, της εξάπλωσης, δηλαδή, των 3G δικτύων και της εμφάνισης των 4G, οι εταιρείες κινητής τηλεφωνίας χρειάζονται εργαλεία μέτρησης των ταχυτήτων μεταφοράς δεδομένων στο δίκτυό τους.

Στην κατεύθυνση αυτή αναπτύξαμε ένα ολοκληρωμένο σύστημα (Client – Server) μέτρησης και καταγραφής της ποιότητας του δικτύου ως προς την ταχύτητα μετάδοσης (downlink) και το χρόνο καθυστέρησης (latency). Στην πλευρά του Client αποφασίσαμε να αναπτύξουμε μια εφαρμογή (KampRate) για κινητά με λειτουργικό σύστημα Android, λόγω της δυναμικής που έχει, αλλά και των πολλών δυνατοτήτων που προσφέρει. Τόσο στον Client, όσο και στον Server, η γλώσσα προγραμματισμού που χρησιμοποιήθηκε ήταν η Java. Στόχος μας ήταν η δημιουργία ενός συστήματος που θα παρέχει αξιόπιστα και συγκρίσιμα αποτελέσματα αποτυπώνοντας τις δυνατότητες του δικτύου και σκιαγραφόντας την εμπειρία του χρήστη.

Στο σημείο αυτό θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες προς τον καθηγητή, κ. Ευστάθιο Συκά, που μου έδωσε τη δυνατότητα να ασχοληθώ με αυτό το πολύ ενδιαφέρον θέμα.

Επίσης, θέλω να ευχαριστήσω την κα. Ευγενία Αδαμοπούλου και τον κ. Κωνσταντίνο Δεμέστιχα, διδάκτορες της σχολής ΗΜΜΥ και μέλη του Εργαστηρίου Κινητών και Προσωπικών Επικοινωνιών, για την πολύτιμη βοήθεια τους και την αγαστή συνεργασία κατά τη διάρκεια εκπόνησης της διπλωματικής εργασίας μου.

Εύχομαι η εργασία αυτή να αποδειχθεί χρήσιμη στους αναγνώστες της.

Περίληψη

Η μεταφορά δεδομένων με υψηλές ταχύτητες ανοίγει νέους δρόμους για την κινητή τηλεφωνία, με την εξάπλωση των 3G και τη σχεδίαση των 4G δικτύων. Παρουσιάζεται, λοιπόν, η ανάγκη για προσδιορισμό των ταχυτήτων που παρέχει το δίκτυο μιας εταιρείας κινητής τηλεφωνίας στα διάφορα σημεία της επικράτειας, ώστε να καταγραφεί η εμπειρία του χρήστη και να γίνουν οι απαραίτητες ενέργειες βελτίωσης του δικτύου.

Σκοπός της παρούσης εργασίας είναι η ανάπτυξη ενός ολοκληρωμένου συστήματος χαρτογράφησης της εμπειρίας κινητών χρηστών ως προς το ρυθμό μετάδοσης δεδομένων (downlink) και το χρόνο καθυστέρησης (latency) σε δίκτυα ασύρματων, και ειδικότερα κινητών, επικοινωνιών. Αναπτύξαμε μια εφαρμογή (KampRate) για κινητά με λειτουργικό Android, η οποία παρέχει τη δυνατότητα μέτρησης του ρυθμού μετάδοσης (downlink) και του χρόνου καθυστέρησης (latency), τόσο σε εσωτερικό όσο και σε εξωτερικό χώρο, με αποστολή και λήψη πακέτων πάνω από UDP. Η εφαρμογή διαθέτει ένα πολύ εύχρηστο γραφικό περιβάλλον και παρουσιάζει τις καταγραφές σε χάρτη (Google Maps) με επισήμανση των μετρούμενων ταχυτήτων μέσω χρωματικής κλίμακας.

Ο Server είναι υπεύθυνος για την αποστολή πακέτων προς τους Clients, ώστε να γίνει η μέτρηση της ταχύτητας στην πλευρά του κινητού τερματικού. Επιπλέον, για τον Server σχεδιάσαμε ένα σύστημα υποδοχής των καταγραφών από τους διάφορους Clients, εισαγωγής των μετρήσεων σε μια κεντρική βάση δεδομένων και συγκεντρωτικής παρουσίασης των αποτελεσμάτων σε χάρτη.

Με αυτό τον τρόπο, ο χρήστης της κινητής συσκευής μπορεί κάνοντας μια καταγραφή να προσδιορίσει τμήματα ή σημεία ιδιαίτερου ενδιαφέροντος ή προσοχής, όπως πολύ χαμηλών ταχυτήτων, και ο διαχειριστής στον Server να αποκτήσει μια πλήρη εικόνα της ποιότητας του δικτύου συνθέτοντας τις καταγραφές που έχουν αποσταλεί από τους Clients. Τελικά, συγκροτείται μια βάση δεδομένων που αποτυπώνει αναλυτικά τις δυνατότητες του δικτύου, όπως αυτές φανερώθηκαν στις καταγραφές που πραγματοποιούν οι χρήστες.

Η εργασία χωρίζεται σε τρία κεφάλαια. Στο πρώτο γίνεται ανασκόπηση των συστημάτων κινητών επικοινωνιών, με έμφαση στις τεχνολογίες, τους ρυθμούς μετάδοσης και τις προσφερόμενες υπηρεσίες. Αναφέρονται ακόμη κάποια επαγγελματικά εργαλεία μέτρησεων και αναλύονται οι τεχνικές x-DMA. Στο δεύτερο κεφάλαιο περιγράφεται αναλυτικά το ολοκληρωμένο σύστημα που αναπτύξαμε και στο τρίτο παρουσιάζονται αποτελέσματα δοκιμών. Στο παράρτημα μπορεί κανείς να βρει το μεγαλύτερο τμήμα του κώδικα των εφαρμογών.

Λέξεις Κλειδιά
UDP, Download, Ping, Ποιότητα Δικτύου, Μετρήσεις Δικτύου, Συστήματα Κινητών Επικοινωνιών, Εμπειρία Χρήστη, Android, Google Maps

Summary

High speed data transfer defines a new reality for mobile telephony, with the growth of 3G networks and the upcoming 4G networks. It is consequently necessary for a telecom enterprise to measure data exchange rates through its network, in order to determine user's experience and proceed to improvements on the infrastructure.

Objective of this diploma thesis is the development of a mapping system related to the downlink data transfer rate (bit rate) and time delay (latency) as they are perceived by mobile users in mobile networks. We developed an application for Android phones which supports bit rate and latency measurements, both indoors and outdoors, by sending and receiving packets over UDP. The application offers an advanced but handy graphical user interface and shows recorded paths on the map (Google Maps) marking measured segments with use of a colored scale.

Server is responsible for sending packets to Clients, where the measurements are conducted. Additionally, on Server's side we designed a system capable of receiving measurement files from Clients, importing them into a main database and presenting results on the map.

Thereby, the user of the mobile device is able to locate segments or points that need special attention, like low data speed segments or points, and the Server's administrator to obtain a global view of network quality by combining the various measurement paths sent from the Clients. Eventually, a database is filled up with Client's measurements, therefore holding information that depicts network's capabilities.

The diploma thesis is divided into three chapters. The first one includes a historical review of mobile networks' technologies emphasizing on transfer rates and offered services. Also, some professional networking measurement tools are mentioned as well as the x-DMA methods. The second chapter contains a thorough description of the system we developed and in the third we present testing results. Finally, in the appendix one can find most parts of the programming code.

Key Words

UDP, Download, Ping, Network Quality, Network Measurements, Mobile Telecommunications Standard, User Experience, Android, Google Maps

Κεφάλαιο 1

Συστήματα Κινητών Επικοινωνιών

1.1 Εξέλιξη των συστημάτων κινητών επικοινωνιών

Μέχρι το τέλος της δεκαετίας του 1970 είχαν εγκατασταθεί μόνο αναλογικά συστήματα κινητών επικοινωνιών που χρησιμοποιούσαν αρχικά αναλογική διαμόρφωση πλάτους (AM) και αργότερα διαμόρφωση συχνότητας (FM). Η ανάγκη, όμως, για εξυπηρέτηση ολοένα περισσότερων χρηστών γινόταν επιτακτική. Στη δεκαετία του 1960, η AT&T Bell Labs, καθώς και άλλες εταιρίες τηλεπικοινωνιών, είχαν αναπτύξει τις θεμελιώδεις αρχές των κυψελωτών συστημάτων. Η βασική ιδέα ήταν ο χωρισμός μιας περιοχής κάλυψης σε μικρές κυψέλες, κάθε μία από τις οποίες επαναχρησιμοποιεί διαύλους, ώστε να αυξηθεί η χωρητικότητα των συστημάτων. Μια δεκαετία και πλέον αργότερα, η τεχνολογία ήταν έτοιμη να υποστηρίξει τα κυψελωτά συστήματα, οδηγώντας στα κυψελωτά συστήματα 1^{ης} γενιάς.

Κυψελωτά συστήματα 1^{ης} γενιάς

Πρόκειται για αναλογικά συστήματα στα οποία η φωνή διαμόρφωνε ένα υψίσυχνο φέρον. Τα περισσότερα χρησιμοποιούσαν μια απόσταση 45 MHz μεταξύ των συχνοτήτων εκπομπής και λήψης, τέτοια ώστε να είναι εφικτή η υλοποίηση του διπλέκτη με επαρκή απομόνωση των δύο συχνοτήτων.

- **NTT (Nippon Telephone and Telegraph)**: Το 1979 λειτούργησε στην Ιαπωνία το πρώτο παγκοσμίως κυψελωτό σύστημα, το οποίο χρησιμοποιούσε 600 FM duplex διαύλους με εύρος 25 KHz στα 925-940/870-885 MHz.

- **NMT450 (Nordic Mobile Telephone)**: Το πρώτο ευρωπαϊκό κυψελωτό σύστημα αναπτύχθηκε από την Ericsson το 1981, στη ζώνη 450-470 MHz. Το 1986 μετεξελίχθηκε στο **NMT900**, στη ζώνη 890-915/917-950 MHz.

- **AMPS (American Mobile Phone System)**: Το 1982 αναπτύχθηκε και στην Αμερική το πρώτο κυψελωτό σύστημα από την AT&T. Σε αυτό το σύστημα υπήρχαν 832 μονόδρομα κανάλια μετάδοσης από τα 824 έως τα 849 MHz και 832 μονόδρομα κανάλια λήψης από τα 869 έως τα 894 MHz. Κάθε ένα από αυτά τα μονόδρομα κανάλια είχε εύρος 30 kHz. Έτσι, το AMPS χρησιμοποιούσε FDM (Frequency-division Multiplexing) για το διαχωρισμό των καναλιών. Χρησιμοποιήθηκε επίσης και στην Αγγλία, όπου ονομαζόταν TACS, καθώς και στην Ιαπωνία, όπου ονομαζόταν MCS-L1.

Συνοπτικά, τα παραπάνω συστήματα είχαν ως βασικά χαρακτηριστικά την αναλογική διαμόρφωση FM, την τεχνική πολλαπλής πρόσβασης FDMA (Frequency-division Multiple Access) και την τεχνική FDD (Frequency-division Duplexing). Η φασματική πυκνότητα ισχύος του διαμορφωμένου σήματος στα συστήματα FDD πρέπει να ελέγχεται προσεκτικά, ώστε η παρασιτικά ακτινοβολούμενη ισχύς σε γειτονικούς διαύλους να είναι 60-80 dB χαμηλότερη από την επιθυμητή.

Κυψελωτά συστήματα 2^{ης} γενιάς

Τα 2^{ης} γενιάς κυψελωτά δίκτυα βασίζονται όλα σε ψηφιακές τεχνικές, εκμεταλλευόμενα τα πολλαπλά τους πλεονεκτήματα, όπως την αυξημένη ανοσία στο θόρυβο, τις αποδοτικές τεχνικές μετάδοσης, την καλύτερη ποιότητα υπηρεσιών, τη δυνατότητα εφαρμογής τεχνικών κρυπτογράφησης για την ασφάλεια της μετάδοσης, τη χαμηλότερη κατανάλωση ισχύος, την παροχή ταυτόχρονα υπηρεσιών φωνής και δεδομένων κ.α. Στηρίζονται στις τεχνικές TDMA (Time-division Multiple Access) ή DS-CDMA (Direct-sequence Code Division Multiple Access).

- **GSM (Global System for Mobile Communications)**: Σχεδιάστηκε από τον Ευρωπαϊκό Οργανισμό Προτυποποίησης (ETSI), λειτούργησε το 1992 και είναι το πλέον πετυχημένο κυψελωτό σύστημα παγκοσμίως. Στηρίζεται σε FDMA τεχνική, με 200 kHz απόσταση φερόντων, αλλά συνδυάζει και την TDMA τεχνική με FDD. Κάθε φέρον έχει 8 διαύλους – χρονοσχισμές, με διάρκεια χρονοσχισμής τα 0,577 msec. Η μετάδοση και η λήψη δεν πραγματοποιούνται στην ίδια χρονική υποδοχή, επειδή οι πομποδέκτες του GSM δεν μπορούν ταυτόχρονα να μεταδίδουν και να λαμβάνουν και χρειάζονται χρόνο για να αλλάξουν κατάσταση. Χρησιμοποιεί την τεχνική ψηφιακής διαμόρφωσης GMSK με τελικό ρυθμό μετάδοσης τα 270,8 kbps (ένα πολυπλαίσιο των 32500 bit στέλνεται σε 120 msec). Τα πλαίσια δεδομένων μεταδίδονται σε 547 μsec, ο πομπός όμως μπορεί να στείλει ένα πλαίσιο δεδομένων μόνο κάθε 4,615 msec, αφού μοιράζεται το κανάλι με 8 άλλους σταθμούς. Αυτό δίνει μια μικτή ταχύτητα 33,854 kbps. Ωστόσο, η επιβάρυνση δαπανά μεγάλο ποσοστό του εύρους ζώνης, αφήνοντας τελικά ωφέλιμο φορτίο 24,7 kbps ανά χρήστη πριν τη διόρθωση σφαλμάτων. Μετά τη διόρθωση σφαλμάτων απομένουν 13 kbps για ομιλία. Για δεδομένα οι ταχύτητες φθάνουν μέχρι 9,6 kbps. Στο δεύτερο τετράμηνο του 2009 οι συνδρομές στα δίκτυα GSM ήταν περισσότερες από 3,4 δισεκατομμύρια.

- **D-AMPS (Digital AMPS)**: Υποστηρίζει ψηφιακή σηματοδοσία βασισμένη στο TDMA (3 χρονοσχισμές ανά φέρον), με απόσταση φερόντων τα 30 kHz, διαμόρφωση π/4-DQPSK, με τελικό ρυθμό μετάδοσης τα 48,6 kbps. Χρησιμοποιεί, δηλαδή, τα ίδια κανάλια όπως το AMPS και στις ίδιες συχνότητες, οπότε ένα κανάλι μπορεί να είναι αναλογικό και τα γειτονικά του να είναι ψηφιακά. Κάθε πλαίσιο υποστηρίζει τρεις χρήστες, οι οποίοι χρησιμοποιούν με τη σειρά τις ανερχόμενες και κατερχόμενες συνδέσεις. Κάθε υποδοχή έχει μήκος 324 bit, από τα οποία μόνο 159 bit απομένουν για τη συμπιεσμένη ομιλία. Με 50 υποδοχές / sec, το εύρος ζώνης που είναι διαθέσιμο για τη συμπιεσμένη ομιλία είναι λίγο κάτω από τα 8 kbps. Εξέλιξη του D-AMPS αποτελεί η προδιαγραφή IS-136, που παρέχει τη δυνατότητα αποστολής μικρών μηνυμάτων και υποστηρίζει κλειστές ομάδες χρηστών.

- **IS-95 (CDMA)**: Προτάθηκε από την Qualcomm, υιοθετήθηκε το 1992 και βασίζεται στην τεχνική CDMA (Code Division Multiple Access). Η CDMA επιτρέπει σε κάθε σταθμό να μεταδίδει συνεχώς σε όλο το φάσμα συχνοτήτων. Οι πολλαπλές ταυτόχρονες

μεταδόσεις διαχωρίζονται με βάση τη θεωρία κωδικοποίησης. Το βασικό στοιχείο της τεχνικής CDMA είναι η δυνατότητα να μπορούμε να εξάγουμε το επιθυμητό σήμα, απορρίπτοντας οτιδήποτε άλλο ως τυχαίο θόρυβο. Χρησιμοποιεί διασπορά φάσματος απευθείας ακολουθίας (Spread Spectrum Direct Sequence) και παρουσιάζει ασσυμετρία ζεύξης, χρησιμοποιώντας διαφορετικές τεχνικές για την ευθεία και την αντίστροφη ζεύξη. Σε κάθε Κινητό Σταθμό σε μια κυψέλη, αποδίδεται ένας διαφορετικός κώδικας, παρέχοντας έτσι ορθογωνιότητα μεταξύ των χρηστών. Οι συχνότητες λειτουργίας είναι ίδιες με το AMPS και το D-AMPS, αλλά έχουν αποδοθεί και επιπλέον συχνότητες στην περιοχή 1,8-2 GHz. Το εύρος ζώνης του διεσπαρμένου σήματος είναι 1,25 MHz. Ο ρυθμός μετάδοσης για κάθε χρήστη είναι συνάρτηση του χρόνου που η φωνή είναι ενεργή, αλλά και της συνολικής κίνησης στο δίκτυο. Αποτέλεσε οδηγό και βάση εκκίνησης για τα συστήματα 3ης γενιάς.

- **PDC (Personal Digital Cellular)**: Αναπτύχθηκε στην Ιαπωνία από το 1989 και στηρίζεται στις αρχές του D-AMPS. Υποστηρίζει TDMA τεχνική πολλαπλής πρόσβασης (3 χρονοσχισμές ανά φέρον), απόσταση φερόντων 25 kHz, διαμόρφωση π/4-DQPSK και τελικό ρυθμό μετάδοσης τα 42 kbps. Οι συχνότητες λειτουργίας είναι οι 810-826/940-956 MHz και 1429-1453/1477-1501 MHz.

- **iDen (Integrated Digital Enhanced Network)**: Αναπτύχθηκε από τη Motorola και χρησιμοποιεί κανάλια των 25 KHz. Χρησιμοποιεί την τεχνική TDMA, όπως επίσης και την FDD για να μεταδίδει και να λαμβάνει σήματα ξεχωριστά, με τις δύο περιοχές να διαχωρίζονται από 39 MHz, 45 MHz ή 48 MHz, ανάλογα με την περιοχή συχνότητας που χρησιμοποιείται.

Η 2η γενιά κυψελωτών συστημάτων έδωσε τη δυνατότητα παροχής υπηρεσιών φωνής σε μεγάλους πληθυσμούς και μεγάλες γεωγραφικές εκτάσεις, λόγω της πολύ καλής ποιότητας της φωνητικής υπηρεσίας. Ωστόσο, η δυνατότητα υποστήριξης υπηρεσιών δεδομένων είναι πολύ περιορισμένη στα συστήματα αυτά, με αποτέλεσμα η ανάγκη για παροχή υπηρεσιών με υψηλούς ρυθμούς μετάδοσης, ώστε να μεταδίδονται εικόνες υψηλής ποιότητας και video πραγματικού χρόνου και να παρέχεται πρόσβαση στο Διαδίκτυο με υψηλές ταχύτητες να οδηγήσει στη σχεδίαση των συστημάτων 2.5G.

Κυψελωτά συστήματα 2.5 γενιάς

Τα συστήματα 2.5G στηρίζονται σε συστήματα τεχνολογίας 2ης γενιάς προσφέροντας υπηρεσίες δεδομένων υψηλότερης ταχύτητας υποστηρίζοντας τεχνολογίες μεταγωγής πακέτου. Το βασικό χαρακτηριστικό είναι η παροχή κινητικότητας IP, φέρνοντας τους χρήστες κινητών επικοινωνιών πιο κοντά στο Internet.

- **GPRS (General Packet Radio Service)**: Πρόκειται για ένα δίκτυο μεταγωγής πακέτων πάνω από το GSM. Επιτρέπει στους κινητούς σταθμούς να στέλνουν και να λαμβάνουν

πακέτα IP σε μια κυψέλη η οποία χρησιμοποιεί κάποιο σύστημα φωνής. Μερικές χρονικές υποδοχές σε κάποιες συχνότητες δεσμεύονται για κίνηση πακέτων δεδομένων. Το πλήθος και η θέση των χρονικών υποδοχών μπορεί να ορίζεται δυναμικά από το σταθμό βάσης, ανάλογα με το λόγο κίνησης φωνής προς δεδομένα στην κυψέλη. Θεωρητικά μπορούν να υποστηριχθούν ρυθμοί μέχρι και 112 kbps. Έχουν προβλεφθεί τέσσερις διαφορετικοί τύποι κωδικοποίησης, οι οποίοι υποστηρίζουν ρυθμούς για το χρήστη μέχρι και 21,4 kbps για μια χρονοσχισμή. Η τηλεπικοινωνιακή κίνηση του GPRS έχει πάντα μικρότερη προτεραιότητα από εκείνη των υπηρεσιών του GSM και χρησιμοποιεί τους ραδιοπόρους που δεν χρησιμοποιούνται από το GSM.

- **EDGE (Enhanced Data rates for GSM Evolution)**: Είναι μια προσθήκη στο GPRS και δεν μπορεί να λειτουργήσει αυτόνομα. Αποτελεί την προσπάθεια να αυξηθεί ο ρυθμός μετάδοσης πάνω από τη ραδιοεπαφή του GSM, κάτι που το επιτυγχάνει με περισσότερα bit ανά baud. Αυτό σημαίνει και περισσότερα σφάλματα ανά baud, οπότε έχει εννιά διαφορετικές μεθόδους διαμόρφωσης και διόρθωσης σφαλμάτων, οι οποίες διαφέρουν μεταξύ τους στο ποσοστό του εύρους ζώνης το οποίο αφιερώνεται για τη διόρθωση των σφαλμάτων που εισάγονται λόγω της υψηλότερης ταχύτητας. Η βασική διαφορά από το GPRS είναι ότι χρησιμοποιεί διαφορετικό τρόπο διαμόρφωσης (8-PSK) και διαφορετικούς τύπους κωδικοποίησης διαύλου. Θεωρητικά μπορεί να υποστηρίξει ρυθμούς μέχρι και 384 kbps, ενώ ο μέγιστος ρυθμός μετάδοσης για μια χρονοσχισμή έχει αυξηθεί στα 59,2 kbps. Κάθε χρονοσχισμή μπορεί να χρησιμοποιηθεί από πολλούς χρήστες, αυξάνοντας έτσι την χωρητικότητα του δικτύου.

- **CDMA2000-1xRTT**: Υποστηρίζει ρυθμούς μετάδοσης δεδομένων μέχρι 144 kbps, ενώ το εύρος διαύλου είναι ίδιο με του IS-95, δηλαδή 1,25 MHz. Σχεδόν διπλασιάζει τη χωρητικότητα του IS-95 προσθέτοντας 64 κανάλια στη ζεύξη καθόδου, τα οποία είναι ορθογώνια ως προς το αρχικό σετ των 64 καναλιών του IS-95.

Κυψελωτά συστήματα 3^{ης} γενιάς

Η υποστήριξη εφαρμογών πολυμέσων και η δυνατότητα πρόσβασης σε πληροφορίες και υπηρεσίες από δημόσια ή ιδιωτικά δίκτυα με υψηλούς ρυθμούς μετάδοσης είναι το βασικό χαρακτηριστικό των συστημάτων 3^{ης} γενιάς. Οι ραδιοεπαφές που έχουν αναπτυχθεί για τα συστήματα 3^{ης} γενιάς είναι το WCDMA ή UTRA, το οποίο σχεδιάστηκε ώστε να αξιοποιεί τη δομή του δικτύου κορμού του GSM, και το CDMA2000, το οποίο παρέχει συμβατότητα με το δίκτυο IS-95. Υπάρχει και τρίτη προδιαγραφή που καλείται TD-SCDMA, η οποία αναπτύσσεται κυρίως στην Κίνα.

- **UMTS (Universal Mobile Telecommunications System)**: Πρόκειται για την ονομασία που έδωσε το ETSI το 1990 στα δίκτυα τρίτης γενιάς. Το WCDMA (Wideband Code Division Multiple Access) είναι η ραδιοεπαφή του UMTS και έχει δύο εκδοχές, την FDD και την TDD. Το εύρος διαύλου είναι 5 MHz με ακολουθία διασποράς σε ρυθμό 3,84

Mcps. Κάθε φέρον διαιρείται σε χρονο-πλαίσια των 10 msec και κάθε χρονο-πλαίσιο διαθέτει 15 χρονοσχισμές. Έτσι, το UMTS έχει τη δυνατότητα υποστήριξης πολύ υψηλών ρυθμών μετάδοσης (384 kbps ή και 2 Mbps, ανάλογα με την υλοποίηση και το περιβάλλον), αλλά και τη δυνατότητα υποστήριξης μεταγωγής κυκλώματος και μεταγωγής πακέτου. Οι συχνότητες που έχουν αποδοθεί στα δίκτυα 3ης γενιάς στην Ευρώπη είναι 2*60 MHz (1920-1980 MHz uplink, 2110-2170 MHz downlink) για WCDMA FDD συστήματα, 25 MHz (1900-1920 & 2020-2025 MHz) για TDD συστήματα με υποχρέωση έκδοσης άδειας και 10 MHz (2010-2020 MHz) για TDD συστήματα χωρίς αδειοδότηση.

Παρόμοια με την WCDMA-TDD είναι και η TD-SCDMA, με τη διαφορά ότι το εύρος του διαύλου είναι 1,6 MHz και η ακολουθία διασποράς έχει ρυθμό 1,28 Mcps. Το πρότυπο αυτό αναπτύχθηκε κυρίως διότι η κινέζικη κυβέρνηση και οι κινέζικες εταιρίες ήθελαν να αποφύγουν το υψηλό κόστος των αδειών που απαιτούνταν για τη χρήση τεχνολογιών που είχαν αναπτυχθεί από τις δυτικές εταιρείες.

- **CDMA2000:** Αρχικά υλοποιήθηκε η τεχνολογία CDMA2000-1xRTT η οποία όμως θεωρείται μια 2.5G λύση και αναφέρθηκε παραπάνω.

  Στη συνέχεια προέκυψε η προδιαγραφή **CDMA2000-1xEV-DO (1xEvolution Data Only)**, η οποία λειτουργεί ως επικάλυψη του IS-95 και υποστηρίζει υψηλότερους ρυθμούς δεδομένων, έως και 3,1 Mbps για την ευθεία ζεύξη και μέχρι 1,2 Mbps για την αντίστροφη ζεύξη, σε ξεχωριστό αποκλειστικό δίαυλο. Στηρίζεται στην τεχνολογία High Data Rate (HDR) που είχε αναπτύξει η Qualcomm. Χρησιμοποιεί την τεχνική CDMA (Code-division Multiple Access) αλλά και την τεχνική TDMA (Time-division Multiple Access) για να μεγιστοποιήσει τόσο τη ρυθμαπόδοση (throughput) του χρήστη, όσο και του συστήματος συνολικά. Το εύρος ζώνης κάθε καναλιού είναι 1,25 MHz, όπως στο IS-95 και στο 1xRTT, ο τρόπος, ωστόσο, που αξιοποιεί το φάσμα είναι διαφορετικός. Ακόμη, το δίκτυο κορμού που χρησιμοποιεί βασίζεται εξ ολοκλήρου στη μεταγωγή πακέτου, οπότε δεν δεσμεύεται από τους περιορισμούς ενός δικτύου μεταγωγής κυκλώματος, όπως τα παλαιότερα συστήματα.

  Έπειτα, αναπτύχθηκε η τεχνολογία **CDMA2000-1xEV-DV (1xEvolution Data and Voice)**, η οποία υποστηρίζει υπηρεσίες φωνής και δεδομένων ταυτόχρονα στο ίδιο φέρον, με ρυθμούς μέχρι 3,1 Mbps για την ευθεία ζεύξη και μέχρι 1,8 Mbps για την αντίστροφη.

Κυψελωτά συστήματα 3.5 γενιάς

Οι προηγμένες υπηρεσίες των κυψελωτών δικτύων 3ης γενιάς απαιτούν διαφορετικό επίπεδο υπηρεσίας (QoS) όσον αφορά το ρυθμό μετάδοσης δεδομένων, την καθυστέρηση, το ρυθμό σφαλμάτων, την χωρητικότητα και την κάλυψη. Προς αυτή την κατεύθυνση αναπτύχθηκε η τεχνολογία **HSPA (High-Speed Packet Access)** η οποία αποτελείται από δύο επιμέρους τεχνολογίες, την HSDPA και την HSUPA.

Η τεχνολογία **HSDPA** (High-Speed Downlink Packet Access) ορίζει τρεις καινούριους τύπους καναλιών: **High-Speed Downlink Shared Channel (HS-DSCH)**, **High-Speed Shared Control Channel (HS-SCCH)**, **High-Speed Dedicated Physical Control Channel (HS-DPCCH)**. Το HS-DSCH συσχετίζεται πάντα με ένα DPCH (Dedicated Physical Channel) και ένα ή περισσότερα HS-SCCH. Το HS-SCCH πληροφορεί το χρήστη ότι θα σταλούν δεδομένα στο HS-DSCH. Το HS-DPCCH μεταφέρει μηνύματα ACK/NACK και τον τρέχοντα δείκτη ποιότητας καναλιών (CQI). Τα νέα χαρακτηριστικά που εισήγαγε το HSDPA είναι η προσαρμοστική διαμόρφωση και κωδικοποίηση (AMC: Adaptive Modulation and Coding), η ταχεία υβριδική επανεκπομπή πακέτων (HARQ: Hybrid Automatic Repeat reQuest) και η ταχεία δρομολόγηση κίνησης (Fast Scheduling). Ακόμη, στέλνει πακέτα 500 φορές το δευτερόλεπτο. Αυτό έχει ως αποτέλεσμα τη δραστική μείωση της καθυστέρησης μετάδοσης (Round Trip Delay) και την εξοικονόμηση χρόνου σε περίπτωση απώλειας κάποιου πακέτου. Επιπλέον, επιτρέπει στο δίκτυο να κάνει προσαρμογή της ζεύξης (Link Adaptation) ανά 2 msec και συνακόλουθα να λαμβάνει αποφάσεις με βάση την τρέχουσα κατάσταση. Συνοπτικά, το HSDPA είναι μια τεχνολογία η οποία βασίζεται κυρίως σε βελτιώσεις του λογισμικού, οι οποίες επιτρέπουν το διπλασιασμό της χωρητικότητας της ραδιο-επαφής και δίνουν μια αύξηση 5-10 φορές στο ρυθμό μετάδοσης στην κάτω ζεύξη (7,2 – 14,4 Mbps).

Η τεχνολογία **HSUPA (High-Speed Uplink Packet Access)** προσφέρει υψηλούς ρυθμούς μετάδοσης στην άνω ζεύξη, μέχρι και 5,7 Mbps. Για να επιτευχθεί αυτός ο ρυθμός χρησιμοποιεί ένα ενισχυμένο, αφιερωμένο, φυσικό κανάλι (E-DCH), μικρό TTI (Transmission Time Interval) με ελάχιστη τιμή 2 msec, το οποίο επιτρέπει γρήγορες αποκρίσεις στην αλλαγή ραδιοσυνθηκών και μηνυμάτων λαθών, γρήγορο προγραμματισμό που επιτρέπει στο σταθμό βάσης να διαθέτει αποτελεσματικά τους ραδιοπόρους και την ταχεία υβριδική ARQ που βελτιώνει την απόδοση της διόρθωσης λαθών.

Εξέλιξη του HSPA αποτελεί το **HSPA+ (Evolved HSPA)** το οποίο χρησιμοποιεί την τεχνική πολλαπλών-κεραιών MIMO (Multiple-Input Multiple-Output) και διαμόρφωση 64QAM. Θεωρητικά μπορεί να οδηγήσει σε ρυθμούς μετάδοσης της τάξης των 56 Mbps (downlink) και 22 Mbps (uplink), ωστόσο η πραγματική ταχύτητα για τους χρήστες είναι αρκετά χαμηλότερη.

Κυψελωτά συστήματα 4^{ης} γενιάς

Στα συστήματα αυτά στόχος είναι η ταχύτητα να φθάνει τα 100 Mbps (downlink) για υψηλής κινητικότητας επικοινωνία, όπως σε τρένα και αυτοκίνητα, και το 1 Gbps για χαμηλής κινητικότητας επικοινωνία, όπως για πεζούς και χρήστες σε σταθερή θέση. Οι προϋποθέσεις που έχει θέσει το ITU-R (ITU Radiocommunication Sector) και τις οποίες πρέπει να πληροί ένα IMT-Advanced (International Mobile Telecommunications Advanced) κυψελωτό σύστημα είναι:

- Χρήση του πρωτοκόλλου IP τόσο για τη μετάδοση φωνής όσο και για τη μεταφορά δεδομένων (all-IP packet switched network)

- Μέγιστοι ρυθμοί μετάδοσης 100 Mbps για χρήστες υψηλής κινητικότητας και 1 Gbps για χρήστες χαμηλής κινητικότητας

- Δυναμικός διαμοιρασμός και χρησιμοποίηση των πόρων του δικτύου ώστε να υποστηρίζονται ταυτόχρονα περισσότεροι χρήστες ανά κυψέλη

- Κλιμακωτό εύρος ζώνης καναλιού, ανάμεσα σε 5 MHz και 20 MHz, προεραιτικά μέχρι και 40 MHz

- Μέγιστη Φασματική Αποδοτικότητα Ζεύξης (Link Spectral Efficiency) της τάξης των 15 bit/s/Hz στην ευθεία ζεύξη (downlink) και 6,75 bit/s/Hz στην αντίστροφη ζεύξη (uplink). Αυτό σημαίνει πως 1 Gbps στο downlink θα πρέπει να είναι εφικτό με εύρος ζώνης μικρότερο από 67 MHz.

- Φασματική Αποδοτικότητα Συστήματος (System Spectral Efficiency) μέχρι και 3 bit/s/Hz/cell στο downlink και 2,25 bit/s/Hz/cell για χρήση σε εσωτερικό χώρο

- Ομαλές μεταπομπές (handovers) ανάμεσα σε ετερογενή δίκτυα

- Παροχή υψηλής ποιότητας υπηρεσιών για την υποστήριξη πολυμέσων της επόμενης γενιάς

Για να καλυφθούν οι παραπάνω προϋποθέσεις θα χρησιμοποιηθεί μια σειρά προηγμένων τεχνικών, όπως:

- Χρήση κεραιών τύπου MIMO, οι οποίες βελτιώνουν την ποιότητα της ζεύξης αυξάνοντας το ρυθμό μετάδοσης και την εμβέλεια του συστήματος, δίχως να αυξάνονται οι απαιτήσεις σε ισχύ εκπομπής και εύρος ζώνης

- Χρήση τεχνικών διαμόρφωσης πολλαπλού φέροντος (MCM: Multi-Carrier Modulation), οι οποίες διαιρούν την κύρια ροή δεδομένων σε μικρότερες ροές, κάθε μία από τις οποίες διαμορφώνει ένα ξεχωριστό φέρον. Αυτό εξασφαλίζει καλύτερη αξιοποίηση των χαρακτηριστικών του ασύρματου διαύλου αντιμετωπίζοντας κατά περίπτωση τις δυσμενείς συνθήκες διάδοσης που επικρατούν στις διάφορες ζώνες συχνοτήτων.

- Χρήση τεχνικών δυναμικής απόδοσης διαύλων (DCA: Dynamic Channel Assignment), οι οποίες επιτρέπουν τη βέλτιστη κατανομή και επαναχρησιμοποίηση των διαθέσιμων διαύλων

- Χρήση τεχνικών δρομολόγησης που λαμβάνουν υπ’ όψιν τις απαιτήσεις του κάθε χρήστη σε εύρος ζώνης, αλλά και τις συνθήκες διάδοσης που επικρατούν στον κάθε

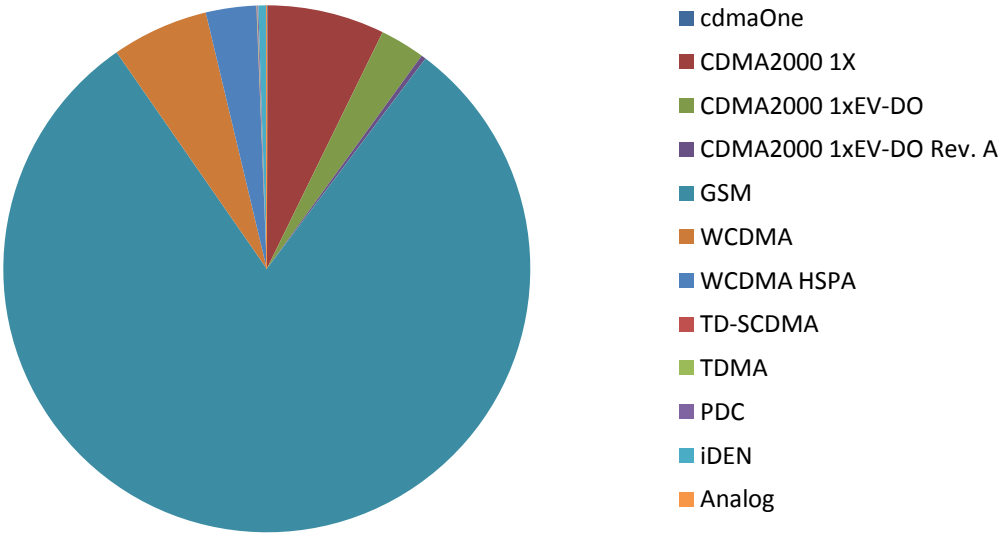
δίαυλο (channel-dependent scheduling), αυξάνοντας τελικά τη ρυθμαπόδοση (throughput) του συστήματος

Το **LTE (Long Term Evolution)** είναι μια σειρά από προσθήκες στο UMTS που, αν και διαφημίζεται ως 4G, η πρώτη του έκδοση δεν πληροί όλες τις προδιαγραφές που είδαμε παραπάνω ώστε να χαρακτηριστεί 4G σύστημα. Παρέχει μέγιστους ρυθμούς μετάδοσης downlink τουλάχιστον 100 Mbps και uplink 50 Mbps. Οι μετρούμενοι RTT (Round Trip Time) φθάνουν κάτω από τα 10 msec. Υποστηρίζει κλιμακωτό εύρος ζώνης από 1,4 MHz έως 20 MHz, καθώς και τις τεχνικές FDD και TDD.

Το **LTE Advanced** είναι ένα πρώιμο πρότυπο κινητών επικοινωνιών, το οποίο κατατέθηκε επισήμως ως υποψήφιο για 4G σύστημα στο τέλος του 2009 και αναμένεται να ολοκληρωθεί μέσα στο 2011. Είναι συμβατό προς τα πίσω με το LTE, ενώ αντίθετα το LTE δεν είναι συμβατό προς τα πίσω με τα 3G συστήματα. Στόχος του LTE Advanced συστήματος είναι να πληροί ή και να ξεπερνά τις προϋποθέσεις του ITU-R. Προς την κατεύθυνση αυτή αξιοποιεί προχωρημένα τοπολογικά δίκτυα. Βρίσκεται στο στάδιο της σχεδίασης και δεν έχουν οριστικοποιηθεί οι προδιαγραφές του.

Market Data Summary (Q2 2009)
Connections by Bearer Technology

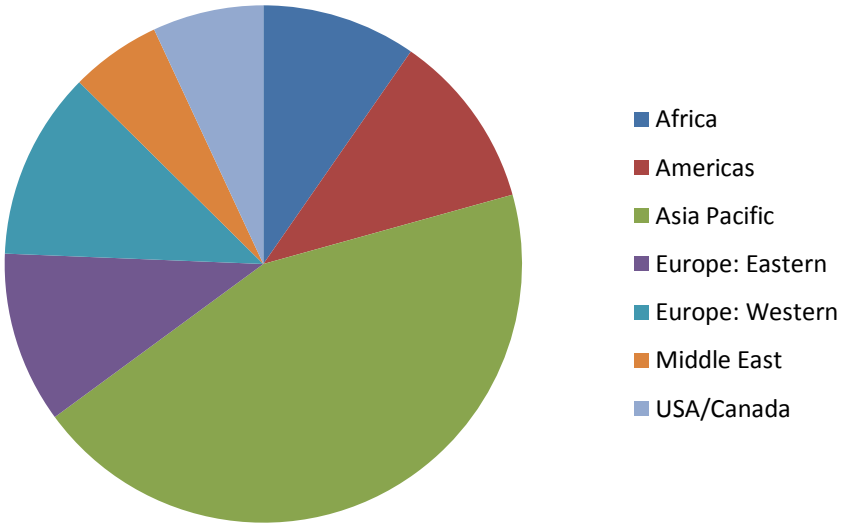
Total	4,310,295,611	%
cdmaOne	2,449,937	0.0568
CDMA2000 1X	309,907,068	7.1899
CDMA2000 1xEV-DO	118,688,849	2.7536
CDMA2000 1xEV-DO Rev. A	12,644,062	0.2933
GSM	3,450,410,548	80.0504
WCDMA	255,630,141	5.9307
WCDMA HSPA	133,286,097	3.0923
TD-SCDMA	825,044	0.0191
TDMA	1,480,766	0.0344
PDC	2,740,320	0.0636
iDEN	22,172,858	0.5144
Analog	9,593	0.0002



Είναι φανερό πως το GSM (2^{η} γενιά) κυριαρχεί σε παγκόσμιο επίπεδο, ενώ τα υπόλοιπα συστήματα 2^{ης} γενιάς έχουν ουσιαστικά εγκαταλειφθεί. Τα 3G συστήματα (CDMA2000, WCDMA) έχουν ήδη καταλάβει ένα σημαντικό μερίδιο της αγοράς και αναμένεται να κυριαρχήσουν στα επόμενα χρόνια, εν αναμονεί και των συστημάτων 4G που θα αλλάξουν ξανά τις ισορροπίες.

Connections by World Region

World	4,310,295,611	%
Africa	416,303,821	9.6584
Americas	475,193,998	11.0246
Asia Pacific	1,906,764,743	44.2374
Europe: Eastern	462,040,510	10.7195
Europe: Western	506,982,364	11.7621
Middle East	243,953,091	5.6598
USA/Canada	299,057,084	6.9382



Σύμφωνα με έρευνα της ITU (International Telecommunication Union – «The world in 2010, Facts and Figures») από τις 5,3 δισεκατομμύρια συνδέσεις κινητής τηλεφωνίας που υπολογίζονταν για το τέλος του 2010, οι 3,8 θα είναι στις αναπτυσσόμενες χώρες. Ένας λόγος της μεγάλης ανάπτυξης των κινητών επικοινωνιών στον αναπτυσσόμενο κόσμο είναι το χαμηλότερο κόστος υποδομών, σε σχέση με τη σταθερή τηλεφωνία. Το ζήτημα, ωστόσο, έχει κοινωνιολογικές, πολιτικές και οικονομικές προεκτάσεις που δεν μπορούν να αναλυθούν σε αυτή την εργασία.

Στις αναπτυγμένες χώρες η αύξηση των συνδρομών έχει μειωθεί αισθητά την τελευταία πενταετία (ποσοστό μόλις +1,6% για το 2009-2010). Σε 100 κατοίκους αντιστοιχούν πλέον 116 συνδρομές. Οι 3G συνδρομές υπολογίζεται ότι θα ανέρχονται σε 940 εκατομμύρια έναντι μόλις 72 εκατομμυρίων το 2005. Η αγορά φαίνεται ότι έχει ωριμάσει και ο ανταγωνισμός θα επικεντρωθεί στην προσφορά των πιο προηγμένων υπηρεσιών με το χαμηλότερο δυνατό κόστος.

Τέλος, ένα εντυπωσιακό στοιχείο είναι ο τριπλασιασμός των sms μηνυμάτων από το 2007 στο 2010 (από 1,8 τρισεκατομμύρια σε 6,1 τρισεκατομμύρια). Με άλλα λόγια, σχεδόν 200.000 sms στέλνονται κάθε δευτερόλεπτο. Εκπεφρασμένα σε χρηματικό κόστος και θεωρώντας μια μέση τιμή 0,07$ ανά μήνυμα, πρόκειται για 14.000$ κάθε δευτερόλεπτο!

1.2 Προσφερόμενες υπηρεσίες

Στην προηγούμενη παράγραφο μελετήσαμε τις γενιές συστημάτων κινητών επικοινωνιών και τις διαφορετικές τεχνολογίες που τις χαρακτηρίζουν. Η εξέλιξη των κινητών επικοινωνιών έχει ως σκοπό την παροχή περισσότερων και καλύτερων υπηρεσιών στους χρήστες, ώστε να ανταποκριθούν σε υπάρχουσες ανάγκες που δεν καλύπτονται ή να δημιουργήσει νέες μέσω καινοτόμων ιδεών. Στο πλαίσιο αυτό, θα αναφερθούμε στις προσφερόμενες υπηρεσίες τριών τεχνολογιών που κυριάρχησαν στη γενιά τους, του GSM ($2^η$ γενιά), του GPRS (2.5 γενιά) και του UMTS ($3^η$ γενιά).

Οι υπηρεσίες που παρέχει το **GSM** είναι πολλές περισσότερες και ποιοτικά αναβαθμισμένες σε σχέση με τα αναλογικά συστήματα που διαδέχτηκε. Τις διακρίνουμε σε τρεις κατηγορίες:

- **Bearer Services**: Εξασφαλίζουν τη μετάδοση των σημάτων μεταξύ σημείων πρόσβασης των δικτύων και καθορίζουν τις υπηρεσίες που μπορεί να υποστηρίξει ένα δίκτυο. Προϋπήρξαν οι υπηρεσίες αυτές που επιτρέπουν τη σύνδεση των συνδρομητών σε άλλα δίκτυα (PSTN, ISDN, Data Networks). Είναι υπηρεσίες μεταγωγής κυκλώματος με ρυθμό μετάδοσης από 900 bps έως 9600 bps, ενώ υπάρχει και η υπηρεσία μετάδοσης δεδομένων «υψηλής» ταχύτητας με τεχνολογία μεταγωγής κυκλώματος (HSCSD), όπου ο ρυθμός μπορεί να φθάσει και τα 57600 bps.

- **Tele Services**: Εξασφαλίζουν την επικοινωνία μεταξύ χρηστών με βάση τα πρωτόκολλα που έχουν συμφωνηθεί από τους παρόχους των δικτύων.

    - Φωνητικές κλήσεις

    - Κλήσεις Ανάγκης (Emergency Calls), στις οποίες δεν είναι απαραίτητη η ύπαρξη έγκυρης κάρτας SIM

    - Dual Tone Multi Frequency (DTMF), μια μέθοδος σηματοδοσίας που επιτρέπει στους συνδρομητές τον έλεγχο απομακρυσμένων συσκευών που είναι συνδεδεμένες σε μια τηλεφωνική γραμμή (π.χ. τηλεφωνητής)

    - Fax, είτε μεταξύ δύο κινητών τερματικών fax, είτε μεταξύ σταθερού και κινητού τερματικού (μέγιστη ταχύτητα 9,6 kbps)

- Short Message Service – SMS, μηνύματα κειμένου τα οποία μπορούν να περιλαμβάνουν το πολύ μέχρι 160 αλφαριθμητικούς χαρακτήρες και αποθηκεύονται στο δίκτυο, στο SMS Center. Προωθούνται στον Κινητό Σταθμό, μόλις αυτός ενεργοποιηθεί και γίνει γνωστή η θέση του.

- SMS – Cell Broadcast, μηνύματα μέχρι 93 αλφαριθμητικούς χαρακτήρες τα οποία εκπέμπονται από ένα BTS προς όλα τα κινητά της κυψέλης

- Τηλεφωνητής (Voice Mail) και Fax Mail

- **Supplementary Services & Value Added Services** : Πρόσθετες υπηρεσίες που εξαρτώνται από τον παροχέα υπηρεσιών ή του δικτύου.

    - Προώθηση Κλήσεων (Call Forwarding)

    - Φραγή Εξερχομένων Κλήσεων (Blocking of Outgoing Calls)

    - Φραγή Εισερχομένων Κλήσεων (Blocking of Ingoing Calls)

    - Πληροφορίες Χρέωσης

    - Αναμονή Κλήσης (Call Waiting)

    - Κράτηση Κλήσης (Call Hold)

    - Τηλεδιάσκεψη (Conference Call)

    - Αναγνώριση καλούντος (Caller ID)

    - Κλειστές Ομάδες Χρηστών (Closed Groups)

Το GSM ικανοποιεί τη βασική ανάγκη του χρήστη για φωνητική επικοινωνία, προσφέροντας και μια σειρά συμπληρωματικών υπηρεσιών. Δεν μπορεί, ωστόσο, να ανταποκριθεί στις ανάγκες του σύγχρονου χρήστη, ο οποίος επιθυμεί υπηρεσίες γύρω από το χώρο της μεταφοράς δεδομένων. Ένα πρώτο βήμα προς τη νέα πραγματικότητα έγινε με το GPRS, ένα σύστημα που χαρακτηρίστηκε ως 2.5G και υλοποιούσε ένα δίκτυο μεταγωγής πακέτων πάνω από το GSM.

Διακρίνουμε τις προσφερόμενες από το **GPRS** υπηρεσίες στις κατηγορίες:

- **Point-to-Point**: Υπάρχουν πάντα επιβεβαιώσεις λήψης των πακέτων.

- Υπηρεσίες χωρίς σύνδεση (PTP ConnectionLess Network Services, PTP-CLNS): Δεν απαιτείται η ύπαρξη εγκατάστασης σύνδεσης και κάθε πακέτο στέλνεται ανεξάρτητα από προηγούμενα ή επόμενα πακέτα, δυνατότητα που είναι πολύ χρήσιμη για περιπτώσεις όπου έχουμε καταιγιστική (bursty) μετάδοση δεδομένων. Στις υπηρεσίες αυτές υποστηρίζεται το Internet Protocol (IP).

- Υπηρεσίες με σύνδεση (PTP Connection Oriented Network Service PTP-CONS): Απαιτείται η ύπαρξη σύνδεσης πριν την αποστολή πακέτων. Τα πακέτα στέλνονται σε μια σειρά και είναι συνδεδεμένα. Υπάρχει μια λογική σύνδεση μεταξύ πομπού και δέκτη, σε αντίθεση με τη φυσική σύνδεση που υπάρχει μεταξύ αποστολέα και παραλήπτη, σε μια ζεύξη μεταγωγής κυκλώματος.

- **Point-to-Multipoint**: Τα δεδομένα της πηγής αποστέλλονται σε πολλαπλούς αποδέκτες.

  - Point-to-Multipoint Multicast: Ο αποστολέας στέλνει την πληροφορία σε πολλαπλούς αποδέκτες που βρίσκονται σε καθορισμένη περιοχή. Μπορεί επίσης να καθορίσει και μια ομάδα αποδεκτών στην περιοχή. Οι αποδέκτες δεν χρειάζεται να είναι συνδεδεμένοι στο GPRS δίκτυο. Επειδή δεν υπάρχουν επιβεβαιώσεις λήψης, η εκπομπή θεωρείται ανώνυμη, ενώ η λήψη εξασφαλίζεται με πολλαπλές εκπομπές των ίδιων πακέτων.

  - Point-to-Multipoint Group Call: Οι αποδέκτες πρέπει να είναι συνδεδεμένοι σε GPRS δίκτυο. Μπορούν να στείλουν δεδομένα πίσω στον αποστολέα, δημιουργώντας μια αμφίδρομη σύνδεση. Επίσης, μπορούν να επικοινωνούν μεταξύ τους σχηματίζοντας μια σύνδεση με Ν δυνατούς δρόμους (N-way Connection), όπως στις κλήσεις τηλεδιάσκεψης.

  - Internet Protocol Multicast (IP-M): Η υπηρεσία στηρίζεται στο IP. Η ομάδα δεκτών μπορεί να ανήκει είτε στο δίκτυο κινητής τηλεφωνίας είτε οπουδήποτε στο Internet.

- **Υπηρεσίες προστιθέμενης αξίας**:

  - Πρόσβαση στο Internet οποιαδήποτε στιγμή

  - Multimedia Messaging Service (MMS): δυνατότητα αποστολής μηνυμάτων που περιέχουν πολυμεσικό υλικό (φωτογραφίες, βίντεο)

  - Push to talk over cellular (PoC/PTT): Επιτρέπει στους συνδρομητές να χρησιμοποιήσουν το κινητό τους σαν ένα walkie-talkie με απεριόριστη εμβέλεια. Ένα πλεονέκτημα του Poc/PTT είναι ότι επιτρέπει σε ένα άτομο να

μιλήσει σε μια ομάδα συνδιαλεγομένων με το πάτημα ενός κουμπιού. Οι χρήστες δεν χρειάζεται να κάνουν πολλαπλές κλήσεις για να συντονιστούν με μια ομάδα.

- Εφαρμογές Internet για «έξυπνες» συσκευές μέσω WAP (Wireless Application Protocol)

Το GPRS, αν και αποτελούσε ένα βήμα προς τα εμπρός, δεν θα μπορούσε να ανταποκριθεί στις σημερινές ανάγκες των χρηστών που περιλαμβάνουν την πρόσβαση στο διαδίκτυο με υψηλές ταχύτητες, τις διαδραστικές αγορές και τις τραπεζικές συναλλαγές, την ενημέρωση από on-line εφημερίδες, τις on-line βιβλιοθήκες, υπηρεσίες που βασίζονται στην τοποθεσία του χρήστη, video και μουσική, video τηλεφωνία και video τηλεδιάσκεψη, τα διαδραστικά παιχνίδια, το κινητό γραφείο, την τηλεϊατρική, τις υπηρεσίες πιστωτικών καρτών, την κράτηση εισιτηρίων και πολλά άλλα. Τα δίκτυα **3ης γενιάς** είναι σε θέση να παρέχουν όλες αυτές τις υπηρεσίες.

Μια σημαντική διαφοροποίηση από τους προκατόχους τους, άλλωστε, είναι ότι οι παράμετροι των υπηρεσιών δεν είναι σταθερές, αλλά συντελείται δυναμική επαναδιαπραγμάτευση, όποτε αυτό απαιτείται. Διακρίνουμε τις υπηρεσίες στις εξής κατηγορίες (**UMTS**):

- **Tele services**: Εξασφαλίζουν την επικοινωνία από άκρο σε άκρο μεταξύ χρηστών με βάση προσυμφωνημένα πρωτόκολλα. Χρησιμοποιούν όλη τη στοίβα πρωτοκόλλων του OSI και συμπεριλαμβάνουν τις σχετικές λειτουργίες του τερματικού εξοπλισμού. Περιλαμβάνονται:
    - Φωνητικές κλήσεις
    - Κλήσεις Ανάγκης (Emergency Calls)
    - Fax
    - Μηνύματα Κειμένου (SMS)
    - Cell Broadcast προς όλα τα κινητά της κυψέλης
    - Τηλεφωνητής (Voice Mail)
    - Υπηρεσίες Κλήσεων Ομάδας (Voice Group Service)
    - Πρόσβαση στο Internet

- **Bearer Services**: Εξασφαλίζουν τη μετάδοση των σημάτων μεταξύ σημείων πρόσβασης των δικτύων και καθορίζουν τις υπηρεσίες που μπορεί να υποστηρίξει ένα δίκτυο. Μια υπηρεσία bearer καθορίζεται από ένα σύνολο χαρακτηριστικών που τη διαφοροποιούν από μια άλλη bearer υπηρεσία. Τα χαρακτηριστικά αυτά περιέχουν:

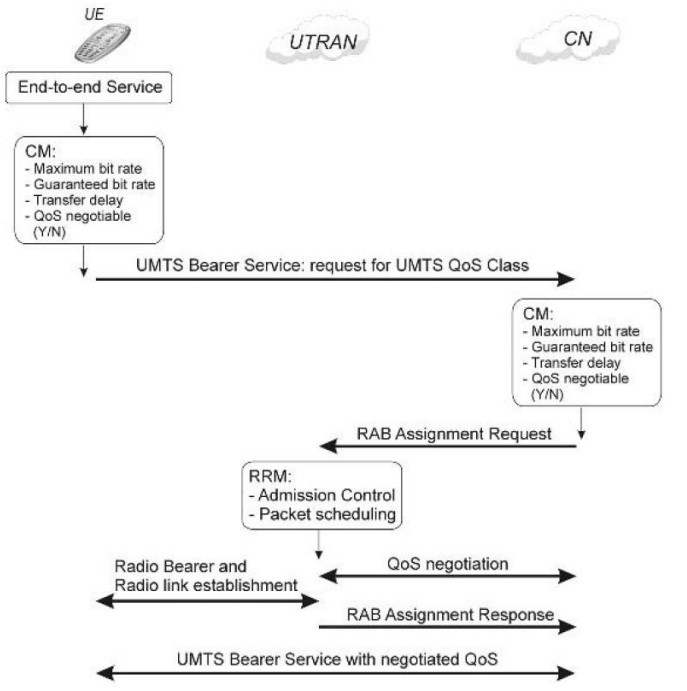
  - τον τύπο της τηλεπικοινωνιακής κίνησης (π.χ. connection-oriented ή connectionless, constant bit rate – CBR ή variable bit rate – VBR, point-to-point ή point-to-multipoint)

  - την ποιότητα της πληροφορίας (π.χ. ρυθμός μετάδοσης, BER, μέγιστη καθυστέρηση, μεταβολή στην καθυστέρηση)

  - τους υποστηριζόμενους ρυθμούς μετάδοσης

  Όταν μια εφαρμογή ζητήσει από το δίκτυο μια υπηρεσία με συγκεκριμένες παραμέτρους, το δίκτυο ελέγχει τους διαθέσιμους πόρους και αποδίδει την απαιτούμενη υπηρεσία ή προτείνει μια υπηρεσία χαμηλότερης ποιότητας. Η εφαρμογή είτε αποδέχεται είτε απορρίπτει την απόδοση του δικτύου (*σχήμα 1*).

- **Supplementary and Value Added Services**: Υπηρεσίες που εξαρτώνται από τον παροχέα υπηρεσιών ή του δικτύου. Στις προηγούμενες που έχουμε αναφέρει, προστίθεται η δυνατότητα ταυτόχρονων πολλαπλών κλήσεων.



Bearer management—principle diagram

Σχήμα 1

1.3 Επαγγελματικά εργαλεία μέτρησης και ανάλυσης δικτύων κινητών επικοινωνιών (State-of-the-art)

TEMS PORTFOLIO (Ascom)

Το τμήμα Network Testing της Ascom προσφέρει το TEMS PORTFOLIO, ένα ολοκληρωμένο σύνολο λύσεων για έλεγχο, αποτύπωση και ανάλυση της απόδοσης του δικτύου, διευκολύνοντας την ανάπτυξη, τη βελτιστοποίηση και τη συντήρηση δικτύων κινητών επικοινωνιών. Πιο αναλυτικά, προσφέρει τα ακόλουθα:

- **LTE Offerings**: Προϊόντα που αφορούν κυρίως την τεχνολογία LTE (Long Term Evolution):

    - **TEMS Discovery**: Αφορά τη μετεπεξεργασία των μετρούμενων δεδομένων για τη ραδιοεπαφή. Υποστηρίζει τις τεχνολογίες CDMA2000, 1xEV, GSM/GPRS/EDGE, WCDMA, HSDPA/HSUPA/HSPA+, TD-SCDMA, WinMAX και LTE. Περιέχει πάνω από 200 στοιχεία πληροφοριών για το LTE. Αποκωδικοποιεί LTE L2/3 μηνύματα και εισάγει τα δεδομένα της κάθε κυψέλης σε TEMS Xml ή csv/tab δομές.

    - **TEMS Investigation**: Χρησιμοποιείται για συλλογή δεδομένων κατά τη διάρκεια της οδήγησης. Παρέχει πληροφορίες για LTE DL και LTE UL, για ρυθμαπόδοση (throughput) και χρόνους καθυστέρησης, όπως επίσης και για τις γειτονικές κυψέλες. Στο παράθυρο του χάρτη εμφανίζει μετρήσεις και συμβάντα. Υποστηρίζει μετρήσεις FTP, HTTP, Ping και UDP.

    - **TEMS Symphony Suite**: Εργαλείο για LTE Benchmarking και έλεγχο δικτύου με απαιτήσεις QoS.

- **Test and Measurement**: Συσκευές για τον έλεγχο και την αξιολόγηση της απόδοσης και της ποιότητας των ασύρματων δικτύων και υπηρεσιών:

    - **TEMS Pocket**: Hand-held εργαλείο για την επαλήθευση, τη διατήρηση και την αντιμετώπιση σφαλμάτων δικτύων κινητών επικοινωνιών, καθώς και για βασικές διεργασίες οργάνωσης κυψελών. Συλλέγει δεδομένα και συμβάντα και τα παρουσιάζει στην οθόνη της συσκευής. Λαμβάνει μετρήσεις για δίκτυα WCDMA/HSPA 850/900/1700/1900/2100 MHz, GSM/GPRS/EDGE 850/900/1800/1900/2100 MHz, CDMA 1x/EVDO Rev A 850/1900 MHz (US). Αποθηκεύει τις μετρήσεις σε logfiles, ενώ διαθέτει επιλογές για FTP και ρυθμίσεις για τις κυψέλες.

    - **TEMS Investigation**: Προσφέρει συλλογή και μετεπεξεργασία δεδομένων, καθώς και ανάλυση πραγματικού χρόνου. Χρησιμοποιείται για την αποτύπωση της συμπεριφοράς στο τερματικό, με μετρήσεις όπως η κάλυψη της κυψέλης, η χωρητικότητα, η προσβασιμότητα κ.α. Υποστηρίζει indoor, outdoor και pedestrian μετρήσεις.

- **Benchmarking and monitoring**: Συστήματα αξιολόγησης και αποτύπωσης της συμπεριφοράς και της ποιότητας των ασύρματων δικτύων και των υπηρεσιών τους:

    - **TEMS Automatic**: Παρέχει πληροφορίες απ' άκρο σε άκρο για την QoS. Δίνει μια εικόνα της ποιότητας του δικτύου, όπως αυτή γίνεται αντιληπτή από τους συνδρομητές, ενώ συλλέγει τις απαραίτητες λεπτομέρειες για αναφορές, αντιμετώπιση σφαλμάτων και ανάλυση. Συλλέγει δεδομένα για το δίκτυο με αυτόματατο τρόπο, ελέγχει την ποιότητα φωνής και μετάδοσης δεδομένων, εντοπίζει σφάλματα, όρια χωρητικότητας και προβλήματα ρυθμίσεων. Ακόμη, συλλέγει δεδομένα από ανταγωνιστικά δίκτυα για συγκριτική αξιολόγηση.

    - **TEMS Monitor Master**: Πρόκειται για μια πλατφόρμα για τον έλεγχο, την εμφάνιση και την αναφορά δραστηριοτήτων που μπορεί να κάνει ο χρήστης σε ένα κινητό τηλέφωνο, όπως ενασχόληση με mark-up based υπηρεσίες (HTML, WAP, WAP2, i-mode), υπηρεσίες μηνυμάτων (SMS, MMS), υλικό για download (συμπεριλαμβανομένης της J2ME, φωτογραφιών και ήχων κλήσης), οπτικοακουστικό υλικό (streaming, 3G βιντεοκλήσεις) κ.α.

    - **TEMS Symphony**: Πλατφόρμα για μετρήσεις διαφορετικού τύπου (in-vehicle, indoor, nomadic). Υποστηρίζει LTE, με ταχείς, πολυπύρηνους επεξεργαστές και μεγάλη χωρητικότητα μνήμης. Διαθέτει γραφικό περιβάλλον που επιτρέπει στον απομακρυσμένο χρήστη να βλέπει και να αναλύει τα δεδομένα. Έχει

ευελιξία ως προς την υποστήριξη πολλών τύπων συσκευών, περιλαμβανομένων των παραδοσιακών handsets και modems.

- **Reporting and Analysis**: Λογισμικό για την οπτικοποίηση, την ανάλυση και τη σύνταξη αναφορών της απόδοσης και της ποιότητας δικτύων κινητών επικοινωνιών:

    - **TEMS Box Office**: Πρόκειται για την πλατφόρμα σύνταξης αναφορών του TEMS Monitor Master. Λόγω της ευελιξίας και των πολλών ρυθμίσεων προσφέρει στο χρήστη τη δυνατότητα να δημιουργεί τις δικές του αναφορές, όπως αυτός επιθυμεί. Οι αναφορές παραδίδονται μέσω Web Browser, δηλαδή δεν απαιτείται κάποιος ειδικός Client.

    - **TEMS Discovery**: Αναλύθηκε στα LTE Offerings

    - **TEMS Visualization**: Υποστηρίζει την ανάλυση δεδομένων που έχουν αποσπαστεί απευθείας από την υποδομή του δικτύου με βάση συντελεσμένα γεγονότα. Αυτό επιτρέπει τον άμεσο προσδιορισμό προβλημάτων που σχετίζονται με την QoS. Για παράδειγμα μπορεί να εντοπίσει την αιτία προβλημάτων στη ραδιοσυχνότητα ή στη χωρητικότητα του δικτύου ή να προσδιορίσει αν κάποιο πρόβλημα αφορά μια συγκεκριμένη συσκευή ή ένα τύπο συσκευών.

Στα παραπάνω προστίθεται και μια σειρά συμβουλευτικών υπηρεσιών που δεν υπάρχει λόγος να αναλυθούν σε αυτή την εργασία.

Nemo (Anite)

Η Anite προσφέρει το Nemo portfolio, το οποίο περιλαμβάνει εργαλεία μέτρησης, αξιολόγησης και ελέγχου για δίκτυα κινητών επικοινωνιών.

- **LTE Testing**: Παρέχει τη δυνατότητα για συγχρονισμένες και παράλληλες μετρήσεις (in-vehicle) αξιολογούμενες με βάση δείκτες απόδοσης. Υποστηρίζει ανάλυση δεδομένων για LTE και καλύπτει όλα τα στάδια, από τη διόρθωση σφαλμάτων και το σχεδιασμό, μέχρι την ανάπτυξη και τη βελτιστοποίηση. Το Nemo Outdoor υποστηρίζει παραμέτρους LTE UE για φυσικό δίαυλο, μετρήσεις κυψέλης, μετρήσεις CQI, RRC/NAS σηματοδοσία, διαδικασίες RACH, στατιστικά στο MAC στρώμα, EMM/ESM συνόδους κ.α. Ενεργοποιεί το LTE scanning σε συγχρονισμό με τις μετρήσεις του εξοπλισμού του χρήστη, με υψηλή ταχύτητα scanning και ακρίβεια μετρήσεων. Επίσης, υποστηρίζει RSS scanning και OFDMA scanning σε P-SCH/S-SCH λειτουργίες.

- **Drive Testing**: Το Nemo Outdoor, το Nemo Compact-i και το Nemo Handy συνδυάζονται για να σχηματίσουν ένα ολοκληρωμένο σύστημα μετρήσεων (drive test tools). Το Nemo Outdoor (laptop-based drive test tool) υποστηρίζει σχεδόν όλες τις τεχνολογίες (GSM, CDMA2000, EV-DO, TD-SCDMA, WCDMA, HSDPA, HSUPA, HSPA+, WiMAX, LTE) και επιτρέπει πολλά είδη μετρήσεων, όπως ποιότητας φωνής και βίντεο. Τα Nemo Handy και Nemo Compact-i είναι handheld εργαλεία τα οποία προσφέρουν δυνατότητες μετρήσεων με ευχρηστία και οικονομία.

- **Benchmarking**: Το Nemo Outdoor χρησιμοποιείται για να εκτελεί benchmarking σε πάνω από 200 τερματικά και scanners, για σχεδόν κάθε τεχνολογία δικτύου και για πολλαπλές παράλληλες συνδέσεις δεδομένων. Στο Nemo Multi Handy, ένα κεντρικό (Master) τερματικό λειτουργεί σαν κέντρο εντολών, διενεργώντας μετρήσεις ελέγχοντας συγχρονισμένες μετρήσεις σε 1-6 μονάδες (Slaves), με τις μονάδες να μοιράζονται ένα GPS δέκτη. Ακόμη, το Nemo Autonomous χρησιμοποιείται για την εκτέλεση μεγάλης διάρκειας αυτοματοποιημένων benchmarking μετρήσεων.

- **Quality Testing**: Ενσωματώνει λύσεις για μετρήσεις voice, video streaming και ποιότητας βιντεοκλήσεων στις ραδιοεπαφές των CDMA, GSM, WCDMA και TD-SCDMA δικτύων. Αναλυτικά, το Nemo Outdoor εκτελεί fixed-to-mobile, mobile-to-mobile και παράλληλη mobile-to-mobile αξιολόγηση ποιότητας φωνής, αλλά και μετρήσεις βελτιστοποίησης, στηριζόμενο στον PESQ (Perceptual Evaluation of Speech Quality) αλγόριθμο. Το Nemo Handy επιτρέπει mobile-to-mobile και mobile-to-fixed uplink benchmarking για την ποιότητα φωνής, στηριζόμενο στον PESQ και downlink στηριζόμενο στον PSM Mobile αλγόριθμο. Επίσης, τα δύο αυτά εργαλεία επιτρέπουν τη μέτρηση της ποιότητας του video streaming με χρήση του αλγορίθμου PVI (Psytechnics' Video IP). Το Nemo Autonomous εκτελεί αυτοματοποιημένο mobile-to-mobile και mobile-to-fixed uplink benchmarking για την ποιότητα φωνής με βάση τον PESQ αλγόριθμο και downlink με βάση τον PSM, για μεγάλης διάρκειας μετρήσεις. Ακόμη, το Nemo Outdoor αξιολογεί την ποιότητα των βιντεοκλήσεων χρησιμοποιώντας τον VQI (Video Quality Index) αλγόριθμο. Τέλος, το Nemo Analyze παρέχει οπτικοποιημένη μετεπεξεργασία των δεδομένων (Voice: MOS, Call events and statistics, Call setup time,

End-to-end call setup time / Video Call: MOS, Call events and statistics, Call setup time, End-to-end call setup time / Video streaming: MOS, Packet error rate, Jitter, Events and statistics, Streaming initial buffering time, Streaming service access time, Streaming service access time to buffering, Streaming session time).

- **Indoor Testing**: Το Nemo Indoor υπηρετεί την ανάγκη για μεγιστοποίηση και βελτιστοποίηση της κάλυψης του δικτύου, σε θέσεις όπως κτίρια εταιρειών και άλλες indoor τοποθεσίες, όπου η κίνηση κλήσεων και μεταφοράς δεδομένων είναι πυκνή και συγκεντρωμένη. Ενδείκνυται για Qos και QoE (Quality of Experience) μετρήσεις και υποστηρίζει όλες τις τελευταίες τεχνολογίες συμπεριλαμβανομένων των CDMA2000, EV-DO, GSM, WCDMA, HSDPA, HSUPA, HSPA+, TD-SCDMA TETRA, DVB-H, WiMAX και LTE.

Ένα παρόμοιο με τα παραπάνω εμπορικά συστήματα έχει αναπτύξει και η SwissQual (Diversity, QualiPoc, NetQual). Οι λύσεις αυτές, ωστόσο, είναι ιδιαίτερα ακριβές και προσφέρουν πλήθος δυνατοτήτων που μπορεί να μην είναι όλες αναγκαίες σε κάθε εταιρεία κινητών επικοινωνιών. Προς την κατεύθυνση αυτή αναπτύξαμε μια ολοκληρωμένη λύση για μέτρηση bitrate (downlink) και latency που διαθέτει χαρακτηριστικά που είδαμε στα προηγούμενα συστήματα, με προτερήματα την ευχρηστία, την έξυπνη δομή και την ευκολία εγκατάστασης και εκτέλεσης μετρήσεων.

1.4 Μέθοδοι πολλαπλής πρόσβασης καναλιού

Οι τεχνικές πολλαπλής πρόσβασης αποδίδουν αποκλειστικούς διαύλους σε πολλαπλούς χρήστες.

Στην **FDMA (Frequency-division Multiple Access)**, το ολικό εύρος ζώνης του συστήματος χωρίζεται σε διαύλους μη επικαλυπτόμενους στη συχνότητα, που αποδίδονται σε διαφορετικούς χρήστες. Απαιτεί πολύ αυστηρά προκαθορισμένα ζωνοπερατά φίλτρα. Η απομόνωση των διαύλων και κατά συνέπεια η ορθογωνιότητα των σημάτων εξασφαλίζονται με την εισαγωγή συχνοτικών διαστημάτων φύλαξης μεταξύ των διαύλων. Κάθε δίαυλος αποδίδεται σε ένα χρήστη και δεν μπορεί να χρησιμοποιηθεί από άλλους κατά τη διάρκεια μιας τηλεφωνικής συνδιάλεξης. Αφενός, λοιπόν, ο χρήστης περιορίζεται σε ένα μικρό εύρος ζώνης, ακόμα και αν δεν υπάρχουν πολλοί χρήστες στο σύστημα και αφετέρου, το γεγονός της αποκλειστικής διάθεσης του διαύλου ακόμα και για διαστήματα όπου δεν υπάρχουν δεδομένα προς μετάδοση (για παράδειγμα περίοδοι σιωπών) οδηγεί σε μικρή φασματική απόδοση.

Συνήθως συνδυάζεται με TDD τεχνική, ώστε να αποφεύγεται η ταυτόχρονη εκπομπή και λήψη και επιπλέον να είναι δυνατή η χρήση τεχνικών διαφορισμού στο Σταθμό Βάσης, τόσο για την ευθεία όσο και για την αντίστροφη ζεύξη, λόγω της ομοιότητας των διαύλων στις δύο

κατευθύνσεις. Ως συνέπεια, η πολυπλοκότητα της επεξεργασίας του σήματος μεταφέρεται στο σταθμό βάσης.

Στην **TDMA (Time-division Multiple Access)**, ο χρόνος χωρίζεται σε μη επικαλυπτόμενες χρονοσχισμές που αποδίδονται σε διαφορετικούς χρήστες. Απαιτεί το συγχρονισμό μεταξύ των χρηστών. Η ορθογωνιότητα μεταξύ των χρηστών επηρεάζεται σημαντικά από τη διασυμβολική παρεμβολή που προκαλεί ένα σήμα, από μια χρονοσχισμή σε επόμενες χρονοσχισμές. Σε σχέση με την τεχνική FDMA, ο ρυθμός μετάδοσης είναι πολλαπλάσιος κατά ένα παράγοντα, ίσο με τον αριθμό των χρηστών που μοιράζονται το συχνοτικό δίαυλο. Αν ένας δίαυλος δεν χρησιμοποιείται μια χρονική στιγμή, η αντίστοιχη χρονοσχισμή δεν μπορεί να χρησιμοποιηθεί από άλλους χρήστες του συστήματος.

Μερικά πλεονεκτήματα της TDMA είναι:

- Δεν υπάρχουν προβλήματα ενδοδιαμόρφωσης.
- Μπορεί να χρησιμοποιηθεί σε υπηρεσίες φωνής και δεδομένων.
- Υποστηρίζει μεταβαλλόμενο ρυθμό μετάδοσης.
- Είναι φασματικά πιο αποδοτική, αφού δεν χρησιμοποιούνται συχνοτικά διαστήματα φύλαξης.
- Προσφέρει μικρότερη κατανάλωση ισχύος, επειδή η μετάδοση γίνεται σε ριπές.

FDMA και TDMA έχουν, ωστόσο, κάποια βασικά μειονεκτήματα:

- Κατά τη διάρκεια περιόδων σιωπής σε μια τηλεφωνική συνδιάλεξη (50-65% του χρόνου) δεν είναι δυνατή η επαναχρησιμοποίηση του διαύλου από άλλους χρήστες.
- Η επαναχρησιμοποίηση συχνοτήτων που μπορεί να επιτευχθεί έχει σχετικά αραιή δομή.
- Το φαινόμενο των διαλείψεων δημιουργεί σημαντικά προβλήματα.

Στην **CDMA (Code-division Multiple Access)**, ο χρόνος και το εύρος ζώνης χρησιμοποιούνται ταυτόχρονα από διαφορετικούς χρήστες που διαμορφώνονται από ορθογωνικούς ή ημι-ορθογωνικούς κώδικες διασποράς. Η τεχνική αυτή επιτρέπει σε κάθε σταθμό να μεταδίδει συνεχώς σε όλο το φάσμα συχνοτήτων. Οι πολλαπλές ταυτόχρονες μεταδόσεις διαχωρίζονται με βάση τη θεωρία της κωδικοποίησης. Η CDMA δεν κάνει την υπόθεση ότι τα πλαίσια που

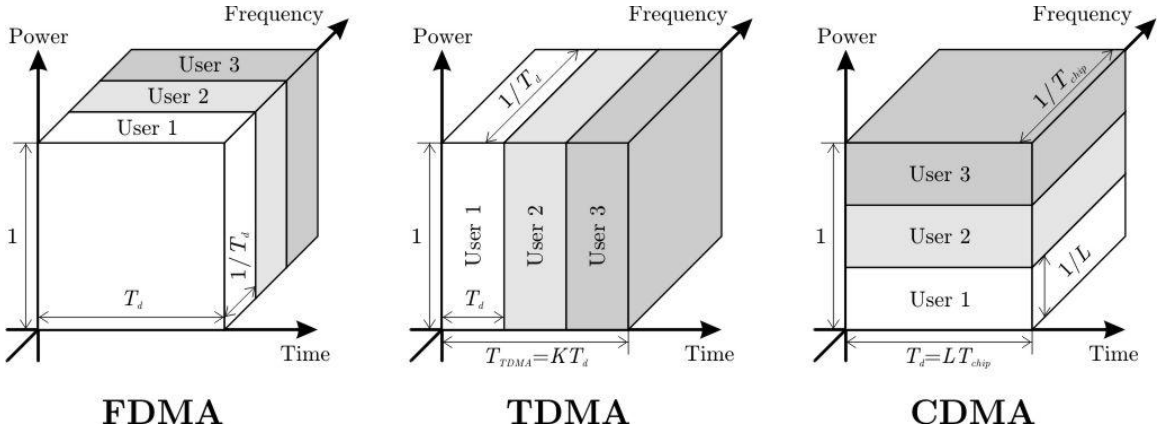
παρουσιάζουν διένεξη αλλοιώνονται ολοκληρωτικά. Αντιθέτως, υποθέτει ότι τυχόν πολλαπλά σήματα προστίθενται γραμμικά.

Το βασικό στοιχείο της είναι πως μπορούμε να εξάγουμε το επιθυμητό σήμα, απορρίπτοντας οτιδήποτε άλλο ως τυχαίο θόρυβο. Στην πράξη, το στενής ζώνης σήμα πληροφορίας διαμορφώνεται από ένα ευρείας ζώνης σήμα διασποράς. Κάθε χρήστης έχει το δικό του σήμα διασποράς (κώδικα). Κάθε σήμα διασποράς είναι ορθογωνικό ως προς τα υπόλοιπα, οπότε με την τεχνική της συσχέτισης μπορούμε να αναγνωρίσουμε στο δέκτη τον επιθυμητό χρήστη. Για κάθε δεδομένο εύρος ζώνης υπάρχει μόνο περιορισμένος αριθμός ορθογωνικών κωδίκων διασποράς. Με τους ημι-ορθογωνικούς κώδικες, δεν υπάρχει περιορισμός στον αριθμό των κωδίκων σε δεδομένο εύρος ζώνης, ωστόσο παραμένει μια παρεμβολή μεταξύ των χρηστών και ο δέκτης δεν μπορεί να ξεχωρίσει τελείως το επιθυμητό σήμα από τους άλλους χρήστες. Η μέθοδος που χρησιμοποιείται για την παραγωγή των ημι-ορθογωνικών CDMA σημάτων είναι η **DS-CDMA** (Direct Sequence – CDMA).

Μερικά πλεονεκτήματα της CDMA είναι:

- Αξιοποιεί τις σιωπηλές περιόδους στις τηλεφωνικές συνομιλίες.
- Δεν απαιτείται η χρήση ισοσταθμιστή (equalizer) για την αντιμετώπιση διασυμβολικής παρεμβολής.
- Στα κυψελωτά συστήματα, όπου γειτονικές κυψέλες χρησιμοποιούν τον ίδιο δίαυλο, δεν απαιτείται μεταπομπή στη συχνότητα.
- Δεν απαιτούνται διαστήματα φύλαξης.
- Επιτυγχάνει πολύ μεγαλύτερες χωρητικότητες σε επίπεδο χρηστών ανά κυψέλη.
- Μπορεί να παρέχει εύρος ζώνης κατά απαίτηση (bandwidth on demand).
- Δεν απαιτούνται πολύπλοκες τεχνικές απόδοσης και διαχείρισης του φάσματος.

Τέλος, στην **SDMA (Space-division Multiple Access)** γίνεται χρήση έξυπνων κεραιών που δίνουν τη δυνατότητα του χωρικού διαχωρισμού των χρηστών από τους Σταθμούς Βάσης με αποτέλεσμα να είναι δυνατή η χρήση των ίδιων πόρων του συστήματος για χωρικά διασκορπισμένους χρήστες. Γίνεται έλεγχος της ακτινοβολούμενης Η/Μ ενέργειας για κάθε χρήστη στην περιοχή κάλυψης. Ο έλεγχος επιτυγχάνεται με τη χρήση έξυπνων, προσαρμοστικών κεραιών.



Κεφάλαιο 2

Σύστημα Καταγραφής της Εμπειρίας Κινητών Χρηστών σε Δίκτυα Ασύρματων Επικοινωνιών

2.1 Λειτουργία του UDP Download

2.1.1 Περιγραφή λειτουργίας

Ο Client (εφαρμογή KampRate στην κινητή συσκευή) αποστέλλει ένα «TodoPacket» πακέτο στον Server, το οποίο φέρει την πληροφορία για το μέγεθος πακέτου (packet size) που θέλει να χρησιμοποιηθεί (200, 500, 1000, 1400 Bytes). Ο Server λαμβάνει το πακέτο και ξεκινά ένα νέο thread (DLProcessRequest), όπου επεξεργάζεται την πληροφορία που φέρει το ληφθέν πακέτο. Αφού έλαβε «TodoPacket» πακέτο, ξεκινά ένα νέο thread (DLSendData) μέσω του οποίου στέλνει πακέτα στον Client με το ζητούμενο μέγεθος. Ο Client, μόλις λάβει το πρώτο πακέτο από τον Server, θεωρεί πως έχει «συνδεθεί» με αυτόν (EstablishedConnection == true) και για κάθε πακέτο που υποδέχεται αυξάνει την τιμή του μετρητή με τον αριθμό των ληφθέντων πακέτων κατά ένα. Σε κάθε διάστημα υπολογίζει το bit rate και μηδενίζει το μετρητή για να ξεκινήσει το επόμενο.

$$bit\ rate = \frac{αριθμός\ ληφθέντων\ πακέτων * μέγεθος\ πακέτου}{\text{χρόνος}}$$

Όταν ο Client τερματίσει την καταγραφή, αποστέλλονται «FIN» πακέτα στον Server. Ο Server, που κάθε στιγμή μπορεί να λάβει πακέτα από τους Clients υποδεχόμενός τα σε ένα thread ειδικά γι' αυτό το σκοπό, περιεργάζεται το πακέτο και βλέπει ότι πρόκειται για «FIN» πακέτο. Διατρέχει, λοιπόν, το διάνυσμα με όλες τις ενεργές «συνδέσεις» με τους Clients, και με βάση την IP Address του Client τερματίζει τη «σύνδεση», στέλνοντας πίσω στον Client «ACK» πακέτα για επιβεβαίωση. Ο Client περιμένει να λάβει τα «ACK» πακέτα εντός ενός χρονικού περιθωρίου (timeout = 5 sec). Αν δεν τα λάβει, ξαναστέλνει «FIN» πακέτα στον Server, ο οποίος διατρέχει πάλι το διάνυσμα με τις «συνδέσεις», δεν βρίσκει «σύνδεση» αυτή τη φορά για να την τερματίσει, αφού την είχε τερματίσει προηγουμένως, και στέλνει εκ νέου «ACK» πακέτα στον Client. Συνολικά, ο Client προσπαθεί 5 φορές μέχρι να λάβει ένα «ACK» πακέτο, διαφορετικά εγκαταλείπει.

Στην περίπτωση που ο Client έχει στείλει «TodoPacket» πακέτο στον Server και δεν έχει λάβει κάποιο πακέτο εντός ενός χρονικού περιθωρίου (timeout = 3 sec), στέλνει «FIN» πακέτα στον Server ώστε αυτός, αν έχει ξεκινήσει να στέλνει, να σταματήσει τη «σύνδεση» και προσπαθεί ξανά από την αρχή. Συνολικά δοκιμάζει 5 φορές και εμφανίζει μήνυμα «Server Unavailable», αν αποτύχουν όλες του οι προσπάθειες.

Ακολουθεί σχηματικό παράδειγμα που περιλαμβάνει τις περισσότερες από τις προαναφερθείσες περιπτώσεις:

2.1.2 Παράδειγμα

Client (KampRate Application)

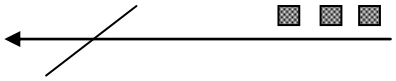
Server

Στέλνει TodoPacket στον Server

TodoPacket →

Από την πληροφορία που φέρει το TodoPacket επιλέγει το μέγεθος πακέτου που θα χρησιμοποιήσει

Περιμένει να λάβει το πρώτο πακέτο. Το χρονικό περιθώριο παρέρχεται δίχως να λάβει κάποιο πακέτο από τον Server

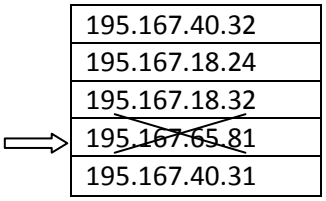


Αρχίζει να στέλνει πακέτα στον Client

Διατρέχει το διάνυσμα με τις «συνδέσεις» για να τερματίσει τη σύνδεση με τον Client

Στέλνει FinPacket για να σταματήσει να στέλνει πακέτα ο Server (αν έχει ξεκινήσει)

FinPacket →

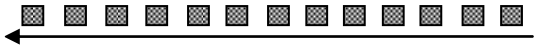


Στέλνει εκ νέου TodoPacket στον Server

TodoPacket →

Από την πληροφορία που φέρει το TodoPacket επιλέγει το μέγεθος πακέτου που θα χρησιμοποιήσει

Λαμβάνει το πρώτο πακέτο (θέτει EstablishedConnection == true). Αυξάνει το μετρητή για κάθε πακέτο που λαμβάνει.



Αρχίζει να στέλνει πακέτα στον Client

Στο τέλος του διαστήματος υπολογίζει το bit rate:

$$bit\ rate = \frac{\text{Αριθμός πακέτων} * \text{μέγεθος πακέτου}}{\text{χρόνος}}$$

Τερματίζει την καταγραφή

FinPacket →

Διατρέχει το διάνυσμα με τις «συνδέσεις» για να τερματίσει τη «σύνδεση» με τον Client και τερματίζει τη «σύνδεση»

Περιμένει να λάβει AckPacket. Δεν το λαμβάνει εντός 5sec.

← AckPacket

Στέλνει πακέτα επιβεβαίωσης

Ξαναστέλνει FinPacket

FinPacket →

Διατρέχει το διάνυσμα με τις «συνδέσεις». Έχει ήδη τερματίσει τη «σύνδεση».

Λαμβάνει AckPacket και τερματίζει

← AckPacket

Στέλνει πακέτα επιβεβαίωσης

2.1.3 Σημειώσεις επί του κώδικα

Server και Client «ακούν» στη θύρα (port) 5001.

```
class Download extends Thread {
   private static final int SERVERPORT = 5001;
   private static final int CLIENTPORT = 5001;
   String msg;
   Boolean EstablishedConnection = false;
   DatagramSocket socket;
   int PacketSize;
   String ServerIP;
   InetAddress serverAddr;
   String clientAddr;

   public Download(String msg1, int PacketSize1, String ServerIPChoice) {
      msg = msg1;
      PacketSize = PacketSize1;
      ServerIP = ServerIPChoice;
   }

   public void run() {
      try {
         // Sending message to initiate, with packet selection info
         serverAddr = InetAddress.getByName(ServerIP);

         int TryToConnect = 0;
         while ((EstablishedConnection.equals(false)) && (TryToConnect < 5)) {
            socket = new DatagramSocket();
            byte[] buf = (msg).getBytes();
            DatagramPacket TodoPacket = new DatagramPacket(buf, buf.length,
                  serverAddr, SERVERPORT);
            socket.send(TodoPacket);
            Log.e("UDP_Download", "C: Sent TodoPacket. Waiting...");
            final String clientAddr = getLocalIpAddress();
            socket.close();

            // Receiving first packet to start Timer
            socket = new DatagramSocket(CLIENTPORT,
                  InetAddress.getByName(clientAddr));
            byte[] buf1 = new byte[PacketSize];
            DatagramPacket first_packet = new DatagramPacket(buf1, buf1.length);
            socket.setSoTimeout(3000);
            try {
               socket.receive(first_packet);
               EstablishedConnection = true;
            } catch (IOException e) {
               // If no packet was received in 3 seconds timeout, send to
               // Server "FIN" packet
               // to stop sending (if Server has started sending data),
               // close socket and try again.
               socket.close();
               socket = new DatagramSocket();
               byte[] buf2 = ("FIN ").getBytes();
               DatagramPacket FinishPacket = new DatagramPacket(buf2,
                     buf2.length, serverAddr, SERVERPORT);
               for (int i = 0; i < 3; i++) {
                  socket.send(FinishPacket);
               }
               socket.close();
               TryToConnect++;
            }
         }
```

Όσο δεν έχει εγκατασταθεί «σύνδεση» και οι προσπάθειες είναι λιγότερες από 5, στέλνονται TodoPacket στον Server.

Αναζητεί στα Network Interfaces την IP Address της κινητής συσκευής (θα επιστρέψει null, αν βρει την Loopback Address)

3 sec περιμένει ο Client για να λάβει κάποιο πακέτο από τον Server. Αν παρέλθει το χρονικό διάστημα, στέλνει «FIN» πακέτα και επιχειρεί ξανά.

```
         if (EstablishedConnection == true) {
            d.setTrue();
            socket.setSoTimeout(0);

            // Call DLIndoorsTimer for Indoors Measurement
            if (IndoorsSelection == true) {
               double lat = IndoorsGeopointSingleton.getInstance(0, 0).latitude();
               double lon = IndoorsGeopointSingleton.getInstance(0, 0).longitude();
               DLIndoorsTimer dlITimer = new DLIndoorsTimer(lat, lon);
               new Thread(dlITimer).start();
            }

            StartTime = System.currentTimeMillis();

            // Receiving packets
            byte[] buf3 = new byte[PacketSize];
            DatagramPacket IncomingPacket = new DatagramPacket(buf3,buf3.length);
            while (d.getValue() == true) {
               socket.receive(IncomingPacket);
               c1.AddCount();
               // Log.e( "GPSLoggerService", " Packet number:" + c1.GetCount() );
            }
            socket.close();
         } else {
            ServerUnavailable();
         }
      } catch (Exception e) {
         Log.e("UDP_Download", "S: ERROR", e);
      }
   }

   public void stopDownload() {
      stopDL stDL = new stopDL(serverAddr, clientAddr);
      new Thread(stDL).start();
   }
}

class stopDL extends Thread {
   private static final int SERVERPORT = 5001;
   private static final int CLIENTPORTSTOP = 5002;
   private InetAddress serverAddr;
   private String clientAddr;
   private DatagramSocket socketT;
   private DatagramSocket socketR;

   public stopDL(InetAddress serverAddr1, String clientAddr1) {
      serverAddr = serverAddr1;
      clientAddr = clientAddr1;
   }

   public void run() {
      if (d.getValue() == true) {
         d.setFalse();
         Boolean ReceivedACK = false;
         String clientAddrT = clientAddr;

         int trytostop = 0;
         try {
            // Log.e("GPSLoggerServcice", InetAddress.getByName(clientAddrT).toString());
            clientAddrT = getLocalIpAddress();
            socketR = new DatagramSocket(CLIENTPORTSTOP,
                  InetAddress.getByName(clientAddrT));
```

Στην περίπτωση που έχουμε Indoors μέτρηση, εκκινεί ένας timer (DLIndoorsTimer).

Count++ για κάθε ληφθέν πακέτο

Αν αποτύχουν όλες οι προσπάθειες για «σύνδεση», καλείται η μέθοδος ServerUnavailable() για να ειδοποιηθεί ο χρήστης με μήνυμα στην οθόνη.

Ο Client ακούει στη θύρα (port) 5002 στην περίπτωση του stopDL. Σε αυτή στέλνει ο Server τα «ACK» πακέτα, ότι, δηλαδή, η «σύνδεση» τερματίστηκε επιτυχώς.

```
            socketR.setSoTimeout(5000);
         } catch (SocketException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
         } catch (UnknownHostException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
         }
         byte[] buf5 = new byte[3];
         DatagramPacket ACKpacket = new DatagramPacket(buf5, buf5.length);

         while ((ReceivedACK == false) && (trytostop < 5)) {
            try {
               if (socketT != null) {
                  socketT.close();
               }
               socketT = new DatagramSocket();
               byte[] buf4 = ("FIN ").getBytes();
               DatagramPacket FinPacket = new DatagramPacket(buf4,
                     buf4.length, serverAddr, SERVERPORT);
               for (int i = 0; i < 3; i++) {
                  socketT.send(FinPacket);
               }
               // socketT.close();
               // Log.e( "UDP_Download", "FinPackets Sent" );

               try {
                  socketR.receive(ACKpacket);
                  String ACKpacketString = new String(ACKpacket.getData());
                  if (ACKpacketString.equals("ACK")) {
                     ReceivedACK = true;
                     socketR.close();
                     ReceivedACKfromServer();
                     Log.e("UDP_Download", "Received ACK packet");
                  } else {
                     Log.e("UDP_Download", "Received a packet but not ACK");
                  }
               } catch (SocketException e) {
                  e.printStackTrace();
                  Log.e("UDP_Download", e.toString(), e);
               }
            } catch (IOException e) {
               e.printStackTrace();
               Log.e("UDP_Download", "Other Error" + e.toString(), e);
            }
            trytostop++;
         }
         if (socketR != null) socketR.close();
         if (socketT != null) { socketT.close(); }

         if (ReceivedACK == false) {
            NoACKReplyfromServer();
         }
      }
   }
}

class DLIndoorsTimer extends Thread {
   private double latitude;
   private double longitude;
   private Location fixedLocation = new Location("fixedloc");
```

Όσο δεν έχει λάβει «ACK» πακέτα από τον Server και ο αριθμός των προσπαθειών είναι μικρότερος από 5, στέλνει «FIN» πακέτα στον Server για να τερματίσει τη «σύνδεση».

Έλεγχος για το αν το ληφθέν πακέτο είναι «ACK».

Όταν ληφθεί το πακέτο επιβεβαίωσης από τον Server, ειδοποιεί το χρήστη με μήνυμα στην οθόνη.

Εάν τελικά δεν ληφθεί «ACK» πακέτο, ειδοποιείται ο χρήστης με μήνυμα στην οθόνη, ώστε να γνωρίζει πως πιθανόν έχει παραμείνει ανοιχτή η σύνδεση στην πλευρά του Server και να επικοινωνήσει με τον Administrator.

```
   public DLIndoorsTimer(double latitude1, double longitude1) {
      latitude = latitude1;
      longitude = longitude1;
      fixedLocation.setLatitude(latitude);
      fixedLocation.setLongitude(longitude);
      fixedLocation.removeAccuracy();
      fixedLocation.removeAltitude();
      fixedLocation.removeBearing();
   }

   public void run() {
      while (d.getValue() == true) {
         try {
            Thread.sleep(10000);
            if (d.getValue() == true) {
               if (mStartNextSegment) {
                  mStartNextSegment = false;
                  startNewSegment();
               }
               storeLocation(fixedLocation);
            }
         } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
         }
      }
   }
}
```

«Περνάμε» στον Constructor τις συντεταγμένες της Fixed Location που θα χρησιμοποιηθεί για όλη την καταγραφή.

10 sec είναι ο χρόνος του κάθε διαστήματος. Ωστόσο, όπως αναφέρεται και στο API, η εντολή sleep δεν τηρεί αυτό το χρόνο με ακρίβεια, οπότε μπορεί να είναι λίγο μεγαλύτερος ή μικρότερος. Σε κάθε περίπτωση, στον υπολογισμό του bit rate διαιρούμε με το χρόνο που έχει παρέλθει και όχι με σταθερό χρόνο 10 sec, οπότε εξασφαλίζεται η αξιοπιστία των μετρήσεων.

Καλείται η μέθοδος στην οποία γίνονται οι υπολογισμοί και αποθηκεύεται η μέτρηση. Έπειτα ξεκινά το επόμενο διάστημα μετρήσεων, αν ο χρήστης δεν έχει τερματίσει την καταγραφή.

2.2 Λειτουργία του UDP Ping

2.2.1 Περιγραφή λειτουργίας

Το χρονικό διάστημα των 10 sec διαιρείται σε 4 υποδιαστήματα (0 - 2,5 / 2,5 – 5 / 5 – 7,5 / 7,5 – 10 sec). Στην αρχή κάθε υποδιαστήματος ο Client αποστέλλει στον Server έναν αριθμό πακέτων που προκύπτει από την επιλογή του χρήστη για μέγιστο αριθμό προσπαθειών (ping attempts) στις ρυθμίσεις (settings) της εφαρμογής. Για παράδειγμα, αν ο χρήστης επιλέξει αριθμό προσπαθειών ίσο με 16, τότε θα στέλνονται 4 πακέτα στην αρχή κάθε υποδιαστήματος, έτσι ώστε 4 * 4 = 16 πακέτα στα 10sec (16 προσπάθειες στα 10sec). Ο Server λαμβάνει το πακέτο που του έστειλε ο Client και στέλνει στην IP Address του Client ένα πακέτο ίδιου μεγέθους και με την ίδια πληροφορία (ίδιο όνομα - string).

Ο Client λαμβάνει το πακέτο και υπολογίζει τη διαφορά χρόνων, δηλαδή τη χρονική στιγμή που έλαβε το πακέτο «απάντηση» του Server μείον τη χρονική στιγμή που του το είχε στείλει. Για να θεωρηθεί επιτυχημένη η προσπάθεια και να καταγραφεί ο χρόνος της, θα πρέπει η ληφθείσα απάντηση να ανήκει στο ίδιο time segment με την αποστολή του πακέτου από τον Client. Αν, δηλαδή, ο Client έκανε μια προσπάθεια στο 1^{ο} time segment θα πρέπει να λάβει το πακέτο «απάντηση» μέσα στο 1ο time segment (t’ – t < 2500 msec), διαφορετικά θεωρείται χαμένη προσπάθεια και καταγράφεται ως lost.

Για την επίτευξη αυτού του τρόπου λειτουργίας - υπολογισμών επιλέχθηκε η εξής ονοματοδοσία:

1st Time Segment (0-2,5 sec)	2nd Time Segment (2,5-5 sec)	3rd Time Segment (5-7,5 sec)	4th Time Segment (7,5-10 sec)
0-0	1-0	2-0	3-0
0-1	1-1	2-1	3-1
0-2	1-2	2-2	3-2
0-3	1-3	2-3	3-3

Ο Client λαμβάνει τα πακέτα «απαντήσεις» σε κάθε time segment και στο τέλος του segment υπολογίζει τα AvgLatency, MinLatency, MaxLatency:

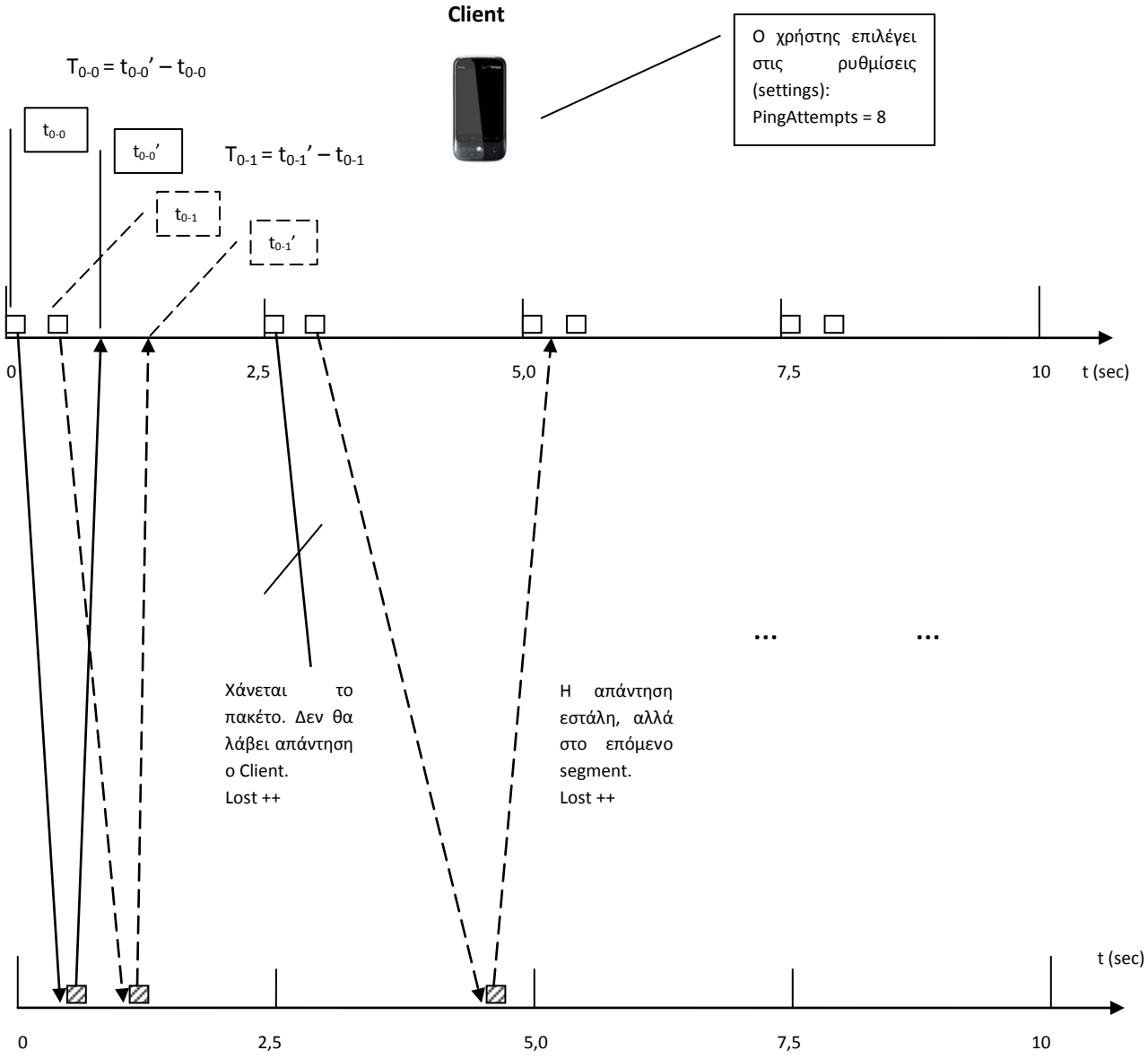
$AvgLatency = \frac{\sum_{i=0}^{R-1} \text{Latency(i)}}{R}$, όπου R = ο αριθμός των ληφθέντων πακέτων (received packets)

$MinLatency = \min_{0 \le i \le R-1} \text{Latency(i)}$

$MaxLatency = \max_{0 \le i \le R-1} \text{Latency(i)}$

Ακολουθεί σχηματικό παράδειγμα που αποτυπώνει τη λειτουργία που περιγράψαμε:

2.2.2 Παράδειγμα



Results:

$$AvgLatency = \frac{T1 + T2 + T5 + T7}{4}$$

MinLatency = T2
MaxLatency = T7

Received = 4
Lost = 4

Server

☐ Πακέτα που στέλνει ο Client

▨ Πακέτα «απαντήσεις» του Server

2.2.3 Σημειώσεις επί του κώδικα

Ο Server «ακούει» στη θύρα (port) 5000 για το Ping, ώστε να μπορεί παράλληλα να εξυπηρετεί Clients τόσο για Download όσο και για Ping (διαφορετικές θύρες).

```
class PingSend extends Thread {
   private static final int SERVERPORT = 5000;
   private int PingAttempts;
   private String ServerIP;
   private InetAddress serverAddr;

   public PingSend(int PingAttempts1, String ServerIPChoice) {
      PingAttempts = PingAttempts1;
      ServerIP = ServerIPChoice;
   }

   public void run() {
      try {
         serverAddr = InetAddress.getByName(ServerIP);
      } catch (UnknownHostException e1) {
         // TODO Auto-generated catch block
         e1.printStackTrace();
      }

      while (p.getValue() == true) {
         if (snts.getValue() == true) {
            snts.setFalse();
            c.ZeroCount();
            StartTime = System.currentTimeMillis();
            try {
               DatagramSocket socket = new DatagramSocket();
               for (int j = 0; j < 4; j++) {
                  long StartTimeSegment = System.currentTimeMillis();
                  cts.setValue(j);
                  for (int i = 0; i < (PingAttempts / 4); i++) {
                     String PacketString = "" + j + "-" + i;
                     // Log.e("UDP_Ping ", "Packet Sent: " + PacketString);
                     byte[] buf = PacketString.getBytes();
                     DatagramPacket SendPacket = new DatagramPacket(buf,
                           buf.length, serverAddr, SERVERPORT);
                     socket.send(SendPacket);
                     st.AddElement(j, i, System.currentTimeMillis());
                     sleep(50);
                  }
                  sleep(StartTimeSegment + 2500 - System.currentTimeMillis());
               }
               socket.close();
            } catch (SocketException e) {
               // TODO Auto-generated catch block
               e.printStackTrace();
            } catch (IOException e) {
               // TODO Auto-generated catch block
               e.printStackTrace();
            } catch (InterruptedException e) {
               // TODO Auto-generated catch block
               e.printStackTrace();
            }

         }
      }
   }
}
```

Την εντολή να ξεκινήσει ένα νέο χρονικό διάστημα (snts: start next time segment) τη δίνει η storeLocation, αφότου έχει τελειώσει με τους υπολογισμούς στο προηγούμενο διάστημα.

Ορίζεται το τρέχον cts (current time segment).

Αφότου στείλει τα πακέτα για το παρόν time segment, περιμένει τον κατάλληλο χρόνο μέχρι να ξεκινήσει το επόμενο.

π.χ. $i < \frac{\text{PingAttempts}}{4} = \frac{16}{4} = 4$

j / i	0	1	2	3
0	0-0	0-1	0-2	0-3
1	1-0	1-1	1-2	1-3
2	2-0	2-1	2-2	2-3
3	3-0	3-1	3-2	3-3

Send Time Table (4 x 8)

i / j	0	1	2	3	4	5	6	7
0								
1			t_{12}					
2								
3								

```
class PingReceive extends Thread {
   private static final int CLIENTPORT = 5000;
   private String clientAddr;

   public void run() {
      clientAddr = getLocalIpAddress();

      try {
         DatagramSocket socket = new DatagramSocket(CLIENTPORT,
               InetAddress.getByName(clientAddr));
         socket.setSoTimeout(3000);
         while (p.getValue() == true) {
            byte[] buf = new byte[3];
            DatagramPacket ReceivePacket = new DatagramPacket(buf, buf.length);
            try {
               socket.receive(ReceivePacket);
               long ReceiveTime = System.currentTimeMillis();
               new Thread(new PingCalc(ReceivePacket, ReceiveTime)).start();
            } catch (IOException e) {
               // TODO Auto-generated catch block
               e.printStackTrace();
            }
         }
         socket.close();
      } catch (SocketException e) {
         // TODO Auto-generated catch block
         e.printStackTrace();
      } catch (UnknownHostException e) {
         // TODO Auto-generated catch block
         e.printStackTrace();
      }
   }
}

class PingCalc extends Thread {
   DatagramPacket ReceivePacket;
   long ReceiveTime;

   public PingCalc(DatagramPacket ReceivePacket1, long ReceiveTime1) {
      ReceivePacket = ReceivePacket1;
      ReceiveTime = ReceiveTime1;
   }

   public void run() {
      byte[] PacketStringData = ReceivePacket.getData();
      String PacketString = new String(PacketStringData);
      String[] PacketStringParts = PacketString.split("-");
      String PacketTimeSegment = PacketStringParts[0];
      String PacketID = PacketStringParts[1];

      int j = Integer.parseInt(PacketTimeSegment);
      int i = Integer.parseInt(PacketID);
      if (j == cts.getValue()) {
         long PacketStartTime = st.ReturnElement(j, i);
         long TotalPingTime = ReceiveTime - PacketStartTime;
         // < 2500 dioti mporei na trexei h PingCalc kai apo tin alli na
         // ypologizontai apotelesmata (storeLocation), opote na allaksei to c (count)
         // xwris na to theloume
         if (TotalPingTime < 2500) {
            r.AddElement(c.GetCount(), TotalPingTime);
            c.AddCount();
```

Σε αυτό το Thread γίνεται η υποδοχή των «απαντήσεων» του Server. Για κάθε ληφθείσα «απάντηση» ξεκινά ένα νέο Thread για την εκτέλεση των υπολογισμών.

Στην PingCalc «περνάμε» το ληφθέν πακέτο και τη χρονική στιγμή που αυτό έφθασε, για την εκτέλεση των υπολογισμών.

Έλεγχος για το αν η «απάντηση» που λάβαμε ανήκει στο τρέχον time segment

TotalPingTime = Latency = Χρόνος που ελήφθη η «απάντηση» - χρόνος που εστάλη το πακέτο (για κάθε προσπάθεια)

Results (msec)
145

```
         // Log.e( "UDP_Ping", "CALCULATE: " + PacketTimeSegment + "-" +PacketID + "
@Latency: " + TotalPingTime );
         }
      }
   }
}

class PingSendIndoors extends Thread {
   private static final int SERVERPORT = 5000;
   private int PingAttempts;
   private String ServerIP;
   private InetAddress serverAddr;
   private Location fixedLocation = new Location("fixedloc");

   public PingSendIndoors(int PingAttempts1, String ServerIPChoice) {
      PingAttempts = PingAttempts1;
      ServerIP = ServerIPChoice;
   }

   public void run() {
      try {
         serverAddr = InetAddress.getByName(ServerIP);
      } catch (UnknownHostException e1) {
         // TODO Auto-generated catch block
         e1.printStackTrace();
      }

      double lat = IndoorsGeopointSingleton.getInstance(0, 0).latitude();
      double lon = IndoorsGeopointSingleton.getInstance(0, 0).longitude();
      fixedLocation.setLatitude(lat);
      fixedLocation.setLongitude(lon);
      fixedLocation.removeAccuracy();
      fixedLocation.removeAltitude();
      fixedLocation.removeBearing();

      while (p.getValue() == true) {
         if (snts.getValue() == true) {
            snts.setFalse();
            c.ZeroCount();
            StartTime = System.currentTimeMillis();
            try {
               DatagramSocket socket = new DatagramSocket();
               for (int j = 0; j < 4; j++) {
                  long StartTimeSegment = System.currentTimeMillis();
                  cts.setValue(j);
                  for (int i = 0; i < (PingAttempts / 4); i++) {
                     String PacketString = "" + j + "-" + i;
                     // Log.e( "Packet Sent: ", PacketString );
                     byte[] buf = PacketString.getBytes();
                     DatagramPacket SendPacket = new DatagramPacket(buf,
                           buf.length, serverAddr, SERVERPORT);
                     socket.send(SendPacket);
                     st.AddElement(j, i, System.currentTimeMillis());
                     sleep(50);
                  }
                  sleep(StartTimeSegment + 2500 - System.currentTimeMillis());
               }
               socket.close();
            } catch (SocketException e) {
               // TODO Auto-generated catch block
               e.printStackTrace();
```

Παίρνουμε τις συντεταγμένες της Fixed Location από το Singleton IndoorsGeopointSingleton.

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

            if (p.getValue() == true) {
               if (mStartNextSegment) {
                  mStartNextSegment = false;
                  startNewSegment();
               }
               storeLocation(fixedLocation);
            }
         }
      }
   }
}
```

Εάν έχουμε διακόψει την καταγραφή, δεν θέλουμε να κληθεί η storeLocation και να αποθηκευτεί το ημιτελές τελευταίο κομμάτι, οπότε ελέγχουμε πρώτα αν έχει τερματιστεί.

Αφότου τελειώσει το χρονικό διάστημα των 10sec καλείται η storeLocation για να γίνουν οι υπολογισμοί και να αποθηκευτεί το καταγεγραμμένο τμήμα. Έπειτα, αν ο χρήστης δεν έχει διακόψει τη μέτρηση (p == true) αρχίζει ένα νέο χρονικό διάστημα των 10sec κ.ο.κ.

2.3 UDP Download και UDP Ping στην πλευρά του Server

2.3.1 Λειτουργία του UDP Download

2.3.1.1 Περιγραφή λειτουργίας

Εκτελώντας το αρχείο DLServer.java ο διαχειριστής καλείται να επιλέξει το χρονικό διάστημα (time interval) που θα μεσολαβεί από την αποστολή ενός πακέτου προς τον Client με το επόμενο. Αυτή η επιλογή καθορίζει το εξερχόμενο bit rate από τον Server. Στην οθόνη εμφανίζεται μια σειρά από συνηθισμένες επιλογές που αφορούν μέγεθος πακέτου 1400 B, ο διαχειριστής, ωστόσο, είναι ελεύθερος να επιλέξει όποια τιμή επιθυμεί. Επιλέγοντας «0», ο Server θα στέλνει πακέτα στους Clients όσο πιο γρήγορα μπορεί. Σε αυτή την περίπτωση δεν καθορίζεται κάποιο άνω όριο στο εξερχόμενο bit rate. Μετά από αυτή την επιλογή, ξεκινά ένα thread (DLReceive) το οποίο λαμβάνει πακέτα «εντολές» από τους Clients. Δημιουργείται, ακόμη, ένα διάνυσμα (DLConnectionsVector) στο οποίο καταγράφονται με δυναμικό τρόπο οι «συνδέσεις» με τους Clients.

Όταν ληφθεί κάποιο πακέτο, ξεκινά ένα νέο Thread (DLProcessRequest) το οποίο επεξεργάζεται την «εντολή» του Client. Εάν πρόκειται για πακέτο «σύνδεσης», ξεκινά ένα νέο thread (DLSendData) μέσω του οποίου στέλνονται πακέτα στον Client για μέτρηση του bit rate. Επιπλέον, στο διάνυσμα με τις «συνδέσεις» προστίθεται μία reference προς το thread αποστολής πακέτων (DLSendData) που χρησιμεύει στον τερματισμό της αποστολής, συνακόλουθα και του thread, όταν τερματιστεί η μέτρηση από τον Client. Συγκεκριμένα, όταν ο Server λάβει ένα «FIN» πακέτο αναζητά στο DLConnectionsVector τη reference προς το αντίστοιχο thread, με βάση την IP Address από την οποία εστάλη το «FIN» πακέτο (το πακέτο φέρει αυτή την πληροφορία), και τερματίζει την αποστολή πακέτων τερματίζοντας το thread (DLSendData). Κατόπιν στέλνει «ACK» πακέτα πίσω στον Client, για επιβεβαίωση πως η

«σύνδεση» τερματίστηκε. Αν του ζητηθεί να τερματίσει «σύνδεση» που δεν υπάρχει (μπορεί να την τερμάτισε μόλις προηγουμένως), διατρέχει το διάνυσμα χωρίς να βρει κάποια να τερματίσει και στέλνει «ACK» πακέτα πίσω στον Client.

2.3.1.2 Σημειώσεις επί του κώδικα

```
public class DLServer{

   public static void main (String args[]) {
      System.out.println( "Please insert Time Interval (msec)" + "\n" +
            "Common Choices:" + "\n" +
            "200 ... Output Bitrate: 56    kbps (1400B) " + "\n" +
            "100 ... Output Bitrate: 112   kbps (1400B) " + "\n" +
            "50  ... Output Bitrate: 224   kbps (1400B) " + "\n" +
            "20  ... Output Bitrate: 560   kbps (1400B) " + "\n" +
            "10  ... Output Bitrate: 1120  kbps (1400B) " + "\n" +
            "5   ... Output Bitrate: 2240  kbps (1400B) " + "\n" +
            "2   ... Output Bitrate: 5600  kbps (1400B) " + "\n" +
            "1   ... Output Bitrate: 11200 kbps (1400B) " + "\n" +
            "0   ... Output Bitrate: Maximum Bitrate (send as fast as you can) " );
      BufferedReader bufferedreader = new BufferedReader( new InputStreamReader( System.in
) );

      try {
         String TimeInterval = bufferedreader.readLine();
         int TimeInterval_msec = Integer.parseInt( TimeInterval );

         new Thread( new DLReceive( TimeInterval_msec ) ).start();
         System.out.println( "\n" + "Starting UDP Download Service..." );
      }
      catch (IOException e) {
         System.out.println( "An error occurred:" + e.toString() );
      }
   }
}

class DLReceive extends Thread{
   private static final int SERVERPORT = 5001;
   private int TimeInterval_msec;

   public DLReceive( int TimeInterval_msec1 ){
      TimeInterval_msec = TimeInterval_msec1;
   }
   public void run() {
      try {
        // A new Vector recording our connections ( reason -> access from multiple threads
)
        DLConnectionsVector CV = new DLConnectionsVector();

        InetAddress serverAddr = InetAddress.getLocalHost();
        System.out.println( "SERVER IP: " + serverAddr + "\n" +
              "Time Interval: " + TimeInterval_msec + " msec" + "\n" +
              "-----------------------------------------" );
          DatagramSocket socket = new DatagramSocket( SERVERPORT, serverAddr );

          while ( true ){
              byte[] buf = new byte[4];
              DatagramPacket TodoPacket = new DatagramPacket( buf, buf.length );
              //Receiving packets
```

Παράδειγμα:
Time Interval = 5 msec
$Output\ Bitrate = \frac{1\ \text{packet}}{5\ \text{msec}} =$
$= \frac{1400\text{B}}{5\ \text{msec}} = \frac{1400*8\ \text{bits}}{5*10^{-3}\text{sec}} = \frac{2240000\ \text{kbps}}{1000} =$
= 2240 kbps

Ο Server λαμβάνει τα πακέτα «εντολές» των Clients και ξεκινά ένα DLProcessRequest thread για κάθε μία από αυτές.

```
              socket.receive( TodoPacket );
              new Thread( new DLProcessRequest( TodoPacket, CV, TimeInterval_msec )
).start();
           }
      }
      catch (SocketException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
      } catch (UnknownHostException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
     } catch (IOException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
     }
   }
}

class DLProcessRequest extends Thread{
   private static final int SERVERPORT = 5001;
   private static final int CLIENTPORT = 5001;
   private static final int CLIENTPORTSTOP = 5002;
   private DatagramPacket TodoPacket;
   private DLConnectionsVector CV;
   private int TimeInterval_msec = 0;
   private int TimeInterval_nsec = 0;
   private int PacketSize;

   public DLProcessRequest( DatagramPacket TodoPacket1, DLConnectionsVector CV1, int
TimeInterval_msec1 ){
      TodoPacket = TodoPacket1;
      CV = CV1;
      TimeInterval_msec = TimeInterval_msec1;
   }

   public void run(){
      //Getting packet size info and setting TimeInterval
          byte[] data = TodoPacket.getData();
          String PacketSizeString = new String( data );

          InetAddress clientAddr = TodoPacket.getAddress();

        if ( !PacketSizeString.equals( "FIN " ) ){
            if ( PacketSizeString.equals( "200 " ) ) {
              PacketSize = 200 ;
             }
             else if ( PacketSizeString.equals( "500 " ) ) {

              PacketSize = 500 ;
             }
             else if ( PacketSizeString.equals( "1000" ) ) {
              PacketSize = 1000;
             }
             else if ( PacketSizeString.equals( "1400" ) ) {
              PacketSize = 1400;
             }

            DLSendData sd = new DLSendData( clientAddr, PacketSize, TimeInterval_msec,
TimeInterval_nsec );
            synchronized ( CV ){
                CV.Add( sd );
                CV.LastElement().start(); // sd.start();
```

Σε άλλη θύρα (port) στέλνονται τα πακέτα προς μέτρηση του bit rate και σε άλλη τα «ACK» πακέτα για επιβεβαίωση τερματισμού της «σύνδεσης».

Για ακόμη μεγαλύτερη ακρίβεια έχει προβλεφθεί και μεταβλητή για nanoseconds στο Time Interval, ωστόσο δεν την χρησιμοποιούμε.

Ο Server λαμβάνει «εντολή» για έναρξη αποστολής πακέτων προς τον Client. Ξεκινά ένα νέο thread τροφοδοτώντας το με τις πληροφορίες:
- Client Address
- PacketSize
- TimeInterval_msec
- TimeInterval_nsec

DLConnectionsVector
sd (195.138.32.64, 1000, 5, 0)
sd (195.125.109.3, 1400, 5, 0)
sd (85.167.112.72, 1000, 5, 0)

```
                System.out.println( "Packet received from Client: " + PacketSizeString + "
with Client Address: " + clientAddr.toString() + "\n" +
                    "Packet size: " + PacketSize + "\n" +
                    "**Connected clients: " + CV.Size() + "\n" +
                    "----------------------------------------" );
            }
         }
         else {
          synchronized ( CV ){
            if ( !CV.IsEmpty() ) {
             for ( int i=0; i<CV.Size(); i++ ){
                DLSendData temporary = CV.getElement( i );
                //Finding which connection to stop, using client's IP Address
                if ( temporary.clientAddr.equals( clientAddr ) ) {
                   temporary.StopSending();
                   CV.Remove( i );
                   System.out.println( "Connection with " + temporary.clientAddr.toString()
+ " is terminated!" + "\n" +
                         "**Connected clients: " + CV.Size() + "\n" +
                         "----------------------------------------" );
                }
             }
            }
             //System.out.println( "**Connected clients: " + CV.Size() );
          }

          //Send ACK to Client. Either there was no connection to stop either connection
has been terminated
          try {
             DatagramSocket socket = new DatagramSocket();
             byte[] buf = ( "ACK" ).getBytes();
             DatagramPacket ACKpacket = new DatagramPacket( buf, buf.length, clientAddr,
CLIENTPORTSTOP );
             for ( int i=0; i<3; i++ ){
                    socket.send( ACKpacket );
             }
                socket.close();
          } catch (SocketException e) {
             // TODO Auto-generated catch block
             e.printStackTrace();
          } catch (IOException e) {
             // TODO Auto-generated catch block
             e.printStackTrace();
          }
         }
   }
}

class DLSendData extends Thread{
   private static final int SERVERPORT = 5001;
   private static final int CLIENTPORT = 5001;
   public InetAddress clientAddr;
   private int PacketSize;
   private Boolean SendBytes = true;
   private int TimeInterval_msec;
   private int TimeInterval_nsec;

   public DLSendData( InetAddress clientAddr1, int PacketSize1, int TimeInterval_msec1,
int TimeInterval_nsec1 ){
     clientAddr = clientAddr1;
     PacketSize = PacketSize1;
     TimeInterval_msec = TimeInterval_msec1;
```

Το διάνυσμα είναι κοινό οπότε πρέπει να διασφαλίσουμε πως δεν θα το τροποποιούν την ίδια στιγμή διαφορετικά DLProcessRequest threads (synchronized).

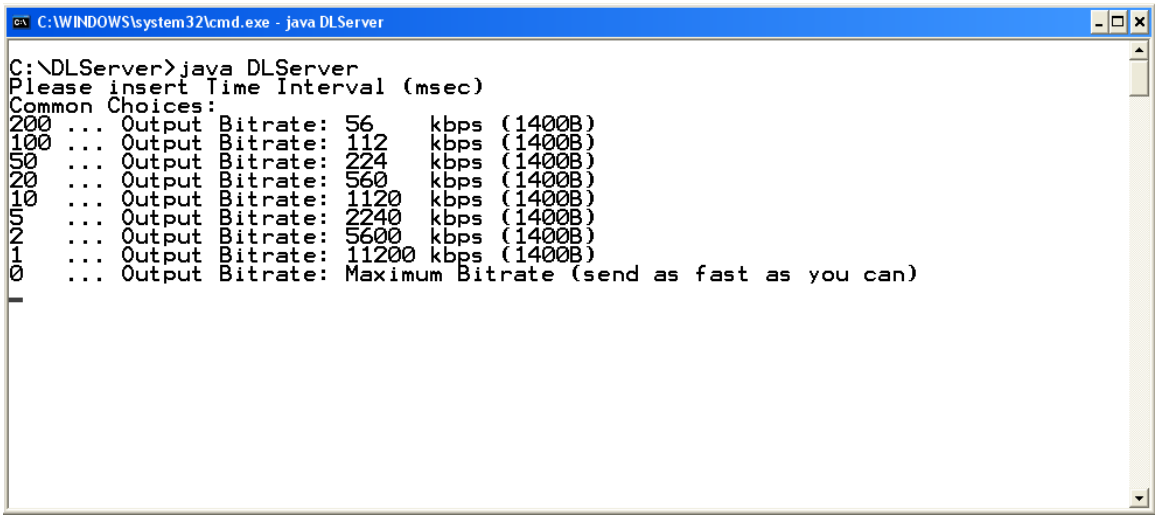
Σταματάει να στέλνει στον Client και αφαιρεί από το διάνυσμα την εν λόγω «σύνδεση».

Στέλνονται «ACK» πακέτα πίσω στον Client για επιβεβαίωση πως η εντολή του ελήφθη και η «σύνδεση» διεκόπη.

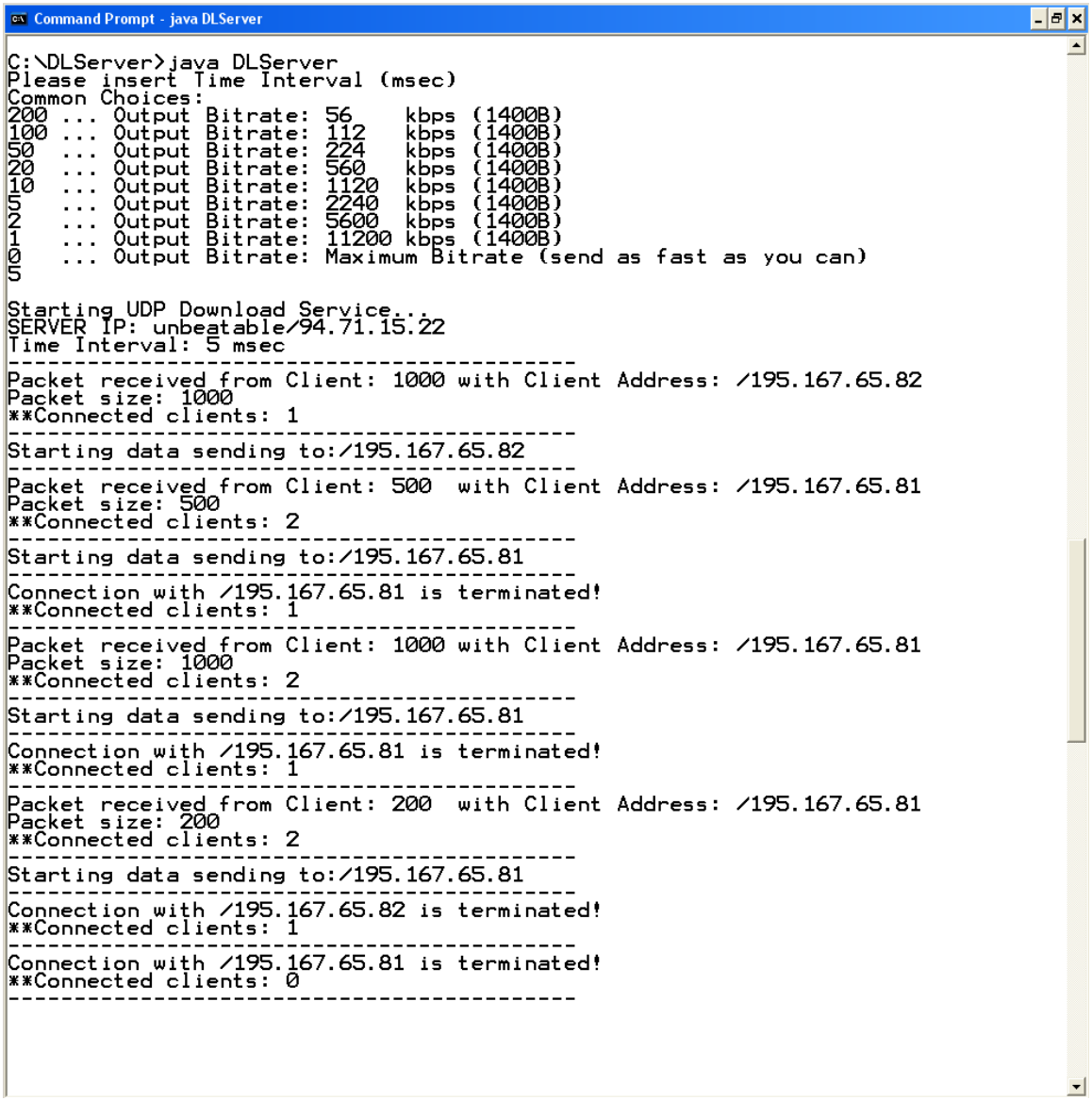
```
     TimeInterval_nsec = TimeInterval_nsec1;
   }

   public void run() {

         try {
         DatagramSocket socket = new DatagramSocket();
         System.out.println( "Starting data sending to:" + clientAddr.toString() + "\n" +
               "-----------------------------------------" );
         //System.out.println( "TimeInterval: " + TimeInterval_msec + "." +
TimeInterval_nsec );

         byte[] buf1 = new byte[PacketSize];
         DatagramPacket packet = new DatagramPacket( buf1, buf1.length, clientAddr,
CLIENTPORT );
         while ( SendBytes == true ){
               socket.send( packet );
               Thread.sleep( TimeInterval_msec, TimeInterval_nsec );
               //System.out.println("Packet Sent.");
         }
      } catch (SocketException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
        } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
         } catch (InterruptedException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
        }
   }

    public void StopSending(){
      SendBytes = false;
    }
}
```

Ρυθμίζεται το εξερχόμενο bit rate ανάλογα με την επιλογή του διαχειριστή (Time Interval).

2.3.1.3 Παράδειγμα (Screenshots)



Ο διαχειριστής επιλέγει το επιθυμητό εξερχόμενο bit rate προς τους Clients. Εκτός από τις παραπάνω τιμές μπορεί να εισαχθεί οποιοδήποτε έγκυρο Time Interval που εξυπηρετεί τις ανάγκες των μετρήσεων. Συνηθίζεται η επιλογή «0» που μας δίνει το μέγιστο εξερχόμενο bit rate, διότι ενδιαφερόμαστε για τις δυνατότητες του δικτύου ως προς το bit rate, οπότε δεν θέλουμε να υπάρχουν περιορισμοί ταχύτητας στην πλευρά του Server.



Σε αυτό το παράδειγμα επιλέξαμε Time Interval = 5 msec που ισοδυναμεί με εξερχόμενο bit rate = 2,24 Mbps. Η υπηρεσία του UDP Download ξεκίνησε και η IP Address του Server εκτυπώθηκε στην οθόνη (94.71.15.22). Ο Server έλαβε ένα πακέτο από τον Client «195.167.65.82» με την εντολή να ξεκινήσει την αποστολή πακέτων μεγέθους 1000 B. Κατόπιν, ξεκίνησε η αποστολή και ενημερωθήκαμε για τον αριθμό των συνδεδεμένων Clients. Κάποια επόμενη στιγμή, ένας ακόμη Client, ο «195.167.65.81», ζήτησε από τον Server να ξεκινήσει την

αποστολή πακέτων μεγέθους 500 B προς αυτόν. Οι δύο Clients λάμβαναν πακέτα από τον Server για τις μετρήσεις τους μέχρι τη στιγμή που ένας αποχώρησε. Ειδοποίησε το Server, ο οποίος ανταποκρίθηκε τερματίζοντας τη σύνδεση και εκτυπώνοντας «Connection with /195.167.65.81 is terminated!». Ο αριθμός των Clients μειώθηκε κατά έναν. Ο αποχωρήσας Client επανήλθε ζητώντας αποστολή πακέτων διαφορετικού μεγέθους (1000 B). Αποχώρησε ξανά και επανήλθε για τελευταία φορά ζητώντας αποστολή πακέτων μεγέθους 200 B. Σε όλο αυτό το χρονικό διάστημα ο έτερος Client παρέμενε «συνδεδεμένος». Τελικά, αποχώρησαν και οι δύο, με πρώτο τον «195.167.65.82» και τον «195.167.65.81» να έπεται.

2.3.2 Λειτουργία του UDP Ping

2.3.2.1 Περιγραφή λειτουργίας

Εκτελώντας το αρχείο PingServer.java ο Server είναι έτοιμος κάθε στιγμή να ανταποκριθεί σε εισερχόμενο πακέτο από τον Client, στέλνοντας σε αυτόν ένα ίδιο πακέτο. Ένα thread (PingReceive) λαμβάνει τα εισερχόμενα πακέτα και για κάθε ένα ξεκινά ένα νέο thread (PingReply) «περνώντας» του το ληφθέν πακέτο. Εκεί, το πακέτο προωθείται πίσω στην IP Address του αποστολέα. Ο λόγος που υλοποιήθηκαν δύο ξεχωριστά threads είναι πως με αυτό τον τρόπο το thread που «ακούει» για εισερχόμενα πακέτα δεν χρονοτριβεί με το να προωθεί το πακέτο πίσω στον Client, οπότε δεν χάνονται πακέτα που έρχονται με πολύ μικρή χρονική διαφορά. Βέβαια, αυτό οδηγεί σε αύξηση του χρόνου προώθησης, καθώς δημιουργείται ένα καινούριο thread, αλλά η χρονική καθυστέρηση θεωρείται αμελητέα. Για τον έλεγχο της αντιστοιχίας σταλθέντων πακέτων και «απαντήσεων», εκτυπώνεται στον Server το όνομα του πακέτου που προωθείται πίσω στον Client. Συνοπτικά, ο PingServer δεν παράγει παρά την «ηχώ» του πακέτου που έστειλε ο Client και η διαφορά εκπομπής και λήψης του πακέτου στην πλευρά του Client λογίζεται ως το μετρούμενο latency.

2.3.2.2 Σημειώσεις επί του κώδικα

```
public class PingServer {

   public static void main(String args[]) {
      new Thread(new PingReceive()).start();
      System.out.println("Starting UDP Ping Service...");
   }
}

class PingReceive extends Thread {
   public static final int SERVERPORT = 5000;
   public static final int CLIENTPORT = 5000;
   DatagramSocket socket;

   public void run() {
      try {
         InetAddress serverAddr = InetAddress.getLocalHost();
         System.out.println("SERVER IP: " + serverAddr + "\n"
               + "------------------------------------------");
```

```
            socket = new DatagramSocket(SERVERPORT, serverAddr);
            while (true) {
               byte[] buf = new byte[3];
               DatagramPacket IncomingPacket = new DatagramPacket(buf,
                     buf.length);
               // System.out.println("Waiting for Incoming Packet...");
               socket.receive(IncomingPacket);
               new Thread(new PingReply(IncomingPacket)).start();
            }
         } catch (UnknownHostException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
            System.out.println("ERROR: Couldn't get localhost address");
         } catch (SocketException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
            socket.close();
         } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
            socket.close();
         }

      }
   }

   class PingReply extends Thread {
      public static final int CLIENTPORT = 5000;
      DatagramPacket IncomingPacket;

      public PingReply(DatagramPacket IncomingPacket1) {
         IncomingPacket = IncomingPacket1;
      }

      public void run() {
         // Sending reply
         InetAddress clientAddr = IncomingPacket.getAddress();
         // System.out.println("CLIENT IP: " + clientAddr);
         try {
            DatagramSocket socket = new DatagramSocket();
            byte[] data = IncomingPacket.getData();
            String PacketString = new String(data);

            byte[] buf1 = PacketString.getBytes();
            DatagramPacket packet = new DatagramPacket(buf1, buf1.length,
                  clientAddr, CLIENTPORT);
            socket.send(packet);
            socket.close();
            System.out.println("Reply Sent: " + PacketString
                  + " to Client with IP: " + clientAddr.toString());
         } catch (SocketException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
         } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
         }
      }
   }
```

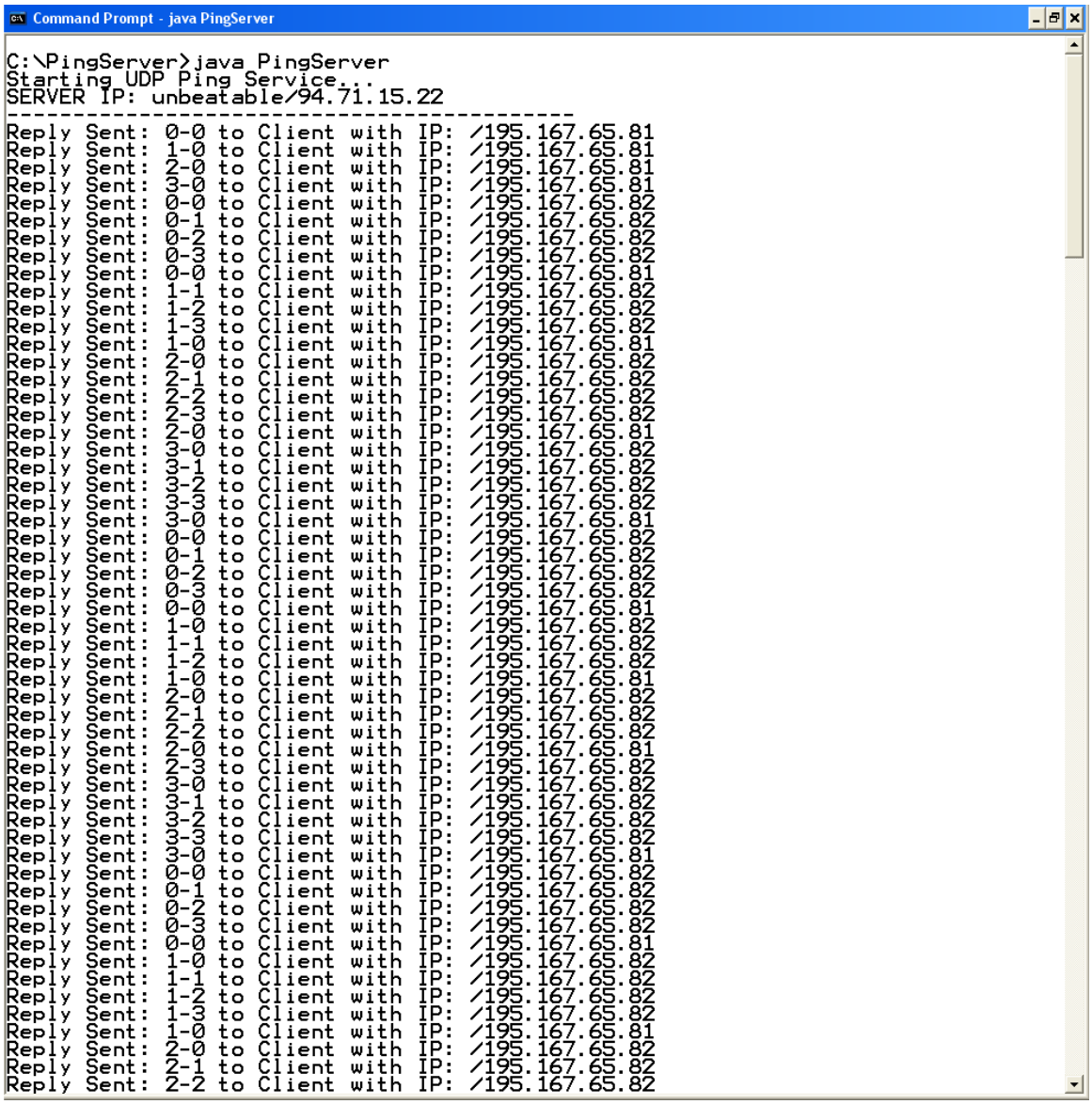
Ο Server λαμβάνει πακέτα από τους Clients και για κάθε ένα εξ αυτών ξεκινά ένα νέο thread (PingReply).

Από το ληφθέν πακέτο αποσπάται η πληροφορία για την IP Address του αποστολέα, στην οποία και θα σταλεί το πακέτο «απάντηση».

Το πακέτο «απάντηση» αποστέλλεται στην IP Address του Client και τερματίζεται το Thread.

2.3.2.3 Παράδειγμα (Screenshots)

Σε αυτό το παράδειγμα δύο Clients στέλνουν στον Server πακέτα για τα οποία περιμένουν την απάντηση, ώστε να υπολογίσουν το latency. Ο πελάτης «195.167.65.81» έχει επιλέξει να εκτελεί 4 προσπάθειες (ping attempts) στο διάστημα των 10 sec, ενώ ο «195.167.65.82» έχει επιλέξει 16 προσπάθειες. Αυτό γίνεται εμμέσως αντιληπτό από το γεγονός πως ο Server στον μεν «195.167.65.81» επιστρέφει τα πακέτα «0-0, 1-0, 2-0, 3-0», ενώ στον «195.167.65.82» επιστρέφει τα «0-0, 0-1, 0-2, 0-3, 1-0, 1-1, 1-2, 1-3, 2-0, 2-1, 2-2, 2-3, 3-0, 3-1, 3-2, 3-3». Για κάθε πακέτο «απάντηση» που στέλνει, εκτυπώνει στην οθόνη το όνομα του πακέτου και τη διεύθυνση προορισμό.



2.4 Σύστημα καταγραφής των μετρήσεων

Μέσω των λειτουργιών του UDP Download και του UDP Ping λαμβάνουμε μετρήσεις, τις οποίες χρησιμοποιούμε για δύο σκοπούς:

- Για την απεικόνιση στο χάρτη της κινητής συσκευής των «ταχυτήτων» (bit rate, latency) που μετρήσαμε στα διάφορα τμήματα και την εμφάνιση στατιστικών στοιχείων

- Για την αποθήκευση των στοιχείων της διαδρομής σε μια database σε έναν κεντρικό Server, όπου και συλλέγονται όλες οι καταγραφές από τους διάφορους Clients για περαιτέρω ανάλυση (queries, απεικόνιση κλπ)

Σε επίπεδο εφαρμογής στην κινητή συσκευή (KampRate), οφείλουμε να μελετήσουμε τις εξής περιπτώσεις:

- **Download (Outdoors)**: Η κινητή συσκευή ζητάει από τον GPS Provider ενημέρωση της θέσης της, όσο πιο γρήγορα γίνεται. Όταν λάβει ενημέρωση και παράλληλα έχει ξεπεραστεί το χρονικό διάστημα των 10 sec, καλείται η μέθοδος storeLocation(location) όπου γίνονται οι απαραίτητοι υπολογισμοί (υπολογισμός bit rate) και μηδενισμοί (μηδενίζεται ο μετρητής πακέτων) και ξεκινά η διαδικασία αποθήκευσης του διαστήματος (χρονικού και χωρικού) που μόλις μετρήθηκε, όπως επίσης αρχίζει ένα νέο διάστημα. Η νέα εγγραφή που γίνεται στη database της κινητής συσκευής χρησιμοποιείται ώστε στο χάρτη να σχεδιαστεί, με τον αντίστοιχο χρωματισμό, μία γραμμή που αντιστοιχεί στο χωρικό κομμάτι της μέτρησης και για να ανανεωθούν τα στατιστικά στοιχεία.

- **Download (Indoors)**: Επειδή βρισκόμαστε σε εσωτερικό χώρο, δεν μπορεί να χρησιμοποιηθεί το GPS για τον εντοπισμό της θέσης, οπότε χρησιμοποιείται ο Network Provider. Αφότου ληφθεί ενημέρωση για τη θέση, ο χρήστης επιλέγει την ακριβή του τοποθεσία στο χάρτη (υλοποιημένο στην IndoorsSelectLocation) και η καταγραφή ξεκινά. Στην περίπτωση αυτή έχουμε ως σταθερή θέση (fixed location) αυτή που επέλεξε ο χρήστης manually ή τη θέση που βρήκε ο Network Provider, αν ικανοποιούσε το χρήστη και δεν την άλλαξε. Ένα καινούριο thread (DLIndoorsTimer) αποτελεί το «ρολόι» για αυτή την καταγραφή. Κάθε 10sec καλεί τη storeLocation(location <- fixed location) για να γίνουν οι υπολογισμοί και να αποθηκευτεί το καταγεγραμμένο χρονικό διάστημα, όπως επίσης αρχίζει ένα νέο διάστημα, έχοντας μηδενιστεί ο μετρητής των ληφθέντων πακέτων. Όπως είναι αναμενόμενο, στο χάρτη δεν σχεδιάζονται χρωματισμένα τμήματα, παρά ο χρήστης απλά βλέπει τη θέση του σε αυτόν.

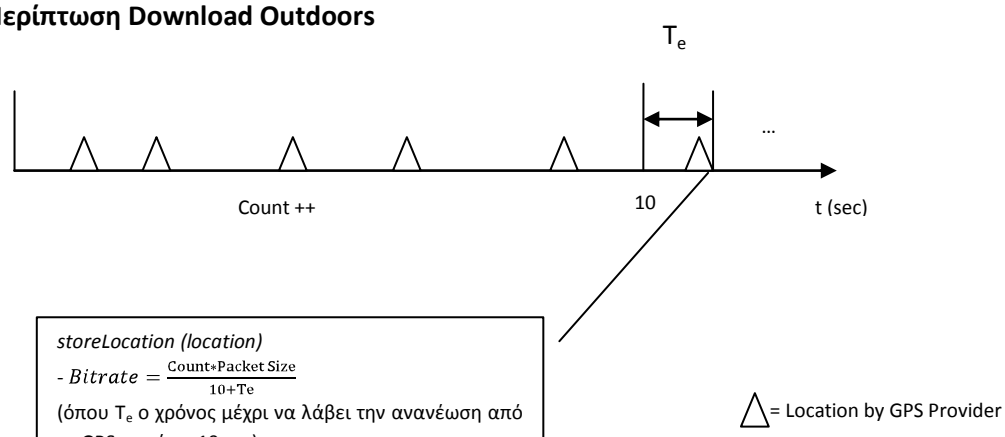
- **Ping (Outdoors)**: Το Ping στην περίπτωση που καταγράφουμε μια διαδρομή (Outdoors) λειτουργεί όπως και το Download. Όταν ληφθεί ενημέρωση θέσης από τον GPS Provider και παράλληλα έχει ξεπεραστεί το χρονικό διάστημα των 10 sec καλείται η

storeLocation(location) και γίνονται οι απαραίτητοι υπολογισμοί (AvgLatency, MaxLatency, MinLatency). Αποθηκεύονται οι πληροφορίες για το συγκεκριμένο χρονικό και χωρικό διάστημα και αρχίζει το επόμενο χρονικό κομμάτι των 10 sec. Στο χάρτη κάθε τμήμα χρωματίζεται ανάλογα με το μέσο latency που μετρήθηκε και ανανεώνονται τα στατιστικά στοιχεία.

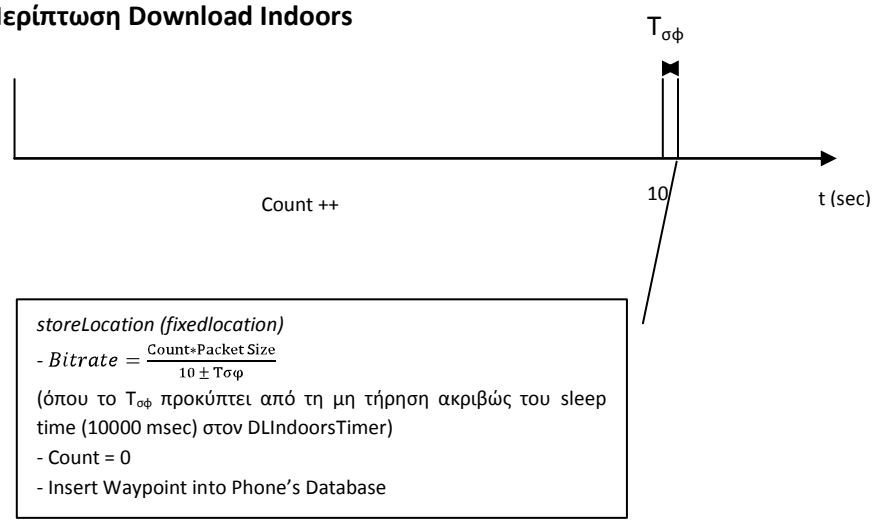
- **Ping (Indoors)**: Όπως και στο Download Indoors, επιλέγεται πρώτα η σταθερή θέση (fixed location) του χρήστη. Στο τέλος των 10 sec γίνονται οι υπολογισμοί και ξεκινά ένα νέο χρονικό διάστημα των 10 sec (PingSendIndoors). Στο χάρτη ο χρήστης βλέπει τη θέση του.

Οι ιδιαιτερότητες που παρουσιάζουν οι παραπάνω περιπτώσεις φαίνονται και στα ακόλουθα σχήματα:

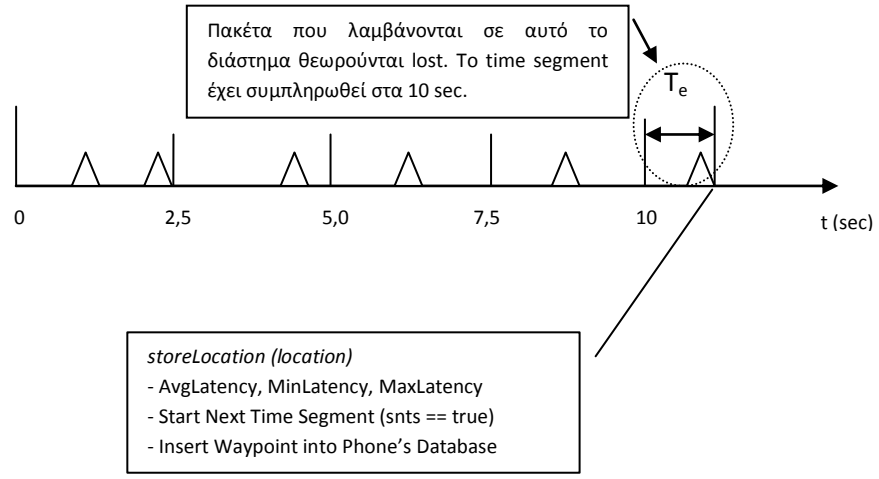
Περίπτωση Download Outdoors



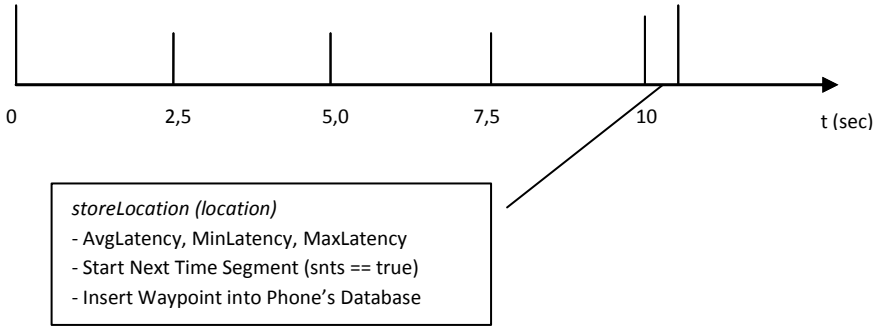
Περίπτωση Download Indoors



Περίπτωση Ping Outdoors



Περίπτωση Ping Indoors



2.5 Σύστημα αποθήκευσης των μετρήσεων από τον Client στον Server

2.5.1 Περιγραφή λειτουργίας

Ο Client μέσω της εφαρμογής KampRate που εκτελείται στην κινητή συσκευή, λαμβάνει μετρήσεις επί διαδρομών (Outdoors) ή επί σημείων (Indoors). Έχει τη δυνατότητα να βλέπει τις διαδρομές σχεδιασμένες στο χάρτη, με συνημμένη την πληροφορία για την ταχύτητα της μέτρησης μέσω διαφορετικών χρωματισμών, όπως επίσης και στατιστικά στοιχεία με διαγράμματα και άλλες πληροφορίες. Σκοπός, όμως, του ολοκληρώμενου αυτού συστήματος, είναι αφενός ο Client να έχει εικόνα των μετρήσεων που έχει λάβει, αφετέρου να υπάρξει συγκεντρωτική απεικόνιση και χαρτογράφηση της ποιότητας του δικτύου. Για το τελευταίο απαιτείται η συλλογή των μετρήσεων από όλες τις καταγραφές που εκτέλεσαν οι διάφοροι Clients. Ο Client, λοιπόν, πρέπει να έχει τη δυνατότητα να αποστείλει τις μετρήσεις του σε έναν κεντρικό Server και αντίστοιχα ο Server πρέπει να είναι σε θέση να τις υποδεχθεί και να τις

διαχειριστεί. Στην κατεύθυνση αυτή σχεδιάστηκε μια δομή από τον Client στον Server που υλοποιεί τα παραπάνω με ταχύτητα και αξιοπιστία.

Η εφαρμογή KampRate στον Client έχει μια εσωτερική βάση δεδομένων, στην οποία καταγράφονται οι πληροφορίες για τα χωρικά-χρονικά τμήματα των διαδρομών στην περίπτωση εξωτερικής μέτρησης, όπως επίσης και για τα χρονικά τμήματα για τις στατικές εσωτερικές μετρήσεις. Από αυτή τη βάση δεδομένων, αντλούν πληροφορίες πολλές λειτουργίες της εφαρμογής για δυναμική (live) απεικόνιση των διαδρομών, αλλά και για την εκτέλεση των απαιτούμενων υπολογισμών. Παράλληλα, οι εγγραφές της χρησιμοποιούνται για την εξαγωγή xml αρχείων τα οποίο έχουν μια συγκεκριμένη δομή για τις ξεχωριστές περιπτώσεις των μετρήσεων Download και των μετρήσεων Ping. Τα xml αρχεία αποθηκεύονται στην SD Card της κινητής συσκευής και επιπλέον παρέχεται η δυνατότητα - επιλογή της αποστολής τους (upload) στον κεντρικό Server. Ο Server υποδέχεται τα xml αρχεία («τρέχοντας» το script handle_upload.php) και τα συγκεντρώνει σε έναν φάκελο ονόματι «Temp». Ο τρόπος που γίνεται αυτή η διαδικασία θα αναλυθεί σε επόμενη παράγραφο.

Ο διαχειριστής στον Server δύναται να εισάγει τις πληροφορίες που φέρουν τα ληφθέντα xml αρχεία στη βάση δεδομένων του Server. Αυτό επιτυγχάνεται εύκολα μέσω ενός PHP Script που έχουμε φτιάξει (update.php). Παράλληλα με την εισαγωγή τους στη βάση δεδομένων του Server, τα xml αρχεία μεταφέρονται στους τελικούς τους φακέλους («DL» και «Ping»), ανάλογα με το αν πρόκειται για μέτρηση Download ή Ping. Μέσω του phpMyAdmin που μας παρέχει ένα εύχρηστο γραφικό περιβάλλον (GUI), ο διαχειριστής μπορεί να βρει εγγραφές που τον ενδιαφέρουν (μέσω queries), να κάνει διάφορες επεξεργασίες ή να εξάγει τη βάση με τη δομή που επιθυμεί. Παραδείγματος χάρην, η δυνατότητα εξαγωγής του συνόλου (ή μέρους) της βάσης σε xml δομή είναι πολύ σημαντική για την απεικόνιση πολλαπλών διαδρομών–καταγραφών στο χάρτη στην πλευρά του Server με χρήση του εργαλείου NetPerfTool.

Συνοψίζοντας, οι διάφοροι Clients μπορούν να αποστέλουν τις καταγεγραμμένες μετρήσεις τους στον Server, όπου εισάγονται σε μια κεντρική βάση δεδομένων για περαιτέρω ανάλυση. Η άνωθεν περιγραφείσα δομή φαίνεται σχηματικά παρακάτω.

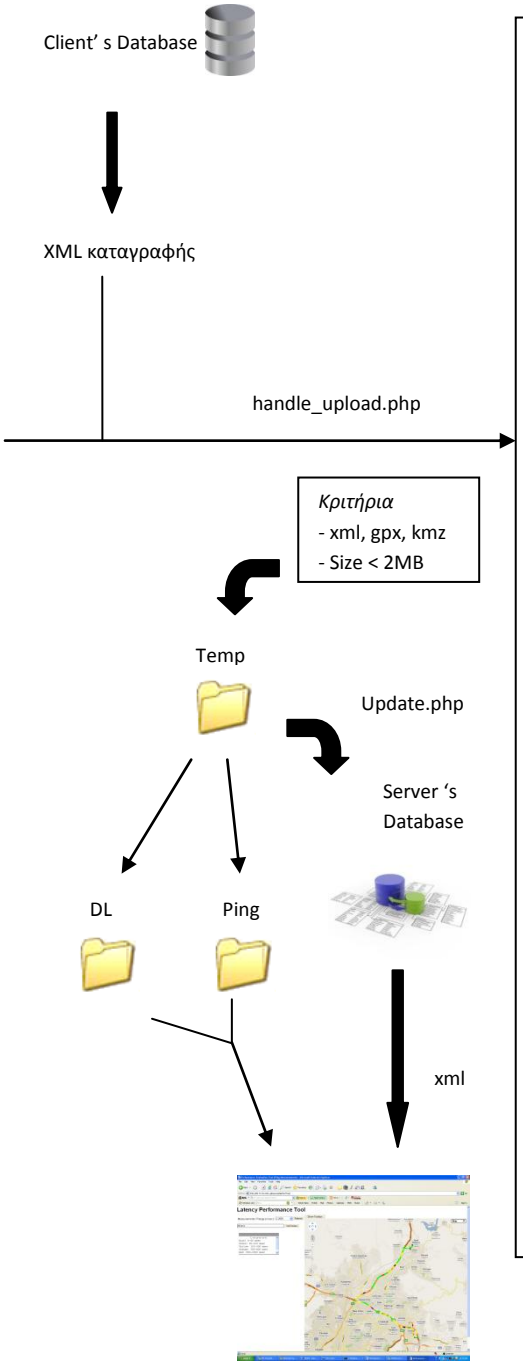
Client (KampRate Application)

Server

1. Επιλέγει την τρέχουσα καταγραφή (διαδρομή στην περίπτωση outdoors) ή κάποια παλαιότερη.

2. Η καταγραφή αποθηκεύεται με μια συγκεκριμένη xml δομή στην SD Card της κινητής συσκευής (Store to SD Card).

3. Ο χρήστης επιλέγει να «ανεβάσει» την καταγραφή στον Server, στην IP Address που θα ορίσει (Upload to Server).



4. Ο Server υποδέχεται την καταγραφή και αν πληροί τα κριτήρια ελέγχου τοποθετείται στον φάκελο «Temp».

5. Οι καταγραφές εισάγονται στην database του Server και μετακινούνται από τον προσωρινό φάκελο «Temp» στους φακέλους «DL» και «Ping».

6. Ο διαχειριστής κάνει αναζητήσεις στην ανανεωμένη βάση δεδομένων και την εξάγει σε μορφή xml.

7. Το νέο αρχείο χρησιμοποιείται για την απεικόνιση πολλαπλών διαδρομών στην πλευρά του Server με το εργαλείο NetPrefTool

2.5.2 Client

2.5.2.1 Η βάση δεδομένων του Client

Η εφαρμογή KampRate αξιοποιεί τη δυνατότητα που παρέχει το Android για δημιουργία εσωτερικής βάσης δεδομένων και συγκεκριμένα την κλάση SQLiteDatabase η οποία περιλαμβάνει μεθόδους για τη δημιουργία, τη διαγραφή, την εκτέλεση SQL εντολών, καθώς και άλλων συνηθισμένων εργασιών διαχείρισης βάσεων δεδομένων. Οι κυριότεροι πίνακες στη βάση δεδομένων είναι:

Table: *tracks*

Field	Type
_id	INTEGER PRIMARY KEY AUTOINCREMENT
name	TEXT
creationtime	INTEGER NOT NULL
networkingmeasurementchoice	TEXT
serverip	TEXT
packetsize	INTEGER
pingattempts	INTEGER
area	TEXT

- _id : Αύξων αριθμός που χρησιμοποιείται ως ταυτότητα για κάθε track. Διαδικασίες, όπως η προσπέλαση, η εγγραφή και η διαγραφή, χρησιμοποιούν αυτό τον αριθμό.

- name : Η ονομασία που έχει δώσει ο χρήστης στην καταγραφή. Προτείνεται η default που προβλέπεται, ώστε να διασφαλιστεί αργότερα η απρόσκοπτη εισαγωγή στη database του Server.

- creationtime : Η χρονική στιγμή (σε μορφή ημερομηνίας) που ξεκίνησε η συγκεκριμένη καταγραφή

- networkingmeasurementchoice : Το είδος της καταγραφής που πραγματοποιήθηκε (Download ή Ping)

- serverip : Η διεύθυνση του Server που χρησιμοποιήθηκε για την καταγραφή

- packetsize : Το μέγεθος του πακέτου που επιλέχθηκε (αφορά την περίπτωση Download)

- pingattempts : Ο αριθμός των προσπαθειών που θα επιχειρηθούν (αφορά την περίπτωση ping)

- area : Η πληροφορία για το αν πραγματοποιήθηκε εξωτερική (Outdoors) ή εσωτερική (Indoors) μέτρηση

Ο πίνακας *tracks*, λοιπόν, εμπεριέχει όλες εκείνες τις πληροφορίες που αφορούν τις ρυθμίσεις και το είδος μιας καταγραφής. Οι πληροφορίες αυτές παραμένουν αμετάβλητες κατά την πραγματοποίηση της καταγραφής.

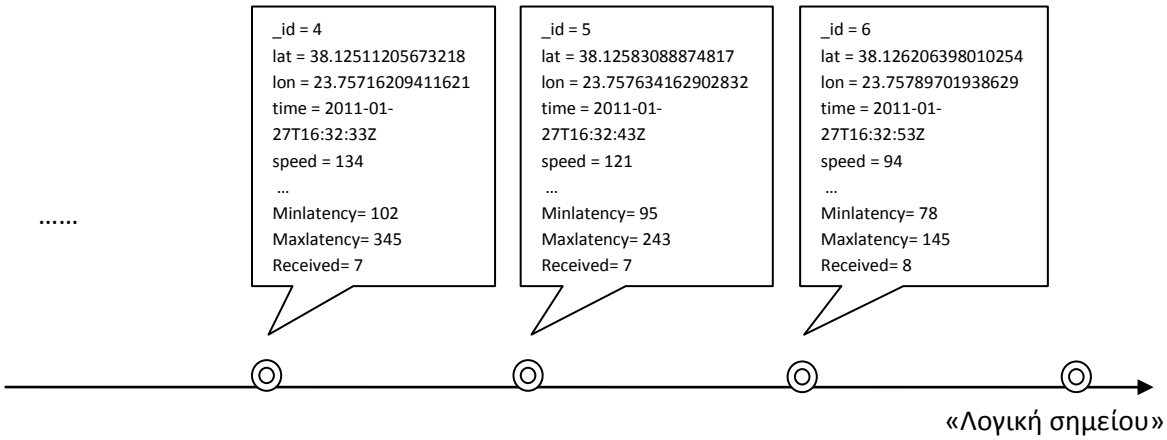
Table: *waypoints*

Field	**Type**
_id	INTEGER PRIMARY KEY AUTOINCREMENT
latitude	REAL NOT NULL
longitude	REAL NOT NULL
time	INTEGER NOT NULL
speed	speed
tracksegment	INTEGER NOT NULL
accuracy	REAL
altitude	REAL
bearing	REAL
minlatency	INTEGER
maxlatency	INTEGER
received	INTEGER
lost	INTEGER
cellid	INTEGER
lac	INTEGER
networktype	INTEGER

- _id : αύξων αριθμός που αποτελεί την ταυτότητα της κάθε σειράς (row)
- latitude : γεωγραφικό πλάτος της θέσης που αποθηκεύτηκε
- longitude : γεωγραφικό μήκος της θέσης που αποθηκεύτηκε
- time : η χρονική στιγμή του τέλους του διαστήματος της συγκεκριμένης μέτρησης
- speed : μεταβλητή «μπαλαντέρ» που χρησιμοποιείται για την αποθήκευση του bit rate και του Average Latency (εναλλακτικά)
- tracksegment : ο αριθμός - ταυτότητα του segment στο οποίο ανήκει το waypoint
- accuracy : η ακρίβεια της θέσης που παρείχε ο δορυφόρος

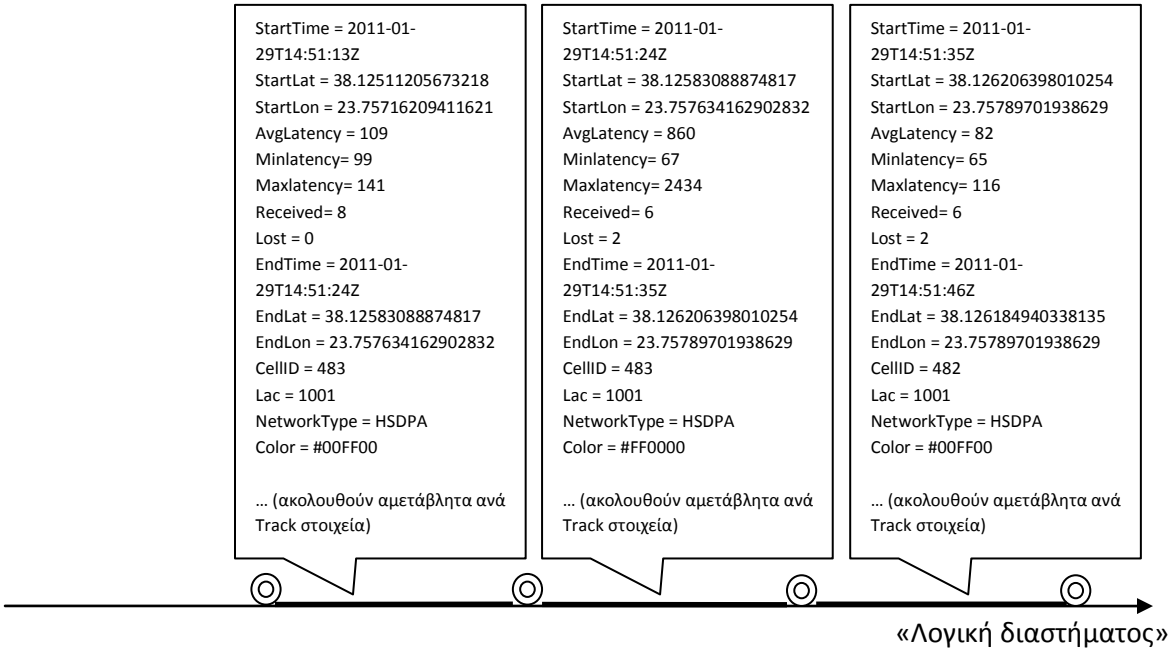
- altitude : το υψόμετρο της καταγραφείσας θέσης
- bearing : η διεύθυνση της κίνησης (degrees East of true North)
- minlatency : η μικρότερη τιμή latency που κατεγράφη σε αυτό το διάστημα (αφορά την περίπτωση Ping)
- maxlatency : η μεγαλύτερη τιμή latency που κατεγράφη σε αυτό το διάστημα (αφορά την περίπτωση Ping)
- received : ο αριθμός των επιτυχημένων προσπαθειών στην περίπτωση του Ping (πόσα πακέτα «απαντήσεις» ελήφθησαν) σε αυτό το διάστημα
- lost : ο αριθμός των αποτυχημένων προσπαθειών στην περίπτωση του Ping (πόσα πακέτα «απαντήσεις» χάθηκαν) σε αυτό το διάστημα
- cellid : η κυψέλη στην οποία βρισκόμασταν, όταν αποθηκεύτηκε η θέση με τις μετρήσεις στο τέλος του διαστήματος
- lac : ο κωδικός τοποθεσίας (location area code), όταν αποθηκεύτηκε η θέση με τις μετρήσεις στο τέλος του διαστήματος
- networktype : το είδος του δικτύου στο οποίο ήταν συνδεδεμένος ο Client, όταν αποθηκεύτηκε η θέση με τις μετρήσεις στο τέλος του διαστήματος

Ο πίνακας *waypoints* περιλαμβάνει εκείνες τις πληροφορίες που μπορεί να μεταβάλονται κατά τη διάρκεια της καταγραφής. Στην ουσία πρόκειται για τις πληροφορίες που έχει συγκεντρώσει στο τέλος μιας μέτρησης των 10 sec και πριν προχωρήσει στην επόμενη. Η «λογική σημείου» που ακολουθεί, δηλαδή τι μετρήσεις έχει πάρει μέχρι εκείνο το χωρικό και χρονικό σημείο, μεταλλάσσεται σε «λογική διαστήματος» κατά τη δημιουργία του xml αρχείου που αποτυπώνει συνολικά την καταγραφή. Για το σκοπό αυτό, το χωρικό και χρονικό σημείο που προσδιορίζει ένα waypoint στη βάση δεδομένων της εφαρμογής, αποτελεί την έναρξη του επόμενου διαστήματος στην περίπτωση του xml. Με αυτό τον τρόπο, τα σημεία γίνονται διαστήματα. Στην περίπτωση μιας εξωτερικής καταγραφής (Outdoors), το σύνολο των χωρικών – χρονικών διαστημάτων αποτελεί μια διαδρομή. Σε μια εσωτερική καταγραφή (Indoors), αφού η θέση παραμένει σταθερή (fixed location), το σύνολο των χρονικών διαστημάτων προσδιορίζει μια καταγραφή.



2.5.2.2 Δημιουργία των xml αρχείων

Η δομή της καταγραφής, όπως αυτή έχει πραγματωθεί στη βάση δεδομένων της εφαρμογής (KampRate), δεν εξυπηρετεί τους σκοπούς του συστήματός μας. Κρίθηκε, λοιπόν, αναγκαίο να βρεθεί ένας τρόπος αποτύπωσης των καταγραφών που θα ανταποκρινόταν στη «λογική διαστήματος». Επιπλέον, η νέα δομή έπρεπε να είναι εύκολα διαχειρίσιμη, σαφής και ευέλικτη, ώστε τα διάφορα αρχεία των καταγραφών να συλλέγονται στον Server, να εισάγονται σε μια κεντρική βάση δεδομένων και να μπορούν να χρησιμοποιηθούν για αναλύσεις και απεικονίσεις στην πλευρά του. Με γνώμονα τα παραπάνω, καταλήξαμε στο σχεδιασμό μιας xml δομής, η οποία χρησιμοποιεί τις καταγεγραμμένες πληροφορίες της βάσης δεδομένων της εφαρμογής KampRate στην κινητή συσκευή. Δημιουργεί διαστήματα μετρήσεων (lines), κάθε ένα από τα οποία φέρει μια ομάδα αμετάβλητων (για κάθε ξεχωριστό Track) και μεταβλητών πληροφοριών.



Παρακάτω παρουσιάζονται και αναλύονται οι 4 περιπτώσεις καταγραφών ως προς τα εξαγόμενα xml αρχεία:

A. Download Outdoors

```
<?xml version='1.0' encoding='UTF-8' standalone='yes' ?>
<Track>
<line>
<StartTime>2011-01-29T14:38:39Z</StartTime>      -> χρονική έναρξη του διαστήματος
<StartLat>38.12638878822327</StartLat>           -> γεωγραφικό πλάτος αρχής διαστήματος
<StartLon>23.757998943328857</StartLon>          -> γεωγραφικό μήκος αρχής διαστήματος
<Bitrate>223</Bitrate>                           -> μετρούμενο bit rate στο διάστημα
<EndTime>2011-01-29T14:38:49Z</EndTime>          -> χρονική λήξη διαστήματος
<EndLat>38.12637269496918</EndLat>               -> γεωγραφικό πλάτος τέλους διαστήματος
<EndLon>23.758004307746887</EndLon>              -> γεωγραφικό μήκος τέλους διαστήματος
<CellID>482</CellID>                             -> cell id στο τέλος του διαστήματος
<Lac>1001</Lac>                                  -> lac στο τέλος του διαστήματος
<NetworkType>HSDPA</NetworkType>                 -> network type στο τέλος του διαστήματος
<Color>#FF8040</Color>                           -> χρωματισμός διαστήματος

<NMC>DOWNLOAD</NMC>
<AREA>outdoors</AREA>
<SERVERIP>85.72.36.243</SERVERIP>
<PACKETSIZE>1400</PACKETSIZE>
<TRACKNAME>IMEI:9838 Tr 2011-01-29 16:38 !DL</TRACKNAME>
<TRACKTIME>2011-01-29T14:38:28Z</TRACKTIME>
</line>

<line>
<StartTime>2011-01-29T14:38:49Z</StartTime>
<StartLat>38.12637269496918</StartLat>
<StartLon>23.758004307746887</StartLon>
<Bitrate>223</Bitrate>
<EndTime>2011-01-29T14:39:00Z</EndTime>
<EndLat>38.12629759311676</EndLat>
<EndLon>23.75791847705841</EndLon>
<CellID>482</CellID>
<Lac>1001</Lac>
<NetworkType>HSDPA</NetworkType>
<Color>#FF8040</Color>
```

Μεταβλητά στοιχεία για κάθε line. Κάθε line αποτελεί ένα διάστημα μετρήσεων. Το σύνολο των lines μιας καταγραφής συγκροτούν μια διαδρομή ή μια καταγραφή εσωτερικού χώρου.

```
<NMC>DOWNLOAD</NMC>                              -> είδος καταγραφής (Download – Ping)
<AREA>outdoors</AREA>                            -> χώρος καταγραφής (Indoors – Outdoors)
<SERVERIP>85.72.36.243</SERVERIP>                -> IP Address του Server
<PACKETSIZE>1400</PACKETSIZE>                    -> μέγεθος πακέτου
<TRACKNAME>IMEI:9838 Tr 2011-01-29 16:38 !DL</TRACKNAME> -> όνομα καταγραφής
<TRACKTIME>2011-01-29T14:38:28Z</TRACKTIME>      -> χρονική έναρξη καταγραφής
</line>

<line>
<StartTime>2011-01-29T14:39:00Z</StartTime>
<StartLat>38.12629759311676</StartLat>
<StartLon>23.75791847705841</StartLon>
<Bitrate>224</Bitrate>
<EndTime>2011-01-29T14:39:11Z</EndTime>
<EndLat>38.12628149986267</EndLat>
<EndLon>23.75790238380432</EndLon>
<CellID>482</CellID>
<Lac>1001</Lac>
```

Αμετάβλητα στοιχεία για κάθε line. Είναι οι σταθερές πληροφορίες – ρυθμίσεις κάθε καταγραφής. Είναι απαραίτητες, καθώς στην database του Server εισάγονται lines από τις καταγραφές. Πρέπει, λοιπόν, για κάθε line να γνωρίζουμε υπό ποιες ρυθμίσεις μετρήθηκε και πού ανήκει.

```
<NetworkType>HSDPA</NetworkType>
<Color>#FF8040</Color>

<NMC>DOWNLOAD</NMC>
<AREA>outdoors</AREA>
<SERVERIP>85.72.36.243</SERVERIP>
<PACKETSIZE>1400</PACKETSIZE>
<TRACKNAME>IMEI:9838 Tr 2011-01-29 16:38 !DL</TRACKNAME>
<TRACKTIME>2011-01-29T14:38:28Z</TRACKTIME>
</line>

...
</Track>
```

B. Download Indoors

```
<?xml version='1.0' encoding='UTF-8' standalone='yes' ?>
<Track>
<line>
<StartTime>2011-01-29T13:26:49Z</StartTime>
<StartLat>38.114472</StartLat>
<StartLon>23.767668</StartLon>
<Bitrate>225</Bitrate>
<EndTime>2011-01-29T13:26:59Z</EndTime>
<EndLat>38.114472</EndLat>
<EndLon>23.767668</EndLon>
<CellID>482</CellID>
<Lac>1001</Lac>
<NetworkType>HSDPA</NetworkType>
<Color>#FF8040</Color>

<NMC>DOWNLOAD</NMC>
<AREA>indoors</AREA>
<SERVERIP>85.72.36.243</SERVERIP>
<PACKETSIZE>1400</PACKETSIZE>
<TRACKNAME>IMEI:9838 Tr 2011-01-29 15:26 !DL</TRACKNAME>
<TRACKTIME>2011-01-29T13:26:39Z</TRACKTIME>
</line>

<line>
<StartTime>2011-01-29T13:26:59Z</StartTime>
<StartLat>38.114472</StartLat>
<StartLon>23.767668</StartLon>
<Bitrate>223</Bitrate>
<EndTime>2011-01-29T13:27:09Z</EndTime>
<EndLat>38.114472</EndLat>
<EndLon>23.767668</EndLon>
<CellID>482</CellID>
<Lac>1001</Lac>
<NetworkType>HSDPA</NetworkType>
<Color>#FF8040</Color>

<NMC>DOWNLOAD</NMC>
<AREA>indoors</AREA>
<SERVERIP>85.72.36.243</SERVERIP>
<PACKETSIZE>1400</PACKETSIZE>
<TRACKNAME>IMEI:9838 Tr 2011-01-29 15:26 !DL</TRACKNAME>
<TRACKTIME>2011-01-29T13:26:39Z</TRACKTIME>
</line>
```

Πρόκειται για Indoors καταγραφή, οπότε StartLat == EndLat και StartLon == EndLon, αφού η θέση της κινητής συσκευής είναι σταθερή (fixed location).

Στο όνομα της καταγραφής δεν φαίνεται αν πρόκειται για Indoors ή Outdoors. Τόσο οι Indoors όσο και οι Outdoors καταγραφές θα εισαχθούν στη database του Server ομαδοποιημένες κατά είδος καταγραφής (Download – Ping) και όχι κατά χώρο καταγραφής (Area).

Η χρονική λήξη ενός διαστήματος αποτελεί παράλληλα τη χρονική έναρξη για το επόμενο διάστημα.

```
<line>
<StartTime>2011-01-29T13:27:09Z</StartTime>
<StartLat>38.114472</StartLat>
<StartLon>23.767668</StartLon>
<Bitrate>224</Bitrate>
<EndTime>2011-01-29T13:27:19Z</EndTime>
<EndLat>38.114472</EndLat>
<EndLon>23.767668</EndLon>
<CellID>482</CellID>
<Lac>1001</Lac>
<NetworkType>HSDPA</NetworkType>
<Color>#FF8040</Color>

<NMC>DOWNLOAD</NMC>
<AREA>indoors</AREA>
<SERVERIP>85.72.36.243</SERVERIP>
<PACKETSIZE>1400</PACKETSIZE>
<TRACKNAME>IMEI:9838 Tr 2011-01-29 15:26 !DL</TRACKNAME>
<TRACKTIME>2011-01-29T13:26:39Z</TRACKTIME>
</line>

...
</Track>
```

Το πεδίο Color προσδιορίζει το χρωματισμό του διαστήματος ανάλογα με το μετρούμενο bit rate. Χρησιμεύει ιδιαίτερα στο NetPerfTool για μεγαλύτερη αποδοτικότητα στην απεικόνιση πολλαπλών καταγραφών στο χάρτη στο Server.

C. Ping Outdoors

```
<?xml version='1.0' encoding='UTF-8' standalone='yes' ?>
<Track>
<line>
<StartTime>2011-01-29T14:45:38Z</StartTime>
<StartLat>38.13521862030029</StartLat>
<StartLon>23.751593828201294</StartLon>
<AvgLatency>123</AvgLatency>
<MinLatency>96</MinLatency>
<MaxLatency>152</MaxLatency>
<Received>8</Received>
<Lost>0</Lost>
<EndTime>2011-01-29T14:45:49Z</EndTime>
<EndLat>38.13630759716034</EndLat>
<EndLon>23.75093936920166</EndLon>
<CellID>8163</CellID>
<Lac>1001</Lac>
<NetworkType>HSDPA</NetworkType>
<Color>#FFFF00</Color>

<NMC>PING</NMC>
<AREA>outdoors</AREA>
<SERVERIP>85.72.36.243</SERVERIP>
<PINGATTEMPTS>8</PINGATTEMPTS>
<TRACKNAME>IMEI:9838 Tr 2011-01-29 16:44 !PING</TRACKNAME>
<TRACKTIME>2011-01-29T14:44:32Z</TRACKTIME>
</line>

<line>
<StartTime>2011-01-29T14:45:49Z</StartTime>
<StartLat>38.13630759716034</StartLat>
<StartLon>23.75093936920166</StartLon>
<AvgLatency>323</AvgLatency>
<MinLatency>134</MinLatency>
<MaxLatency>652</MaxLatency>
<Received>6</Received>
```

Τα αποτελέσματα των μετρήσεων που ελήφθησαν σε αυτό το διάστημα. Οι καταγραφές Download από την άλλη, έχουν ένα μόνο πεδίο σε αυτή τη θέση, που προσδιορίζει την ταχύτητα του downloading (bit rate)

Τα received και lost πακέτα κάθε διαστήματος είναι ιδιαίτερα σημαντικά μεγέθη, καθώς συμπληρώνουν την πληροφορία για τη μέτρηση του latency. Η χρησιμότητά τους γίνεται αντιληπτή και από το εξής παράδειγμα: Ας υποθέσουμε πως έχει μετρηθεί AvgLatency = 67 msec σε ένα διάστημα (που αποτελεί σχετικά μικρό χρόνο). Αν αυτός ο χρόνος, ωστόσο, έχει προκύψει από μόλις μία επιτυχημένη προσπάθεια και 7 χαμένες (έστω ότι pingatempts = 8) δεν μπορεί να αποτελεί έγκυρη πληροφορία, διότι τα υπόλοιπα 7 πακέτα χάθηκαν. Τέτοιες περιπτώσεις μπορεί να εξετάσει ο διαχειριστής στην πλευρά του Server και να τις αποκλείσει ή απλά να τις επισημάνει με queries προς την database που επιστρέφει τα επιθυμητά πεδία των lines.

```
<Lost>2</Lost>
<EndTime>2011-01-29T14:46:00Z</EndTime>
<EndLat>38.136908411979675</EndLat>
<EndLon>23.750617504119873</EndLon>
<CellID>473</CellID>
<Lac>1001</Lac>
<NetworkType>HSDPA</NetworkType>
<Color>#FF8040</Color>

<NMC>PING</NMC>
<AREA>outdoors</AREA>
<SERVERIP>85.72.36.243</SERVERIP>
<PINGATTEMPTS>8</PINGATTEMPTS>
<TRACKNAME>IMEI:9838 Tr 2011-01-29 16:44 !PING</TRACKNAME>
<TRACKTIME>2011-01-29T14:44:32Z</TRACKTIME>
</line>

<line>
<StartTime>2011-01-29T14:46:00Z</StartTime>
<StartLat>38.136908411979675</StartLat>
<StartLon>23.750617504119873</StartLon>
<AvgLatency>314</AvgLatency>
<MinLatency>105</MinLatency>
<MaxLatency>774</MaxLatency>
<Received>8</Received>
<Lost>0</Lost>
<EndTime>2011-01-29T14:46:12Z</EndTime>
<EndLat>38.13760042190552</EndLat>
<EndLon>23.750027418136597</EndLon>
<CellID>473</CellID>
<Lac>1001</Lac>
<NetworkType>HSDPA</NetworkType>
<Color>#FF8040</Color>

<NMC>PING</NMC>
<AREA>outdoors</AREA>
<SERVERIP>85.72.36.243</SERVERIP>
<PINGATTEMPTS>8</PINGATTEMPTS>
<TRACKNAME>IMEI:9838 Tr 2011-01-29 16:44 !PING</TRACKNAME>
<TRACKTIME>2011-01-29T14:44:32Z</TRACKTIME>
</line>

...
</Track>
```

Το γεωγραφικό πλάτος και μήκος του τέλους ενός διαστήματος ταυτίζονται με το γεωγραφικό πλάτος και μήκος της αρχής του επόμενου διαστήματος. Αυτό ισχύει για κάθε τύπο Outdoors καταγραφής. Στο Indoors η θέση είναι fixed, οπότε δεν υπάρχει καμία αλλαγή καθ'όλη τη διάρκεια της καταγραφής.

Το Cell ID στο τέλος του διαστήματος. Παρατηρούμε πως έχει αλλάξει σε σχέση με εκείνο που κατεγράφη δυο διαστήματα νωρίτερα.

D. Ping Indoors

```
<?xml version='1.0' encoding='UTF-8' standalone='yes' ?>
<Track>
<line>
<StartTime>2011-01-29T13:21:17Z</StartTime>
<StartLat>38.114472</StartLat>
<StartLon>23.767668</StartLon>
<AvgLatency>108</AvgLatency>
<MinLatency>85</MinLatency>
<MaxLatency>138</MaxLatency>
<Received>8</Received>
<Lost>0</Lost>
<EndTime>2011-01-29T13:21:27Z</EndTime>
<EndLat>38.114472</EndLat>
```

```
<EndLon>23.767668</EndLon>
<CellID>482</CellID>
<Lac>1001</Lac>
<NetworkType>HSDPA</NetworkType>
<Color>#00FF00</Color>

<NMC>PING</NMC>
<AREA>indoors</AREA>
<SERVERIP>85.72.36.243</SERVERIP>
<PINGATTEMPTS>8</PINGATTEMPTS>
<TRACKNAME>IMEI:9838 Tr 2011-01-29 15:21 !PING</TRACKNAME>
<TRACKTIME>2011-01-29T13:21:07Z</TRACKTIME>
</line>

<line>
<StartTime>2011-01-29T13:21:27Z</StartTime>
<StartLat>38.114472</StartLat>
<StartLon>23.767668</StartLon>
<AvgLatency>119</AvgLatency>
<MinLatency>80</MinLatency>
<MaxLatency>204</MaxLatency>
<Received>8</Received>
<Lost>0</Lost>
<EndTime>2011-01-29T13:21:37Z</EndTime>
<EndLat>38.114472</EndLat>
<EndLon>23.767668</EndLon>
<CellID>482</CellID>
<Lac>1001</Lac>
<NetworkType>HSDPA</NetworkType>
<Color>#00FF00</Color>

<NMC>PING</NMC>
<AREA>indoors</AREA>
<SERVERIP>85.72.36.243</SERVERIP>
<PINGATTEMPTS>8</PINGATTEMPTS>
<TRACKNAME>IMEI:9838 Tr 2011-01-29 15:21 !PING</TRACKNAME>
<TRACKTIME>2011-01-29T13:21:07Z</TRACKTIME>
</line>

<line>
<StartTime>2011-01-29T13:21:37Z</StartTime>
<StartLat>38.114472</StartLat>
<StartLon>23.767668</StartLon>
<AvgLatency>121</AvgLatency>
<MinLatency>78</MinLatency>
<MaxLatency>159</MaxLatency>
<Received>8</Received>
<Lost>0</Lost>
<EndTime>2011-01-29T13:21:47Z</EndTime>
<EndLat>38.114472</EndLat>
<EndLon>23.767668</EndLon>
<CellID>482</CellID>
<Lac>1001</Lac>
<NetworkType>HSDPA</NetworkType>
<Color>#FFFF00</Color>

...
</Track>
```

Το πεδίο με την IP Address του Server μπορεί να φανεί χρήσιμο σε πιθανή αλλαγή του Server ή αν χρησιμοποιούνται περισσότεροι του ενός. Κρίνεται σκόπιμο να γίνουν μετρήσεις για το ποσοστό των πακέτων που χάνονται με ευθύνη των δρομολογητών του υποδικτύου στο οποίο βρίσκεται ο Server.

2.5.2.3 Αποστολή των xml αρχείων στον Server

Για την αποστολή των xml αρχείων που φέρουν τις πληροφορίες των καταγεγραμμένων διαδρομών ή των μετρήσεων εσωτερικού χώρου από τον Client στον Server υλοποιήθηκε η UploadTo Server Activity. Εγκαθίσταται μια HTTP σύνδεση ανάμεσα στον Client και τον Server και ο Client «ανεβάζει» το xml αρχείο με HTTP POST. POST είναι μια από τις request methods που υποστηρίζει το HTTP πρωτόκολλο. Χρησιμοποιείται όταν ο Client χρειάζεται να στείλει δεδομένα (data) στον Server ως μέρος της request, όπως για παράδειγμα να «ανεβάσει» ένα αρχείο. Σε αντίθεση με το GET όπου μόνο η URL και οι επικεφαλίδες (headers) στέλνονται στον Server, στο POST περιλαμβάνεται και ένα μήνυμα. Αυτό επιτρέπει δεδομένα αυθαίρετου μεγέθους και τύπου να αποσταλούν στον Server. Είναι, λοιπόν, το ενδεδειγμένο για τις ανάγκες μας.

Στην πλευρά του Server, το handle_upload.php script διαχειρίζεται το εισερχόμενο αρχείο. Η σύνδεση γίνεται στη θύρα 80, καθώς σε αυτήν «ακούει» ο Web Server που έχουμε ρυθμίσει. Εναλλακτικά, μπορεί να χρησιμοποιηθούν και άλλες θύρες (για παράδειγμα η 8080) με την αντίστοιχη τροποποίηση στον κώδικα του KampRate:

```
String urlServer = "http://" + ServerIP + ":8080" + /handle_upload.php";
```

αλλά και στις ρυθμίσεις του Server (Apache, httpd.conf):

```
Listen 8080
```

Αν το αρχείο «ανέβει» με επιτυχία στον Server, εμφανίζεται στην οθόνη της εφαρμογής KampRate ένα μήνυμα, ώστε να ειδοποιηθεί ο χρήστης. Σε περίπτωση κάποιου σφάλματος (π.χ. αν ο Server είναι unavailable ή αν διακοπεί η σύνδεση) θα εμφανιστεί μήνυμα σφάλματος.

2.5.3 Server

2.5.3.1 Η βάση δεδομένων του Server

Τα xml αρχεία με τις καταγραφές των Clients εισάγονται στη βάση δεδομένων του Server. Συγκεκριμένα, οι τιμές των πεδίων των xml αρχείων εισάγονται στα αντίστοιχα πεδία στη βάση δεδομένων του Server, η οποία έχει δύο πίνακες (tables), τον «download» και τον «ping». Τους πίνακες αυτούς μπορούμε να τους κατασκευάσουμε είτε με εντολή από τη γραμμή εντολών της MySQL είτε μέσω του γραφικού περιβάλλοντος του phpMyAdmin. Έχουν τα ακόλουθα πεδία με τους εξής τύπους δεδομένων:

Table: *download*

Field	Type
StartTime	datetime
StartLat	double
StartLon	double
Bitrate	mediumint(9)
EndTime	datetime
EndLat	double
EndLon	double
CellID	smallint(6)
Lac	smallint(6)
NetworkType	tinytext
Color	tinytext
AREA	tinytext
SERVERIP	tinytext
PACKETSIZE	smallint(6)
TRACKNAME	tinytext
TRACKTIME	datetime

Table: *ping*

Field	Type
StartTime	datetime
StartLat	double
StartLon	double
AvgLatency	smallint(6)
MinLatency	smallint(6)
MaxLatency	smallint(6)
Received	tinyint(4)
Lost	tinyint(4)
EndTime	datetime
EndLat	double
EndLon	double
CellID	smallint(6)
Lac	smallint(6)
NetworkType	tinytext
Color	tinytext
AREA	tinytext
SERVERIP	tinytext
PINGATTEMPTS	tinyint(4)
TRACKNAME	tinytext

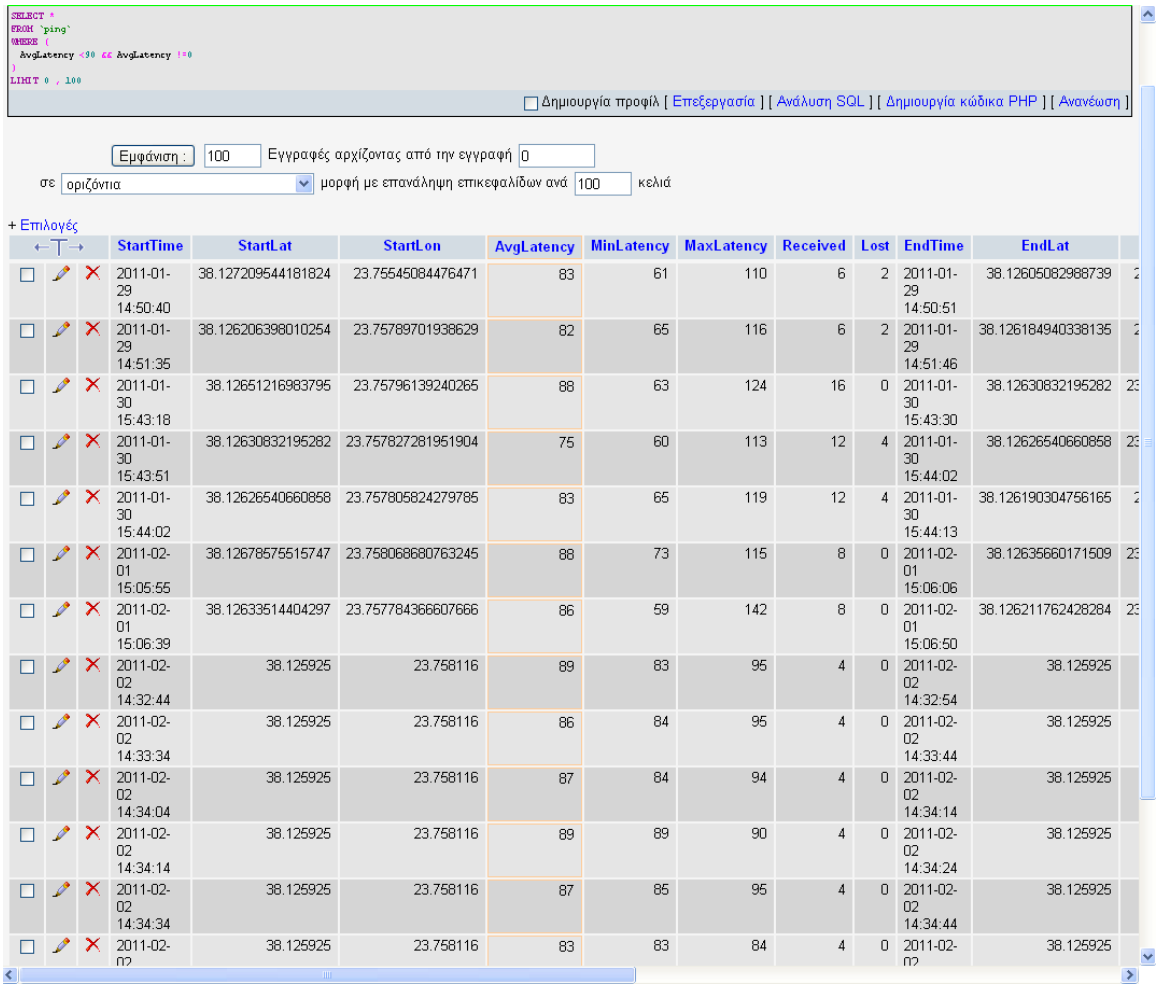
TRACKTIME	datetime

Οι παραπάνω πίνακες «γεμίζουν» με τα στοιχεία των τμημάτων (lines) από τις καταγραφές των Clients συγκροτώντας μια πηγή πληροφοριών για την ποιότητα και τις «ταχύτητες» του δικτύου στα διάφορα σημεία της επικράτειας. Ο διαχειριστής δύναται να εκτελέσει queries προς τη βάση δεδομένων με τα κριτήρια που επιθυμεί, όπως για παράδειγμα να απομονώσει τμήματα ή σημεία με χαμηλές ταχύτητες.

Στο παράδειγμα που ακολουθεί, έχουμε εισάγει 4 καταγραφές στην database, 3 Outdoors διαδρομές και 1 Indoors καταγραφή. Εκτελούμε μια σειρά από queries για να δείξουμε τις δυνατότητες του διαχειριστή ως προς την αξιοποίηση των μετρήσεων που έχουν καταγραφεί και συλλεχθεί.

Αναζήτηση 1: Βρίσκουμε εκείνα τα σημεία ή διαστήματα στα οποία μετρήθηκε Average Latency μικρότερο από 90 msec, εκτός από εκείνα τα σημεία ή διαστήματα στα οποία βρέθηκε 0 msec (δηλαδή όλες οι προσπάθειες απέτυχαν).

```
SELECT *
FROM `ping`
WHERE (
AvgLatency <90
AND AvgLatency !=0
)
```



```
SELECT *
FROM `ping`
WHERE (
  AvgLatency <90 && AvgLatency !=0
)
LIMIT 0 , 100
```

Δημιουργία προφίλ [Επεξεργασία] [Ανάλυση SQL] [Δημιουργία κώδικα PHP] [Ανανέωση]

Εμφάνιση : 100 Εγγραφές αρχίζοντας από την εγγραφή 0

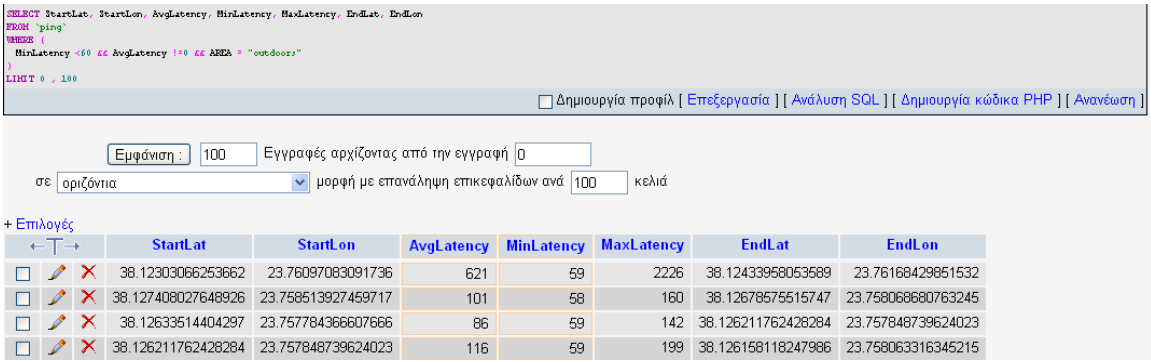
σε οριζόντια μορφή με επανάληψη επικεφαλίδων ανά 100 κελιά

+ Επιλογές

StartTime	StartLat	StartLon	AvgLatency	MinLatency	MaxLatency	Received	Lost	EndTime	EndLat
2011-01-29 14:50:40	38.127209544181824	23.75545084476471	83	61	110	6	2	2011-01-29 14:50:51	38.12605082988739
2011-01-29 14:51:35	38.126206398010254	23.75789701938629	82	65	116	6	2	2011-01-29 14:51:46	38.126184940338135
2011-01-30 15:43:18	38.12651216983795	23.75796139240265	88	63	124	16	0	2011-01-30 15:43:30	38.12630832195282
2011-01-30 15:43:51	38.12630832195282	23.757827281951904	75	60	113	12	4	2011-01-30 15:44:02	38.12626540660858
2011-01-30 15:44:02	38.12626540660858	23.757805824279785	83	65	119	12	4	2011-01-30 15:44:13	38.126190304756165
2011-02-01 15:05:55	38.12678575515747	23.758068680763245	88	73	115	8	0	2011-02-01 15:06:06	38.12635660171509
2011-02-01 15:06:39	38.12633514404297	23.757784366607666	86	59	142	8	0	2011-02-01 15:06:50	38.126211762428284
2011-02-02 14:32:44	38.125925	23.758116	89	83	95	4	0	2011-02-02 14:32:54	38.125925
2011-02-02 14:33:34	38.125925	23.758116	86	84	95	4	0	2011-02-02 14:33:44	38.125925
2011-02-02 14:34:04	38.125925	23.758116	87	84	94	4	0	2011-02-02 14:34:14	38.125925
2011-02-02 14:34:14	38.125925	23.758116	89	89	90	4	0	2011-02-02 14:34:24	38.125925
2011-02-02 14:34:34	38.125925	23.758116	87	85	95	4	0	2011-02-02 14:34:44	38.125925
2011-02-02	38.125925	23.758116	83	83	84	4	0	2011-02-02	38.125925

Αναζήτηση 2: Βρίσκουμε τις συντεταγμένες και τα Average Latency, Minimum Latency, Maximum Latency από εκείνα τα διαστήματα που είχαν Minimum Latency μικρότερο από 60 msec (δεν επιθυμούμε να βρούμε σημεία που να πληρούν αυτό το κριτήριο, δηλαδή αποκλείουμε τις μετρήσεις που πραγματοποιήθηκαν σε εσωτερικό χώρο).

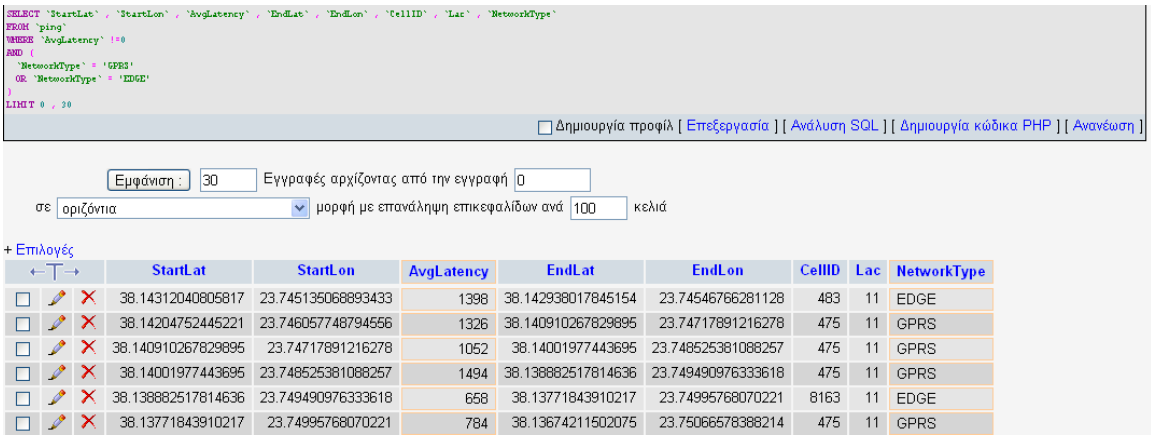
```
SELECT `StartLat` , `StartLon` , `AvgLatency` , `MinLatency` , `MaxLatency` , `EndLat` ,
`EndLon`
FROM `ping`
WHERE `AvgLatency` !=0
AND `MinLatency` <60
AND `AREA` = 'outdoors'
```



StartLat	StartLon	AvgLatency	MinLatency	MaxLatency	EndLat	EndLon
38.12303066253662	23.76097083091736	621	59	2226	38.12433958053589	23.76168429851532
38.127408027648926	23.758513927459717	101	58	160	38.12678575515747	23.758068680763245
38.12633514404297	23.757784366607666	86	59	142	38.126211762428284	23.757848739624023
38.126211762428284	23.757848739624023	116	59	199	38.126158118247986	23.758063316345215

Αναζήτηση 3: Βρίσκουμε συντεταγμένες, Average Latency, Cell ID, LAC και Network Type για εκείνα τα διαστήματα στο τέλος των οποίων το δίκτυο ήταν «GPRS» ή «EDGE».

```
SELECT `StartLat` , `StartLon` , `AvgLatency` , `EndLat` , `EndLon` , `CellID` , `Lac` ,
`NetworkType`
FROM `ping`
WHERE `AvgLatency` !=0
AND (
`NetworkType` = 'GPRS'
OR `NetworkType` = 'EDGE'
)
```

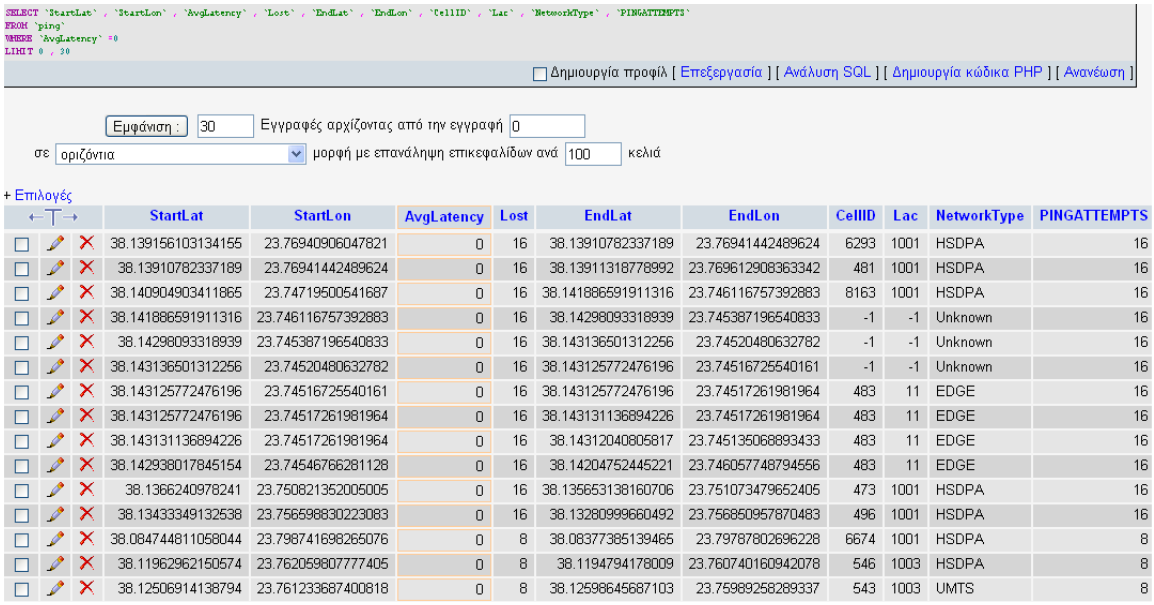


StartLat	StartLon	AvgLatency	EndLat	EndLon	CellID	Lac	NetworkType
38.14312040805817	23.745135068893433	1398	38.142938017845154	23.74546766281128	483	11	EDGE
38.14204752445221	23.746057748794556	1326	38.140910267829895	23.74717891216278	475	11	GPRS
38.140910267829895	23.74717891216278	1052	38.14001977443695	23.748525381088257	475	11	GPRS
38.14001977443695	23.748525381088257	1494	38.138882517814636	23.749490976333618	475	11	GPRS
38.138882517814636	23.749490976333618	658	38.13771843910217	23.74995768070221	8163	11	EDGE
38.13771843910217	23.74995768070221	784	38.13674211502075	23.75066578388214	475	11	GPRS

Όπως αναμενόταν, τα μετρούμενα Average Latency είναι πολύ μεγάλα, δηλαδή οι χρόνοι καθυστέρησης ήταν σημαντικοί, αφού σε εκείνα τα διαστήματα ήμαστε σε 2.5G δίκτυο με περιορισμένες ταχύτητες μεταφοράς δεδομένων.

Αναζήτηση 4: Βρίσκουμε σημεία και διαστήματα στα οποία καμία προσπάθεια Ping δεν ήταν επιτυχημένη, κάτι που οφείλεται είτε σε κακή ποιότητα δικτύου σε εκείνη τη θέση είτε σε διαπομπή (handover) και μετάβαση από μια κυψέλη σε μια άλλη.

```
SELECT `StartLat` , `StartLon` , `AvgLatency` , `Lost` , `EndLat` , `EndLon` , `CellID` ,
 `Lac` , `NetworkType` , `PINGATTEMPTS`
FROM `ping`
WHERE `AvgLatency` =0
```



Δημιουργία προφίλ [Επεξεργασία] [Ανάλυση SQL] [Δημιουργία κώδικα PHP] [Ανανέωση]

Εμφάνιση : 30 Εγγραφές αρχίζοντας από την εγγραφή 0

σε οριζόντια μορφή με επανάληψη επικεφαλίδων ανά 100 κελιά

+ Επιλογές

StartLat	StartLon	AvgLatency	Lost	EndLat	EndLon	CellID	Lac	NetworkType	PINGATTEMPTS
38.139156103134155	23.76940906047821	0	16	38.13910782337189	23.76941442489624	6293	1001	HSDPA	16
38.13910782337189	23.76941442489624	0	16	38.13911318778992	23.769612908363342	481	1001	HSDPA	16
38.140904903411865	23.74719500541687	0	16	38.141886591911316	23.746116757392883	8163	1001	HSDPA	16
38.141886591911316	23.746116757392883	0	16	38.14298093318939	23.745387196540833	-1	-1	Unknown	16
38.14298093318939	23.745387196540833	0	16	38.143136501312256	23.74520480632782	-1	-1	Unknown	16
38.143136501312256	23.74520480632782	0	16	38.143125772476196	23.74516725540161	-1	-1	Unknown	16
38.143125772476196	23.74516725540161	0	16	38.143125772476196	23.74517261981964	483	11	EDGE	16
38.143125772476196	23.74517261981964	0	16	38.143131136894226	23.74517261981964	483	11	EDGE	16
38.143131136894226	23.74517261981964	0	16	38.14312040805817	23.745135068893433	483	11	EDGE	16
38.142938017845154	23.74546766281128	0	16	38.14204752445221	23.746057748794556	483	11	EDGE	16
38.1366240978241	23.750821352005005	0	16	38.135653138160706	23.751073479652405	473	1001	HSDPA	16
38.13433349132538	23.756598830223083	0	16	38.13280999660492	23.756850957870483	496	1001	HSDPA	16
38.084744811058044	23.798741698265076	0	8	38.08377385139465	23.79787802696228	6674	1001	HSDPA	8
38.11962962150574	23.762059807777405	0	8	38.1194794178009	23.760740160942078	546	1003	HSDPA	8
38.12506914138794	23.761233687400818	0	8	38.12598645687103	23.75989258289337	543	1003	UMTS	8

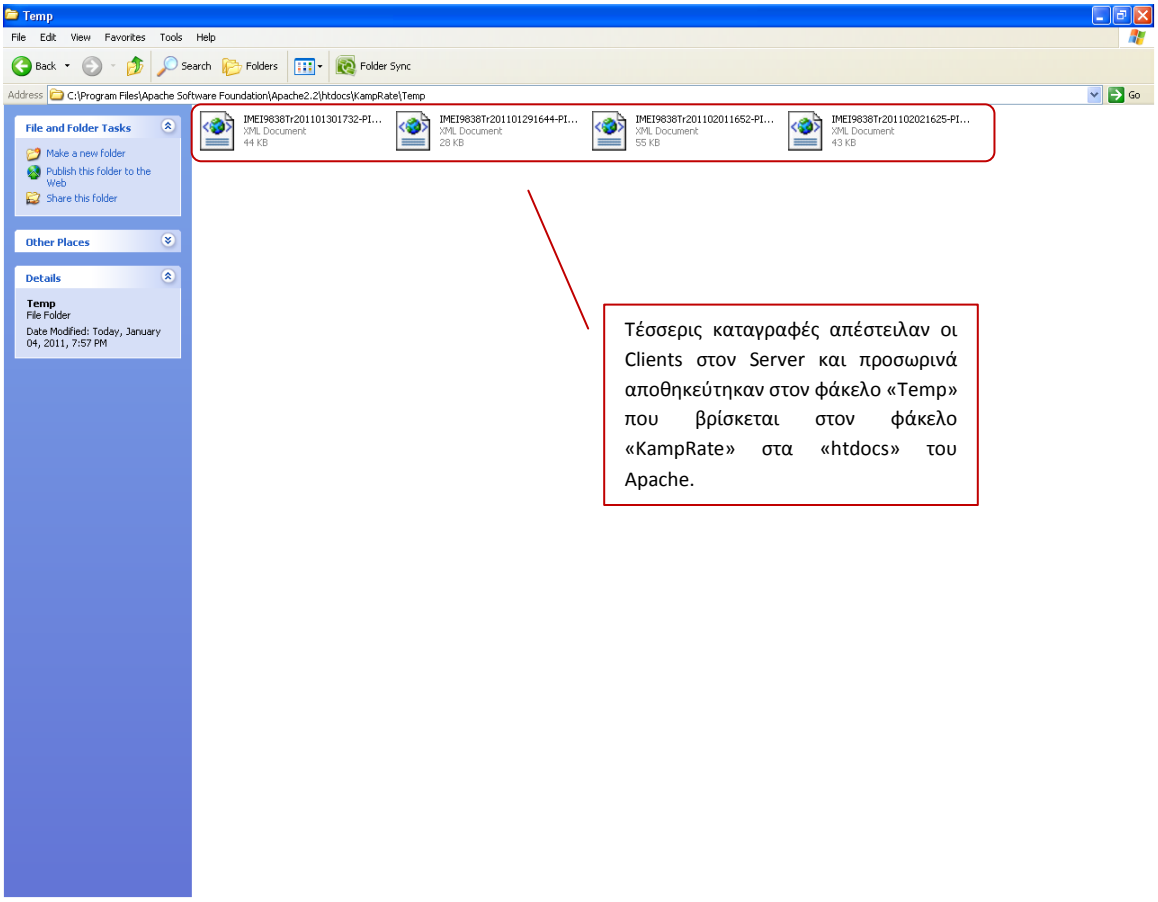
Παρατηρούμε πως στην 4η, 5η και 6η εγγραφή τα Cell ID, LAC και Network Type είναι άγνωστα. Αυτό οφείλεται σε διαπομπή (handover), οπότε για μερικά sec η ροή των δεδομένων διακόπτεται. Στις υπόλοιπες περιπτώσεις δεν είναι βέβαιο ότι δεν υπήρξε handover και οι αποτυχίες οφείλονται στην κακή ποιότητα του δίκτυου ή στο βαρύ φορτίο εκείνης της χρονικής περιόδου, διότι τα Cell ID, LAC και Network Type λαμβάνουν τιμές στο τέλος του διαστήματος των 10 sec, οπότε μπορεί μέσα στο διάστημα να έγινε το handover και στο τέλος του να είχε αλλάξει η κυψέλη στην οποία βρίσκεται η κινητή συσκευή.

2.5.3.2 Υποδοχή των xml αρχείων και εισαγωγή στη βάση δεδομένων

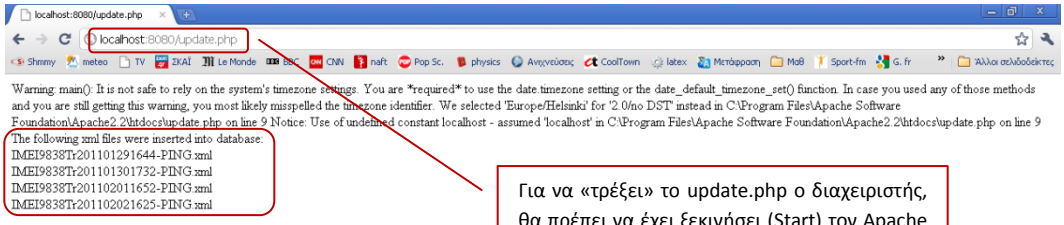
Όπως είδαμε, ο Client αποστέλλει τις καταγραφές του στον Server, ο οποίος τις υποδέχεται και τις τοποθετεί στον φάκελο «Temp». Για να επιτευχθεί αυτό έχουμε δημιουργήσει ένα php script, το handle_upload.php. Αυτό είναι υπεύθυνο για την υποδοχή του αρχείου, τον έλεγχο του τύπου αρχείου (xml, gpx, kmz), καθώς και το μέγεθός του (μικρότερο από 2MB) και τέλος

τη μεταφορά του στο φάκελο «Temp». Τα δύο κριτήρια διασφαλίζουν σε ένα πρώτο επίπεδο πως ο Server δεν θα δεχθεί μη σχετικά με το σκοπό του αρχεία.

Οι Clients «ανεβάζουν» τα αρχεία των καταγραφών τους στον Server, τα οποία καταλήγουν στο φάκελο «Temp»:

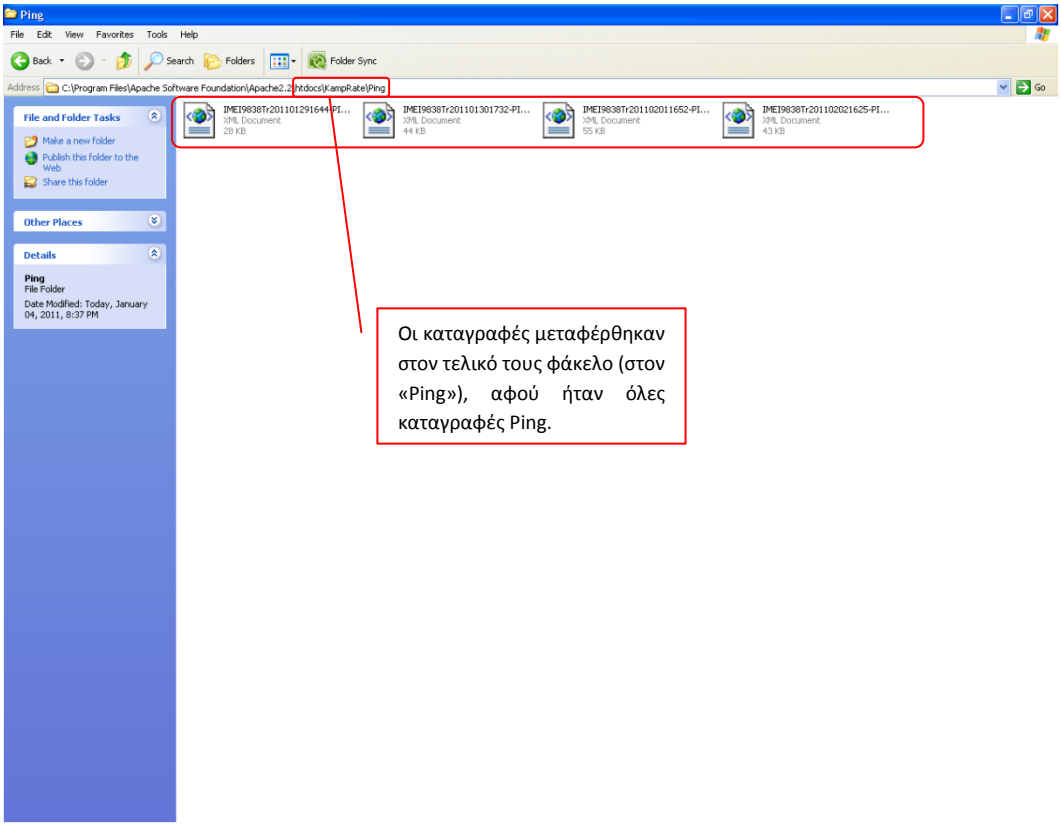


Έπειτα, θα πρέπει οι πληροφορίες που φέρουν αυτά τα xml αρχεία να εισαχθούν στη βάση δεδομένων. Για το σκοπό αυτό έχουμε δημιουργήσει ένα ακόμη script, το update.php. Σε αυτό πραγματοποιείται αρχικά σύνδεση με τη MySQL βάση δεδομένων. Σε περίπτωση αποτυχίας της σύνδεσης θα ειδοποιηθούμε με μήνυμα σφάλματος στην οθόνη. Έπειτα, για κάθε αρχείο που βρίσκεται στον φάκελο Temp ελέγχει από το όνομά του αν είναι καταγραφή Download ή Ping και εκτελεί μία query προς τη βάση δεδομένων που αφορά τη φόρτωση του xml αρχείου σε αυτήν, στον αντίστοιχο πίνακά της (download ή ping). Κατόπιν, τα xml αρχεία μεταφέρονται στους τελικούς τους φακέλους («DL» και «Ping»). Η διαδικασία παρουσιάζεται παρακάτω:



Για να «τρέξει» το update.php ο διαχειριστής, θα πρέπει να έχει ξεκινήσει (Start) τον Apache και να πληκτρολογήσει στον Web Browser την παραπάνω διεύθυνση. Στην περίπτωσή μας, έχουμε ρυθμίσει τον Web Server να ακούει στη θύρα (port) 8080.

Ειδοποιούμαστε για τις καταγραφές που εισήχθησαν με επιτυχία στην database. Και οι τέσσερις καταγραφές εισήχθησαν με επιτυχία.

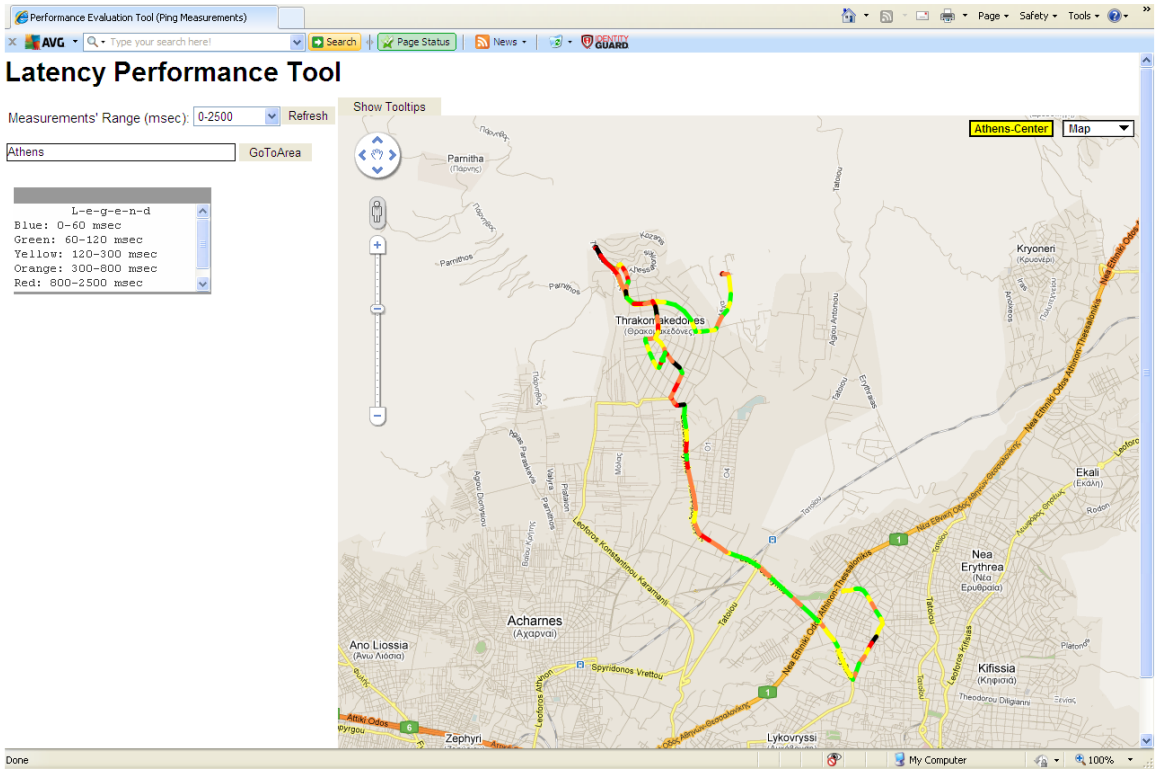


Οι καταγραφές μεταφέρθηκαν στον τελικό τους φάκελο (στον «Ping»), αφού ήταν όλες καταγραφές Ping.

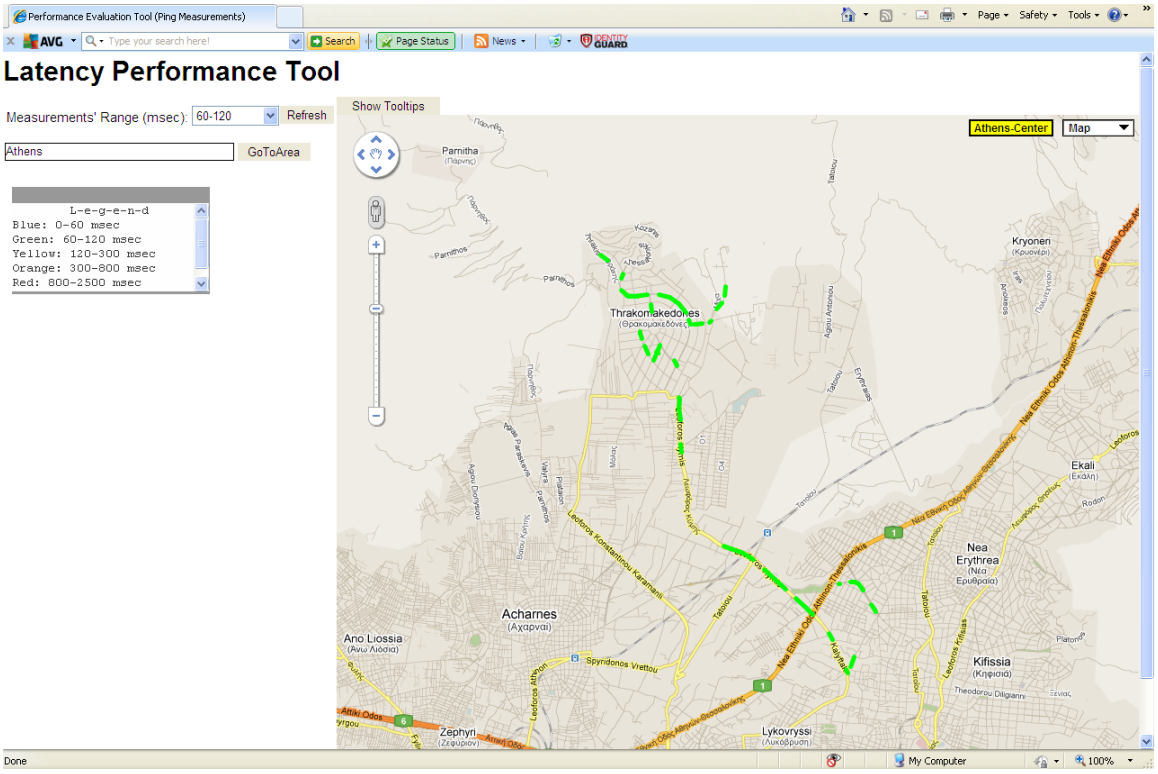
2.5.3.3 Το NetPerfTool για οπτική παρουσίαση των καταγεγραμμένων διαδρομών

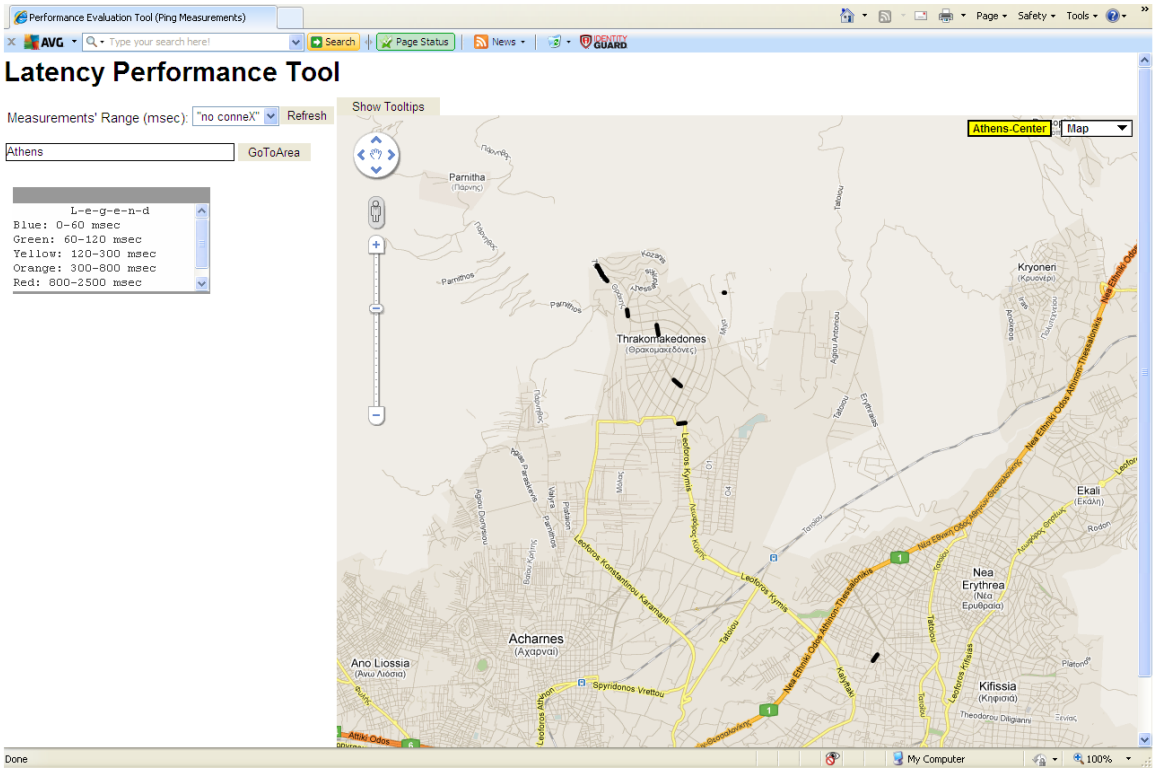
Με το NetPerfTool μπορούμε να κάνουμε μια συνολική χαρτογράφηση της ποιότητας του δικτύου με βάση τις ληφθείσες μετρήσεις από τους διάφορους Clients. Στην ουσία πρόκειται για μια απεικόνιση των μετρήσεων στο χάρτη, όπως γίνεται και στην εφαρμογή KampRate. Η βασική διαφορά είναι πως με αυτό το εργαλείο πετυχαίνουμε να εμφανίσουμε παράλληλα πολλές καταγραφές, στοιχείο που προσδίδει εποπτικότητα και ευκολία στον εντοπισμό περιοχών που χρήζουν προσοχής.

Στο ακόλουθο παράδειγμα έχουμε εισάγει ένα xml με τρεις διαδρομές Ping. Ανοίγουμε το NetPerfTool.html με τον Internet Explorer 8.

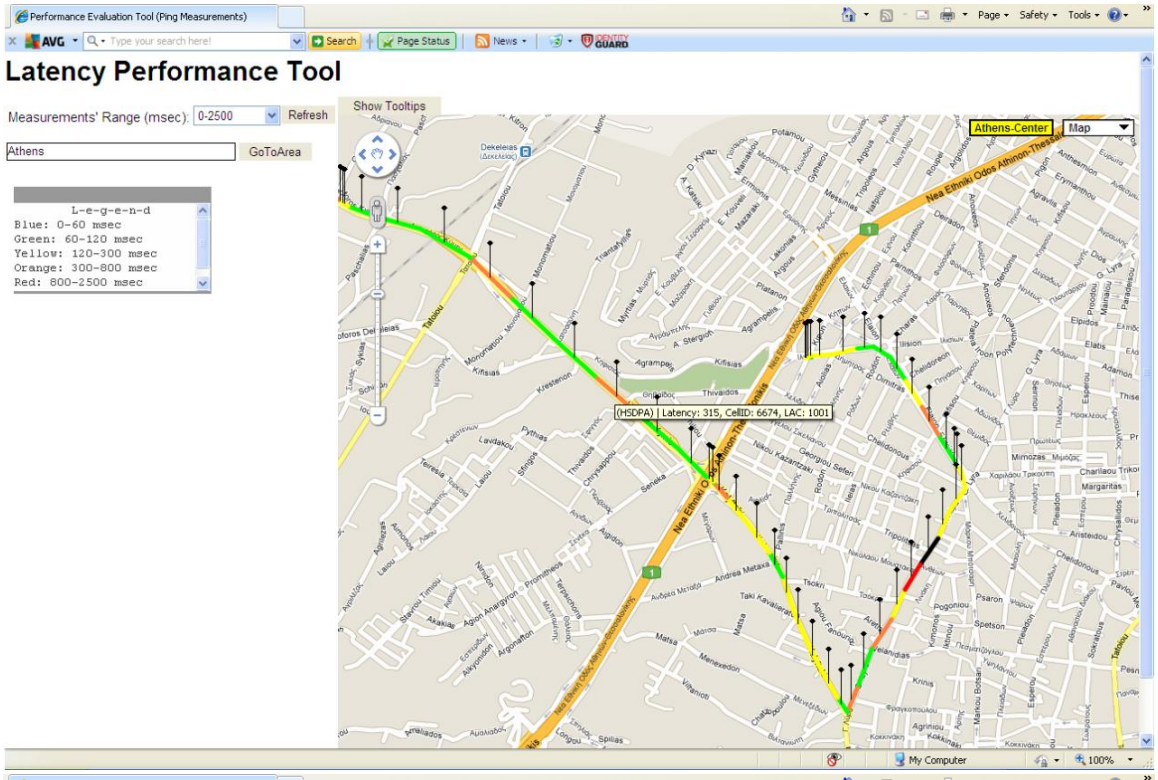


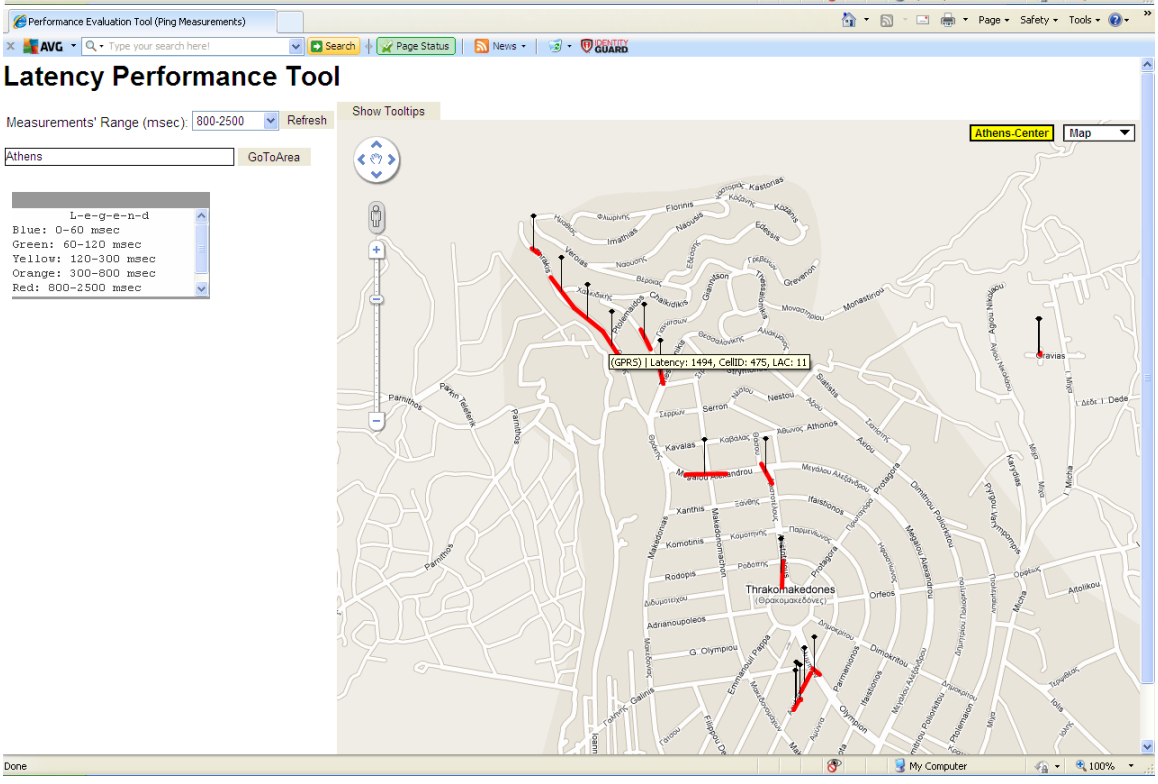
Στο αριστερό μέρος της οθόνης της εφαρμογής έχουμε τη δυνατότητα επιλογής του εύρους των μετρήσεων που επιθυμούμε να εμφανιστούν στο χάρτη, με την κλίμακα να βρίσκεται ακριβώς από κάτω. Στο δεξί μέρος παρουσιάζεται ο χάρτης με τις μετρήσεις των διαδρομών που έχουμε εισάγει. Αν επιλέξουμε εύρος μετρήσεων «60-120» msec και πατήσουμε «Refresh», θα δούμε μόνο εκείνα τα διαστήματα στα οποία μετρήθηκε Average Latency από 60-120 msec, ενώ, αν επιλέξουμε «no conneX», θα εμφανιστούν στο χάρτη μόνο εκείνα τα διαστήματα στα οποία απέτυχαν όλες οι προσπάθειες (ping attempts), δηλαδή εκείνα τα διαστήματα στα οποία συνέβη διαπομπή (handover) ή είχαν χαμηλή ποιότητα δικτύου:


Performance Evaluation Tool (Ping Measurements)
Latency Performance Tool
Measurements' Range (msec): 60-120
Refresh
Athens
GoToArea
L-e-g-e-n-d
Blue: 0-60 msec
Green: 60-120 msec
Yellow: 120-300 msec
Orange: 300-800 msec
Red: 800-2500 msec
Show Tooltips
Athens-Center
Map
Parnitha
Thrakomakedones
Kryoneri
Ekali
Nea Erythrea
Acharnes
Ano Liossia
Kifissia
Zephyri
Lykovryssi
Done
My Computer
100%


Performance Evaluation Tool (Ping Measurements)
Latency Performance Tool
Measurements' Range (msec): "no conneX"
Refresh
Athens
GoToArea
L-e-g-e-n-d
Blue: 0-60 msec
Green: 60-120 msec
Yellow: 120-300 msec
Orange: 300-800 msec
Red: 800-2500 msec
Show Tooltips
Athens-Center
Map
Parnitha
Thrakomakedones
Kryoneri
Ekali
Nea Erythrea
Acharnes
Ano Liossia
Kifissia
Done
My Computer
100%

Μια επιπλέον επιλογή είναι να εμφανιστούν για κάθε καταγεγραμμένο διάστημα οι πληροφορίες για το είδος του δικτύου, το Average Latency, το Cell ID και το LAC. Η λειτουργία αυτή ενεργοποιείται πατώντας «Show Tooltips».



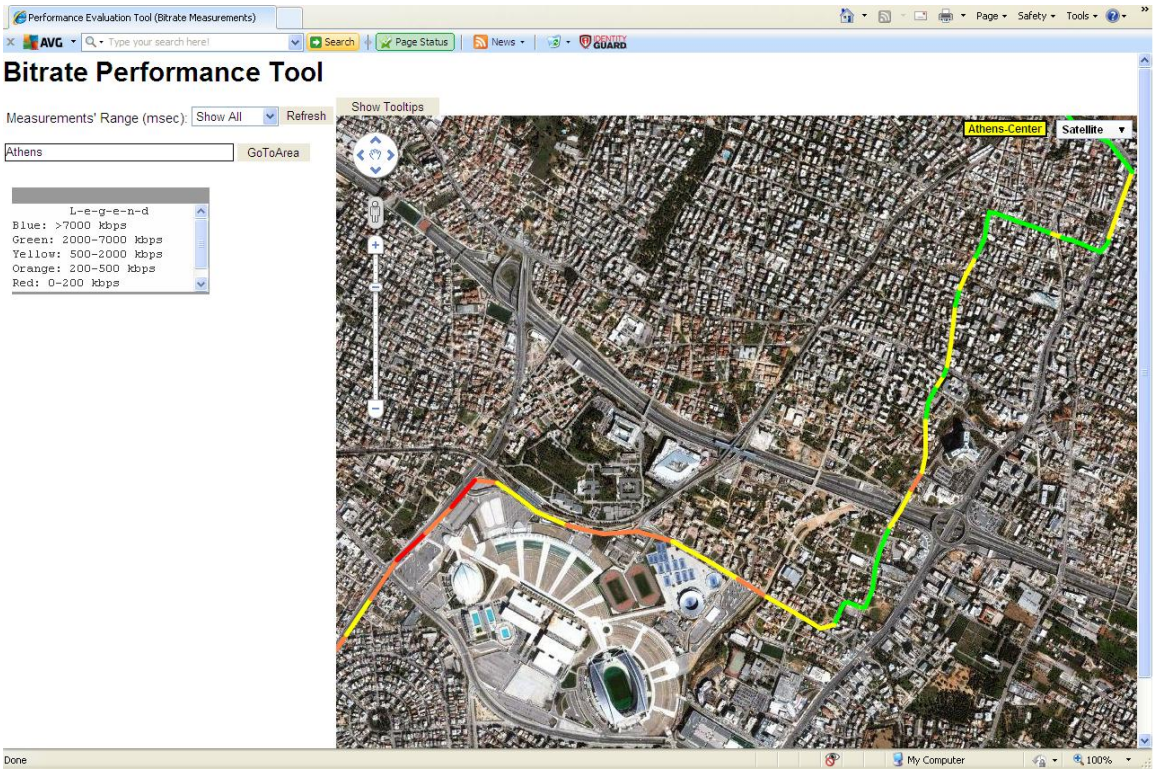


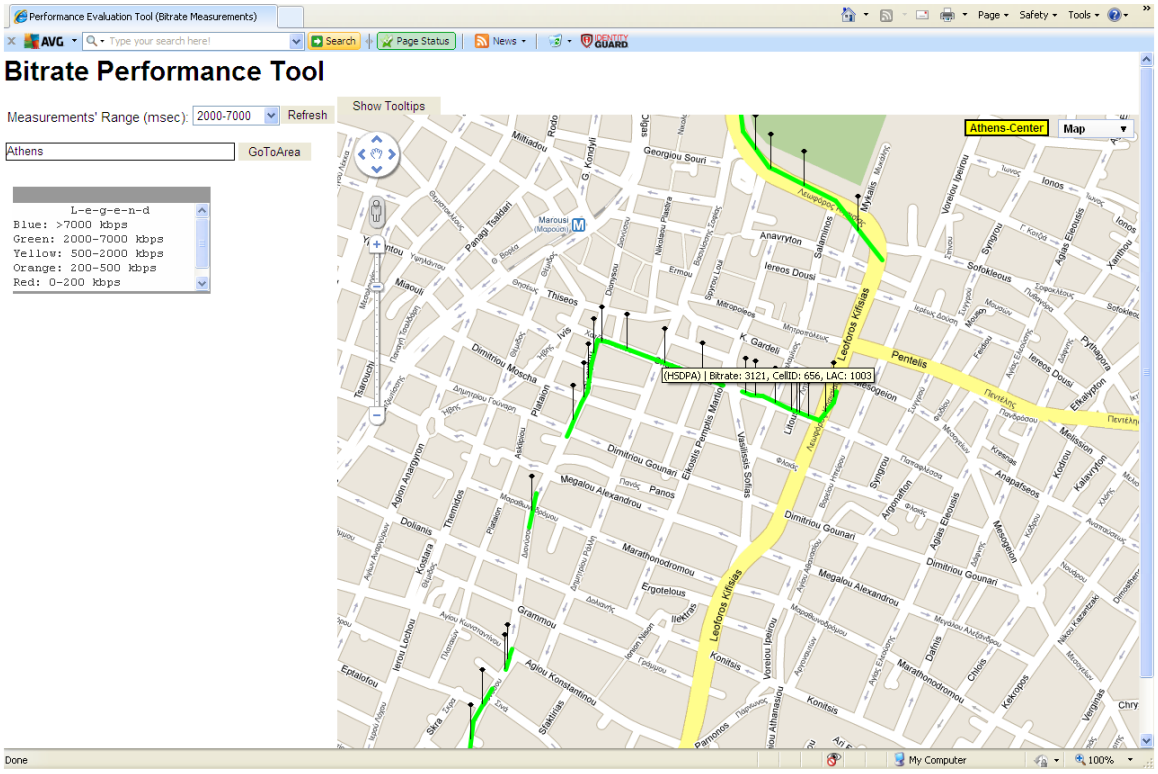
Το NetPerfTool αποτελείται από τα εξής αρχεία:

- Latency.html
- Download.html
- loadxmldoc.js
- myMarker.gif

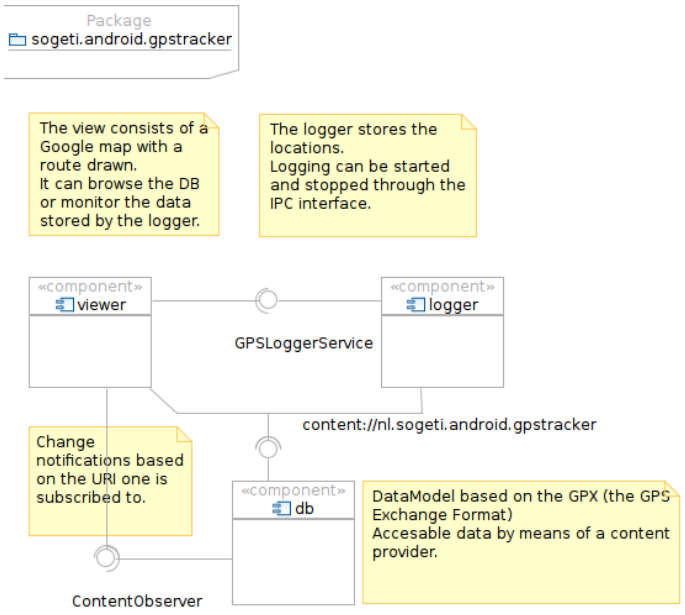
και είναι γραμμένο σε javascript. Η λογική είναι πως διαβάζουμε κατά DOM (Document Object Model) τα στοιχεία του xml αρχείου και για κάθε line εμφανίζουμε στο χάρτη το χρωματισμένο διάστημα. Το DOM μας βοηθά να θεωρήσουμε τη δενδρική μορφή του xml κειμένου με αντικειμενοστραφή τρόπο. Αν ο χρήστης πατήσει το «Show Tooltips», καλείται η συνάρτηση showMarkers() που προσθέτει στα διαστήματα τις πληροφορίες που αναφέραμε προηγουμένως. Ο πηγαίος κώδικας για το NetPerfTool βρίσκεται στο παράρτημα της εργασίας.

Ακολούθως παρουσιάζεται για πληρότητα και μια περίπτωση Download με επιλογή για εμφάνιση του χάρτη, όπως φαίνεται από το δορυφόρο (satellite).





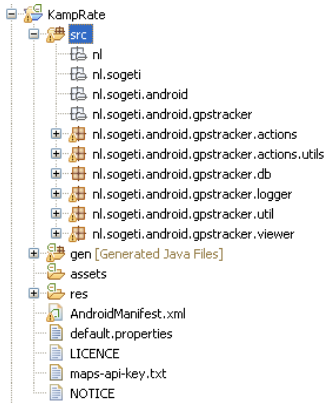
2.6 Διάρθρωση του κώδικα της εφαρμογής KampRate



Όπως φαίνεται και στο σχήμα, τα βασικά στοιχεία της εφαρμογής είναι ο logger, ο viewer και η db. Ο logger τρέχει στο background και είναι υπεύθυνος για τις μετρήσεις και την πυροδότητηση της αποθήκευσής τους. Η αποθήκευση γίνεται στην db, στην οποία και

συλλέγονται σε δομημένη μορφή όλες οι μετρήσεις και η πληροφορία που φέρουν. Ο viewer οπτικοποιεί τις μετρήσεις δημιουργώντας χρωματισμένες διαδρομές πάνω στον Google Map, εμφανίζοντας παράλληλα πληροφορίες γι' αυτές. Όσον αφορά τις μεταξύ τους συσχετίσεις και τη λογική σειρά διεργασιών, ο logger διεξάγει τις μετρήσεις και τις στέλνει στη db για να καταγραφούν. Ο viewer από την άλλη, παρακολουθεί για νέες εγγραφές στη db και, όταν διαπιστώσει πως συντελείται κάποια, ανανεώνει το χάρτη του και τα στοιχεία που εμφανίζει στην οθόνη.

Ο logger, ο viewer και η db αποτελούνται από κλάσεις που κάθε μία επιτελεί ένα ξεχωριστό ρόλο. Ακολούθως τις αναλύουμε:



- **actions**: Υλοποιούν συγκεκριμένες λειτουργίες της εφαρμογής, καλούμενες από τα βασικά μέρη της.

- **actions.utils**: Είναι υπεύθυνες για το σχηματισμό δομών και την εκτέλεση βοηθητικών υπολογισμών.

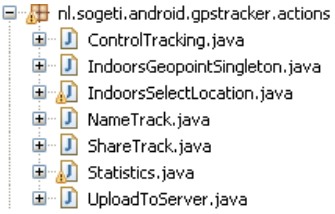
- **db**: Δημιουργούν τη βάση δεδομένων και υλοποιούν όλες τις λειτουργίες της (εγγραφή – διαγραφή διαδρομής, waypoints, σημειώσεις κ.α.)

- **logger**: Αποτελούν τον πυρήνα της εφαρμογής και υλοποιούν τον τρόπο διεξαγωγής μετρήσεων.

- **util**: Οι σταθερές (constants) της εφαρμογής, καθώς και κάποιες βοηθητικές κλάσεις.

- **viewer**: Είναι υπεύθυνες για ό,τι βλέπει ο χρήστης στην οθόνη. Το περιβάλλον που δημιουργούν είναι εν μέρει στατικό, όπως η δομή του interface και οι επιλογές – ρυθμίσεις του χρήστη, και εν μέρει δυναμικό, όπως ο χάρτης που ανανεώνεται με τις τελευταίες λαμβανόμενες μετρήσεις.

Εστιάζουμε τώρα στις κυριότερες κλάσεις:



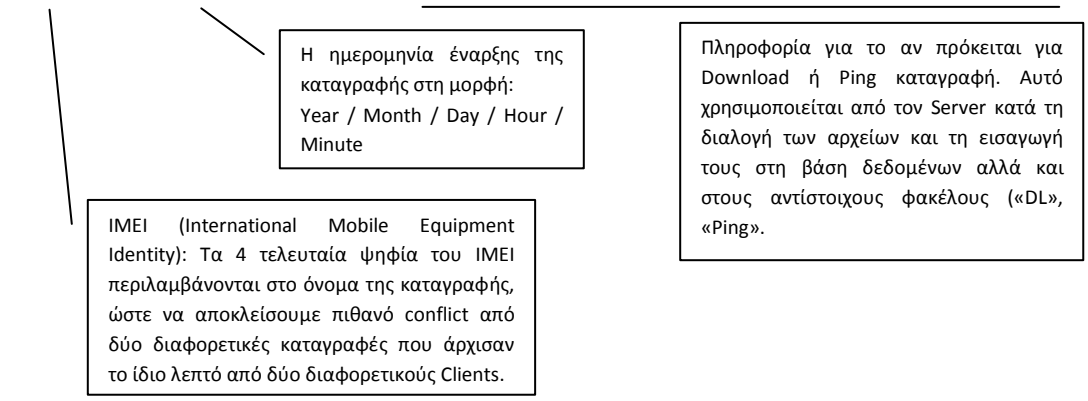
- **ControlTracking**: Πρόκειται για μια Activity που επιτρέπει στο χρήστη να ξεκινήσει ή να σταματήσει μια καταγραφή. Στην περίπτωση που ξεκινά μια καταγραφή ελέγχει αν πρόκειται για Indoors ή Outdoors και, αν διαπιστώσει το πρώτο, καλεί την IndoorsSelectLocation από την οποία περιμένει να λάβει μια απάντηση. Αν η απάντηση είναι πως ο χρήστης επέλεξε κάποια θέση στο χάρτη (fixed location), καλεί τη μέθοδο startGPSLogging() για να αρχίσει η καταγραφή. Διαφορετικά, δεν προχωρά σε κάποια ενέργεια. Αν έχει ήδη αρχίσει μια καταγραφή και ο χρήστης

επιλέξει να την σταματήσει, καλεί τη μέθοδο stopGPSLogging(). Όταν τελειώσει, επιστρέφει μήνυμα (μέσω intent) στην Activity που την κάλεσε (LoggerMap) με το loggerTrackId.

- **IndoorsGeopointSingleton**: Δημιουργήσαμε αυτό το Singleton προκειμένου να μπορούμε να ανακτούμε τη θέση (συντεταγμένες) που επέλεξε ο χρήστης στην IndoorsSelectLocation για την περίπτωση Indoors, μέσα από την GPSLoggerService. Το σχήμα αυτό μας παρέχει πρόσβαση στο αντικείμενο από οποιοδήποτε σημείο της εφαρμογής.

- **IndoorsSelectLocation:** Είναι μια Map Activity η οποία καλείται από την ControlTracking στην περίπτωση που ο χρήστης έχει επιλέξει να εκτελέσει Indoors μέτρηση. Επειδή σε εσωτερικό χώρο δεν είναι δυνατό να λάβουμε σήμα προσδιορισμού της θέσης μας από το δορυφόρο (GPS), γίνεται αναζήτηση της θέσης από τον Network Provider. Όταν ληφθεί η πρώτη ενημέρωση, εμφανίζεται με ένδειξη (μια κουκίδα) η θέση στο χάρτη. Αν αυτή ταυτίζεται με την πραγματική θέση του χρήστη, ο χρήστης «πατάει» το κουμπί «OK» και επιστρέφεται στην ControlTracking μήνυμα πετυχημένης επιλογής θέσης. Διαφορετικά, ο χρήστης έχει τη δυνατότητα να επιλέξει με το χέρι την ακριβή του θέση στο χάρτη. Για το σκοπό αυτό έχουμε υλοποιήσει ένα Overlay. Τέλος, ο χρήστης μπορεί να εγκαταλείψει την activity πατώντας «Cancel», οπότε και θα επιστραφεί στην IndoorsSelectLocation πως ο χρήστης δεν επέλεξε θέση στο χάρτη.

- **NameTrack:** Η ControlTracking επιστρέφει μήνυμα (intent) στη LoggerMap, η οποία, μόλις το λαμβάνει, ξεκινά την NameTrack Activity. Μέσω αυτής ο χρήστης ονοματίζει την καταγραφή. Η ονοματοδοσία έχει προβλεφθεί να είναι συγκεκριμένη, ώστε να εξυπηρετεί την ανάγκη στην πλευρά του Server για διαχείριση των xml αρχείων με βάση χαρακτηριστικά του ονόματός τους. Η δομή του ονόματος είναι ως εξής:



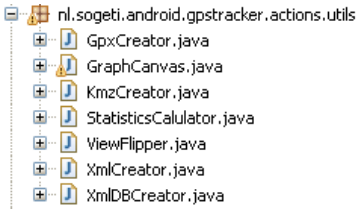
Βεβαίως, ο χρήστης μπορεί να εισάγει το όνομα που επιθυμεί για την καταγραφή, είναι όμως σημαντικό να μην αλλοιώσει την πληροφορία που αφορά το είδος της μέτρησης («DL» και «PING» αντίστοιχα) και τον χαρακτήρα «-» που το διαχωρίζει από την πληροφορία για το IMEI και τη χρονική στιγμή έναρξης της καταγραφής.

Όταν ο χρήστης πατήσει «OK», θα γίνει ενημέρωση του ονόματος της καταγραφής στη βάση δεδομένων και θα τερματίσει η Activity.

- **ShareTrack**: Μέσω αυτής της Activity ο χρήστης επιλέγει σε ποια μορφή επιθυμεί να αποθηκεύσει μια καταγραφή. Μετά την επιλογή δημιουργείται ένας XmlCreator που θα μετασχηματίσει τη δομή της database στην επιθυμητή, για την καταγραφή που έχουμε «φορτώσει». Αναμένει να διεκπεραιωθεί αυτή η διαδικασία, ώστε να ξεκινήσει έπειτα την UploadToServer, απ' όπου ο χρήστης δύναται να «ανεβάσει» την καταγραφή στον κεντρικό Server. Τέλος, από την UploadToServer περιμένει μια ειδοποίηση (intent) ώστε να τερματίσει η διαδικασία του Share Track.

- **Statistics**: Είναι μια Activity, υπεύθυνη για την εμφάνιση των στατιστικών στοιχείων. Δημιουργεί έναν StatisticsCalculator από τον οποίο λαμβάνει τις διάφορες τιμές που χρειάζονται για να εμφανίσει τα στατιστικά στο πάνω μέρος της οθόνης και τα γραφήματα στο κάτω μέρος. Επίσης, υλοποιεί ένα menu, από το οποίο ο χρήστης μπορεί να επιλέξει το είδος του γραφήματος που επιθυμεί να εμφανιστεί (Graph Type), να «φορτώσει» μια άλλη καταγραφή (List tracks) ή να αποθηκεύσει την καταγραφή και να την αποστείλει στον κεντρικό Server (Share track).

- **UploadToServer**: Είναι μια activity που ενεργοποιείται από την ShareTrack, αφότου έχει αποθηκευτεί το xml αρχείο της καταγραφής στην SD Card της κινητής συσκευής. Υλοποιεί δύο διαδοχικά alert dialogs. Το πρώτο ερωτά το χρήστη αν επιθυμεί να «ανεβάσει» την καταγραφή στον Server. Σε περίπτωση θετικής απάντησης ξεκινά το δεύτερο alert dialog που τον προτρέπει να εισάγει την IP Address του Server και να πατήσει «OK» για να ξεκινήσει η αποστολή. Για το σκοπό αυτό ανοίγει μια HttpURLConnection, συνδεόμενη στο `http://ServerIP:80/handle_upload.php`
  Έπειτα, ανοίγει ένα FileInputStream για να διαβάσει τη ροή δεδομένων του xml αρχείου και ένα DataOutputStream για να στείλει τη ροή στον Server. Η διαδικασία αυτή πραγματώνεται με HTTP POST. Κατά τη διάρκεια της αποστολής εμφανίζεται στην οθόνη ένδειξη «loading». Όταν ολοκληρωθεί η διαδικασία, είτε επιτυχώς είτε ανεπιτυχώς, εμφανίζεται το αντίστοιχο μήνυμα στην οθόνη και τερματίζεται η activity.



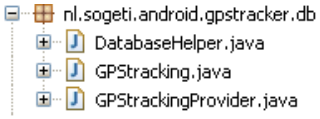
- **GpxCreator**: Ένας XmlCreator που καλείται από την ShareTrack προκειμένου να μετασχηματίσει τη δομή της καταγραφής στην database σε GPX (GPS Exchange Format) δομή. Δεν έχει υλοποιηθεί ο πλήρης μετασχηματισμός.

- **GraphCanvas**: Πρόκειται για ένα View που υλοποιεί το σχεδιασμό των γραφημάτων στο κάτω μέρος της οθόνης των Statistics.

- **KmzCreator**: Ένας XmlCreator που καλείται από την ShareTrack προκειμένου να μετασχηματίσει τη δομή της καταγραφής στην database σε Kmz (Keyhole Markup Language) δομή. Δεν έχει υλοποιηθεί ο πλήρης μετασχηματισμός.

- **StatisticsCalculator**: Ένα σύνολο μεθόδων για την εκτέλεση των υπολογισμών που απαιτούνται για την εμφάνιση των στατιστικών. Χρησιμοποιεί τις εγγραφές της βάσης δεδομένων, μέσω queries, για να υπολογίσει average, minimum και maximum τιμές.

- **XmlDBCreator**: Ένας XmlCreator που καλείται από την ShareTrack προκειμένου να μετασχηματίσει τη δομή της καταγραφής στην database σε μια Xml δομή, η οποία έχει σχεδιαστεί κατάλληλα, ώστε να εξασφαλίζει την εισαγωγή στη βάση δεδομένων του Server και βέβαια την πλήρη και σαφή παρουσίαση των μετρήσεων των επιμέρους τμημάτων.



- **DatabaseHelper**: Αποτελεί βοηθητική κλάση (SQLiteOpenHelper) για τη διαχείριση της βάσης δεδομένων. Είναι υπεύθυνση για τη δημιουργία της βάσης, για την εγγραφή μιας νέας διαδρομής ή ενός νέου waypoint, για τη διαγραφή μιας καταγραφής κ.α. Οι μέθοδοί της καλούνται από την GPSTrackingProvider με τις παραμέτρους που θα χρησιμοποιηθούν για την εκτέλεση μια λειτουργίας ή την εισαγωγή τους στη βάση.

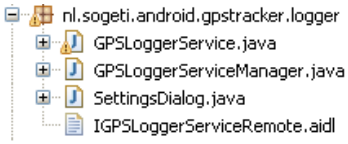
- **GPSTracking**: Περιλαμβάνει όλες τις σταθερές που χρησιμοποιούνται για τη δημιουργία της βάσης, όπως τα ονόματα, τους τύπους των πεδίων και τις δηλώσεις δημιουργίας (creation statements). Για την επέκταση της βάσης, πρέπει κανείς να αλλάξει αυτές τις δηλώσεις προσθέτοντας τα αντίστοιχα πεδία και να επαναεγκαταστήσει την εφαρμογή, ώστε να δημιουργηθούν οι νέοι πίνακες με τα ανανεωμένα πεδία.

- **GPSTrackingProvider**: Πρόκειται για έναν Content Provider, ο οποίος ενθυλακώνει δεδομένα καθιστώντας τα προσβάσιμα από κάθε σημείο της εφαρμογής μέσω του Content Resolver Interface. Μία διαδρομή είναι ένας «δρόμος» από την αρχή ως το τέλος. Οι GPS θέσεις που λαμβάνονται προσδιορίζουν τα waypoints. Τα waypoints που αποθηκεύονται σε μια σειρά καθιστούν ένα segment. Ένας «δρόμος» απαρτίζεται από ένα ή περισσότερα segments. Μερικά παραδείγματα από τα χρησιμοποιούμενα URI είναι:

  `content://nl.sogeti.android.gpstracker/tracks/2` -> Θα επιστρέψει μία εγγραφή (σε μια σειρά), την καταγραφή με ID = 2.

  `content://nl.sogeti.android.gpstracker/tracks/2/segments/1/waypoints/52` -> Θα επιστρέψει από το πρώτο segment της δεύτερης καταγραφής το waypoint υπ'αριθμόν 52.

  Ο Content Provider, λοιπόν, αντιστοιχίζει URI schemes με δεδομένα, ώστε να είναι δυνατή η ανάκτησή τους από οποιοδήποτε σημείο της εφαρμογής.

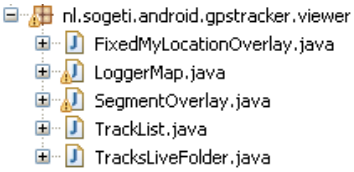


- **GPSLoggerService**: Είναι η υπηρεσία που τρέχει στο background και στην οποία έχουν υλοποιηθεί τα UDP Download και UDP Ping. Η καταγραφή ξεκινά, όταν κληθεί η startLogging() από την ControlTracking και σταματά, όταν κληθεί η stopLogging(). Οι μετρήσεις αποθηκεύονται όταν καλείται η storeLocation(). Επίσης, «ακούει» για αλλαγές στη θέση με βάση λήψεις από το δορυφόρο, αλλά και για αλλαγές του Cell ID, ώστε να αποθηκεύει και να εμφανίζει (η LoggerMap) στο κάτω μέρος της οθόνης το Cell ID και το LAC. Ακόμη, ανακτά τις ρυθμίσεις που έχει κάνει ο χρήστης στα settings για την τρέχουσα καταγραφή και τις χρησιμοποιεί για τους υπολογισμούς που εκτελεί. Είναι ο πυρήνας της εφαρμογής.

- **GPSLoggerServiceManager**: Είναι μια κλάση με βοηθητικές μεθόδους αλληλεπίδρασης με την GPSLoggerService.

- **SettingsDialog**: Πρόκειται για την Preference Activity της εφαρμογής μέσω της οποίας ο χρήστης βλέπει την οθόνη των ρυθμίσεων (settings).

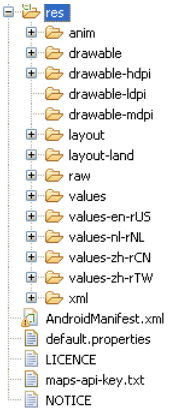
- **Constants**: Είναι η κλάση που περιέχει όλες εκείνες τις σταθερές της εφαρμογής που υπεισέρχονται σε διάφορα σημεία του κώδικα. Έχει υλοποιηθεί για να γίνονται εύκολα οι αλλαγές και οι τροποποιήσεις.



- **LoggerMap**: Είναι η κύρια Map Activity της εφαρμογής και υλοποιεί την κεντρική οθόνη που βλέπει ο χρήστης. Περιέχει το κύριο μενού της εφαρμογής, από το οποίο ο χρήστης επιλέγει τις διάφορες λειτουργίες που επιθυμεί να εκτελέσει.

Παρατηρεί τις νέες εγγραφές (ContentObserver) που συντελούνται κατά τη διάρκεια των μετρήσεων και ανανεώνει το χάρτη που βλέπει ο χρήστης με το τελευταίο διάστημα που προστίθεται κάθε φορά δημιουργώντας γι' αυτό το σκοπό Segment Overlays. Έτσι, σχηματίζεται διαδοχικά μια διαδρομή. Παράλληλα, εμφανίζει πληροφορίες (bit rate ή Average Latency, Cell ID, LAC, RSSI) στην οθόνη. Ειδικά για το RSSI έχει υλοποιηθεί ένας ξεχωριστός TelephonyManager που εμφανίζει τη νέα τιμή, όταν υπάρξει αλλαγή. Αντίθετα, τα Cell ID και LAC είναι οι τιμές που αποσπάστηκαν (GPSLoggerService), όταν γινόταν η αποθήκευση του σημείου, στο τέλος του διαστήματος. Τέλος, συντονίζει τα επιμέρους συστατικά της εφαρμογής, αφού αποτελεί το κέντρο ελέγχου του χρήστη.

- **SegmentOverlay**: Είναι ένα Overlay που σχεδιάζει πάνω στο χάρτη τα μετρούμενα διαστήματα με τις σωστές συντεταγμένες και τον κατάλληλο χρωματισμό.



- **Drawable**: τα διάφορα εικονίδια της εφαρμογής

- **Layout**: xml αρχεία που συνθέτουν το GUI της εφαρμογής

- **Values**: Σταθερές τιμές που αποσπώνται σε διάφορα σημεία του κώδικα

- **AndroidManifest.xml**: Περιλαμβάνει τη δήλωση των Activities, Services, Content Providers και Broadcast Receivers που έχουν υλοποιηθεί, καθώς και τα δικαιώματα (permissions) που είναι αναγκαία για την πρόσβαση της εφαρμογής στις λειτουργίες της κινητής συσκευής.

Βιβλιογραφία

[1] Α. Κανάτας, Φ. Κωνσταντίνου, Γ. Πάντος, «Συστήματα Κινητών Επικοινωνιών», Εκδόσεις Παπασωτηρίου, 2008

[2] Andrew S. Tanenbaum, «Δίκτυα Υπολογιστών», Εκδόσεις Κλειδάριθμος, 2003

[3] Kaaranen H., Ahtiainen A., Laitinen L., Naghian S., Niemi V., «UMTS Networks - Architecture, Mobility and Services», 2005

[4] ITU, «The World in 2010» http://www.itu.int/ITU-D/ict/material/FactsFigures2010.pdf

[5] Α. Ανδρονικάκης, «Σύστημα Προσδιορισμού Θέσης Σταθμών Βάσης σε Δίκτυα Κινητών Επικοινωνιών», 2010

[6] Ε. Αυγέρη, Μ. Καλλένου, «Μετρήσεις και Ανάλυση Ποιότητας Υπηρεσιών σε Ευρυζωνικά Δίκτυα Κινητών Επικοινωνιών», 2009

[7] Mark Murphy, «Beginning Android» Apress, 2009

[8] Reto Meier, «Professional Android 2 Application Development» Wiley Publishing, 2010

[9] Γ.Ι. Στασινόπουλος, «Διαδίκτυο και Εφαρμογές», 2008

[10] Luke Welling, Laura Thompson, «PHP and MySQL Web Development (4th Edition)», Pearson Education, 2009

[11] Y. Daniel Liang, «Introduction to Java Programming, Comprehensive (8th Edition), 2010

[12] Joyce Farrell, «Java Programming», 2011

[13] James F. Kurose, Keith W. Ross, «Computer Networking: A Top-Down Approach Featuring the Internet (5th edition)», 2009

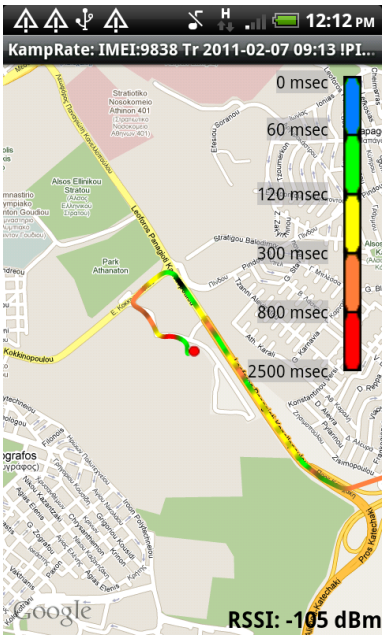
[14] David Flanagan, «JavaScript: The Definitive Guide», 2006

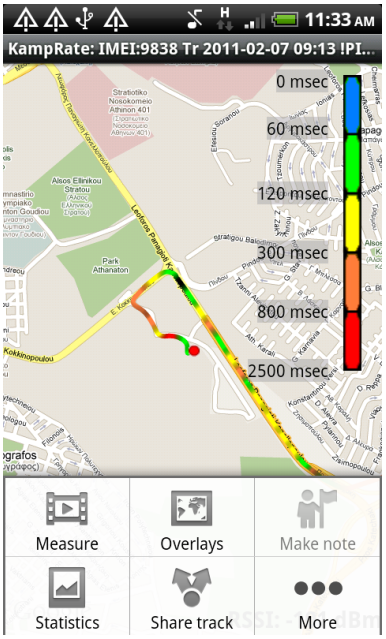
Κεφάλαιο 3

Παρουσίαση Εφαρμογής KampRate – Αποτελέσματα Δοκιμών

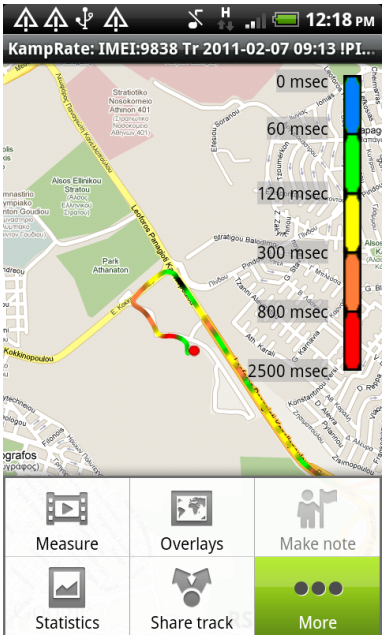
3.1 Η εφαρμογή KampRate

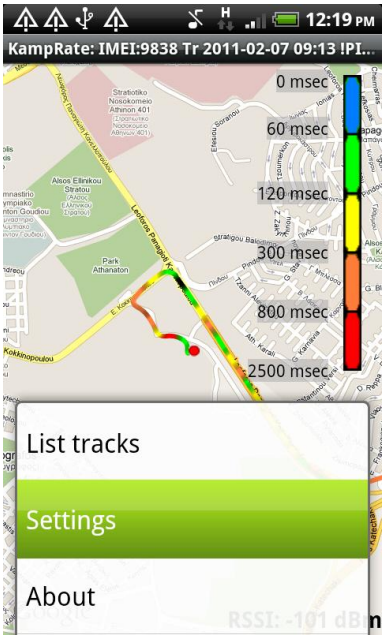
Σε αυτή την ενότητα θα παρουσιαστεί η εφαρμογή KampRate και θα αναφερθούν οι επιμέρους δυνατότητές της. Αρχικά, η κεντρική οθόνη της εφαρμογής που βλέπει ο χρήστης, όταν «τρέξει» την εφαρμογή, φαίνεται παρακάτω. Στο χάρτη έχει «φορτωθεί» η τελευταία καταγεγραμμένη διαδρομή. Πατώντας το πλήκτρο «menu» του κινητού, εμφανίζεται το κεντρικό μενού.



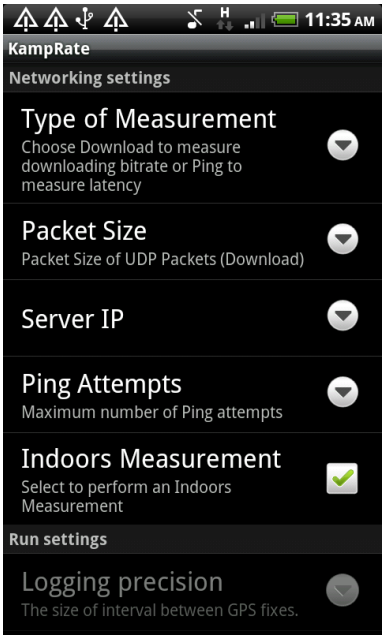


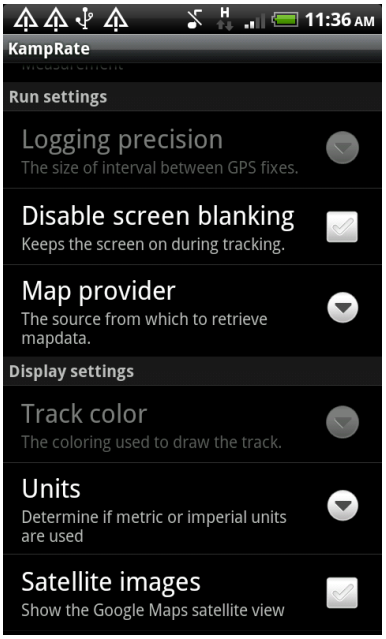
Για να ρυθμίσει τα settings ο χρήστης, θα πρέπει να πατήσει «More» και έπειτα «Settings».

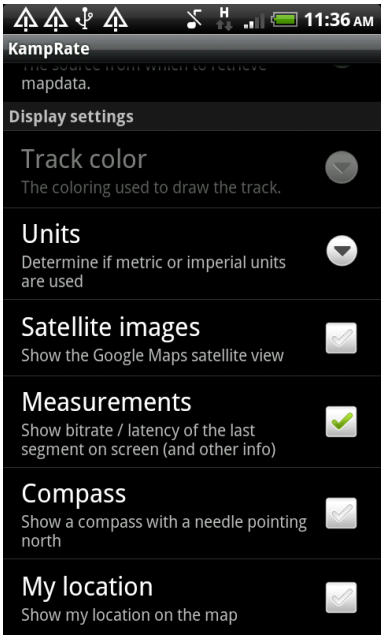




Τα settings φαίνονται στις επόμενες εικόνες.







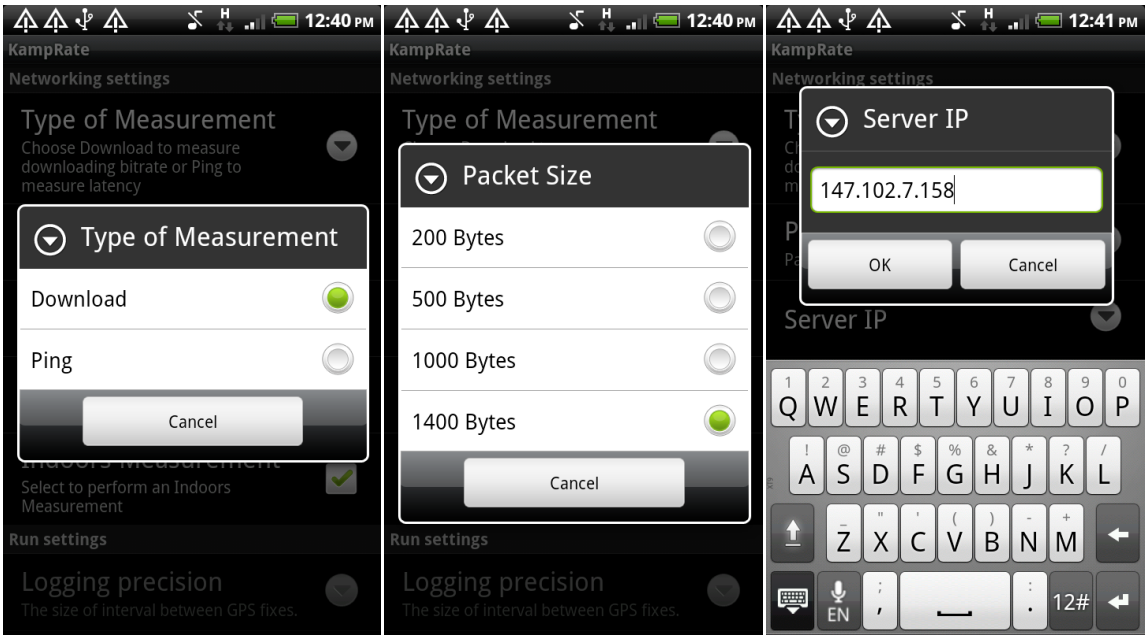
Για κάθε μία επιλογή υπάρχει ένδειξη για το τι ακριβώς ρυθμίζει και πού χρησιμεύει. Διακρίνονται σε τρεις κατηγορίες:

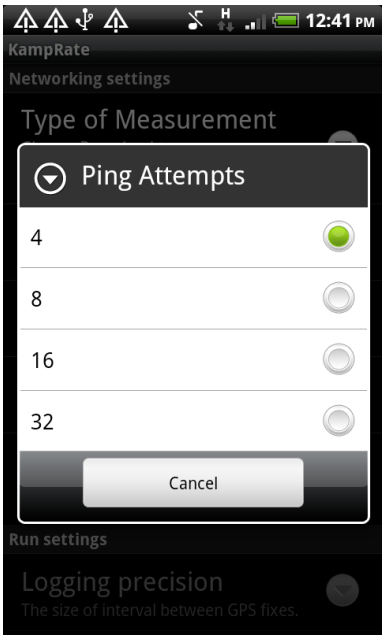
- **Networking Settings**: Είναι ρυθμίσεις που αφορούν τη διενέργεια των μετρήσεων του UDP Download και του UDP Ping. Στο «Type of Measurement» ο χρήστης επιλέγει αν θα πραγματοποιήσει «Download» ή «Ping» μέτρηση. Στο «Packet Size» επιλέγει μέγεθος πακέτου για την περίπτωση του «Download». Στο «Server IP» εισάγει την IP Address

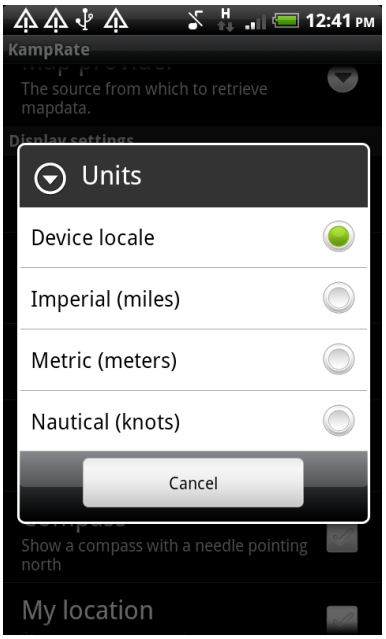
του Server ή αφήνει την default ως έχει. Στο «Ping Attempts» επιλέγει το μέγιστο αριθμό προσπαθειών Ping που επιχειρούνται στο διάστημα των 10 sec. Τέλος, «τσεκάροντας» την επιλογή «Indoors Measurement» πραγματοποιείται Indoors μέτρηση, διαφορετικά πραγματοποιείται Outdoors μέτρηση.

- **Run Settings**: Αφορούν λειτουργίες του τηλεφώνου κατά τη διάρκεια που εκτελούνται οι μετρήσεις. Η «Disable Screen Blanking» κρατάει αναμμένη την οθόνη κατά τη διάρκεια των μετρήσεων.

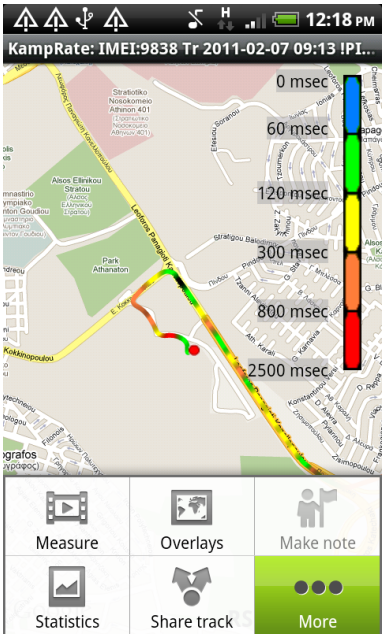
- **Display Settings**: Είναι επιλογές που αφορούν την οθόνη που βλέπει ο χρήστης κατά τη διάρκεια των μετρήσεων. Με το «Units» επιλέγει μετρική μονάδα, με το «Satellite images» ρυθμίζει αν επιθυμεί να βλέπει τους χάρτες όπως από τον δορυφόρο ή όπως από το GPS, με το «Measurements» επιλέγει αν θα εμφανίζονται στην κεντρική οθόνη τα bit rate (ή Average Latency), Cell ID και LAC, με το «Compass» εμφανίζεται μια μικρή πυξίδα στο πάνω μέρος της οθόνης και τέλος, με το «My location» επισημαίνεται η θέση της κινητής συσκευής στο χάρτη κάθε στιγμή.

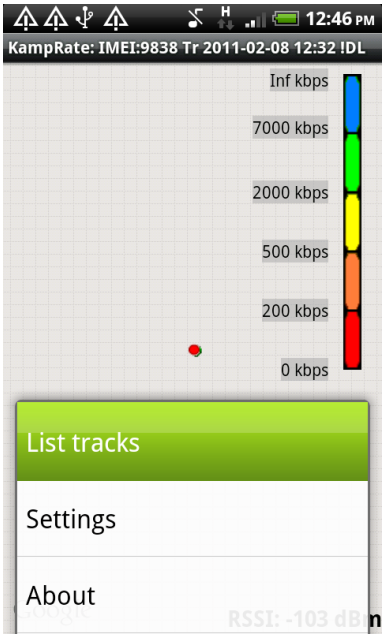




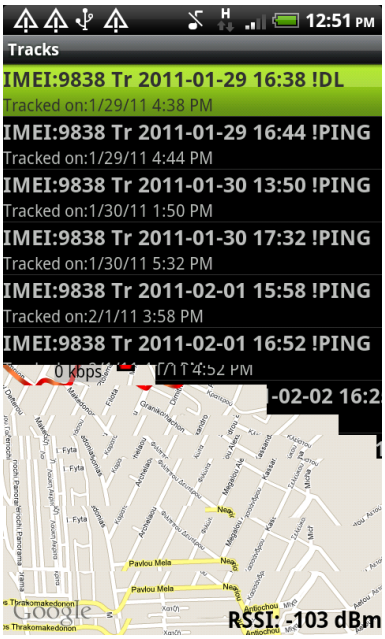


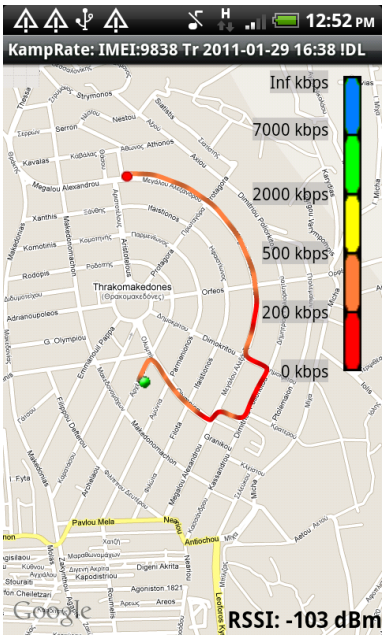
Προκειμένου ο χρήστης να δει όλες τις καταγραφές που έχει κάνει, επιλέγει «List tracks» από το «More» του κεντρικού μενού.



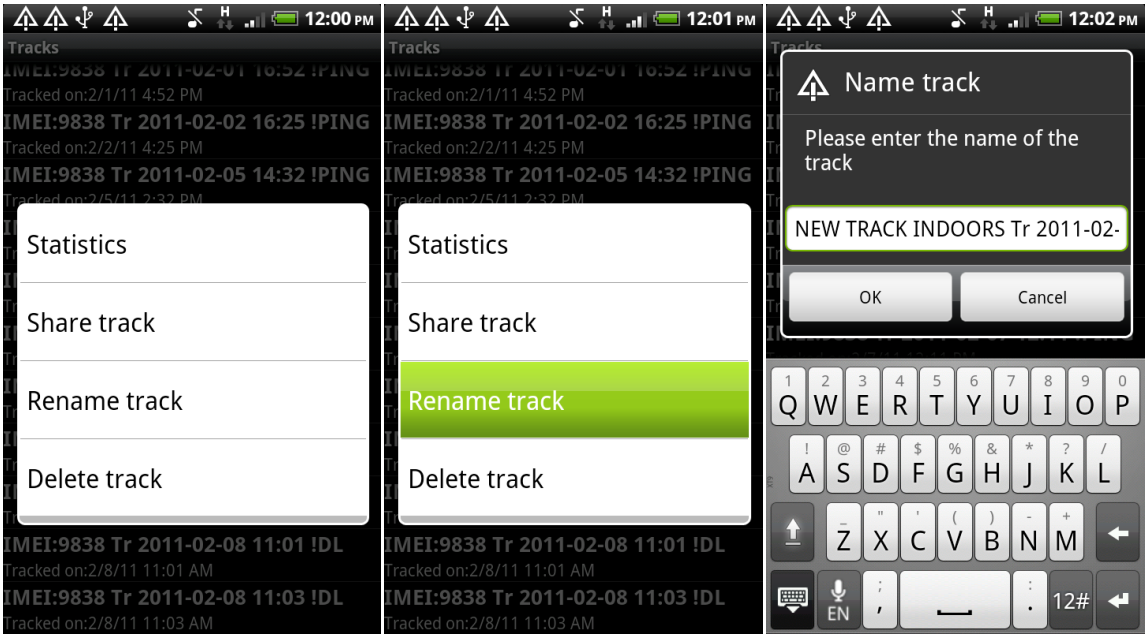


Εμφανίζονται τότε όλες οι καταγραφές του με την ονοματολογία που έχουμε αναλύσει σε προηγούμενη ενότητα. Επιλέγοντας μια καταγραφή, αυτή «φορτώνει» στο χάρτη και στα statistics.



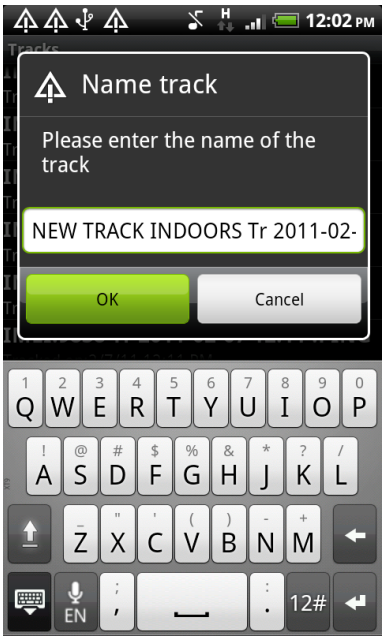


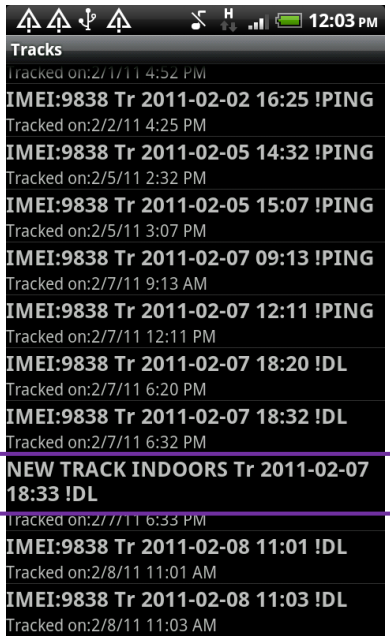
Μπορεί, όμως, να εκτελέσει και μια σειρά ενεργειών επί των καταγραφών πατώντας παρατεταμένα για μερικά δευτερόλεπτα πάνω σε μια καταγραφή.

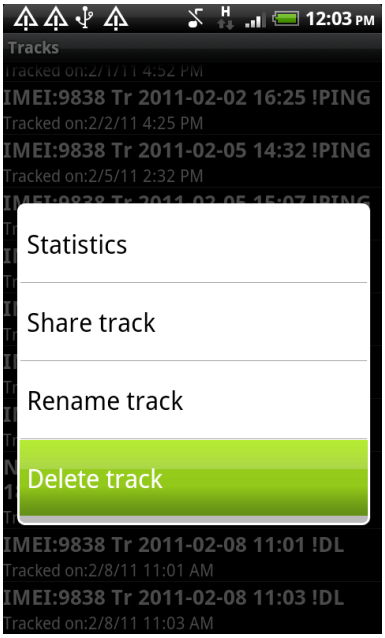


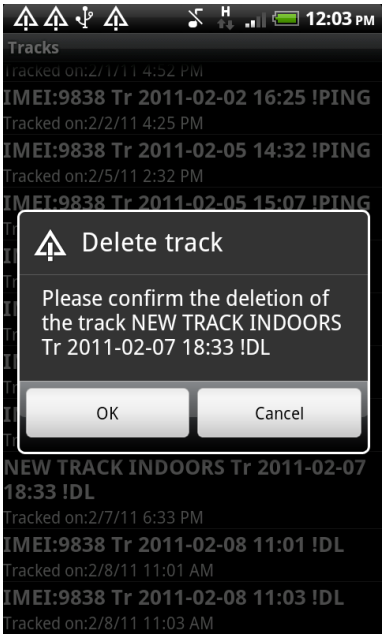
Πατώντας «Statistics» εμφανίζονται τα στατιστικά στοιχεία της καταγραφής. Είναι το ίδιο με το να είχε φορτώσει τη διαδρομή και να επέλεγε «Statistics» από το κεντρικό μενού. Παρομοίως με το «Share track» εκτελεί την ίδια ενέργεια με το «Share track» του κεντρικού μενού. Με το «Rename track» έχει τη δυνατότητα να μετονομάσει την καταγραφή του, με τους κινδύνους που ελλοχεύουν αν αλλάξει τη δομή του ονόματος, όπως είχαμε επισημάνει και σε

προηγούμενη ενότητα. Ακόμη, με το «Delete track» εκτελείται πράξη διαγραφής στην database και η καταγραφή χάνεται οριστικά.



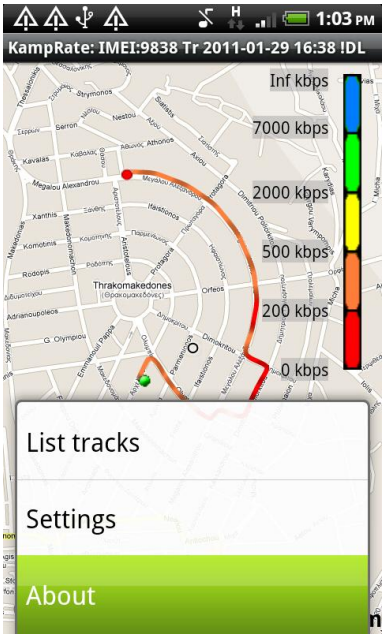


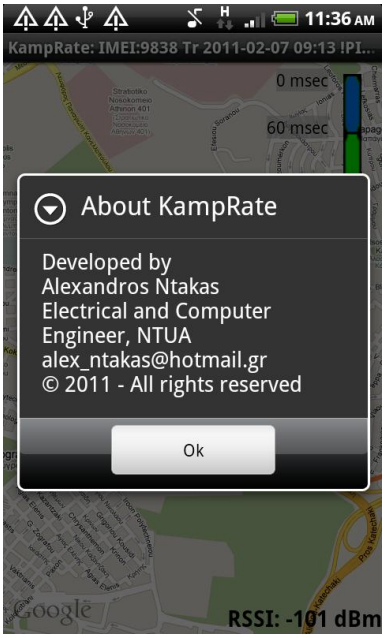




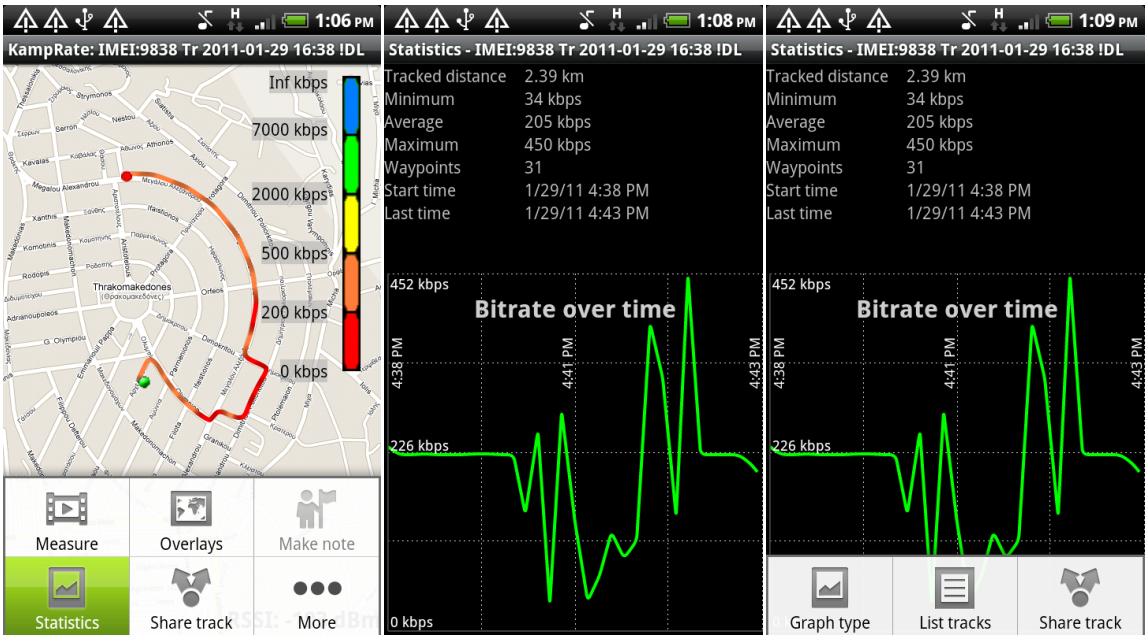
Τέλος, πατώντας το κουμπί «menu» του κινητού εμφανίζεται επιλογή για αναζήτηση μιας καταγραφής.

Για να ολοκληρώσουμε τις επιλογές του tab «More», πατώντας «About» ο χρήστης μπορεί να δει τα στοιχεία του γράφοντος που ανέπτυξε την εφαρμογή.



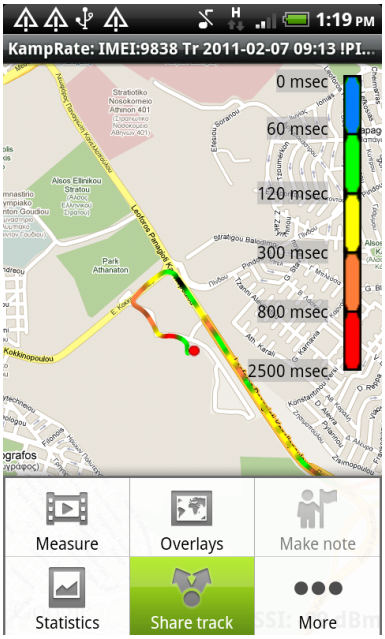


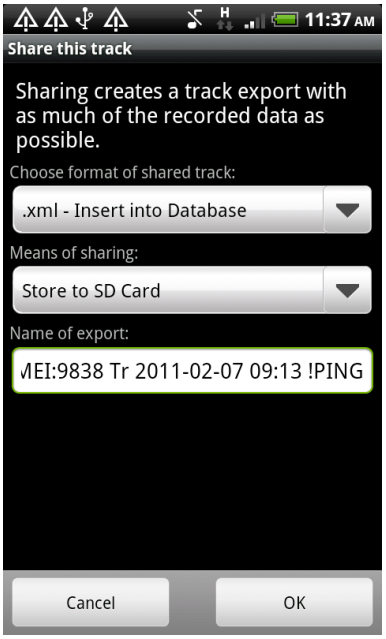
Έστω τώρα ότι ο χρήστης επιθυμεί να δει τα στατιστικά μιας καταγραφής που έχει «φορτώσει». Θα πρέπει τότε να πατήσει το «Statistics» του κεντρικού μενού και η οθόνη των στατιστικών θα εμφανιστεί. Πατώντας το πλήκτρο «menu» του κινητού εμφανίζονται κάποιες επιλογές.



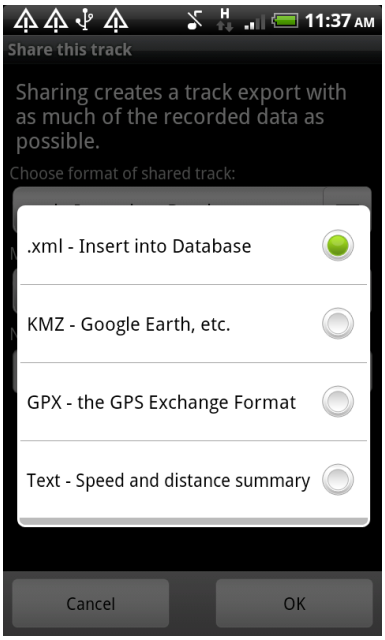
Με το «Graph Type» ο χρήστης μπορεί να επιλέξει αν θέλει να εμφανιστεί το διάγραμμα που δείχνει το μετρούμενο μέγεθος (bit rate, Average Latency) σε συνάρτηση με το χρόνο ή σε συνάρτηση με το διάστημα. Την ίδια επιλογή μπορεί να κάνει ακουμπώντας το δάχτυλο του στο διάγραμμα και σέρνοντάς το προς τα αριστερά ή τα δεξιά, οπότε εμφανίζεται το επόμενο διάγραμμα. Οι άλλες δύο επιλογές («List tracks» και «Share track») κάνουν ακριβώς ό,τι και αυτές του κεντρικού μενού.

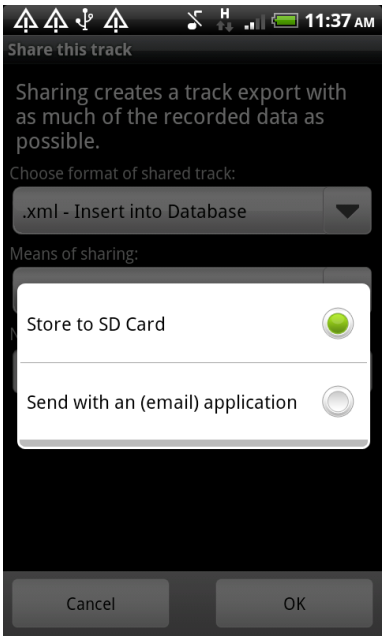
Για να αποθηκεύσει την καταγραφή στην SD Card σε κάποια από τις αναλυθείσες δομές (xml, gpx, kmz) ο χρήστης επιλέγει το «Share track» από το κεντρικό μενού, έχοντας βέβαια πρώτα «φορτώσει» τη διαδρομή που τον ενδιαφέρει.





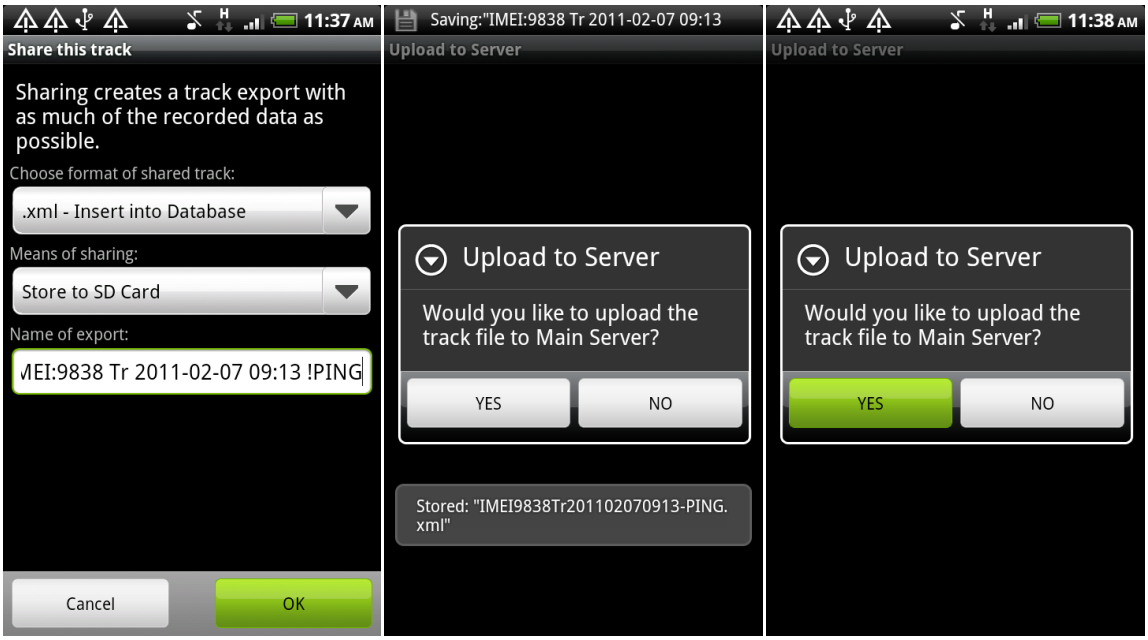
Επιλέγει την επιθυμητή δομή και αν θα αποθηκεύσει την καταγραφή στην SD Card του κινητού ή αν θα την αποστείλει μέσω e-mail.



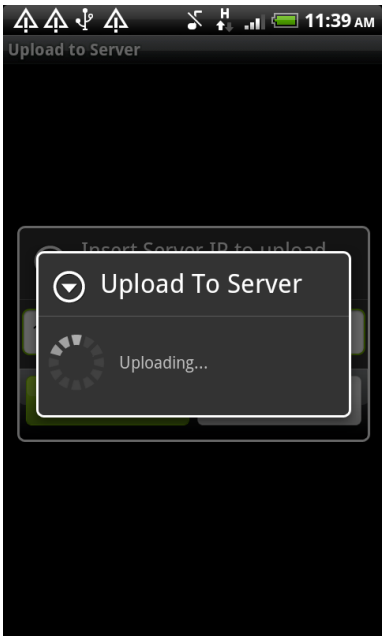


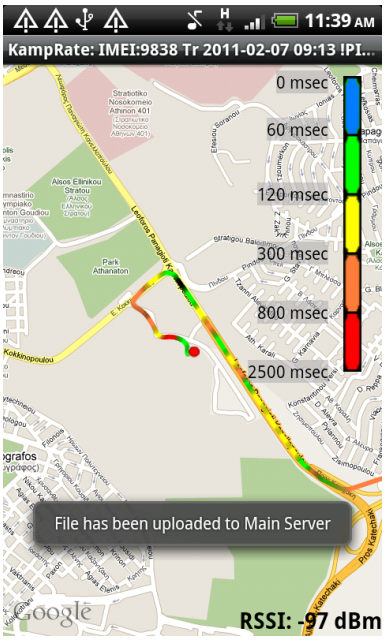
Έστω ότι επιλέγει να αποθηκεύσει την καταγραφή στην SD Card με την xml δομή που έχουμε σχεδιάσει (αυτή η επιλογή είναι και η συνήθης). Πατάει «OK» και λαμβάνει ειδοποίηση για την

επιτυχημένη αποθήκευση. Παράλληλα ερωτάται αν επιθυμεί να «ανεβάσει» την καταγραφή σε κάποιον Server.

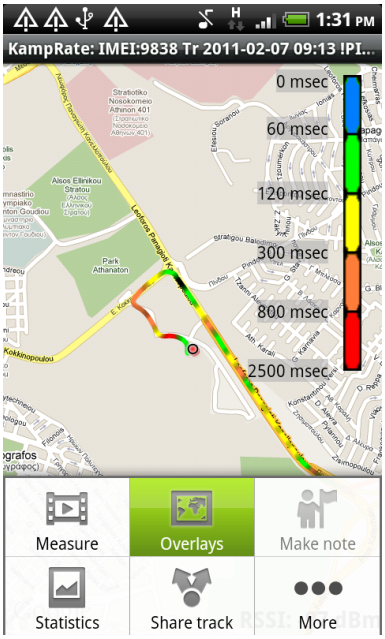


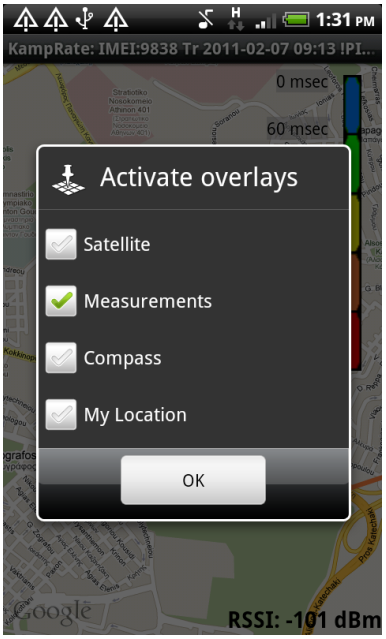
Ο χρήστης επιλέγει «YES» και ξεκινά η διαδικάσια του uploading. Κατά τη διάρκειά της βλέπει στην οθόνη την ένδειξη «Uploading» και όταν ολοκληρωθεί, λαμβάνει μήνυμα επιτυχίας ή αποτυχίας.



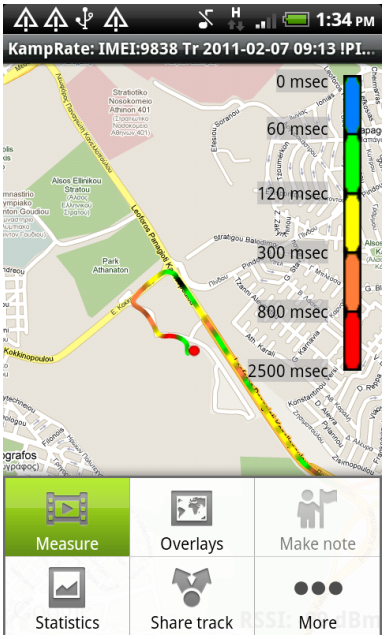


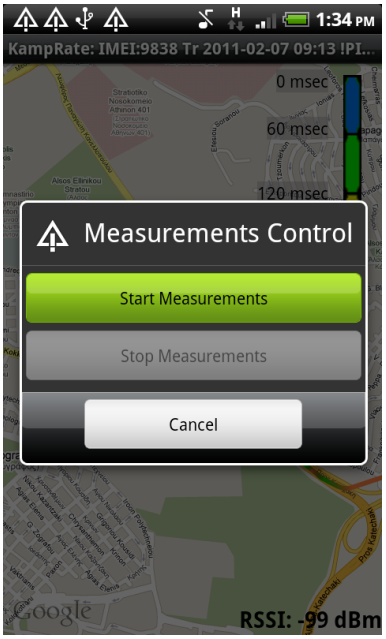
Με την επιλογή «Overlays» στο κεντρικό μενού, ο χρήστης μπορεί να κάνει κάποιες ρυθμίσεις εύκολα και γρήγορα με αυτόματη ανανέωση της κεντρικής οθόνης που βλέπει με βάση τις τελευταίες επιλογές του. Οι ρυθμίσεις αυτές μπορούν να γίνουν και από τα «Settings».





Τελειώνοντας αυτή την περιήγηση στο interface και στις επιλογές του χρήστη, η έναρξη μιας νέας καταγραφής σηματοδοτείται από την επιλογή «Start Measurements» στο «Measure» του κεντρικού μενού.



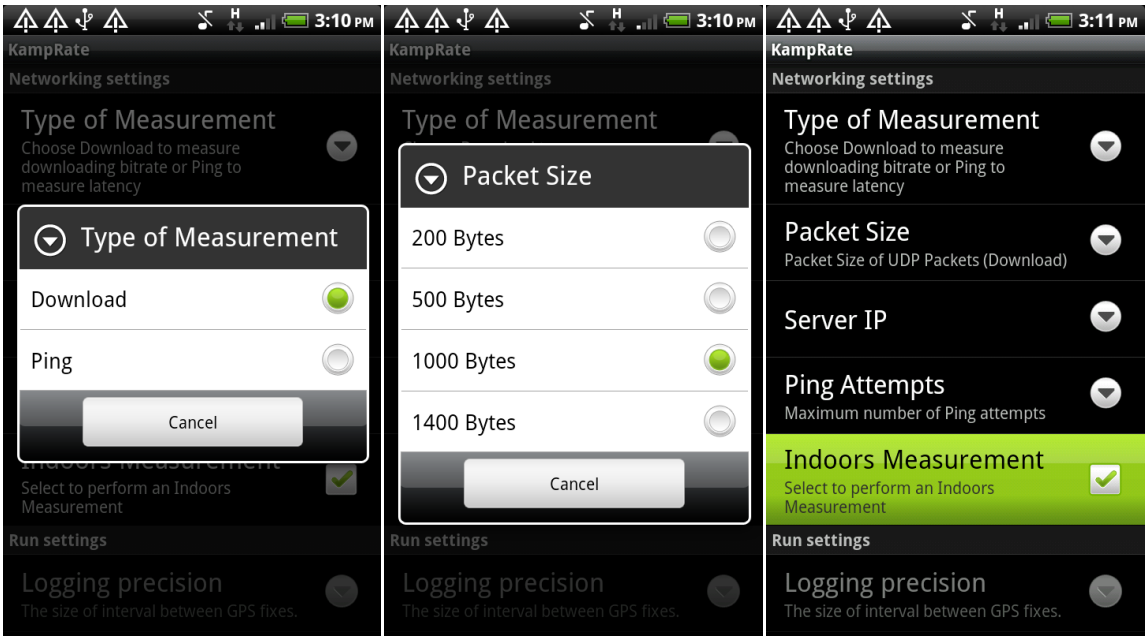


3.2 Αποτελέσματα δοκιμών

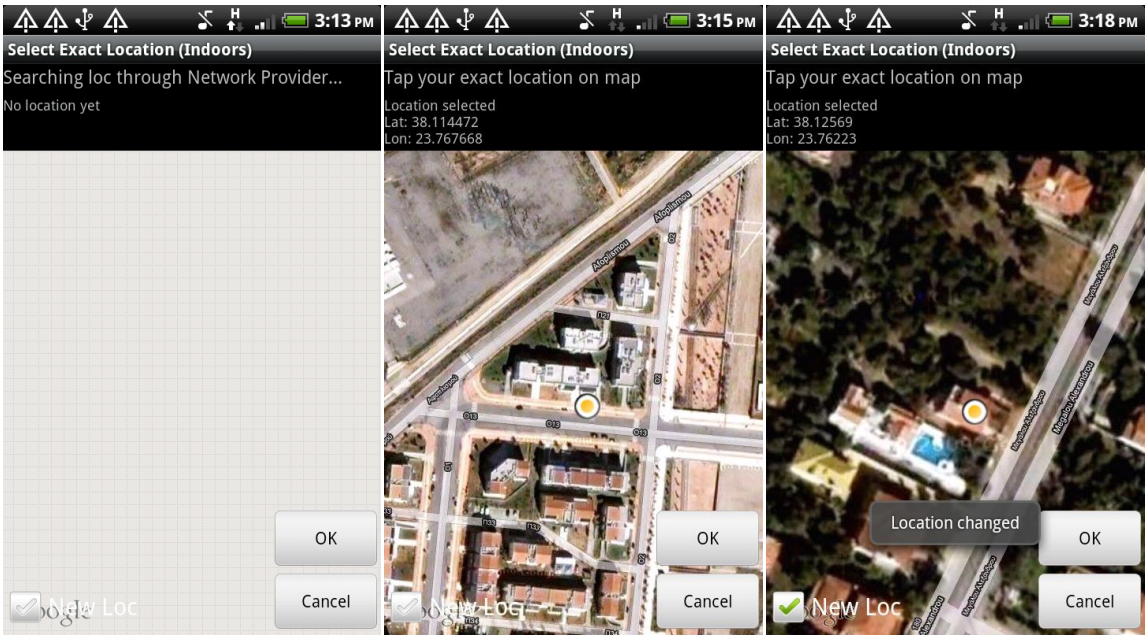
Στο KampRate, ο χρήστης μπορεί να εκτελέσει τέσσερα διαφορετικά είδη μετρήσεων: Download Indoors, Download Outdoors, Ping Indoors και Ping Outdoors. Στις σελίδες που ακολουθούν παρουσιάζουμε μία δοκιμή για κάθε είδος μετρήσεων. Επιπρόσθετα, καταλήγουμε σε κάποια συμπεράσματα που προέκυψαν από τη διεξαγωγή πολλών μετρήσεων.

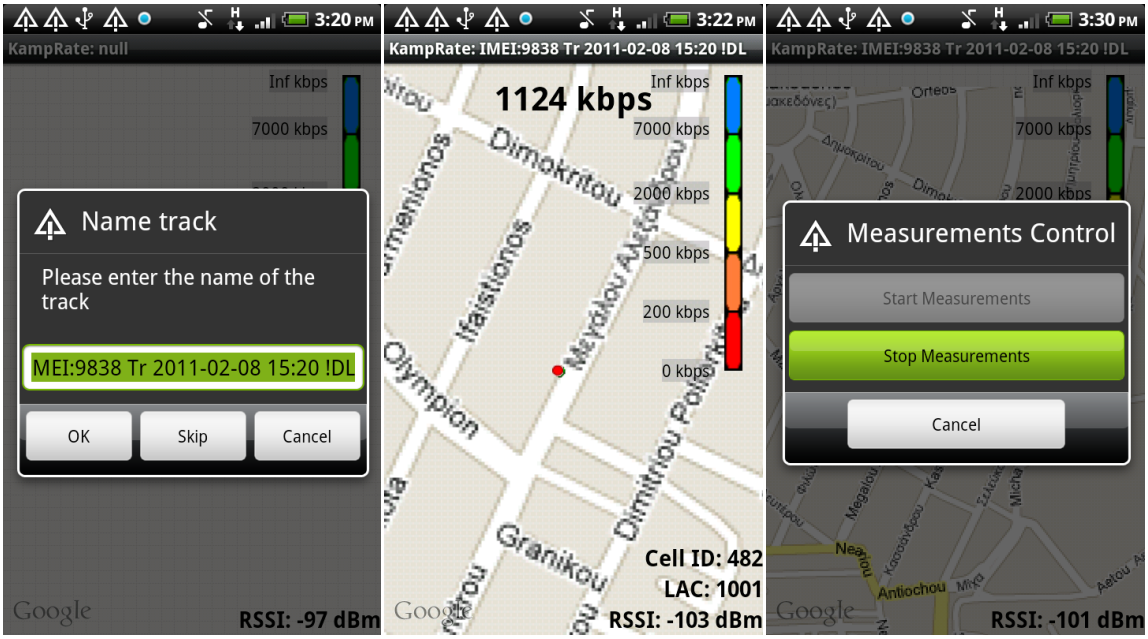
Download Indoors

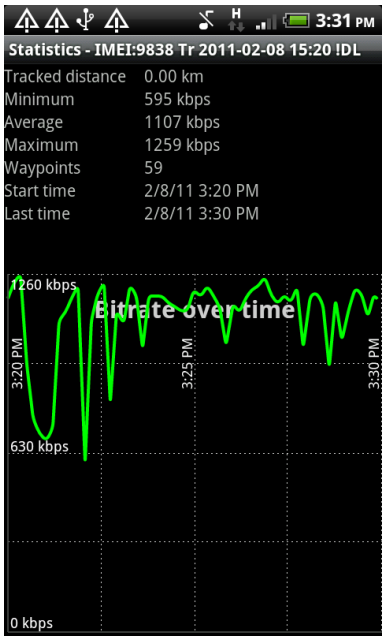
Στο παράδειγμά μας ρυθμίζουμε το KampRate ώστε να εκτελεστεί Download Indoors μέτρηση με μέγεθος πακέτου 1000 B.



Εν συνεχεία, επιλέγουμε «Measure» και «Start Measurements». Αναζητείται η θέση μας από τον Network Provider και μόλις βρεθεί επισημαίνεται στο χάρτη με ένα δείκτη. Επειδή δεν μας ικανοποιεί η ακρίβεια προσδιορισμού (απέχει αρκετά από την πραγματική μας θέση στη συγκεκριμένη περίπτωση), επιλέγουμε το «New Loc» που βρίσκεται στο κάτω μέρος της οθόνης και ακουμπάμε (tap) στην πραγματική μας θέση στο χάρτη. Πατάμε το «OK», ονοματίζουμε την καταγραφή (δεν θα αλλάξουμε το προτεινόμενο όνομα) και οι μετρήσεις ξεκινούν. Αφήνουμε τις μετρήσεις να «τρέξουν» για 10 λεπτά. Πατάμε «Stop Measurements» για να τερματίσει η καταγραφή και ανοίγουμε τα στατιστικά στοιχεία για να έχουμε μια εικόνα των μετρήσεων που πήραμε.


3:13 PM
Select Exact Location (Indoors)
Searching loc through Network Provider...
No location yet
OK
Cancel
New Loc
3:15 PM
Select Exact Location (Indoors)
Tap your exact location on map
Location selected
Lat: 38.114472
Lon: 23.767668
OK
Cancel
New Loc
3:18 PM
Select Exact Location (Indoors)
Tap your exact location on map
Location selected
Lat: 38.12569
Lon: 23.76223
Location changed
OK
Cancel
New Loc


3:20 PM
KampRate: null
Inf kbps
7000 kbps
Name track
Please enter the name of the track
MEI:9838 Tr 2011-02-08 15:20 !DL
OK
Skip
Cancel
Google
RSSI: -97 dBm
3:22 PM
KampRate: IMEI:9838 Tr 2011-02-08 15:20 !DL
1124 kbps
Inf kbps
7000 kbps
2000 kbps
500 kbps
200 kbps
0 kbps
Dimokritou
Ifaistionos
Olympion
Granikou
Cell ID: 482
LAC: 1001
RSSI: -103 dBm
3:30 PM
KampRate: IMEI:9838 Tr 2011-02-08 15:20 !DL
Measurements Control
Start Measurements
Stop Measurements
Cancel
Antiochou
RSSI: -101 dBm

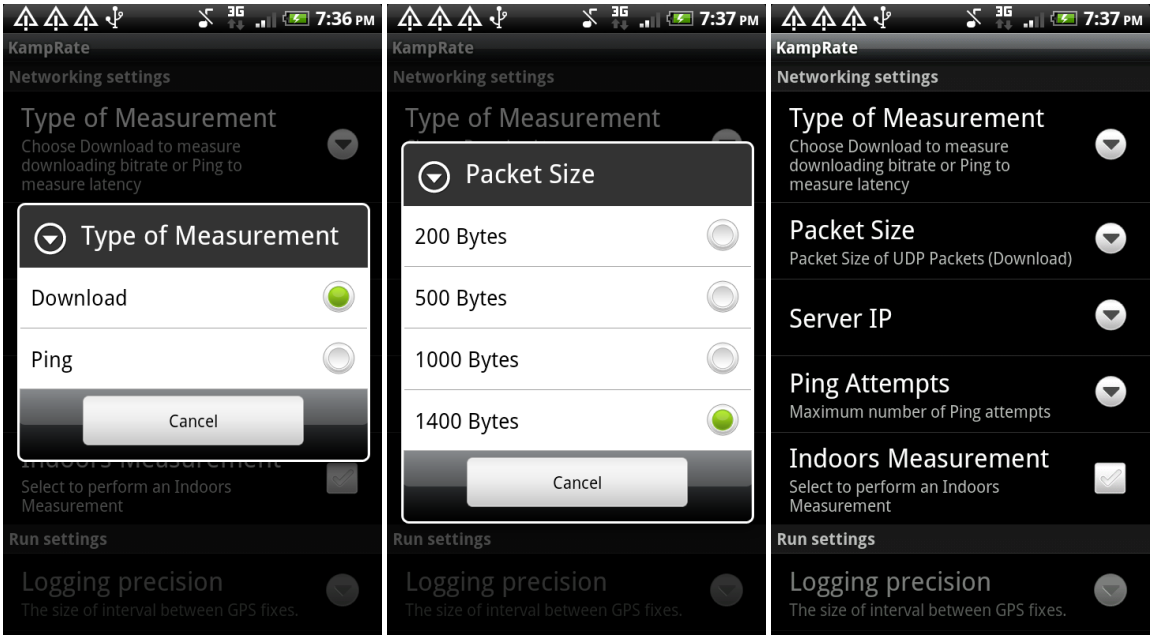


Η διανυόμενη απόσταση είναι 0 km, αφού η καταγραφή ήταν εσωτερικού χώρου, οπότε η θέση μας ήταν σταθερή. Το ελάχιστο bit rate που μετρήθηκε ήταν 595 kbps, ενώ το μέγιστο 1259 kbps. Το μέσο bit rate ήταν 1107 kbps. Συνολικά δημιουργήθηκαν 59 waypoints.

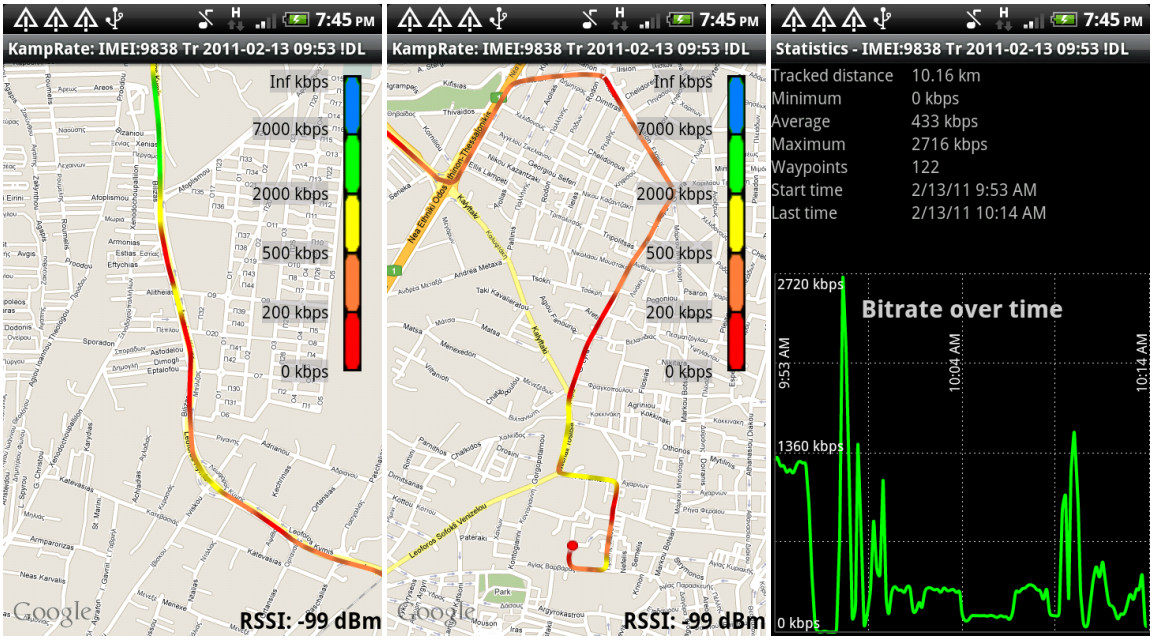
(Η θέση που επιλέξαμε στο χάρτη δεν ανταποκρίνεται στην πραγματικότητα και έχει χρησιμοποιηθεί μόνο για λόγους επίδειξης)

Download Outdoors

Εκτελούμε μέτρηση Download σε εξωτερικό χώρο, επιλέγοντας μέγεθος πακέτου 1400 B.

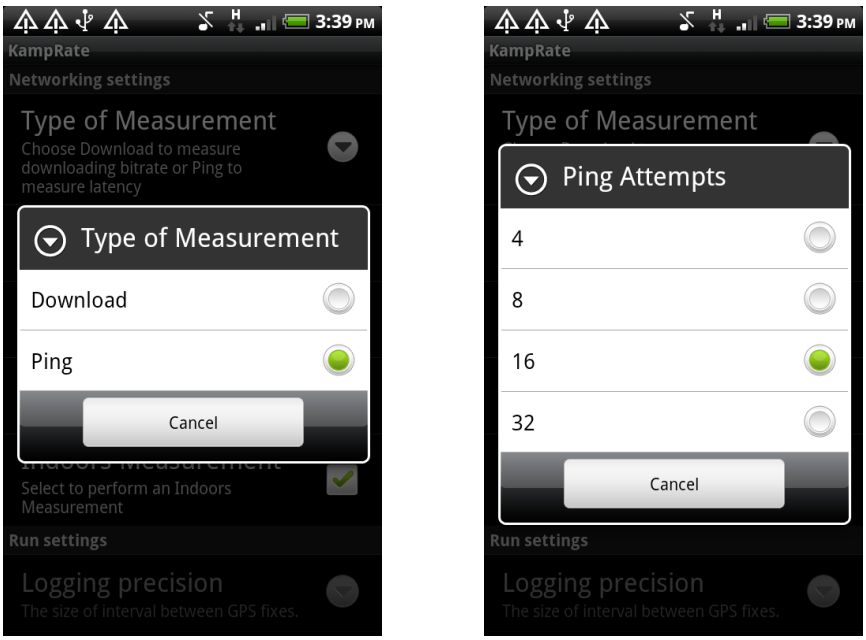


Από τα αποτελέσματα παρατηρούμε πως στη διαδρομή των 10,16 km που διανύσαμε, σημειώθηκε μέγιστο bitrate ίσο με 2716 kbps, ενώ το μέσο bitrate ήταν 433 kbps.

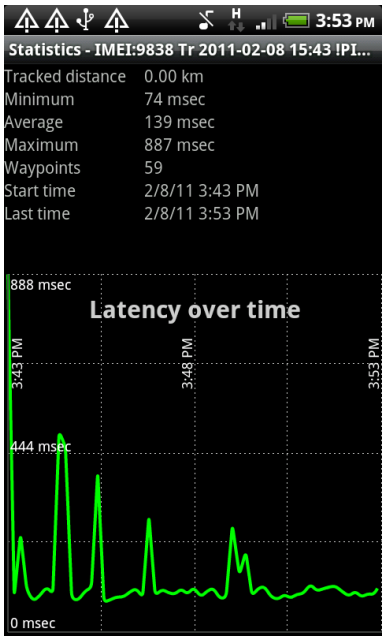


Ping Indoors

Σε αυτό το παράδειγμα εκτελούμε μέτρηση Ping εσωτερικού χώρου με μέγιστο αριθμό επιχειρούμενων ανά segment προσπαθειών ίσο με 16.



Γίνεται η επιλογή της θέσης, όπως προηγουμένως, και λαμβάνονται μετρήσεις για 10 λεπτά. Για να σταματήσουμε την καταγραφή επιλέγουμε «Stop Measurements».

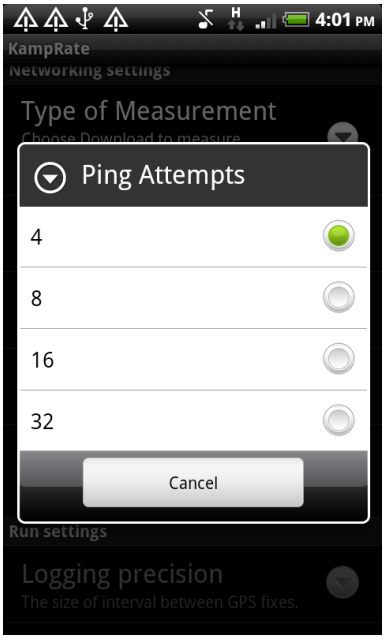


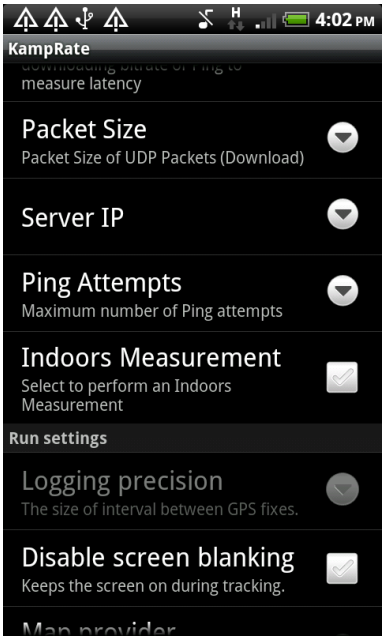
Η διανυόμενη απόσταση είναι 0 km, αφού βρισκόμαστε στο ίδιο σημείο (εσωτερικός χώρος). Ελάχιστο latency μετρήθηκε 74 msec, ενώ μέγιστο 887 msec (που αφορά όμως την πρώτη μέτρηση που δεν περιλαμβάνεται στο εξαγόμενο xml και συνακόλουθα δεν εισάγεται στη βάση δεδομένων). Το μέσο latency βρέθηκε 139 msec και δημιουργήθηκαν 59 waypoints.

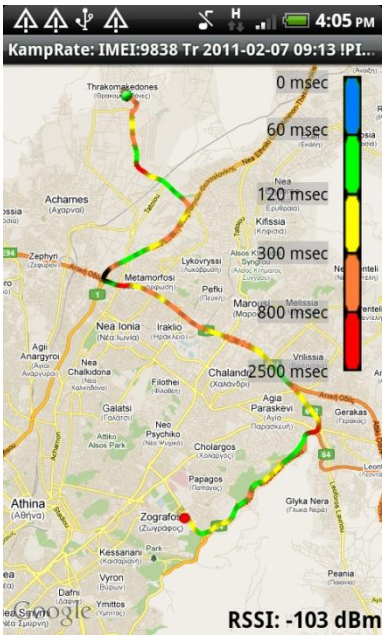
(Η θέση που επιλέξαμε στο χάρτη δεν ανταποκρίνεται στην πραγματικότητα και έχει χρησιμοποιηθεί μόνο για λόγους επίδειξης)

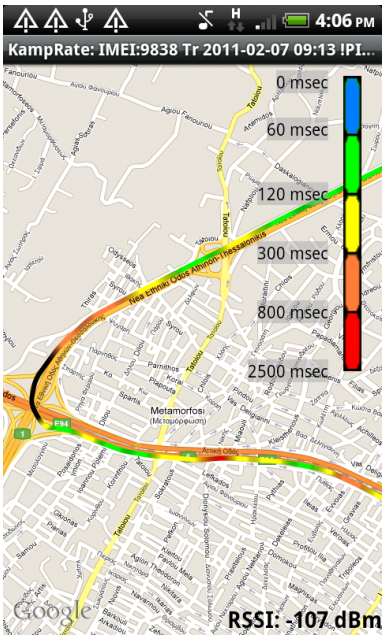
Ping Outdoors

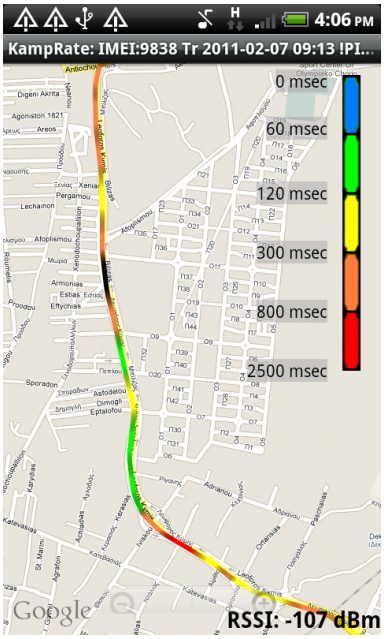
Στις ρυθμίσεις της εφαρμογής επιλέγουμε αριθμό προσπαθειών ίσο με 4 και αποεπιλέγουμε το «Indoors Measurement».

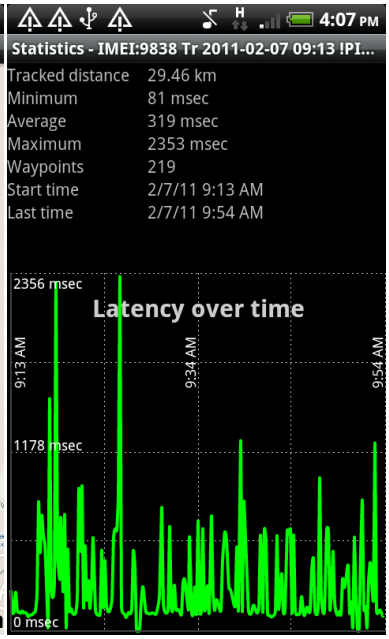


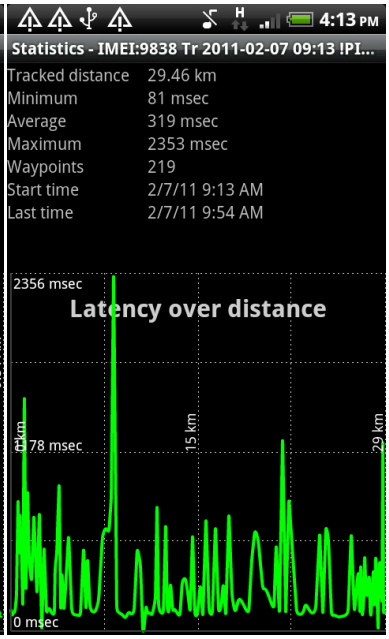








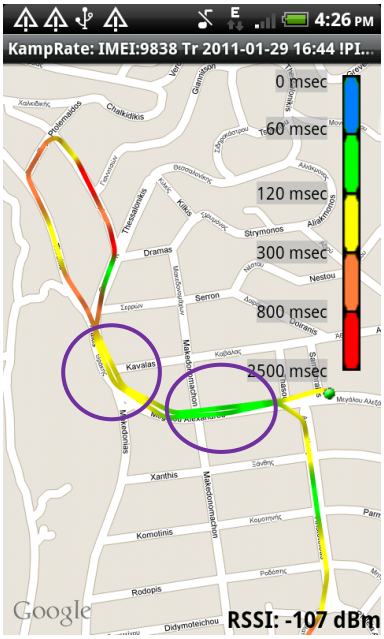




Πρόκειται για μια μεγάλη διαδρομή 29.46 km που διήρκησε περίπου 40 λεπτά. Το ελάχιστο latency στη διαδρομή ήταν 81 msec, ενώ το μέγιστο πολύ κοντά στο όριο των 2500 msec και ίσο με 2353 msec. Μέσο latency υπολογίστηκε 319 msec.

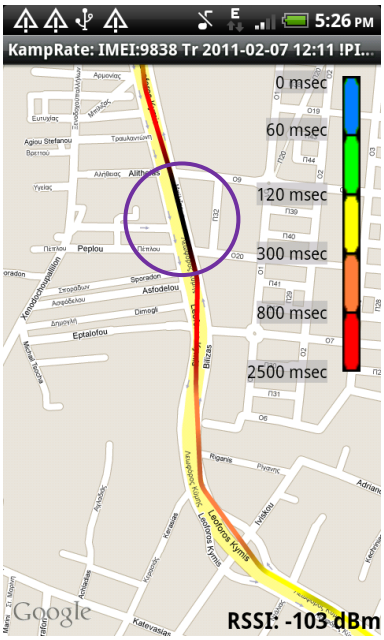
Παρατηρήσεις – Επισημάνσεις

- Σε αντίθετη κατεύθυνση και για τα ίδια τμήματα μετρήθηκαν περίπου ίδιοι χρόνοι Average Latency. Η χρονική απόσταση ανάμεσα στην άνοδο και στην κάθοδο του δρόμου ήταν μικρή. Η παρατήρηση αυτή ενισχύεται και συμπληρώνεται από μια δοκιμή που πραγματοποιήσαμε με δύο κινητά στο ίδιο αυτοκίνητο, τα οποία διαπιστώθηκε ότι μετρούσαν σχεδόν ίδιους χρόνους στις μετρήσεις που εκτελέσαμε.



- Από δοκιμές που πραγματοποιήσαμε στην Πολυτεχνειούπολη, παρατηρήσαμε πως το bit rate αυξανόταν πολύ, όταν περνούσαμε κοντά από την κεραία της κινητής τηλεφωνίας. Αντίστοιχα το latency μειωνόταν αισθητά. Η συμπεριφορά αυτή εμφανίστηκε σε όλες τις καταγραφές που πραγματοποιήσαμε. Η αύξηση της ταχύτητας είναι αναμενόμενη, όταν πλησιάζουμε στην κεραία, και δεν υπάρχουν σημαντικά εμπόδια ανάμεσα.

- Το μαύρο χρώμα χρησιμοποιείται όταν μετράται bit rate ίσο με μηδέν ή όταν δεν επιτυγχάνει καμία προσπάθεια Ping στο διάστημα των 10 sec. Αυτές οι δύο περιπτώσεις έχουν δύο πιθανές αιτίες: Η πρώτη είναι το ενδεχόμενο διαπομπής (handover), οπότε για μερικά second χάνεται η σύνδεση και μηδενίζεται η ταχύτητα. Αυτό δεν είναι απαραίτητο πως συμβαίνει σε κάθε handover. Άλλωστε, γνωρίζουμε πως στην περίπτωση της διαπομπής, το κινητό «ακούει» και από τους δύο σταθμούς μέχρι να γίνει η μετάβαση. Στην πράξη, ωστόσο, βλέπουμε πως πολλές φορές η διαπομπή έχει ως αποτέλεσμα μηδενισμό του bit rate ή αποτυχία στις προσπάθειες Ping. Η δεύτερη πιθανή αιτία, είναι το χαμηλής ποιότητας δίκτυο στο μετρούμενο διάστημα. Για να προσδιορίσουμε το αίτιο κάθε φορά, πρέπει να εντοπίζουμε στις πληροφορίες που αποθηκεύονται για κάθε segment, αν υπήρξε αλλαγή στο Cell ID, οπότε και πρόκειται σίγουρα για



handover. Αν δεν έγινε αλλαγή στο Cell ID και παραμείναμε στην ίδια κυψέλη, σημαίνει πως το δίκτυο ήταν χαμηλής ποιότητας σε εκείνο το διάστημα.

Παράρτημα:

Πηγαίος Κώδικας των Εφαρμογών

KampRate

ControlTracking.java

```
package nl.sogeti.android.gpstracker.actions;

import nl.sogeti.android.gpstracker.R;
import nl.sogeti.android.gpstracker.db.GPStracking.Tracks;
import nl.sogeti.android.gpstracker.logger.GPSLoggerServiceManager;
import nl.sogeti.android.gpstracker.util.Constants;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.AlertDialog.Builder;
import android.app.Dialog;
import android.content.ComponentName;
import android.content.ContentUris;
import android.content.DialogInterface;
import android.content.DialogInterface.OnClickListener;
import android.content.DialogInterface.OnDismissListener;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.Button;

/**
 * Activity with options to Start tracking and Stop tracking
 *
 * @author Alexandros Ntakas
 */
public class ControlTracking extends Activity {
   private static final int DIALOG_LOGCONTROL = 26;
   private static final String TAG = "OGT.ControlTracking";

   private GPSLoggerServiceManager mLoggerServiceManager;
   private Button start;
   private Button stop;
   private boolean paused;

   private final View.OnClickListener mLoggingControlListener = new View.OnClickListener()
{
      public void onClick(View v) {
         int id = v.getId();
         switch (id) {
         case R.id.logcontrol_start:
            Intent receivedIntent = getIntent();
            Boolean IndoorsCheck = receivedIntent.getBooleanExtra(
                  Constants.INDOORSCHECK, false);

            if (IndoorsCheck == true) {
               Intent selectLocIntent = new Intent(ControlTracking.this,
                     IndoorsSelectLocation.class);
               startActivityForResult(selectLocIntent, 0);
            } else {
               long loggerTrackId = mLoggerServiceManager.startGPSLogging(null);
               Intent intent = new Intent();
```

```
                intent.setData(ContentUris.withAppendedId(Tracks.CONTENT_URI,
loggerTrackId));
                ComponentName caller = ControlTracking.this.getCallingActivity();
                if (caller != null) {
                   setResult(RESULT_OK, intent);
                }
                finish();
             }
             break;
          case R.id.logcontrol_stop:
             mLoggerServiceManager.stopGPSLogging();
             finish();
          default:
             finish();
          }
       }
    };
    private OnClickListener mDialogClickListener = new OnClickListener() {
       public void onClick(DialogInterface dialog, int which) {
          finish();
       }
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       this.setVisible(false);
       paused = false;
       mLoggerServiceManager = new GPSLoggerServiceManager(this);
    }

    @Override
    protected void onResume() {
       super.onResume();
       mLoggerServiceManager.startup(new Runnable() {
          public void run() {
             showDialog(DIALOG_LOGCONTROL);
          }
       });
    }

    @Override
    protected void onPause() {
       super.onPause();
       mLoggerServiceManager.shutdown();
       paused = true;
    }

    @Override
    protected Dialog onCreateDialog(int id) {
       Dialog dialog = null;
       LayoutInflater factory = null;
       View view = null;
       Builder builder = null;
       switch (id) {
       case DIALOG_LOGCONTROL:
          builder = new AlertDialog.Builder(this);
          factory = LayoutInflater.from(this);
          view = factory.inflate(R.layout.logcontrol, null);
          builder.setTitle(R.string.dialog_tracking_title)
                .setIcon(android.R.drawable.ic_dialog_alert)
                .setNegativeButton(R.string.btn_cancel,
```

```
                mDialogClickListener).setView(view);
      dialog = builder.create();
      start = (Button) view.findViewById(R.id.logcontrol_start);
      stop = (Button) view.findViewById(R.id.logcontrol_stop);
      start.setOnClickListener(mLoggingControlListener);
      stop.setOnClickListener(mLoggingControlListener);
      dialog.setOnDismissListener(new OnDismissListener() {
         public void onDismiss(DialogInterface dialog) {
            if (!paused) {
               finish();
            }
         }
      });
      return dialog;
   default:
      return super.onCreateDialog(id);
   }
}

@Override
protected void onPrepareDialog(int id, Dialog dialog) {
   switch (id) {
   case DIALOG_LOGCONTROL:
      updateDialogState(mLoggerServiceManager.getLoggingState());
      break;
   default:
      break;
   }
   super.onPrepareDialog(id, dialog);
}

private void updateDialogState(int state) {
   switch (state) {
   case Constants.STOPPED:
      start.setEnabled(true);
      stop.setEnabled(false);
      break;
   case Constants.LOGGING:
      start.setEnabled(false);
      stop.setEnabled(true);
      break;
   case Constants.PAUSED:
      start.setEnabled(false);
      stop.setEnabled(true);
      break;
   default:
      Log.w(TAG, String.format(
            "State %d of logging, enabling and hope for the best....", state));
      start.setEnabled(false);
      stop.setEnabled(false);
      break;
   }
}

/**
 * Result returned from IndoorsSelectLocation. If gotLocation == true ->
 * startLogging, return result to LoggerMap. Else, do nothing, finish and
 * return to LoggerMap.
 */
protected void onActivityResult(int requestCode, int resultCode,
      Intent fixedLocIntent) {
```

```
      if (fixedLocIntent.getBooleanExtra(Constants.gotLoc, false) == true) {
         long loggerTrackId = mLoggerServiceManager.startGPSLogging(null);

         Intent intent = new Intent();
         intent.setData(ContentUris.withAppendedId(Tracks.CONTENT_URI,
               loggerTrackId));
         ComponentName caller = ControlTracking.this.getCallingActivity();
         if (caller != null) {
            setResult(RESULT_OK, intent);
         }
         finish();
      } else {
         finish();
      }
   }
}
```

IndoorsGeopointSingleton.java

```
package nl.sogeti.android.gpstracker.actions;

/**
 * Singleton to retrieve location coordinates of fixedLocation (case Indoors)
 * when needed (GPSLoggerService)
 *
 * @author Alexandros Ntakas
 */
public class IndoorsGeopointSingleton {
   private static IndoorsGeopointSingleton instance = null;
   private double latitude;
   private double longitude;

   protected IndoorsGeopointSingleton(double latitude1, double longitude1) {
      latitude = latitude1;
      longitude = longitude1;
   }

   public static IndoorsGeopointSingleton getInstance(double latitude,
         double longitude) {
      if (instance == null) {
         instance = new IndoorsGeopointSingleton(latitude, longitude);
      }
      return instance;
   }

   public double latitude() {
      return latitude;
   }

   public double longitude() {
      return longitude;
   }
}
```

IndoorsSelectLocation.java

```
package nl.sogeti.android.gpstracker.actions;

import java.util.List;
```

```
import nl.sogeti.android.gpstracker.R;
import nl.sogeti.android.gpstracker.actions.IndoorsGeopointSingleton;
import nl.sogeti.android.gpstracker.util.Constants;
import android.content.Context;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Point;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.View.OnTouchListener;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.TextView;
import android.widget.Toast;

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

/**
 * Activity to find location through Network Provider (case Indoors) and select
 * exact location by tapping on the map
 *
 * @author Alexandros Ntakas
 */
public class IndoorsSelectLocation extends MapActivity implements
      OnTouchListener {

   private LocationManager locationManager;
   private LocationListener locationListener;
   private MapView mapView;
   private MapController mc;
   private GeoPoint p;
   private TextView TextviewINDOORS;
   private TextView TextviewCoordinatesINDOORS;
   private CheckBox NewLocINDOORS;
   private List<Overlay> listOfOverlays;

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

      // Getting layout controls
      TextviewINDOORS = (TextView) findViewById(R.id.TextviewINDOORS);
      TextviewINDOORS.setText("Searching loc through Network Provider...");
      TextviewCoordinatesINDOORS = (TextView)
findViewById(R.id.TextviewCoordinatesINDOORS);
      TextviewCoordinatesINDOORS.setText("No location yet");
      NewLocINDOORS = (CheckBox) findViewById(R.id.NewLocINDOORS);
      NewLocINDOORS.setChecked(false);
      NewLocINDOORS.setClickable(false);
```

```
   Button OkINDOORS = (Button) findViewById(R.id.OkINDOORS);
   OkINDOORS.setOnClickListener(OkINDOORSListener);
   Button CancelINDOORS = (Button) findViewById(R.id.CancelINDOORS);
   CancelINDOORS.setOnClickListener(CancelINDOORSListener);

   locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
   locationListener = new MyLocationListener();
   locationManager.requestLocationUpdates(
         locationManager.NETWORK_PROVIDER, 0, 10, locationListener);

   mapView = (MapView) findViewById(R.id.mapviewINDOORS);
   mapView.setSatellite(true);
   mapView.setOnTouchListener(this);

   mc = mapView.getController();
}

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

private class MyLocationListener implements LocationListener {
   public void onLocationChanged(Location loc) {
      if (loc != null) {
         Toast.makeText(getBaseContext(), "Location changed", 2000).show();
         p = new GeoPoint((int) (loc.getLatitude() * 1E6),
               (int) (loc.getLongitude() * 1E6));
         mc.animateTo(p);
         mc.setZoom(19);

         TextviewINDOORS.setText("Tap your exact location on map");
         TextviewCoordinatesINDOORS.setText("Location selected" + "\n"
               + "Lat: " + loc.getLatitude() + "\n" + "Lon: "
               + loc.getLongitude());

         MapOverlay mapOverlay = new MapOverlay(p);
         listOfOverlays = mapView.getOverlays();
         listOfOverlays.clear();
         listOfOverlays.add(mapOverlay);

         mapView.invalidate();

         NewLocINDOORS.setClickable(true);
         // Removing updates of location Listener since location was located
         // Now the user may select his exact location on the map by tapping on it
         locationManager.removeUpdates(locationListener);
      }
   }

   public void onProviderDisabled(String provider) {
   }

   public void onProviderEnabled(String provider) {
   }

   public void onStatusChanged(String provider, int status, Bundle extras) {
   }
}

private class MapOverlay extends Overlay {
   private GeoPoint p;
```

```
   public MapOverlay(GeoPoint p1) {
      p = p1;
   }

   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);

      // ---add the marker---
      Bitmap bmp = BitmapFactory.decodeResource(getResources(), R.drawable.pushpin);
      canvas.drawBitmap(bmp, screenPts.x, screenPts.y, null);
      return true;
   }
}

private OnClickListener OkINDOORSListener = new OnClickListener() {
   public void onClick(View v) {
      Intent fixedLocIntent = new Intent();
      if (p != null) {
         fixedLocIntent.putExtra(Constants.gotLoc, true);
      } else {
         fixedLocIntent.putExtra(Constants.gotLoc, false);
      }
      setResult(0, fixedLocIntent);

      // Create a Singleton with Location coordinates. This will be accessed later
      // by GPSLoggerService (Case Indoors)
      IndoorsGeopointSingleton.getInstance(p.getLatitudeE6() / 1E6,
            p.getLongitudeE6() / 1E6);

      mapView.setSatellite(false);
      finish();
   }
};

private OnClickListener CancelINDOORSListener = new OnClickListener() {
   public void onClick(View v) {
      Intent fixedLocIntent = new Intent();
      fixedLocIntent.putExtra(Constants.gotLoc, false);
      setResult(0, fixedLocIntent);

      mapView.setSatellite(false);
      finish();
   }
};

public boolean onTouch(View v, MotionEvent event) {
   if (NewLocINDOORS.isChecked()) {
      p = mapView.getProjection().fromPixels((int) event.getX(), (int) event.getY());
      Toast.makeText(getBaseContext(), "Location changed", 2000).show();

      TextviewCoordinatesINDOORS.setText("Location selected" + "\n"
            + "Lat: " + (p.getLatitudeE6() / 1E6) + "\n" + "Lon: "
            + (p.getLongitudeE6() / 1E6));

      MapOverlay mapOverlay = new MapOverlay(p);
      listOfOverlays = mapView.getOverlays();
      listOfOverlays.clear();
```

```
         listOfOverlays.add(mapOverlay);

         mapView.invalidate();
      }
      return false;
   }
}
```

NameTrack.java

```
package nl.sogeti.android.gpstracker.actions;

import java.util.Calendar;

import nl.sogeti.android.gpstracker.R;
import nl.sogeti.android.gpstracker.db.GPStracking.Tracks;
import nl.sogeti.android.gpstracker.util.Constants;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.AlertDialog.Builder;
import android.app.Dialog;
import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.Context;
import android.content.DialogInterface;
import android.content.DialogInterface.OnDismissListener;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.telephony.TelephonyManager;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.EditText;

/**
 * Activity that pops up the dialog to name the track. The user should not
 * change the suggested name's structure of the track if he wants to upload the
 * file to Main Server and be inserted automatically into database.
 *
 * @author Alexandros Ntakas
 */
public class NameTrack extends Activity {
   private static final int DIALOG_TRACKNAME = 23;

   protected static final String TAG = "OGT.NameTrack";

   private EditText mTrackNameView;
   private boolean paused;

   protected long mTrackId = -1;

   private final DialogInterface.OnClickListener mTrackNameDialogListener = new
DialogInterface.OnClickListener() {
      public void onClick(DialogInterface dialog, int which) {
         String trackName = null;
         switch (which) {
         case DialogInterface.BUTTON_POSITIVE:
```

```
            trackName = mTrackNameView.getText().toString();
            ContentValues values = new ContentValues();
            values.put(Tracks.NAME, trackName);
            getContentResolver().update(
                  ContentUris.withAppendedId(Tracks.CONTENT_URI,
                        NameTrack.this.mTrackId), values, null, null);
            clearNotification();
            break;
         case DialogInterface.BUTTON_NEUTRAL:
            startDelayNotification();
            break;
         case DialogInterface.BUTTON_NEGATIVE:
            clearNotification();
            break;
         default:
            Log.e(TAG, "Unknown option ending dialog:" + which);
            break;
         }
         finish();
      }
};

private void clearNotification() {
   NotificationManager noticationManager = (NotificationManager) this
         .getSystemService(Context.NOTIFICATION_SERVICE);
   ;
   noticationManager.cancel(R.layout.namedialog);
}

private void startDelayNotification() {
   int resId = R.string.dialog_routename_title;
   int icon = R.drawable.ic_maps_indicator_current_position;
   CharSequence tickerText = getResources().getString(resId);
   long when = System.currentTimeMillis();

   Notification nameNotification = new Notification(icon, tickerText, when);
   nameNotification.flags |= Notification.FLAG_AUTO_CANCEL;

   CharSequence contentTitle = getResources().getString(R.string.app_name);
   CharSequence contentText = getResources().getString(resId);

   Intent notificationIntent = new Intent(this, NameTrack.class);
   notificationIntent.setData(ContentUris.withAppendedId(
         Tracks.CONTENT_URI, mTrackId));

   PendingIntent contentIntent = PendingIntent.getActivity(this, 0,
         notificationIntent, Intent.FLAG_ACTIVITY_NEW_TASK);
   nameNotification.setLatestEventInfo(this, contentTitle, contentText, contentIntent);

   NotificationManager noticationManager = (NotificationManager) this
         .getSystemService(Context.NOTIFICATION_SERVICE);
   noticationManager.notify(R.layout.namedialog, nameNotification);
}

@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   this.setVisible(false);
   paused = false;

   Uri data = this.getIntent().getData();
   if (data != null) {
```

```
      mTrackId = Long.parseLong(data.getLastPathSegment());
   }
}

@Override
protected void onPause() {
   super.onPause();
   paused = true;
}

@Override
protected void onResume() {
   super.onResume();
   if (mTrackId >= 0) {
      showDialog(DIALOG_TRACKNAME);
   }
}

@Override
protected Dialog onCreateDialog(int id) {
   Dialog dialog = null;
   LayoutInflater factory = null;
   View view = null;
   Builder builder = null;
   switch (id) {
   case DIALOG_TRACKNAME:
      builder = new AlertDialog.Builder(this);
      factory = LayoutInflater.from(this);
      view = factory.inflate(R.layout.namedialog, null);
      mTrackNameView = (EditText) view.findViewById(R.id.nameField);
      builder.setTitle(R.string.dialog_routename_title)
            .setMessage(R.string.dialog_routename_message)
            .setIcon(android.R.drawable.ic_dialog_alert)
            .setPositiveButton(R.string.btn_okay,
                  mTrackNameDialogListener)
            .setNeutralButton(R.string.btn_skip,
                  mTrackNameDialogListener)
            .setNegativeButton(R.string.btn_cancel,
                  mTrackNameDialogListener).setView(view);
      dialog = builder.create();
      dialog.setOnDismissListener(new OnDismissListener() {
         public void onDismiss(DialogInterface dialog) {
            if (!paused) {
               finish();
            }
         }
      });
      return dialog;
   default:
      return super.onCreateDialog(id);
   }
}

@Override
protected void onPrepareDialog(int id, Dialog dialog) {
   switch (id) {
   case DIALOG_TRACKNAME:
      String trackName;
      Calendar c = Calendar.getInstance();
      trackName = String
            .format(getString(R.string.dialog_routename_default), c, c,c, c, c);
```

```
         /**
          * Get IMEI number to separate on Server's side different tracks
          * with the same start time (which otherwise would have the same
          * name)
          *
          * Add to the suggested name of the track "!DL" or "!PING" so the
          * server may import the files correctly to the database tables and
          * the user can have extra information in the files' list (List
          * Tracks) in the application
          */
         TelephonyManager telephonyManager = (TelephonyManager)
getSystemService(Context.TELEPHONY_SERVICE);
         String IMEI = telephonyManager.getDeviceId();
         if (IMEI == null)
            IMEI = "";
         String IMEIlastDigits = IMEI.substring(11);
         trackName = "IMEI:" + IMEIlastDigits + " " + trackName;

         int NMChelper = getIntent().getIntExtra( Constants.NETWORKINGMEASUREMENT, 0);
         if (NMChelper == Constants.DOWNLOADCHOICE) {
            trackName = trackName + " !DL";
         } else if (NMChelper == Constants.PINGCHOICE) {
            trackName = trackName + " !PING";
         }

         mTrackNameView.setText(trackName);
         mTrackNameView.setSelection(0, trackName.length());
         break;
      default:
         super.onPrepareDialog(id, dialog);
         break;
      }
   }
}
```

Statistics.java

```
package nl.sogeti.android.gpstracker.actions;

import nl.sogeti.android.gpstracker.R;
import nl.sogeti.android.gpstracker.actions.utils.GraphCanvas;
import nl.sogeti.android.gpstracker.actions.utils.StatisticsCalulator;
import nl.sogeti.android.gpstracker.db.GPStracking.Tracks;
import nl.sogeti.android.gpstracker.util.UnitsI18n;
import nl.sogeti.android.gpstracker.viewer.TrackList;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.app.AlertDialog.Builder;
import android.content.ContentResolver;
import android.content.Intent;
import android.database.ContentObserver;
import android.net.Uri;
import android.os.Bundle;
import android.os.Handler;
import android.view.ContextMenu;
import android.view.GestureDetector;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.MotionEvent;
```

```
import android.view.View;
import android.view.GestureDetector.SimpleOnGestureListener;
import android.view.View.OnClickListener;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.Button;
import android.widget.TextView;
import android.widget.ViewFlipper;

/**
 * Display some calculations based on a track
 *
 * @author Alexandros Ntakas
 */
public class Statistics extends Activity {

   private static final int DIALOG_GRAPHTYPE = 3;
   private static final int MENU_GRAPHTYPE = 11;
   private static final int MENU_TRACKLIST = 12;
   private static final int MENU_SHARE = 41;
   private static final String TRACKURI = "TRACKURI";
   @SuppressWarnings("unused")
   private static final String TAG = "OGT.Statistics";

   private static final int SWIPE_MIN_DISTANCE = 120;
   private static final int SWIPE_MAX_OFF_PATH = 250;
   private static final int SWIPE_THRESHOLD_VELOCITY = 200;

   private Uri mTrackUri = null;
   private boolean calculating;

   private TextView avgSpeedView;
   private TextView distanceView;
   private TextView endtimeView;
   private TextView starttimeView;
   private TextView maxSpeedView;
   private TextView minSpeedView;
   private TextView waypointsView;
   private TextView minAltitudeView;
   private TextView maxAltitudeView;

   private UnitsI18n mUnits;
   private GraphCanvas mGraphTimeSpeed;

   private ViewFlipper mViewFlipper;
   private Animation mSlideLeftIn;
   private Animation mSlideLeftOut;
   private Animation mSlideRightIn;
   private Animation mSlideRightOut;
   private GestureDetector mGestureDetector;
   private GraphCanvas mGraphDistanceSpeed;
   private GraphCanvas mGraphTimeAltitude;
   private GraphCanvas mGraphDistanceAltitude;

   private final ContentObserver mTrackObserver = new ContentObserver(
         new Handler()) {

      @Override
      public void onChange(boolean selfUpdate) {
         if (!calculating) {
            Statistics.this.drawTrackingStatistics();
         }
```

```
   }
};
private OnClickListener mGraphControlListener = new View.OnClickListener() {
   public void onClick(View v) {
      int id = v.getId();
      switch (id) {
      case R.id.graphtype_timespeed:
         mViewFlipper.setDisplayedChild(0);
         break;
      case R.id.graphtype_distancespeed:
         mViewFlipper.setDisplayedChild(1);
         break;
      default:
         break;
      }
      dismissDialog(DIALOG_GRAPHTYPE);
   }
};
private StatisticsCalulator mCalculator;

class MyGestureDetector extends SimpleOnGestureListener {
   @Override
   public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX,
         float velocityY) {
      if (Math.abs(e1.getY() - e2.getY()) > SWIPE_MAX_OFF_PATH)
         return false;
      // right to left swipe
      if (e1.getX() - e2.getX() > SWIPE_MIN_DISTANCE
            && Math.abs(velocityX) > SWIPE_THRESHOLD_VELOCITY) {
         mViewFlipper.setInAnimation(mSlideLeftIn);
         mViewFlipper.setOutAnimation(mSlideLeftOut);
         mViewFlipper.showNext();
      } else if (e2.getX() - e1.getX() > SWIPE_MIN_DISTANCE
            && Math.abs(velocityX) > SWIPE_THRESHOLD_VELOCITY) {
         mViewFlipper.setInAnimation(mSlideRightIn);
         mViewFlipper.setOutAnimation(mSlideRightOut);
         mViewFlipper.showPrevious();
      }
      return false;
   }
}

/**
 * Called when the activity is first created.
 */
@Override
protected void onCreate(Bundle load) {
   super.onCreate(load);
   mUnits = new UnitsI18n(this, new UnitsI18n.UnitsChangeListener() {
      public void onUnitsChange() {
         drawTrackingStatistics();
      }
   });
   setContentView(R.layout.statistics);

   mCalculator = new StatisticsCalulator(this, mUnits);

   mViewFlipper = (ViewFlipper) findViewById(R.id.flipper);
   mSlideLeftIn = AnimationUtils.loadAnimation(this, R.anim.slide_left_in);
   mSlideLeftOut = AnimationUtils.loadAnimation(this, R.anim.slide_left_out);
   mSlideRightIn = AnimationUtils.loadAnimation(this, R.anim.slide_right_in);
   mSlideRightOut = AnimationUtils.loadAnimation(this, R.anim.slide_right_out);
```

```
   mGraphTimeSpeed = (GraphCanvas) mViewFlipper.getChildAt(0);
   mGraphDistanceSpeed = (GraphCanvas) mViewFlipper.getChildAt(1);

   mGraphTimeSpeed.setType(GraphCanvas.TIMESPEEDGRAPH);
   mGraphDistanceSpeed.setType(GraphCanvas.DISTANCESPEEDGRAPH);

   mGestureDetector = new GestureDetector(new MyGestureDetector());

   maxSpeedView = (TextView) findViewById(R.id.stat_maximumspeed);
   minSpeedView = (TextView) findViewById(R.id.stat_minimumspeed);
   minAltitudeView = (TextView) findViewById(R.id.stat_minimalaltitide);
   maxAltitudeView = (TextView) findViewById(R.id.stat_maximumaltitude);

   maxAltitudeView.setVisibility(View.INVISIBLE);
   minAltitudeView.setVisibility(View.INVISIBLE);

   avgSpeedView = (TextView) findViewById(R.id.stat_averagespeed);
   distanceView = (TextView) findViewById(R.id.stat_distance);
   starttimeView = (TextView) findViewById(R.id.stat_starttime);
   endtimeView = (TextView) findViewById(R.id.stat_endtime);
   waypointsView = (TextView) findViewById(R.id.stat_waypoints);

   if (load != null && load.containsKey(TRACKURI)) {
      mTrackUri = Uri.withAppendedPath(Tracks.CONTENT_URI, load.getString(TRACKURI));
   } else {
      mTrackUri = this.getIntent().getData();
   }
   drawTrackingStatistics();
}

@Override
protected void onRestoreInstanceState(Bundle load) {
   if (load != null) {
      super.onRestoreInstanceState(load);
   }
   if (load != null && load.containsKey(TRACKURI)) {
      mTrackUri = Uri.withAppendedPath(Tracks.CONTENT_URI, load.getString(TRACKURI));
   }
   if (load != null && load.containsKey("FLIP")) {
      mViewFlipper.setDisplayedChild(load.getInt("FLIP"));
   }
}

@Override
protected void onSaveInstanceState(Bundle save) {
   super.onSaveInstanceState(save);
   save.putString(TRACKURI, mTrackUri.getLastPathSegment());
   save.putInt("FLIP", mViewFlipper.getDisplayedChild());
}

@Override
protected void onPause() {
   super.onPause();
   mViewFlipper.stopFlipping();
   ContentResolver resolver = this.getApplicationContext().getContentResolver();
   resolver.unregisterContentObserver(this.mTrackObserver);
}

@Override
protected void onResume() {
   super.onResume();
```

```
        ContentResolver resolver = this.getApplicationContext().getContentResolver();
        resolver.unregisterContentObserver(this.mTrackObserver);
        resolver.registerContentObserver(mTrackUri, true, this.mTrackObserver);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        boolean result = super.onCreateOptionsMenu(menu);
        menu.add(ContextMenu.NONE, MENU_GRAPHTYPE, ContextMenu.NONE,
                R.string.menu_graphtype).setIcon(R.drawable.ic_menu_picture)
                .setAlphabeticShortcut('t');
        menu.add(ContextMenu.NONE, MENU_TRACKLIST, ContextMenu.NONE,
                R.string.menu_tracklist).setIcon(R.drawable.ic_menu_show_list)
                .setAlphabeticShortcut('l');
        menu.add(ContextMenu.NONE, MENU_SHARE, ContextMenu.NONE,
                R.string.menu_shareTrack).setIcon(R.drawable.ic_menu_share)
                .setAlphabeticShortcut('s');
        return result;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        boolean handled = false;
        switch (item.getItemId()) {
        case MENU_GRAPHTYPE:
            showDialog(DIALOG_GRAPHTYPE);
            handled = true;
            break;
        case MENU_TRACKLIST:
            Intent tracklistIntent = new Intent(this, TrackList.class);
            tracklistIntent.putExtra(Tracks._ID, mTrackUri.getLastPathSegment());
            startActivityForResult(tracklistIntent, MENU_TRACKLIST);
            break;
        case MENU_SHARE:
            Intent actionIntent = new Intent(Intent.ACTION_RUN);
            actionIntent.setDataAndType(mTrackUri, Tracks.CONTENT_ITEM_TYPE);
            actionIntent.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);
            startActivity(Intent.createChooser(actionIntent,
getString(R.string.share_track)));
            handled = true;
            break;
        default:
            handled = super.onOptionsItemSelected(item);
        }
        return handled;
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        if (mGestureDetector.onTouchEvent(event))
            return true;
        else
            return false;
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent intent) {
        super.onActivityResult(requestCode, resultCode, intent);
        if (resultCode != RESULT_CANCELED) {
            switch (requestCode) {
            case MENU_TRACKLIST:
                mTrackUri = intent.getData();
```

```
            drawTrackingStatistics();
            break;
         }
      }
   }

   @Override
   protected Dialog onCreateDialog(int id) {
      Dialog dialog = null;
      LayoutInflater factory = null;
      View view = null;
      Builder builder = null;
      switch (id) {
      case DIALOG_GRAPHTYPE:
         builder = new AlertDialog.Builder(this);
         factory = LayoutInflater.from(this);
         view = factory.inflate(R.layout.graphtype, null);
         builder.setTitle(R.string.dialog_graphtype_title)
               .setIcon(android.R.drawable.ic_dialog_alert)
               .setNegativeButton(R.string.btn_cancel, null).setView(view);
         dialog = builder.create();
         return dialog;
      default:
         return super.onCreateDialog(id);
      }
   }

   @Override
   protected void onPrepareDialog(int id, Dialog dialog) {
      switch (id) {
      case DIALOG_GRAPHTYPE:
         Button speedtime = (Button) dialog.findViewById(R.id.graphtype_timespeed);
         Button speeddistance = (Button)
dialog.findViewById(R.id.graphtype_distancespeed);
         speedtime.setOnClickListener(mGraphControlListener);
         speeddistance.setOnClickListener(mGraphControlListener);
      default:
         break;
      }
      super.onPrepareDialog(id, dialog);
   }

   private void drawTrackingStatistics() {
      calculating = true;

      mCalculator.updateCalculations(mTrackUri);

      mGraphTimeSpeed.setData(mTrackUri, mCalculator);
      mGraphDistanceSpeed.setData(mTrackUri, mCalculator);

      mViewFlipper.postInvalidate();

      maxSpeedView.setText(mCalculator.getMaxSpeedText());
      minSpeedView.setText(mCalculator.getMinSpeedText());
      maxAltitudeView.setText(mCalculator.getMaxAltitudeText());
      minAltitudeView.setText(mCalculator.getMinAltitudeText());
      avgSpeedView.setText(mCalculator.getAvgSpeedText());
      distanceView.setText(mCalculator.getDistanceText());
      starttimeView.setText(Long.toString(mCalculator.getStarttime()));
      endtimeView.setText(Long.toString(mCalculator.getEndtime()));
      String titleFormat = getString(R.string.stat_title);
      setTitle(String.format(titleFormat, mCalculator.getTracknameText()));
```

```
      waypointsView.setText(mCalculator.getWaypointsText());

      calculating = false;
   }
}
```

UploadToServer.java

```
package nl.sogeti.android.gpstracker.actions;

import java.io.DataOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.net.HttpURLConnection;
import java.net.URL;

import nl.sogeti.android.gpstracker.util.Constants;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.AlertDialog.Builder;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.DialogInterface.OnClickListener;
import android.content.Intent;
import android.os.Bundle;
import android.os.Looper;
import android.util.Log;
import android.widget.EditText;
import android.widget.Toast;

/**
 * Activity to upload a track to Main Server
 *
 * @author Alexandros Ntakas
 */
public class UploadToServer extends Activity {
   private String filepath;
   private String contentType;
   private Context context;

   public void onCreate(Bundle savedInstanceState) {

      super.onCreate(savedInstanceState);

      context = this;

      Builder builder = new AlertDialog.Builder(context);
      builder.setTitle("Upload to Server");
      builder.setMessage("Would you like to upload the track file to Main Server?");
      builder.setPositiveButton("YES", YESListener);
      builder.setNegativeButton("NO", NOListener);

      filepath = getIntent().getStringExtra(Constants.filePath);
      contentType = getIntent().getStringExtra(Constants.contentType);

      builder.show();
   }

   private OnClickListener YESListener = new OnClickListener() {
```

```
      public void onClick(DialogInterface dialog, int which) {
         AlertDialog.Builder builder = new AlertDialog.Builder(context);
         builder.setTitle("Insert Server IP to upload file");

         final EditText inputServerIP = new EditText(context);
         inputServerIP.setText(Constants.NTUAServerIP);
         builder.setView(inputServerIP);

         builder.setPositiveButton("OK",
               new DialogInterface.OnClickListener() {
                  public void onClick(DialogInterface dialog, int whichButton) {
                     HttpURLConnection connection = null;
                     DataOutputStream outputStream = null;
                     String ServerIP = Constants.NTUAServerIP; // default
                     uploadingDialog uploadingDiagol;

                     String pathToOurFile = filepath;
                     Log.e("UploadToServer", "pathToOurFile: " + pathToOurFile);
                     // Obtain user's input (Server IP)
                     ServerIP = inputServerIP.getText().toString();
                     Log.e("UploadToServer", "Server IP:" + ServerIP);

                     String urlServer = "http://" + ServerIP + ":80" +
"/handle_upload.php";
                     Log.e("UploadToServer", "urlServer: " + urlServer);
                     String lineEnd = "\r\n";
                     String twoHyphens = "--";
                     String boundary = "*****";

                     int bytesRead, bytesAvailable, bufferSize;
                     byte[] buffer;
                     int maxBufferSize = 1 * 1024 * 1024;

                     try {

                        FileInputStream fileInputStream = new FileInputStream(
                              new File(pathToOurFile));

                        uploadingDiagol = new uploadingDialog();
                        new Thread(uploadingDiagol).start();

                        URL url = new URL(urlServer);
                        connection = (HttpURLConnection) url.openConnection();
                        // Allow Inputs & Outputs
                        connection.setDoInput(true);
                        connection.setDoOutput(true);
                        connection.setUseCaches(false);

                        // Enable POST method
                        connection.setRequestMethod("POST");
                        connection.setRequestProperty("Connection", "Keep-Alive");
                        connection.setRequestProperty("Content-Type",
                              "multipart/form-data;boundary="  + boundary);

                        outputStream = new DataOutputStream(connection
                              .getOutputStream());

                        outputStream.writeBytes(twoHyphens + boundary + lineEnd);
                        outputStream.writeBytes("Content-Disposition: form-data;
name=\"uploadedfile\";filename=\""
                                    + pathToOurFile
                                    + "\""
```

```
                                    + lineEnd);
                        outputStream.writeBytes(lineEnd);

                        bytesAvailable = fileInputStream.available();
                        bufferSize = Math.min(bytesAvailable, maxBufferSize);
                        buffer = new byte[bufferSize];

                        // Read file
                        bytesRead = fileInputStream.read(buffer, 0, bufferSize);

                        while (bytesRead > 0) {
                           outputStream.write(buffer, 0, bufferSize);
                           bytesAvailable = fileInputStream.available();
                           bufferSize = Math.min(bytesAvailable,  maxBufferSize);
                           bytesRead = fileInputStream.read(buffer, 0, bufferSize);
                        }

                        outputStream.writeBytes(lineEnd);
                        outputStream.writeBytes(twoHyphens + boundary  + twoHyphens +
lineEnd);

                        // Responses from server (code and message)
                        int serverResponseCode = connection.getResponseCode();
                        String serverResponseMessage = connection.getResponseMessage();
                        Log.e("UploadToServer", "response code:"
                              + serverResponseCode + " message:"
                              + serverResponseMessage);

                        fileInputStream.close();
                        outputStream.flush();
                        outputStream.close();
                        connection.disconnect();

                        uploadingDiagol.stopuploadingDialog();
                        Toast.makeText(
                              context,
                              "File has been uploaded to Main Server",
                              10000).show();
                        finishUpload();
                     } catch (Exception ex) {
                        Log.e("UploadToServer", "" + ex, ex);
                        Toast.makeText(context,
                              "An error occured: " + ex.toString(),
                              10000).show();
                        finishUpload();
                     }
                  }
               });

         builder.setNegativeButton("Cancel",
               new DialogInterface.OnClickListener() {
                  public void onClick(DialogInterface dialog,
                        int whichButton) {
                     finishUpload();
                  }
               });

         builder.show();
      }
   };

   private OnClickListener NOListener = new OnClickListener() {
```

```
      public void onClick(DialogInterface dialog, int which) {
         finishUpload();
      }
   };

   public void finishUpload() {
      Intent UploadReply = new Intent();
      UploadReply.putExtra(Constants.filePath, filepath);
      UploadReply.putExtra(Constants.contentType, contentType);
      setResult(0, UploadReply);
      finish();
   }

   private class uploadingDialog extends Thread {
      ProgressDialog progressdialog;

      public void run() {
         Looper.prepare();
         progressdialog = ProgressDialog.show(context, "Upload To Server","Uploading...");
         Looper.loop();
      }

      public void stopuploadingDialog() {
         progressdialog.dismiss();
      }
   }
}
```

GraphCanvas.java

```
package nl.sogeti.android.gpstracker.actions.utils;

import java.text.DateFormat;
import java.util.Date;

import nl.sogeti.android.gpstracker.R;
import nl.sogeti.android.gpstracker.db.GPStracking.Segments;
import nl.sogeti.android.gpstracker.db.GPStracking.Tracks;
import nl.sogeti.android.gpstracker.db.GPStracking.Waypoints;
import nl.sogeti.android.gpstracker.util.UnitsI18n;
import android.content.ContentResolver;
import android.content.Context;
import android.content.res.Resources;
import android.database.Cursor;
import android.graphics.Bitmap;
import android.graphics.Bitmap.Config;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.CornerPathEffect;
import android.graphics.DashPathEffect;
import android.graphics.Paint;
import android.graphics.Path;
import android.graphics.Typeface;
import android.location.Location;
import android.net.Uri;
import android.util.AttributeSet;
import android.view.View;

/**
 * Calculate and draw graphs of track data
```

```
 *
 * @author Alexandros Ntakas
 */
public class GraphCanvas extends View {
   @SuppressWarnings("unused")
   private static final String TAG = "OGT.GraphCanvas";
   public static final int TIMESPEEDGRAPH = 0;
   public static final int DISTANCESPEEDGRAPH = 1;
   public static final int TIMEALTITUDEGRAPH = 2;
   public static final int DISTANCEALTITUDEGRAPH = 3;
   private Uri mUri;
   private Bitmap mRenderBuffer;
   private Canvas mRenderCanvas;
   private Context mContext;
   private UnitsI18n mUnits;
   private int mGraphType = TIMESPEEDGRAPH;
   private long mEndTime;
   private long mStartTime;
   private double mDistance;
   private int mHeight;
   private int mWidth;
   private int mMinAxis;
   private int mMaxAxis;
   private double mMinAlititude;
   private double mMaxAlititude;
   private double mMaxSpeed;
   private double mDistanceDrawn;
   private long mStartTimeDrawn;
   private long mEndTimeDrawn;
   private boolean calculating;
   float density = Resources.getSystem().getDisplayMetrics().density;

   private Paint whiteText;
   private Paint ltgreyMatrixDashed;
   private Paint greenGraphLine;
   private Paint dkgreyMatrixLine;
   private Paint whiteCenteredText;
   private Paint dkgrayLargeType;

   private String NMC;

   public GraphCanvas(Context context, AttributeSet attrs) {
      this(context, attrs, 0);
   }

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

      mContext = context;

      whiteText = new Paint();
      whiteText.setColor(Color.WHITE);
      whiteText.setAntiAlias(true);
      whiteText.setTextSize((int) (density * 12));

      whiteCenteredText = new Paint();
      whiteCenteredText.setColor(Color.WHITE);
      whiteCenteredText.setAntiAlias(true);
      whiteCenteredText.setTextAlign(Paint.Align.CENTER);
      whiteCenteredText.setTextSize((int) (density * 12));

      ltgreyMatrixDashed = new Paint();
```

```
      ltgreyMatrixDashed.setColor(Color.LTGRAY);
      ltgreyMatrixDashed.setStrokeWidth(1);
      ltgreyMatrixDashed.setPathEffect(new DashPathEffect(new float[] { 2, 4 }, 0));

      greenGraphLine = new Paint();
      greenGraphLine.setPathEffect(new CornerPathEffect(8));
      greenGraphLine.setStyle(Paint.Style.STROKE);
      greenGraphLine.setStrokeWidth(4);
      greenGraphLine.setAntiAlias(true);
      greenGraphLine.setColor(Color.GREEN);

      dkgreyMatrixLine = new Paint();
      dkgreyMatrixLine.setColor(Color.DKGRAY);
      dkgreyMatrixLine.setStrokeWidth(2);

      dkgrayLargeType = new Paint();
      dkgrayLargeType.setColor(Color.LTGRAY);
      dkgrayLargeType.setAntiAlias(true);
      dkgrayLargeType.setTextAlign(Paint.Align.CENTER);
      dkgrayLargeType.setTextSize((int) (density * 21));
      dkgrayLargeType.setTypeface(Typeface.DEFAULT_BOLD);
   }

   /**
    * Set the dataset for which to draw data. Also provide hints and helpers.
    *
    * @param uri
    * @param startTime
    * @param endTime
    * @param distance
    * @param minAlititude
    * @param maxAlititude
    * @param maxSpeed
    * @param units
    */
   public void setData(Uri uri, StatisticsCalulator calc) {
      boolean rerender = false;
      if (uri.equals(mUri)) {
         double distanceDrawnPercentage = mDistanceDrawn / mDistance;
         double duractionDrawnPercentage = (double) ((1d + mEndTimeDrawn -
mStartTimeDrawn) / (1d + mEndTime - mStartTime));
         rerender = distanceDrawnPercentage < 0.99d  || duractionDrawnPercentage < 0.99d;
      } else {
         rerender = true;
      }

      mUri = uri;
      mUnits = calc.getUnits();
      mMinAlititude = calc.getMinAltitude();
      mMaxAlititude = calc.getMaxAltitude();
      mMaxSpeed = calc.getMaxSpeed();
      mStartTime = calc.getStarttime();
      mEndTime = calc.getEndtime();
      mDistance = calc.getDistanceTraveled();

      ContentResolver resolver = this.mContext.getApplicationContext()
            .getContentResolver();
      Cursor trackCursor = null;
      trackCursor = resolver.query(uri, new String[] { Tracks.NMC }, null,null, null);

      if (trackCursor != null) {
         trackCursor.moveToFirst();
```

```
      if (trackCursor.getString(0).equals("DOWNLOAD")) {
         NMC = "DOWNLOAD";
      } else if (trackCursor.getString(0).equals("PING")) {
         NMC = "PING";
      }
   } else {
      NMC = "DOWNLOAD";
   }

   if (rerender && !calculating) {
      renderGraph();
   }
   postInvalidate();
}

public void setType(int graphType) {
   if (mGraphType != graphType) {
      mGraphType = graphType;
      renderGraph();
   }
}

public int getType() {
   return mGraphType;
}

@Override
protected void onSizeChanged(int w, int h, int oldw, int oldh) {
   super.onSizeChanged(w, h, oldw, oldh);

   if (mRenderBuffer == null || mRenderBuffer.getWidth() != w
         || mRenderBuffer.getHeight() != h) {
      mRenderBuffer = Bitmap.createBitmap(w, h, Config.ARGB_8888);
      mRenderCanvas = new Canvas(mRenderBuffer);
      renderGraph();
   }
}

@Override
protected void onDraw(Canvas canvas) {
   super.onDraw(canvas);
   canvas.drawBitmap(mRenderBuffer, 0, 0, null);
}

private void renderGraph() {

   calculating = true;
   if (mRenderBuffer != null) {
      mRenderBuffer.eraseColor(Color.TRANSPARENT);
      switch (mGraphType) {
      case (TIMESPEEDGRAPH):
         setupSpeedAxis();
         drawGraphType();
         drawTimeAxisGraphOnCanvas(new String[] { Waypoints.TIME,Waypoints.SPEED });
         drawSpeedsTexts();
         drawTimeTexts();
         break;
      case (DISTANCESPEEDGRAPH):
         setupSpeedAxis();
         drawGraphType();
         drawDistanceAxisGraphOnCanvas(new String[] {
               Waypoints.LONGITUDE, Waypoints.LATITUDE,
```

```
                Waypoints.SPEED });
            drawSpeedsTexts();
            drawDistanceTexts();
            break;
         default:
            break;
         }
         mDistanceDrawn = mDistance;
         mStartTimeDrawn = mStartTime;
         mEndTimeDrawn = mEndTime;
      }
      calculating = false;
   }

   /**
    *
    * @param params
    */
   private void drawDistanceAxisGraphOnCanvas(String[] params) {
      ContentResolver resolver = mContext.getApplicationContext().getContentResolver();
      Uri segmentsUri = Uri.withAppendedPath(mUri, "/segments");
      Uri waypointsUri = null;
      Cursor segments = null;
      Cursor waypoints = null;
      double[][] values;
      int[][] valueDepth;
      double distance = 1;
      try {
         segments = resolver.query(segmentsUri,
               new String[] { Segments._ID }, null, null, null);
         int segmentCount = segments.getCount();
         values = new double[segmentCount][mWidth];
         valueDepth = new int[segmentCount][mWidth];
         if (segments.moveToFirst()) {
            for (int segment = 0; segment < segmentCount; segment++) {
               segments.moveToPosition(segment);
               long segmentId = segments.getLong(0);
               waypointsUri = Uri.withAppendedPath(segmentsUri, segmentId + "/waypoints");
               try {
                  waypoints = resolver.query(waypointsUri, params, null,  null, null);
                  if (waypoints.moveToFirst()) {
                     Location lastLocation = null;
                     Location currentLocation = null;
                     do {
                        currentLocation = new Location(this.getClass()  .getName());
                        currentLocation.setLongitude(waypoints.getDouble(0));
                        currentLocation.setLatitude(waypoints  .getDouble(1));
                        if (lastLocation != null) {
                           distance += lastLocation.distanceTo(currentLocation);
                        }
                        lastLocation = currentLocation;
                        double value = waypoints.getDouble(2);

                        if (segment < values.length) {
                           int x = (int) ((distance) * (mWidth - 1) / mDistance);
                           if (x < valueDepth[segment].length) {
                              valueDepth[segment][x]++;
                              values[segment][x] = values[segment][x]
                                    + ((value - values[segment][x]) /
valueDepth[segment][x]);
                           }
                        }
```

```
                    } while (waypoints.moveToNext());
                 }
              } finally {
                 if (waypoints != null) {
                    waypoints.close();
                 }
              }
           }
        }
     } finally {
        if (segments != null) {
           segments.close();
        }
     }
     for (int segment = 0; segment < values.length; segment++) {
        for (int x = 0; x < values[segment].length; x++) {
           if (valueDepth[segment][x] > 0) {
              values[segment][x] = translateValue(values[segment][x]);
           }
        }
     }
     drawGraph(values, valueDepth);
  }

  private void drawTimeAxisGraphOnCanvas(String[] params) {
     ContentResolver resolver = mContext.getApplicationContext().getContentResolver();
     Uri segmentsUri = Uri.withAppendedPath(mUri, "/segments");
     Uri waypointsUri = null;
     Cursor segments = null;
     Cursor waypoints = null;
     long duration = 1 + mEndTime - mStartTime;
     double[][] values;
     int[][] valueDepth;
     try {
        segments = resolver.query(segmentsUri,
              new String[] { Segments._ID }, null, null, null);
        int segmentCount = segments.getCount();
        values = new double[segmentCount][mWidth];
        valueDepth = new int[segmentCount][mWidth];
        if (segments.moveToFirst()) {
           for (int segment = 0; segment < segmentCount; segment++) {
              segments.moveToPosition(segment);
              long segmentId = segments.getLong(0);
              waypointsUri = Uri.withAppendedPath(segmentsUri, segmentId + "/waypoints");
              try {
                 waypoints = resolver.query(waypointsUri, params, null,  null, null);
                 if (waypoints.moveToFirst()) {
                    do {
                       long time = waypoints.getLong(0);
                       double value = waypoints.getDouble(1);

                       if (segment < values.length) {
                          int x = (int) ((time - mStartTime) * (mWidth - 1) / duration);
                          if (x < valueDepth[segment].length) {
                             valueDepth[segment][x]++;
                             values[segment][x] = values[segment][x]
                                   + ((value - values[segment][x]) /
valueDepth[segment][x]);
                          }
                       }
                    } while (waypoints.moveToNext());
                 }
```

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

      }
   } finally {
      if (segments != null) {
         segments.close();
      }
   }
   for (int p = 0; p < values.length; p++) {
      for (int x = 0; x < values[p].length; x++) {
         if (valueDepth[p][x] > 0) {
            values[p][x] = translateValue(values[p][x]);
         }
      }
   }
   drawGraph(values, valueDepth);
}

private void setupAltitudeAxis() {
   mMinAxis = 4 * (int) mUnits.conversionFromMeterToHeight(mMinAlititude / 4);
   mMaxAxis = 4 + 4 * (int) mUnits.conversionFromMeterToHeight(mMaxAlititude / 4);

   mWidth = mRenderCanvas.getWidth() - 5;
   mHeight = mRenderCanvas.getHeight() - 10;
}

private void setupSpeedAxis() {
   mMinAxis = 0;
   mMaxAxis = 4 + 4 * (int) (mMaxSpeed / 4);

   mWidth = mRenderCanvas.getWidth() - 5;
   mHeight = mRenderCanvas.getHeight() - 10;
}

private void drawAltitudesTexts() {
   mRenderCanvas.drawText(
         String.format("%d %s", mMinAxis, mUnits.getHeightUnit()), 8,
         mHeight, whiteText);
   mRenderCanvas.drawText(
         String.format("%d %s", (mMaxAxis + mMinAxis) / 2,
               mUnits.getHeightUnit()), 8, 5 + mHeight / 2, whiteText);
   mRenderCanvas.drawText(
         String.format("%d %s", mMaxAxis, mUnits.getHeightUnit()), 8,  15, whiteText);
}

private void drawSpeedsTexts() {
   mRenderCanvas.drawText(
         String.format("%d %s", mMinAxis, mUnits.getSpeedUnit(NMC)), 8,
         mHeight, whiteText);
   mRenderCanvas.drawText(
         String.format("%d %s", (mMaxAxis + mMinAxis) / 2,
               mUnits.getSpeedUnit(NMC)), 8, 3 + mHeight / 2,  whiteText);
   mRenderCanvas.drawText(
         String.format("%d %s", mMaxAxis, mUnits.getSpeedUnit(NMC)), 8,
         7 + whiteText.getTextSize(), whiteText);
}
```

```
private void drawTimeTexts() {
   DateFormat timeInstance = DateFormat.getTimeInstance(DateFormat.SHORT);
   String start = timeInstance.format(new Date(mStartTime));
   String half = timeInstance.format(new Date((mEndTime + mStartTime) / 2));
   String end = timeInstance.format(new Date(mEndTime));

   Path yAxis;
   yAxis = new Path();
   yAxis.moveTo(5, 5 + mHeight / 2);
   yAxis.lineTo(5, 5);
   mRenderCanvas.drawTextOnPath(String.format(start), yAxis, 0,
         whiteCenteredText.getTextSize(), whiteCenteredText);
   yAxis = new Path();
   yAxis.moveTo(5 + mWidth / 2, 5 + mHeight / 2);
   yAxis.lineTo(5 + mWidth / 2, 5);
   mRenderCanvas.drawTextOnPath(String.format(half), yAxis, 0, -3,
whiteCenteredText);
   yAxis = new Path();
   yAxis.moveTo(5 + mWidth - 1, 5 + mHeight / 2);
   yAxis.lineTo(5 + mWidth - 1, 5);
   mRenderCanvas.drawTextOnPath(String.format(end), yAxis, 0, -3, whiteCenteredText);
}

private void drawGraphType() {
   // float density = Resources.getSystem().getDisplayMetrics().density;
   String text;
   switch (mGraphType) {
   case (TIMESPEEDGRAPH):
      if (NMC.equals("DOWNLOAD")) {
         text = mContext.getResources().getString(R.string.graphtype_timebitrate);
      } else {
         text = mContext.getResources().getString(R.string.graphtype_timelatency);
      }
      break;
   case (DISTANCESPEEDGRAPH):
      if (NMC.equals("DOWNLOAD")) {
         text = mContext.getResources().getString(R.string.graphtype_distancebitrate);
      } else {
         text = mContext.getResources().getString(R.string.graphtype_distancelatency);
      }
      break;
   default:
      text = "UNKNOWN GRAPH TYPE";
      break;
   }
   mRenderCanvas.drawText(text, 5 + mWidth / 2, 5 + mHeight / 8, dkgrayLargeType);

}

private void drawDistanceTexts() {
   String start = String.format("%.0f %s", mUnits.conversionFromMeter(0),
         mUnits.getDistanceUnit());
   String half = String.format("%.0f %s", mUnits.conversionFromMeter(mDistance / 2),
         mUnits.getDistanceUnit());
   String end = String.format("%.0f %s", mUnits.conversionFromMeter(mDistance),
               mUnits.getDistanceUnit());

   Path yAxis;
   yAxis = new Path();
   yAxis.moveTo(5, 5 + mHeight / 2);
   yAxis.lineTo(5, 5);
   mRenderCanvas.drawTextOnPath(String.format(start), yAxis, 0,
```

```
            whiteText.getTextSize(), whiteText);
      yAxis = new Path();
      yAxis.moveTo(5 + mWidth / 2, 5 + mHeight / 2);
      yAxis.lineTo(5 + mWidth / 2, 5);
      mRenderCanvas.drawTextOnPath(String.format(half), yAxis, 0, -3,  whiteText);
      yAxis = new Path();
      yAxis.moveTo(5 + mWidth - 1, 5 + mHeight / 2);
      yAxis.lineTo(5 + mWidth - 1, 5);
      mRenderCanvas.drawTextOnPath(String.format(end), yAxis, 0, -3, whiteText);
   }

   private double translateValue(double val) {
      switch (mGraphType) {
      case (TIMESPEEDGRAPH):
      case (DISTANCESPEEDGRAPH):
         break;
      default:
         break;
      }
      return val;

   }

   private void drawGraph(double[][] values, int[][] valueDepth) {
      // Matrix
      // Horizontals
      mRenderCanvas.drawLine(5, 5, 5 + mWidth, 5, ltgreyMatrixDashed); // top
      mRenderCanvas.drawLine(5, 5 + mHeight / 4, 5 + mWidth, 5 + mHeight / 4,
            ltgreyMatrixDashed); // 2nd
      mRenderCanvas.drawLine(5, 5 + mHeight / 2, 5 + mWidth, 5 + mHeight / 2,
            ltgreyMatrixDashed); // middle
      mRenderCanvas.drawLine(5, 5 + mHeight / 4 * 3, 5 + mWidth,
            5 + mHeight / 4 * 3, ltgreyMatrixDashed); // 3rd
      // Verticals
      mRenderCanvas.drawLine(5 + mWidth / 4, 5, 5 + mWidth / 4, 5 + mHeight,
            ltgreyMatrixDashed); // 2nd
      mRenderCanvas.drawLine(5 + mWidth / 2, 5, 5 + mWidth / 2, 5 + mHeight,
            ltgreyMatrixDashed); // middle
      mRenderCanvas.drawLine(5 + mWidth / 4 * 3, 5, 5 + mWidth / 4 * 3,
            5 + mHeight, ltgreyMatrixDashed); // 3rd
      mRenderCanvas.drawLine(5 + mWidth - 1, 5, 5 + mWidth - 1, 5 + mHeight,
            ltgreyMatrixDashed); // right

      // The line
      Path mPath;
      int emptyValues = 0;
      mPath = new Path();
      for (int p = 0; p < values.length; p++) {
         int start = 0;
         while (valueDepth[p][start] == 0 && start < values[p].length - 1) {
            start++;
         }

         mPath.moveTo(
               (float) start + 5,
               5f + (float) (mHeight - ((values[p][start] - mMinAxis) * mHeight)
                     / (mMaxAxis - mMinAxis)));
         for (int x = start; x < values[p].length; x++) {
            double y = mHeight - ((values[p][x] - mMinAxis) * mHeight)
                  / (mMaxAxis - mMinAxis);
            if (valueDepth[p][x] > 0) {
               if (emptyValues > mWidth / 10) {
```

```
                mPath.moveTo((float) x + 5, (float) y + 5);
             } else {
                mPath.lineTo((float) x + 5, (float) y + 5);
             }
             emptyValues = 0;
          } else {
             emptyValues++;
          }
       }
    }
    mRenderCanvas.drawPath(mPath, greenGraphLine);

    // Axis's
    mRenderCanvas.drawLine(5, 5, 5, 5 + mHeight, dkgreyMatrixLine);
    mRenderCanvas.drawLine(5, 5 + mHeight, 5 + mWidth, 5 + mHeight,  dkgreyMatrixLine);

 }

}
```

StatisticsCalulator.java

```
package nl.sogeti.android.gpstracker.actions.utils;

import nl.sogeti.android.gpstracker.db.GPStracking.Segments;
import nl.sogeti.android.gpstracker.db.GPStracking.Tracks;
import nl.sogeti.android.gpstracker.db.GPStracking.Waypoints;
import nl.sogeti.android.gpstracker.util.UnitsI18n;
import android.content.ContentResolver;
import android.content.Context;
import android.database.Cursor;
import android.location.Location;
import android.net.Uri;
import android.util.Log;

/**
 * A class containing methods for result processing
 *
 * @author Alexandros Ntakas
 */
public class StatisticsCalulator {

   private Context mContext;
   private String overallavgSpeedText = "Unknown";
   private String avgSpeedText = "Unknown";
   private String maxSpeedText = "Unknown";
   private String minSpeedText = "Unknown";
   private String maxAltitudeText = "Unknown";
   private String minAltitudeText = "Unknown";
   private String tracknameText = "Unknown";
   private String waypointsText = "Unknown";
   private String distanceText = "Unknown";
   private long mStarttime = -1;
   private long mEndtime = -1;
   private UnitsI18n mUnits;
   private double mMaxSpeed;
   private double mMinSpeed;
   private String NMC = null;
   private long NotAsinglePingSuccessful = 0;
   private double mMaxAltitude;
   private double mMinAltitude;
```

```
private double mDistanceTraveled;
private long mDuration;

public StatisticsCalulator(Context ctx, UnitsI18n units) {
   mContext = ctx;
   mUnits = units;
}

public void updateCalculations(Uri mTrackUri) {
   mMaxSpeed = 0;
   mMinSpeed = 0;
   mMaxAltitude = 0;
   mMinAltitude = 0;
   mDistanceTraveled = 0f;
   long duration = 1;

   float TotalWaypointsSpeed = 0;
   float AverageWaypointsSpeed = 0;
   long nrWaypoints = 0;
   Cursor trackCursor = null;
   ContentResolver resolver = mContext.getContentResolver();
   Cursor waypointsCursor = null;
   Cursor waypointsCursor2 = null;

   try {
      waypointsCursor = resolver.query(
            Uri.withAppendedPath(mTrackUri, "waypoints"), new String[] {
                  "max  (" + Waypoints.TABLE + "." + Waypoints.SPEED+ ")",
                  "max  (" + Waypoints.TABLE + "." + Waypoints.ALTITUDE + ")",
                  "min  (" + Waypoints.TABLE + "." + Waypoints.ALTITUDE + ")",
                  "count(" + Waypoints.TABLE + "." + Waypoints._ID      + ")"
      }, null, null, null);
      if (waypointsCursor.moveToLast()) {
         mMaxSpeed = waypointsCursor.getDouble(0);
         mMaxAltitude = waypointsCursor.getDouble(1);
         mMinAltitude = waypointsCursor.getDouble(2);
         nrWaypoints = waypointsCursor.getLong(3);
         waypointsText = nrWaypoints + "";
      }

      // We added a second cursor (waypointsCursor2) to handle case
      // latency == 0 -> min results
      trackCursor = resolver.query(mTrackUri, new String[] { Tracks.NAME,
            Tracks.NMC }, null, null, null);

      if (trackCursor != null) {
         trackCursor.moveToFirst();
         NMC = trackCursor.getString(1);
         Log.e("StatisticsCalculator", "NMC: " + NMC);
      }

      if (NMC.equals("DOWNLOAD")) {
         waypointsCursor2 = resolver.query(
               Uri.withAppendedPath(mTrackUri, "waypoints"),
               new String[] { "min  (" + Waypoints.TABLE + "."
                     + Waypoints.SPEED + ")" }, null, null, null);
         waypointsCursor2.moveToLast();
         mMinSpeed = waypointsCursor2.getDouble(0);
      } else if (NMC.equals("PING")) {
         String selection = "speed > 0"; // Case latency == 0 (we don't
                                 // want this to be considered as
                                 // min value)
```

```
      waypointsCursor2 = resolver.query(
            Uri.withAppendedPath(mTrackUri, "waypoints"),
            new String[] { "min  (" + Waypoints.TABLE + "."
                  + Waypoints.SPEED + ")" }, selection, null,
            null);
      waypointsCursor2.moveToLast();
      mMinSpeed = waypointsCursor2.getDouble(0);
   }
} finally {
   if (waypointsCursor != null) {
      waypointsCursor.close();
   }

   if (waypointsCursor2 != null) {
      waypointsCursor2.close();
   }
}

try {

   if (trackCursor.moveToLast()) {
      tracknameText = trackCursor.getString(0);
   }

} finally {
   if (trackCursor != null) {
      trackCursor.close();
   }
}
Cursor segments = null;
Location lastLocation = null;
Location currentLocation = null;
try {
   Uri segmentsUri = Uri.withAppendedPath(mTrackUri, "segments");
   segments = resolver.query(segmentsUri,
         new String[] { Segments._ID }, null, null, null);
   if (segments.moveToFirst()) {
      do {
         long segmentsId = segments.getLong(0);
         Cursor waypoints = null;
         try {
            Uri waypointsUri = Uri.withAppendedPath(segmentsUri,
                  segmentsId + "/waypoints");

            waypoints = resolver.query(waypointsUri, new String[] {
                  Waypoints._ID, Waypoints.TIME,
                  Waypoints.LONGITUDE, Waypoints.LATITUDE,
                  Waypoints.SPEED }, null, null, null);
            if (waypoints.moveToFirst()) {
               do {
                  if (mStarttime < 0) {
                     mStarttime = waypoints.getLong(1);
                  }
                  currentLocation = new Location(this.getClass()  .getName());
                  currentLocation.setTime(waypoints.getLong(1));
                  currentLocation.setLongitude(waypoints.getDouble(2));
                  currentLocation.setLatitude(waypoints.getDouble(3));
                  currentLocation.setSpeed(waypoints.getFloat(4));

                  // At first lastLocation == null
                  if (lastLocation != null) {
```

```
                        mDistanceTraveled += lastLocation.distanceTo(currentLocation);
                        duration += currentLocation.getTime() - lastLocation.getTime();
                     }

                     // Case "Not A single Ping was successful", latency == 0
                     float locspeed = currentLocation.getSpeed();
                     if (NMC.equals("DOWNLOAD")) {
                        TotalWaypointsSpeed += locspeed;
                     } else if (NMC.equals("PING")) {
                        if (locspeed > 0) {
                           TotalWaypointsSpeed += locspeed;
                        } else {
                           NotAsinglePingSuccessful++;
                        }
                     }
                     lastLocation = currentLocation;

                  } while (waypoints.moveToNext());
                  mEndtime = lastLocation.getTime();
                  mDuration = mEndtime - mStarttime;

                  // Computing Average Speed:
                  // (Sum of Waypoints Speed / number of Waypoints)
                  if (nrWaypoints != 0) {
                     if (NMC.equals("DOWNLOAD")) {
                        AverageWaypointsSpeed = (float) TotalWaypointsSpeed  /
nrWaypoints;
                     } else if (NMC.equals("PING")) {
                        AverageWaypointsSpeed = (float) TotalWaypointsSpeed
                              / (nrWaypoints - NotAsinglePingSuccessful);
                     }
                     Log.e("Statistics Calculator",
                           "AverageWaypointsSpeed:"
                                 + AverageWaypointsSpeed);
                     Log.e("Statistics Calculator", "nrWaypoints"+ nrWaypoints);
                  } else {
                     Log.e("Statistics Calculator ERROR:",
                           "nrWaypoints == 0");
                     AverageWaypointsSpeed = 0;
                  }
               }
            } finally {
               if (waypoints != null) {
                  waypoints.close();
               }
            }
            lastLocation = null;
         } while (segments.moveToNext());
      }
   } finally {
      if (segments != null) {
         segments.close();
      }
   }

   double maxspeed = mMaxSpeed;
   double minspeed = mMinSpeed;
   double maxAltitude = mUnits.conversionFromMeterToHeight(mMaxAltitude);
   double minAltitude = mUnits.conversionFromMeterToHeight(mMinAltitude);
   double avgSpeedfl = AverageWaypointsSpeed;
   double traveled = mUnits.conversionFromMeter(mDistanceTraveled);
```

```
   avgSpeedText = String.format("%.0f %s", avgSpeedfl, mUnits.getSpeedUnit(NMC));
   distanceText = String.format("%.2f %s", traveled, mUnits.getDistanceUnit());
   maxSpeedText = String.format("%.0f %s", maxspeed, mUnits.getSpeedUnit(NMC));
   minSpeedText = String.format("%.0f %s", minspeed, mUnits.getSpeedUnit(NMC));
   minAltitudeText = String.format("%.0f %s", minAltitude, mUnits.getHeightUnit());
   maxAltitudeText = String.format("%.0f %s", maxAltitude, mUnits.getHeightUnit());
}

/**
 * Get the overallavgSpeedText.
 *
 * @return Returns the overallavgSpeedText as a String.
 */
public String getOverallavgSpeedText() {
   return overallavgSpeedText;
}

/**
 * Get the avgSpeedText.
 *
 * @return Returns the avgSpeedText as a String.
 */
public String getAvgSpeedText() {
   return avgSpeedText;
}

/**
 * Get the maxSpeedText.
 *
 * @return Returns the maxSpeedText as a String.
 */
public String getMaxSpeedText() {
   return maxSpeedText;
}

/**
 * Get the minSpeedText.
 *
 * @return Returns the minSpeedText as a String.
 */
public String getMinSpeedText() {
   return minSpeedText;
}

/**
 * Get the maxAltitudeText.
 *
 * @return Returns the maxAltitudeText as a String.
 */
public String getMaxAltitudeText() {
   return maxAltitudeText;
}

/**
 * Get the minAltitudeText.
 *
 * @return Returns the minAltitudeText as a String.
 */
public String getMinAltitudeText() {
   return minAltitudeText;
}
```

```
/**
 * Get the tracknameText.
 *
 * @return Returns the tracknameText as a String.
 */
public String getTracknameText() {
   return tracknameText;
}

/**
 * Get the waypointsText.
 *
 * @return Returns the waypointsText as a String.
 */
public String getWaypointsText() {
   return waypointsText;
}

/**
 * Get the distanceText.
 *
 * @return Returns the distanceText as a String.
 */
public String getDistanceText() {
   return distanceText;
}

/**
 * Get the starttime.
 *
 * @return Returns the starttime as a long.
 */
public long getStarttime() {
   return mStarttime;
}

/**
 * Get the endtime.
 *
 * @return Returns the endtime as a long.
 */
public long getEndtime() {
   return mEndtime;
}

/**
 * Get the maxSpeeddb.
 *
 * @return Returns the maxSpeeddb as a double.
 */
public double getMaxSpeed() {
   return mMaxSpeed;
}

/**
 * Get the maxAltitude.
 *
 * @return Returns the maxAltitude as a double.
 */
public double getMaxAltitude() {
   return mMaxAltitude;
}
```

```
   /**
    * Get the minAltitude.
    *
    * @return Returns the minAltitude as a double.
    */
   public double getMinAltitude() {
      return mMinAltitude;
   }

   /**
    * Get the distanceTraveled.
    *
    * @return Returns the distanceTraveled as a float.
    */
   public double getDistanceTraveled() {
      return mDistanceTraveled;
   }

   /**
    * Get the mUnits.
    *
    * @return Returns the mUnits as a UnitsI18n.
    */
   public UnitsI18n getUnits() {
      return mUnits;
   }

   public String getDurationText() {
      long s = mDuration / 1000;
      String duration = String.format("%dh:%02dm:%02ds", s / 3600,
            (s % 3600) / 60, (s % 60));

      return duration;
   }
}
```

XmlDBCreator.java

```
package nl.sogeti.android.gpstracker.actions.utils;

import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.TimeZone;

import nl.sogeti.android.gpstracker.R;
import nl.sogeti.android.gpstracker.actions.ShareTrack.ProgressMonitor;
import nl.sogeti.android.gpstracker.db.GPStracking.Segments;
import nl.sogeti.android.gpstracker.db.GPStracking.Tracks;
import nl.sogeti.android.gpstracker.db.GPStracking.Waypoints;
import nl.sogeti.android.gpstracker.util.Constants;

import org.xmlpull.v1.XmlSerializer;

import android.content.ContentResolver;
import android.content.Context;
```

```
import android.database.Cursor;
import android.net.Uri;
import android.os.Environment;
import android.os.Looper;
import android.util.Log;
import android.util.Xml;
import android.widget.Toast;

/**
 * Create the xml file with route's measurements (specific structure that Main
 * Server can handle and insert values into Server's database)
 *
 * @author Alexandros Ntakas
 */
public class XmlDBCreator extends XmlCreator {

   public static final SimpleDateFormat ZULU_DATE_FORMAT = new SimpleDateFormat(
         "yyyy-MM-dd'T'HH:mm:ss'Z'");
   static {
      TimeZone utc = TimeZone.getTimeZone("UTC");
      ZULU_DATE_FORMAT.setTimeZone(utc); // ZULU_DATE_FORMAT format ends with
                                 // Z for UTC so make that true
   }

   private String TAG = "OGT.GpxCreator";

   String NMC;
   String Area;
   String Serverip;
   String Packetsize;
   String Pingattempts;
   String Trackname;
   Date TrackTime;
   Boolean isFirstLine = true;

   public XmlDBCreator(Context context, Uri trackUri,
         String chosenBaseFileName, ProgressMonitor listener) {
      super(context, trackUri, chosenBaseFileName, listener);
   }

   public void run() {
      Looper.prepare();

      String xmlFilePath;
      if (fileName.endsWith(".xml")) {
         setExportDirectoryPath(Environment.getExternalStorageDirectory()
               + Constants.EXTERNAL_DIR
               + fileName.substring(0, fileName.length() - 4));

         xmlFilePath = getExportDirectoryPath() + "/" + fileName;
      } else {
         setExportDirectoryPath(Environment.getExternalStorageDirectory()
               + Constants.EXTERNAL_DIR + fileName);
         xmlFilePath = getExportDirectoryPath() + "/" + fileName + ".xml";
      }

      new File(getExportDirectoryPath()).mkdirs();

      if (mProgressListener != null) {
         mProgressListener.startNotification();
         mProgressListener.updateNotification(getProgress(), getGoal());
      }
```

```
      String resultFilename = null;
      FileOutputStream fos = null;
      BufferedOutputStream buf = null;
      try {
         verifySdCardAvailibility();

         XmlSerializer serializer = Xml.newSerializer();
         File xmlFile = new File(xmlFilePath);
         fos = new FileOutputStream(xmlFile);
         buf = new BufferedOutputStream(fos, 8192);
         serializer.setOutput(buf, "UTF-8");

         serializeTrack(mTrackUri, serializer);
         buf.close();
         buf = null;
         fos.close();
         fos = null;

         if (isNeedsBundling()) {
            resultFilename = bundlingMediaAndXml(xmlFile.getParentFile().getName(),
".zip");
         } else {
            File finalFile = new File(
                  Environment.getExternalStorageDirectory()
                        + Constants.EXTERNAL_DIR + "/"
                        + xmlFile.getName());
            xmlFile.renameTo(finalFile);
            resultFilename = finalFile.getAbsolutePath();

            XmlCreator.deleteRecursive(xmlFile.getParentFile());
         }

         fileName = new File(resultFilename).getName();

         CharSequence text = mContext.getString(R.string.ticker_stored)
               + " \"" + fileName + "\" ";
         Toast toast = Toast.makeText(mContext.getApplicationContext(),
               text, Toast.LENGTH_LONG);
         toast.show();
      } catch (FileNotFoundException e) {
         Log.e(TAG, "Unable to save ", e);
         CharSequence text = mContext.getString(R.string.ticker_failed)
               + " \"" + xmlFilePath + "\" "
               + mContext.getString(R.string.error_filenotfound);
         Toast toast = Toast.makeText(mContext.getApplicationContext(),
               text, Toast.LENGTH_LONG);
         toast.show();
      } catch (IllegalArgumentException e) {
         Log.e(TAG, "Unable to save ", e);
         CharSequence text = mContext.getString(R.string.ticker_failed)
               + " \"" + xmlFilePath + "\" "
               + mContext.getString(R.string.error_filename);
         Toast toast = Toast.makeText(mContext.getApplicationContext(),
               text, Toast.LENGTH_LONG);
         toast.show();
      } catch (IllegalStateException e) {
         Log.e(TAG, "Unable to save ", e);
         CharSequence text = mContext.getString(R.string.ticker_failed)
               + " \"" + xmlFilePath + "\" "
               + mContext.getString(R.string.error_buildxml);
         Toast toast = Toast.makeText(mContext.getApplicationContext(),
```

```
            text, Toast.LENGTH_LONG);
      toast.show();
   } catch (IOException e) {
      Log.e(TAG, "Unable to save ", e);
      CharSequence text = mContext.getString(R.string.ticker_failed)
            + " \"" + xmlFilePath + "\" "
            + mContext.getString(R.string.error_writesdcard);
      Toast toast = Toast.makeText(mContext.getApplicationContext(),
            text, Toast.LENGTH_LONG);
      toast.show();
   } finally {
      if (buf != null) {
         try {
            buf.close();
         } catch (IOException e) {
            Log.e(TAG,
                  "Failed to close buf after completion, ignoring.",
                  e);
         }
      }
      if (fos != null) {
         try {
            fos.close();
         } catch (IOException e) {
            Log.e(TAG,
                  "Failed to close fos after completion, ignoring.",
                  e);
         }
      }
      if (mProgressListener != null) {
         mProgressListener.endNotification(resultFilename,
               getContentType());
      }
      Looper.loop();
   }
}

private void serializeTrack(Uri trackUri, XmlSerializer serializer)
      throws IllegalArgumentException, IllegalStateException, IOException {
   // Retrieving Track's settings
   Cursor trackCursor = null;
   ContentResolver resolver = mContext.getContentResolver();
   trackCursor = resolver.query(trackUri, new String[] { Tracks.NMC,
         Tracks.AREA, Tracks.SERVERIP, Tracks.PACKETSIZE,
         Tracks.PINGATTEMPTS, Tracks.NAME, Tracks.CREATION_TIME }, null,  null, null);
   if (trackCursor.moveToFirst()) {
      NMC = trackCursor.getString(0);
      Area = trackCursor.getString(1);
      Serverip = trackCursor.getString(2);
      Packetsize = Integer.toString(trackCursor.getInt(3));
      Pingattempts = Integer.toString(trackCursor.getInt(4));
      Trackname = trackCursor.getString(5);
      TrackTime = new Date(trackCursor.getLong(6));
   }

   serializer.startDocument("UTF-8", true);
   serializer.text("\n");
   serializer.startTag("", "Track");

   // The list of segments in the track
   serializeSegments(serializer, Uri.withAppendedPath(trackUri, "segments"));
```

```
   serializer.text("\n");
   serializer.endTag("", "Track");
   serializer.endDocument();
}

private void serializeSegments(XmlSerializer serializer, Uri segments)
      throws IOException {
   Cursor segmentCursor = null;
   ContentResolver resolver = mContext.getContentResolver();
   try {
      segmentCursor = resolver.query(segments,
            new String[] { Segments._ID }, null, null, null);
      if (segmentCursor.moveToFirst()) {
         if (mProgressListener != null) {
            mProgressListener.updateNotification(getProgress(),  getGoal());
         }

         do {
            Uri waypoints = Uri.withAppendedPath(segments,
                  segmentCursor.getLong(0) + "/waypoints");
            serializeWaypoints(serializer, waypoints);
         } while (segmentCursor.moveToNext());

      }
   } finally {
      if (segmentCursor != null) {
         segmentCursor.close();
      }
   }
}

private void serializeWaypoints(XmlSerializer serializer, Uri waypoints)
      throws IOException {
   Cursor waypointsCursor = null;
   ContentResolver resolver = mContext.getContentResolver();
   try {
      waypointsCursor = resolver.query(waypoints, new String[] {
            Waypoints.LONGITUDE, Waypoints.LATITUDE, Waypoints.TIME,
            Waypoints.ALTITUDE, Waypoints._ID, Waypoints.SPEED,
            Waypoints.MINLATENCY, Waypoints.MAXLATENCY,
            Waypoints.RECEIVED, Waypoints.LOST, Waypoints.CELLID,
            Waypoints.LAC, Waypoints.NETWORKTYPE }, null, null, null);
      if (waypointsCursor.moveToFirst()) {
         increaseGoal(waypointsCursor.getCount());
         do {
            increaseProgress(1);
            if (mProgressListener != null) {
               mProgressListener.updateNotification(getProgress(),  getGoal());
            }
            serializer.text("\n");

            // We do not include the first measurement in the xml file
            // because we don't know the location of mobile device when
            // logging started
            // On the contrary, results on mobile device (Statistics)
            // include first measurement
            if (isFirstLine == true) {
               isFirstLine = false;

               serializer.startTag("", "line");
               serializer.text("\n");
               serializer.startTag("", "StartTime");
```

```
    Date time = new Date(waypointsCursor.getLong(2));
    serializer.text(ZULU_DATE_FORMAT.format(time));
    serializer.endTag("", "StartTime");
    serializer.text("\n");
    serializer.startTag("", "StartLat");
    serializer.text(Double.toString(waypointsCursor.getDouble(1)));
    serializer.endTag("", "StartLat");
    serializer.text("\n");
    serializer.startTag("", "StartLon");
    serializer.text(Double.toString(waypointsCursor.getDouble(0)));
    serializer.endTag("", "StartLon");
 } else {
    if (NMC.equals("DOWNLOAD")) {
       // Compute color
       long bitrate = waypointsCursor.getLong(5);
       String color;

       if (bitrate <= 0)
          color = "#000000";
       else if (bitrate < 200)
          color = "#FF0000";
       else if (bitrate < 500)
          color = "#FF8040";
       else if (bitrate < 2000)
          color = "#FFFF00";
       else if (bitrate < 7000)
          color = "#00FF00";
       else
          color = "#0000FF";

       serializer.startTag("", "Bitrate");
       serializer.text(Long.toString(waypointsCursor.getLong(5)));
       serializer.endTag("", "Bitrate");
       serializer.text("\n");

       serializer.startTag("", "EndTime");
       Date time = new Date(waypointsCursor.getLong(2));
       serializer.text(ZULU_DATE_FORMAT.format(time));
       serializer.endTag("", "EndTime");
       serializer.text("\n");
       serializer.startTag("", "EndLat");
       serializer.text(Double.toString(waypointsCursor.getDouble(1)));
       serializer.endTag("", "EndLat");
       serializer.text("\n");
       serializer.startTag("", "EndLon");
       serializer.text(Double.toString(waypointsCursor.getDouble(0)));
       serializer.endTag("", "EndLon");
       serializer.text("\n");

       serializer.startTag("", "CellID");
       serializer.text(Integer.toString(waypointsCursor.getInt(10)));
       serializer.endTag("", "CellID");
       serializer.text("\n");
       serializer.startTag("", "Lac");
       serializer.text(Integer.toString(waypointsCursor.getInt(11)));
       serializer.endTag("", "Lac");
       serializer.text("\n");

       // NetworkType
       int networktype = waypointsCursor.getInt(12);
       String networktypeString;
```

```
switch (networktype) {
case 0:
   networktypeString = "Unknown";
   break;
case 1:
   networktypeString = "GPRS";
   break;
case 2:
   networktypeString = "EDGE";
   break;
case 3:
   networktypeString = "UMTS";
   break;
case 4:
   networktypeString = "CDMA";
   break;
case 5:
   networktypeString = "EVDO_0";
   break;
case 6:
   networktypeString = "EVDO_A";
   break;
case 7:
   networktypeString = "1xRTT";
   break;
case 8:
   networktypeString = "HSDPA";
   break;
case 9:
   networktypeString = "HSUPA";
   break;
case 10:
   networktypeString = "HSPA";
   break;
default:
   networktypeString = "Unknown";
   break;
}
serializer.startTag("", "NetworkType");
serializer.text(networktypeString);
serializer.endTag("", "NetworkType");
serializer.text("\n");

// Color
serializer.startTag("", "Color");
serializer.text(color);
serializer.endTag("", "Color");
serializer.text("\n");
serializer.text("\n");

serializer.startTag("", "NMC");
serializer.text(NMC);
serializer.endTag("", "NMC");
serializer.text("\n");
serializer.startTag("", "AREA");
serializer.text(Area);
serializer.endTag("", "AREA");
serializer.text("\n");
serializer.startTag("", "SERVERIP");
serializer.text(Serverip);
serializer.endTag("", "SERVERIP");
serializer.text("\n");
```

```
            serializer.startTag("", "PACKETSIZE");
            serializer.text(Packetsize);
            serializer.endTag("", "PACKETSIZE");
            serializer.text("\n");
            serializer.startTag("", "TRACKNAME");
            serializer.text(Trackname);
            serializer.endTag("", "TRACKNAME");
            serializer.text("\n");
            serializer.startTag("", "TRACKTIME");
            serializer.text(ZULU_DATE_FORMAT.format(TrackTime));
            serializer.endTag("", "TRACKTIME");
            serializer.text("\n");

        } else if (NMC.equals("PING")) {
            // Compute color
            long avgLat = waypointsCursor.getLong(5);
            String color;

            if (avgLat <= 0)
               color = "#000000";
            else if (avgLat < 60)
               color = "#0000FF";
            else if (avgLat < 120)
               color = "#00FF00";
            else if (avgLat < 300)
               color = "#FFFF00";
            else if (avgLat < 800)
               color = "#FF8040";
            else
               color = "#FF0000";

            serializer.startTag("", "AvgLatency");
            serializer.text(Long.toString(waypointsCursor.getLong(5)));
            serializer.endTag("", "AvgLatency");
            serializer.text("\n");
            serializer.startTag("", "MinLatency");
            serializer.text(Long.toString(waypointsCursor.getLong(6)));
            serializer.endTag("", "MinLatency");
            serializer.text("\n");
            serializer.startTag("", "MaxLatency");
            serializer.text(Long.toString(waypointsCursor.getLong(7)));
            serializer.endTag("", "MaxLatency");
            serializer.text("\n");
            serializer.startTag("", "Received");
            serializer.text(Long.toString(waypointsCursor.getLong(8)));
            serializer.endTag("", "Received");
            serializer.text("\n");
            serializer.startTag("", "Lost");
            serializer.text(Long.toString(waypointsCursor.getLong(9)));
            serializer.endTag("", "Lost");
            serializer.text("\n");

            serializer.startTag("", "EndTime");
            Date time = new Date(waypointsCursor.getLong(2));
            serializer.text(ZULU_DATE_FORMAT.format(time));
            serializer.endTag("", "EndTime");
            serializer.text("\n");
            serializer.startTag("", "EndLat");
            serializer.text(Double.toString(waypointsCursor.getDouble(1)));
            serializer.endTag("", "EndLat");
            serializer.text("\n");
            serializer.startTag("", "EndLon");
```

```
serializer.text(Double.toString(waypointsCursor.getDouble(0)));
serializer.endTag("", "EndLon");
serializer.text("\n");

serializer.startTag("", "CellID");
serializer.text(Integer.toString(waypointsCursor.getInt(10)));
serializer.endTag("", "CellID");
serializer.text("\n");
serializer.startTag("", "Lac");
serializer.text(Integer.toString(waypointsCursor.getInt(11)));
serializer.endTag("", "Lac");
serializer.text("\n");

// NetworkType
int networktype = waypointsCursor.getInt(12);
String networktypeString;

switch (networktype) {
case 0:
   networktypeString = "Unknown";
   break;
case 1:
   networktypeString = "GPRS";
   break;
case 2:
   networktypeString = "EDGE";
   break;
case 3:
   networktypeString = "UMTS";
   break;
case 4:
   networktypeString = "CDMA";
   break;
case 5:
   networktypeString = "EVDO_0";
   break;
case 6:
   networktypeString = "EVDO_A";
   break;
case 7:
   networktypeString = "1xRTT";
   break;
case 8:
   networktypeString = "HSDPA";
   break;
case 9:
   networktypeString = "HSUPA";
   break;
case 10:
   networktypeString = "HSPA";
   break;
default:
   networktypeString = "Unknown";
   break;
}
serializer.startTag("", "NetworkType");
serializer.text(networktypeString);
serializer.endTag("", "NetworkType");
serializer.text("\n");

// Color
serializer.startTag("", "Color");
```

```
                    serializer.text(color);
                    serializer.endTag("", "Color");
                    serializer.text("\n");
                    serializer.text("\n");

                    serializer.startTag("", "NMC");
                    serializer.text(NMC);
                    serializer.endTag("", "NMC");
                    serializer.text("\n");
                    serializer.startTag("", "AREA");
                    serializer.text(Area);
                    serializer.endTag("", "AREA");
                    serializer.text("\n");
                    serializer.startTag("", "SERVERIP");
                    serializer.text(Serverip);
                    serializer.endTag("", "SERVERIP");
                    serializer.text("\n");
                    serializer.startTag("", "PINGATTEMPTS");
                    serializer.text(Pingattempts);
                    serializer.endTag("", "PINGATTEMPTS");
                    serializer.text("\n");
                    serializer.startTag("", "TRACKNAME");
                    serializer.text(Trackname);
                    serializer.endTag("", "TRACKNAME");
                    serializer.text("\n");
                    serializer.startTag("", "TRACKTIME");
                    serializer.text(ZULU_DATE_FORMAT.format(TrackTime));
                    serializer.endTag("", "TRACKTIME");
                    serializer.text("\n");
                 }
                 serializer.endTag("", "line");

                 if (waypointsCursor.isLast() != true) {
                    serializer.text("\n");
                    serializer.text("\n");
                    serializer.startTag("", "line");
                    serializer.text("\n");

                    serializer.startTag("", "StartTime");
                    Date time = new Date(waypointsCursor.getLong(2));
                    serializer.text(ZULU_DATE_FORMAT.format(time));
                    serializer.endTag("", "StartTime");
                    serializer.text("\n");
                    serializer.startTag("", "StartLat");
                    serializer.text(Double.toString(waypointsCursor.getDouble(1)));
                    serializer.endTag("", "StartLat");
                    serializer.text("\n");
                    serializer.startTag("", "StartLon");
                    serializer.text(Double.toString(waypointsCursor.getDouble(0)));
                    serializer.endTag("", "StartLon");
                 }
              }
           } while (waypointsCursor.moveToNext());
        }
     } finally {
        if (waypointsCursor != null) {
           waypointsCursor.close();
        }
     }

}
```

```
   private String getContentType() {
      return isNeedsBundling() ? "application/zip" : "text/xml";
   }
}
```

DatabaseHelper.java

```
package nl.sogeti.android.gpstracker.db;

import java.util.Date;

import nl.sogeti.android.gpstracker.db.GPStracking.Media;
import nl.sogeti.android.gpstracker.db.GPStracking.MediaColumns;
import nl.sogeti.android.gpstracker.db.GPStracking.Segments;
import nl.sogeti.android.gpstracker.db.GPStracking.Tracks;
import nl.sogeti.android.gpstracker.db.GPStracking.TracksColumns;
import nl.sogeti.android.gpstracker.db.GPStracking.Waypoints;
import nl.sogeti.android.gpstracker.db.GPStracking.WaypointsColumns;
import android.content.ContentResolver;
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.location.Location;
import android.net.Uri;
import android.util.Log;

/**
 * Class to hold bare-metal database operations exposed as functionality blocks
 * To be used by database adapters, like a content provider, that implement a
 * required functionality set
 *
 * @author Alexandros Ntakas
 */
class DatabaseHelper extends SQLiteOpenHelper {
   private Context mContext;
   private final static String TAG = "OGT.DatabaseHelper";

   public DatabaseHelper(Context context) {
      super(context, GPStracking.DATABASE_NAME, null, GPStracking.DATABASE_VERSION);
      this.mContext = context;
   }

   @Override
   public void onCreate(SQLiteDatabase db) {
      db.execSQL(Waypoints.CREATE_STATEMENT);
      db.execSQL(Segments.CREATE_STATMENT);
      db.execSQL(Tracks.CREATE_STATEMENT);
      db.execSQL(Media.CREATE_STATEMENT);
   }

   /**
    *
    * Will update version 1 through 5 to version 8
    *
    * @see
android.database.sqlite.SQLiteOpenHelper#onUpgrade(android.database.sqlite.SQLiteDatabase
,
    *      int, int)
```

```
     * @see GPStracking.DATABASE_VERSION
     */
    @Override
    public void onUpgrade(SQLiteDatabase db, int current, int targetVersion) {
       Log.i(TAG, "Upgrading db from " + current + " to " + targetVersion);
       if (current <= 5) // From 1-5 to 6 (these before are the same before)
       {
          current = 6;
       }
       if (current == 6) // From 6 to 7 ( no changes )
       {
          current = 7;
       }
       if (current == 7) // From 7 to 8 ( more waypoints data )
       {
          for (String statement : Waypoints.UPGRADE_STATEMENT_7_TO_8) {
             db.execSQL(statement);
          }
          current = 8;
       }
       if (current == 8) // From 8 to 9 ( media Uri data )
       {
          db.execSQL(Media.CREATE_STATEMENT);
          current = 9;
       }
    }

    /**
     * Creates a waypoint under the current track segment with the current time
     * on which the waypoint is reached
     *
     * @param track track
     * @param segment segment
     * @param latitude latitude
     * @param longitude longitude
     * @param time time
     * @param speed the measured speed
     * @return
     */
    long insertWaypoint(long trackId, long segmentId, Location location,
          ContentValues valuesHelp) {
       if (trackId < 0 || segmentId < 0) {
          throw new IllegalArgumentException(
                "Track and segments may not the less then 0.");
       }

       SQLiteDatabase sqldb = getWritableDatabase();

       ContentValues args = new ContentValues();
       args.put(WaypointsColumns.SEGMENT, segmentId);
       args.put(WaypointsColumns.TIME, location.getTime());
       args.put(WaypointsColumns.LATITUDE, location.getLatitude());
       args.put(WaypointsColumns.LONGITUDE, location.getLongitude());
       args.put(WaypointsColumns.SPEED, location.getSpeed());
       args.put(WaypointsColumns.ACCURACY, location.getAccuracy());
       args.put(WaypointsColumns.ALTITUDE, location.getAltitude());
       args.put(WaypointsColumns.BEARING, location.getBearing());

       if (valuesHelp.containsKey(Waypoints.MINLATENCY)) {
          args.put(WaypointsColumns.MINLATENCY,
valuesHelp.getAsLong(Waypoints.MINLATENCY));
```

```
         args.put(WaypointsColumns.MAXLATENCY,
valuesHelp.getAsLong(Waypoints.MAXLATENCY));
         args.put(WaypointsColumns.RECEIVED, valuesHelp.getAsLong(Waypoints.RECEIVED));
         args.put(WaypointsColumns.LOST, valuesHelp.getAsLong(Waypoints.LOST));
      }
      args.put(WaypointsColumns.CELLID, valuesHelp.getAsInteger(Waypoints.CELLID));
      args.put(WaypointsColumns.LAC, valuesHelp.getAsInteger(Waypoints.LAC));

   args.put(WaypointsColumns.NETWORKTYPE,valuesHelp.getAsInteger(Waypoints.NETWORKTYPE));

      long waypointId = sqldb.insert(Waypoints.TABLE, null, args);

      ContentResolver resolver = this.mContext.getContentResolver();
      resolver.notifyChange(
            Uri.withAppendedPath(Tracks.CONTENT_URI, trackId + "/segments/"
                  + segmentId + "/waypoints"), null);

      return waypointId;
   }

   long insertMedia(long trackId, long segmentId, long waypointId,  String mediaUri) {
      if (trackId < 0 || segmentId < 0 || waypointId < 0) {
         throw new IllegalArgumentException(
               "Track, segments and waypoint may not the less then 0.");
      }
      SQLiteDatabase sqldb = getWritableDatabase();

      ContentValues args = new ContentValues();
      args.put(MediaColumns.TRACK, trackId);
      args.put(MediaColumns.SEGMENT, segmentId);
      args.put(MediaColumns.WAYPOINT, waypointId);
      args.put(MediaColumns.URI, mediaUri);

      long mediaId = sqldb.insert(Media.TABLE, null, args);

      ContentResolver resolver = this.mContext.getContentResolver();
      Uri notifyUri = Uri.withAppendedPath(Tracks.CONTENT_URI, trackId
            + "/segments/" + segmentId + "/waypoints/" + waypointId
            + "/media");
      resolver.notifyChange(notifyUri, null);
      resolver.notifyChange(ContentUris.withAppendedId(Media.CONTENT_URI, trackId), null);

      return mediaId;
   }

   /**
    * Deletes a single track and all underlying segments and waypoints
    *
    * @param trackId
    * @return
    */
   int deleteTrack(long trackId) {
      SQLiteDatabase sqldb = getWritableDatabase();
      int affected = 0;
      Cursor cursor = null;
      long segmentId = -1;

      try {
         cursor = sqldb.query(Segments.TABLE, new String[] { Segments._ID },
               Segments.TRACK + "= ?",
               new String[] { String.valueOf(trackId) }, null, null, null, null);
         if (cursor.moveToFirst()) {
```

```
            do {
               segmentId = cursor.getLong(0);
               affected += deleteSegment(sqldb, trackId, segmentId);
            } while (cursor.moveToNext());
         } else {
            Log.e(TAG, "Did not find the last active segment");
         }
      } finally {
         if (cursor != null) {
            cursor.close();
         }
      }

      affected += sqldb.delete(Tracks.TABLE, Tracks._ID + "= ?",
            new String[] { String.valueOf(trackId) });
      ContentResolver resolver = this.mContext.getContentResolver();
      resolver.notifyChange(Tracks.CONTENT_URI, null);
      resolver.notifyChange(ContentUris.withAppendedId(Tracks.CONTENT_URI, trackId),
null);

      return affected;
   }

   /**
    * @param mediaId
    * @return
    */
   int deleteMedia(long mediaId) {
      SQLiteDatabase sqldb = getWritableDatabase();

      Cursor cursor = null;
      long trackId = -1;
      long segmentId = -1;
      long waypointId = -1;
      try {
         cursor = sqldb.query(Media.TABLE, new String[] { Media.TRACK,
               Media.SEGMENT, Media.WAYPOINT }, Media._ID + "= ?",
               new String[] { String.valueOf(mediaId) }, null, null, null,
               null);
         if (cursor.moveToFirst()) {
            trackId = cursor.getLong(0);
            segmentId = cursor.getLong(0);
            waypointId = cursor.getLong(0);
         } else {
            Log.e(TAG, "Did not find the last active segment");
         }
      } finally {
         if (cursor != null) {
            cursor.close();
         }
      }

      int affected = sqldb.delete(Media.TABLE, Media._ID + "= ?",
            new String[] { String.valueOf(mediaId) });

      ContentResolver resolver = this.mContext.getContentResolver();
      Uri notifyUri = Uri.withAppendedPath(Tracks.CONTENT_URI, trackId
            + "/segments/" + segmentId + "/waypoints/" + waypointId
            + "/media");
      resolver.notifyChange(notifyUri, null);
      notifyUri = Uri.withAppendedPath(Tracks.CONTENT_URI, trackId
            + "/segments/" + segmentId + "/media");
```

```
   resolver.notifyChange(notifyUri, null);
   notifyUri = Uri.withAppendedPath(Tracks.CONTENT_URI, trackId + "/media");
   resolver.notifyChange(notifyUri, null);
   resolver.notifyChange(ContentUris.withAppendedId(Media.CONTENT_URI, mediaId), null);

   return affected;
}

/**
 * Delete a segment and all member waypoints
 * @param sqldb The SQLiteDatabase in question
 * @param trackId The track id of this delete
 * @param segmentId The segment that needs deleting
 * @return
 */
int deleteSegment(SQLiteDatabase sqldb, long trackId, long segmentId) {
   int affected = sqldb.delete(Segments.TABLE, Segments._ID + "= ?",
         new String[] { String.valueOf(segmentId) });

   // Delete all waypoints from segments
   affected += sqldb.delete(Waypoints.TABLE, Waypoints.SEGMENT + "= ?",
         new String[] { String.valueOf(segmentId) });
   // Delete all media from segment
   affected += sqldb.delete(Media.TABLE, Media.TRACK + "= ? AND "
         + Media.SEGMENT + "= ?", new String[] {
         String.valueOf(trackId), String.valueOf(segmentId) });

   ContentResolver resolver = this.mContext.getContentResolver();
   resolver.notifyChange(
         Uri.withAppendedPath(Tracks.CONTENT_URI, trackId + "/segments/"
               + segmentId), null);
   resolver.notifyChange(
         Uri.withAppendedPath(Tracks.CONTENT_URI, trackId + "/segments"),
         null);

   return affected;
}

/**
 * Move to a fresh track with a new first segment for this track
 *
 * @return
 */
long toNextTrack(String name, ContentValues valuesHelper) {
   long currentTime = new Date().getTime();
   ContentValues args = new ContentValues();
   args.put(TracksColumns.NAME, name);
   args.put(TracksColumns.CREATION_TIME, currentTime);

   String NMC = valuesHelper.getAsString(Tracks.NMC);
   String ServerIP = valuesHelper.getAsString(Tracks.SERVERIP);
   String Area = valuesHelper.getAsString(Tracks.AREA);

   args.put(TracksColumns.NMC, NMC);
   args.put(Tracks.SERVERIP, ServerIP);
   args.put(Tracks.AREA, Area);

   if (NMC == "DOWNLOAD") {
      int PacketSize = valuesHelper.getAsInteger(Tracks.PACKETSIZE);
      args.put(Tracks.PACKETSIZE, PacketSize);
   } else if (NMC == "PING") {
      int PingAttempts = valuesHelper.getAsInteger(Tracks.PINGATTEMPTS);
```

```
         args.put(Tracks.PINGATTEMPTS, PingAttempts);
      }

      SQLiteDatabase sqldb = getWritableDatabase();
      long trackId = sqldb.insert(Tracks.TABLE, null, args);

      ContentResolver resolver = this.mContext.getContentResolver();
      resolver.notifyChange(Tracks.CONTENT_URI, null);

      return trackId;
   }

   /**
    * Moves to a fresh segment to which waypoints can be connected
    *
    * @return
    */
   long toNextSegment(long trackId) {
      SQLiteDatabase sqldb = getWritableDatabase();

      ContentValues args = new ContentValues();
      args.put(Segments.TRACK, trackId);
      long segmentId = sqldb.insert(Segments.TABLE, null, args);

      ContentResolver resolver = this.mContext.getContentResolver();
      resolver.notifyChange(
            Uri.withAppendedPath(Tracks.CONTENT_URI, trackId + "/segments"),
            null);

      return segmentId;
   }
}
```

GPStracking.java

```
package nl.sogeti.android.gpstracker.db;

import android.net.Uri;
import android.provider.BaseColumns;

/**
 * The GPStracking provider stores all static information about GPStracking.
 *
 * @author Alexandros Ntakas
 */
public final class GPStracking
{
   /** The authority of this provider */
   public static final String AUTHORITY = "nl.sogeti.android.gpstracker";
   /** The content:// style URL for this provider */
   public static final Uri CONTENT_URI = Uri.parse( "content://" + GPStracking.AUTHORITY
);
   /** The name of the database file */
   static final String DATABASE_NAME = "GPSLOG.db";
   /** The version of the database schema */
   static final int DATABASE_VERSION = 9;

   /**
    * This table contains tracks.
    *
    * @author Alexandros Ntakas
```

```
    */
   public static final class Tracks extends TracksColumns implements
android.provider.BaseColumns
   {
      /** The MIME type of a CONTENT_URI subdirectory of a single track. */
      public static final String CONTENT_ITEM_TYPE =
"vnd.android.cursor.item/vnd.nl.sogeti.android.track";
      /** The MIME type of CONTENT_URI providing a directory of tracks. */
      public static final String CONTENT_TYPE =
"vnd.android.cursor.dir/vnd.nl.sogeti.android.track";
      /** The content:// style URL for this provider */
      public static final Uri CONTENT_URI = Uri.parse( "content://" +
GPStracking.AUTHORITY + "/" + Tracks.TABLE );

      /** The name of this table */
      static final String TABLE = "tracks";
      static final String CREATE_STATEMENT =
         "CREATE TABLE " + Tracks.TABLE + "(" + " " + Tracks._ID           + " " +
Tracks._ID_TYPE +
                                          "," + " " + Tracks.NAME          + " " +
Tracks.NAME_TYPE +
                                          "," + " " + Tracks.CREATION_TIME + " " +
Tracks.CREATION_TIME_TYPE +
                                          "," + " " + Tracks.NMC           + " " +
Tracks.NMC_TYPE +
                                          "," + " " + Tracks.SERVERIP      + " " +
Tracks.SERVERIP_TYPE +
                                          "," + " " + Tracks.PACKETSIZE    + " " +
Tracks.PACKETSIZE_TYPE +
                                          "," + " " + Tracks.PINGATTEMPTS  + " " +
Tracks.PINGATTEMPTS_TYPE +
                                          "," + " " + Tracks.AREA          + " " +
Tracks.AREA_TYPE +
                                          ");";
   }

   /**
    * This table contains segments.
    *
    */
   public static final class Segments extends SegmentsColumns implements
android.provider.BaseColumns
   {

      /** The MIME type of a CONTENT_URI subdirectory of a single segment. */
      public static final String CONTENT_ITEM_TYPE =
"vnd.android.cursor.item/vnd.nl.sogeti.android.segment";
      /** The MIME type of CONTENT_URI providing a directory of segments. */
      public static final String CONTENT_TYPE =
"vnd.android.cursor.dir/vnd.nl.sogeti.android.segment";

      /** The name of this table */
      static final String TABLE = "segments";
      static final String CREATE_STATMENT =
         "CREATE TABLE " + Segments.TABLE + "(" + " " + Segments._ID   + " " +
Segments._ID_TYPE +
                                            "," + " " + Segments.TRACK + " " +
Segments.TRACK_TYPE +
                                            ");";
   }

   /**
```

```
    * This table contains waypoints.
    *
    * @author Alexandros Ntakas
    */
   public static final class Waypoints extends WaypointsColumns implements
android.provider.BaseColumns
   {

      /** The MIME type of a CONTENT_URI subdirectory of a single waypoint. */
      public static final String CONTENT_ITEM_TYPE =
"vnd.android.cursor.item/vnd.nl.sogeti.android.waypoint";
      /** The MIME type of CONTENT_URI providing a directory of waypoints. */
      public static final String CONTENT_TYPE =
"vnd.android.cursor.dir/vnd.nl.sogeti.android.waypoint";

      /** The name of this table */
      public static final String TABLE = "waypoints";
      static final String CREATE_STATEMENT = "CREATE TABLE " + Waypoints.TABLE +
      "(" + " " + BaseColumns._ID + " " + WaypointsColumns._ID_TYPE +
      "," + " " + WaypointsColumns.LATITUDE    + " " + WaypointsColumns.LATITUDE_TYPE +
      "," + " " + WaypointsColumns.LONGITUDE   + " " + WaypointsColumns.LONGITUDE_TYPE +
      "," + " " + WaypointsColumns.TIME        + " " + WaypointsColumns.TIME_TYPE +
      "," + " " + WaypointsColumns.SPEED       + " " + WaypointsColumns.SPEED +
      "," + " " + WaypointsColumns.SEGMENT     + " " + WaypointsColumns.SEGMENT_TYPE +
      "," + " " + WaypointsColumns.ACCURACY    + " " + WaypointsColumns.ACCURACY_TYPE +
      "," + " " + WaypointsColumns.ALTITUDE    + " " + WaypointsColumns.ALTITUDE_TYPE +
      "," + " " + WaypointsColumns.BEARING     + " " + WaypointsColumns.BEARING_TYPE +
      "," + " " + WaypointsColumns.MINLATENCY  + " " + WaypointsColumns.MINLATENCY_TYPE +
      "," + " " + WaypointsColumns.MAXLATENCY  + " " + WaypointsColumns.MAXLATENCY_TYPE +
      "," + " " + WaypointsColumns.RECEIVED    + " " + WaypointsColumns.RECEIVED_TYPE +
      "," + " " + WaypointsColumns.LOST        + " " + WaypointsColumns.LOST_TYPE +
      "," + " " + WaypointsColumns.CELLID      + " " + WaypointsColumns.CELLID_TYPE +
      "," + " " + WaypointsColumns.LAC         + " " + WaypointsColumns.LAC_TYPE +
      "," + " " + WaypointsColumns.NETWORKTYPE + " " + WaypointsColumns.NETWORKTYPE_TYPE
+     ");";

      static final String[] UPGRADE_STATEMENT_7_TO_8 =
         {
            "ALTER TABLE " + Waypoints.TABLE + " ADD COLUMN " + WaypointsColumns.ACCURACY
+ " " + WaypointsColumns.ACCURACY_TYPE +";",
            "ALTER TABLE " + Waypoints.TABLE + " ADD COLUMN " + WaypointsColumns.ALTITUDE
+ " " + WaypointsColumns.ALTITUDE_TYPE +";",
            "ALTER TABLE " + Waypoints.TABLE + " ADD COLUMN " + WaypointsColumns.BEARING
+ " " + WaypointsColumns.BEARING_TYPE +";"
         };
   }

   /**
    * This table contains media URI's.
    *
    */
   public static final class Media extends MediaColumns implements
android.provider.BaseColumns
   {

      /** The MIME type of a CONTENT_URI subdirectory of a single media entry. */
      public static final String CONTENT_ITEM_TYPE =
"vnd.android.cursor.item/vnd.nl.sogeti.android.media";
      /** The MIME type of CONTENT_URI providing a directory of media entry. */
      public static final String CONTENT_TYPE =
"vnd.android.cursor.dir/vnd.nl.sogeti.android.media";
```

```
      /** The name of this table */
      public static final String TABLE = "media";
      static final String CREATE_STATEMENT = "CREATE TABLE " + Media.TABLE +
      "(" + " " + BaseColumns._ID       + " " + MediaColumns._ID_TYPE +
      "," + " " + MediaColumns.TRACK    + " " + MediaColumns.TRACK_TYPE +
      "," + " " + MediaColumns.SEGMENT  + " " + MediaColumns.SEGMENT_TYPE +
      "," + " " + MediaColumns.WAYPOINT + " " + MediaColumns.WAYPOINT_TYPE +
      "," + " " + MediaColumns.URI      + " " + MediaColumns.URI_TYPE +
      ");";
      public static final Uri CONTENT_URI = Uri.parse( "content://" +
GPStracking.AUTHORITY + "/" + Media.TABLE );
   }

   /**
    * Columns from the tracks table.
    *
    * @author Alexandros Ntakas
    */
   public static class TracksColumns
   {
      /** The end time */
      public static final String NAME          = "name";
      public static final String CREATION_TIME = "creationtime";
      static final String CREATION_TIME_TYPE   = "INTEGER NOT NULL";
      static final String NAME_TYPE            = "TEXT";
      static final String _ID_TYPE             = "INTEGER PRIMARY KEY AUTOINCREMENT";

      public static final String NMC           = "networkingmeasurementchoice";
      static final String NMC_TYPE             = "TEXT";
      public static final String SERVERIP      = "serverip";
      static final String SERVERIP_TYPE        = "TEXT";
      public static final String PACKETSIZE    = "packetsize";
      static final String PACKETSIZE_TYPE      = "INTEGER";
      public static final String PINGATTEMPTS  = "pingattempts";
      static final String PINGATTEMPTS_TYPE    = "INTEGER";
      public static final String AREA          = "area";
      static final String AREA_TYPE            = "TEXT";
   }

   /**
    * Columns from the segments table.
    *
    */
   public static class SegmentsColumns
   {
      /** The track _id to which this segment belongs */
      public static final String TRACK = "track";
      static final String TRACK_TYPE   = "INTEGER NOT NULL";
      static final String _ID_TYPE     = "INTEGER PRIMARY KEY AUTOINCREMENT";
   }

   /**
    * Columns from the waypoints table.
    *
    * @author Alexandros Ntakas
    */
   public static class WaypointsColumns
   {

      /** The latitude */
      public static final String LATITUDE = "latitude";
      /** The longitude */
```

```
      public static final String LONGITUDE = "longitude";
      /** The recorded time */
      public static final String TIME = "time";
      /** The speed in meters per second */
      public static final String SPEED = "speed";
      /** The segment _id to which this segment belongs */
      public static final String SEGMENT = "tracksegment";
      /** The accuracy of the fix */
      public static final String ACCURACY = "accuracy";
      /** The altitude */
      public static final String ALTITUDE = "altitude";
      /** the bearing of the fix */
      public static final String BEARING = "bearing";

      public static final String MINLATENCY  = "minlatency";
      public static final String MAXLATENCY  = "maxlatency";
      public static final String RECEIVED    = "received";
      public static final String LOST        = "lost";
      public static final String CELLID      = "cellid";
      public static final String LAC         = "lac";
      public static final String NETWORKTYPE = "networktype";

      static final String LATITUDE_TYPE  = "REAL NOT NULL";
      static final String LONGITUDE_TYPE = "REAL NOT NULL";
      static final String TIME_TYPE      = "INTEGER NOT NULL";
      static final String SPEED_TYPE     = "REAL NOT NULL";
      static final String SEGMENT_TYPE   = "INTEGER NOT NULL";
      static final String ACCURACY_TYPE  = "REAL";
      static final String ALTITUDE_TYPE  = "REAL";
      static final String BEARING_TYPE   = "REAL";
      static final String _ID_TYPE       = "INTEGER PRIMARY KEY AUTOINCREMENT";

      static final String MINLATENCY_TYPE  = "INTEGER";
      static final String MAXLATENCY_TYPE  = "INTEGER";
      static final String RECEIVED_TYPE    = "INTEGER";
      static final String LOST_TYPE        = "INTEGER";
      static final String CELLID_TYPE      = "INTEGER";
      static final String LAC_TYPE         = "INTEGER";
      static final String NETWORKTYPE_TYPE = "INTEGER";
   }

   /**
    * Columns from the media table.
    *
    */
   public static class MediaColumns
   {
      /** The track _id to which this segment belongs */
      public static final String TRACK    = "track";
      static final String TRACK_TYPE      = "INTEGER NOT NULL";
      public static final String SEGMENT  = "segment";
      static final String SEGMENT_TYPE    = "INTEGER NOT NULL";
      public static final String WAYPOINT = "waypoint";
      static final String WAYPOINT_TYPE   = "INTEGER NOT NULL";
      public static final String URI      = "uri";
      static final String URI_TYPE        = "TEXT";
      static final String _ID_TYPE        = "INTEGER PRIMARY KEY AUTOINCREMENT";
   }
}
```

GPStrackingProvider.java

```
package nl.sogeti.android.gpstracker.db;

import java.util.List;

import nl.sogeti.android.gpstracker.db.GPStracking.Media;
import nl.sogeti.android.gpstracker.db.GPStracking.Segments;
import nl.sogeti.android.gpstracker.db.GPStracking.Tracks;
import nl.sogeti.android.gpstracker.db.GPStracking.Waypoints;
import android.app.SearchManager;
import android.content.ContentProvider;
import android.content.ContentResolver;
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.UriMatcher;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteQueryBuilder;
import android.location.Location;
import android.net.Uri;
import android.provider.LiveFolders;
import android.util.Log;

/**
 * Goal of this Content Provider is to make the GPS Tracking information
 * uniformly available to this application and even other applications. The
 * GPS-tracking database can hold, tracks, segments or waypoints
 * <p>
 * A track is an actual route taken from start to finish. All the GPS locations
 * collected are waypoints. Waypoints taken in sequence without loss of
 * GPS-signal are considered connected and are grouped in segments. A route is
 * build up out of 1 or more segments.
 * <p>
 * For example:<br>
 * <code>content://nl.sogeti.android.gpstracker/tracks</code> is the URI that
 * returns all the stored tracks or starts a new track on insert
 * <p>
 * <code>content://nl.sogeti.android.gpstracker/tracks/2</code> is the URI
 * string that would return a single result row, the track with ID = 23.
 * <p>
 * <code>content://nl.sogeti.android.gpstracker/tracks/2/segments</code> is the
 * URI that returns all the stored segments of a track with ID = 2 or starts a
 * new segment on insert
 * <p>
 * <code>content://nl.sogeti.android.gpstracker/tracks/2/waypoints</code> is the
 * URI that returns all the stored waypoints of a track with ID = 2
 * <p>
 * <code>content://nl.sogeti.android.gpstracker/tracks/2/segments</code> is the
 * URI that returns all the stored segments of a track with ID = 2
 * <p>
 * <code>content://nl.sogeti.android.gpstracker/tracks/2/segments/3</code> is
 * the URI string that would return a single result row, the segment with ID = 3
 * of a track with ID = 2 .
 * <p>
 * <code>content://nl.sogeti.android.gpstracker/tracks/2/segments/1/waypoints</code>
 * is the URI that returns all the waypoints of a segment 1 of track 2.
 * <p>
 * <code>content://nl.sogeti.android.gpstracker/tracks/2/segments/1/waypoints/52</code>
 * is the URI string that would return a single result row, the waypoint with ID
 * = 52
 * <p>
```

```
 * Media is stored under a waypoint and may be queried as:<br>
 *
<code>content://nl.sogeti.android.gpstracker/tracks/2/segments/3/waypoints/22/media</code
>
 * <p>
 * All media for a segment can be queried with:<br>
 * <code>content://nl.sogeti.android.gpstracker/tracks/2/segments/3/media</code>
 * <p>
 * All media for a track can be queried with:<br>
 * <code>content://nl.sogeti.android.gpstracker/tracks/2/media</code>
 * <p>
 * The whole set of collected media may be queried as:<br>
 * <code>content://nl.sogeti.android.gpstracker/media</code>
 * <p>
 * A single media is stored with an ID, for instance ID = 12:<br>
 * <code>content://nl.sogeti.android.gpstracker/media/12</code>
 *
 * @version $Id: GPStrackingProvider.java 528 2010-05-09 19:49:23Z rcgroot $
 * @author Alexandros Ntakas
 */
public class GPStrackingProvider extends ContentProvider {
   private static final String TAG = "OGT.GPStrackingProvider";

   /* Action types as numbers for using the UriMatcher */
   private static final int TRACKS = 1;
   private static final int TRACK_ID = 2;
   private static final int TRACK_MEDIA = 3;
   private static final int TRACK_WAYPOINTS = 4;
   private static final int SEGMENTS = 5;
   private static final int SEGMENT_ID = 6;
   private static final int SEGMENT_MEDIA = 7;
   private static final int WAYPOINTS = 8;
   private static final int WAYPOINT_ID = 9;
   private static final int WAYPOINT_MEDIA = 10;
   private static final int SEARCH_SUGGEST_ID = 11;
   private static final int LIVE_FOLDERS = 12;
   private static final int MEDIA = 13;
   private static final int MEDIA_ID = 14;
   private static final String[] SUGGEST_PROJECTION = new String[] {
         Tracks._ID,
         Tracks.NAME + " AS " + SearchManager.SUGGEST_COLUMN_TEXT_1,
         "datetime(" + Tracks.CREATION_TIME + "/1000, 'unixepoch') as "
               + SearchManager.SUGGEST_COLUMN_TEXT_2,
         Tracks._ID + " AS " + SearchManager.SUGGEST_COLUMN_INTENT_DATA_ID

   };
   private static final String[] LIVE_PROJECTION = new String[] {
         Tracks._ID + " AS " + LiveFolders._ID,
         Tracks.NAME + " AS " + LiveFolders.NAME,
         "datetime(" + Tracks.CREATION_TIME + "/1000, 'unixepoch') as "
               + LiveFolders.DESCRIPTION };

   private static UriMatcher sURIMatcher = new UriMatcher(UriMatcher.NO_MATCH);

   /**
    * Although it is documented that in addURI(null, path, 0) "path" should be
    * an absolute path this does not seem to work. A relative path gets the
    * jobs done and matches an absolute path.
    */
   static {
      GPStrackingProvider.sURIMatcher = new UriMatcher(UriMatcher.NO_MATCH);
      GPStrackingProvider.sURIMatcher.addURI(GPStracking.AUTHORITY, "tracks",
```

```
            GPStrackingProvider.TRACKS);
      GPStrackingProvider.sURIMatcher.addURI(GPStracking.AUTHORITY,
            "tracks/#", GPStrackingProvider.TRACK_ID);
      GPStrackingProvider.sURIMatcher.addURI(GPStracking.AUTHORITY,
            "tracks/#/media", GPStrackingProvider.TRACK_MEDIA);
      GPStrackingProvider.sURIMatcher.addURI(GPStracking.AUTHORITY,
            "tracks/#/waypoints", GPStrackingProvider.TRACK_WAYPOINTS);
      GPStrackingProvider.sURIMatcher.addURI(GPStracking.AUTHORITY,
            "tracks/#/segments", GPStrackingProvider.SEGMENTS);
      GPStrackingProvider.sURIMatcher.addURI(GPStracking.AUTHORITY,
            "tracks/#/segments/#", GPStrackingProvider.SEGMENT_ID);
      GPStrackingProvider.sURIMatcher.addURI(GPStracking.AUTHORITY,
            "tracks/#/segments/#/media", GPStrackingProvider.SEGMENT_MEDIA);
      GPStrackingProvider.sURIMatcher.addURI(GPStracking.AUTHORITY,
            "tracks/#/segments/#/waypoints", GPStrackingProvider.WAYPOINTS);
      GPStrackingProvider.sURIMatcher.addURI(GPStracking.AUTHORITY,
            "tracks/#/segments/#/waypoints/#",
            GPStrackingProvider.WAYPOINT_ID);
      GPStrackingProvider.sURIMatcher.addURI(GPStracking.AUTHORITY,
            "tracks/#/segments/#/waypoints/#/media",
            GPStrackingProvider.WAYPOINT_MEDIA);
      GPStrackingProvider.sURIMatcher.addURI(GPStracking.AUTHORITY, "media",
            GPStrackingProvider.MEDIA);
      GPStrackingProvider.sURIMatcher.addURI(GPStracking.AUTHORITY,
            "media/#", GPStrackingProvider.MEDIA_ID);

      GPStrackingProvider.sURIMatcher.addURI(GPStracking.AUTHORITY,
            "live_folders/tracks", GPStrackingProvider.LIVE_FOLDERS);
      GPStrackingProvider.sURIMatcher.addURI(GPStracking.AUTHORITY,
            "search_suggest_query", GPStrackingProvider.SEARCH_SUGGEST_ID);
   }

   private DatabaseHelper mDbHelper;

   @Override
   public int delete(Uri uri, String selection, String[] selectionArgs) {
      int match = GPStrackingProvider.sURIMatcher.match(uri);
      int affected = 0;
      switch (match) {
      case GPStrackingProvider.TRACK_ID:
         affected = this.mDbHelper.deleteTrack(new Long(uri
               .getLastPathSegment()).longValue());
         break;
      case GPStrackingProvider.MEDIA_ID:
         affected = this.mDbHelper.deleteMedia(new Long(uri
               .getLastPathSegment()).longValue());
         break;
      default:
         affected = 0;
         break;
      }
      return affected;
   }

   @Override
   public String getType(Uri uri) {
      int match = GPStrackingProvider.sURIMatcher.match(uri);
      String mime = null;
      switch (match) {
      case TRACKS:
         mime = Tracks.CONTENT_TYPE;
         break;
```

```
   case TRACK_ID:
      mime = Tracks.CONTENT_ITEM_TYPE;
      break;
   case SEGMENTS:
      mime = Segments.CONTENT_TYPE;
      break;
   case SEGMENT_ID:
      mime = Segments.CONTENT_ITEM_TYPE;
      break;
   case WAYPOINTS:
      mime = Waypoints.CONTENT_TYPE;
      break;
   case WAYPOINT_ID:
      mime = Waypoints.CONTENT_ITEM_TYPE;
      break;
   }
   return mime;
}

@Override
public Uri insert(Uri uri, ContentValues values) {
   Uri insertedUri = null;
   int match = GPStrackingProvider.sURIMatcher.match(uri);
   List<String> pathSegments = null;
   long trackId;
   long segmentId;
   long waypointId = -1;
   long mediaId;
   switch (match) {
   case WAYPOINTS:
      pathSegments = uri.getPathSegments();
      trackId = Integer.parseInt(pathSegments.get(1));
      segmentId = Integer.parseInt(pathSegments.get(3));

      Location loc = new Location(TAG);

      Double latitude = values.getAsDouble(Waypoints.LATITUDE);
      Double longitude = values.getAsDouble(Waypoints.LONGITUDE);
      Long time = values.getAsLong(Waypoints.TIME);
      Long speed = values.getAsLong(Waypoints.SPEED);
      if (time == null) {
         time = System.currentTimeMillis();
      }
      if (speed == null) {
         speed = 0l;
      }
      loc.setLatitude(latitude);
      loc.setLongitude(longitude);
      loc.setTime(time);
      loc.setSpeed(speed);

      if (values.containsKey(Waypoints.ACCURACY)) {
         loc.setAccuracy(values.getAsFloat(Waypoints.ACCURACY));
      }
      if (values.containsKey(Waypoints.ALTITUDE)) {
         loc.setAltitude(values.getAsDouble(Waypoints.ALTITUDE));

      }
      if (values.containsKey(Waypoints.BEARING)) {
         loc.setBearing(values.getAsFloat(Waypoints.BEARING));
      }
```

```
            ContentValues valuesHelp = new ContentValues();
            if (values.containsKey(Waypoints.MINLATENCY)) {
               valuesHelp.put(Waypoints.MINLATENCY, values.getAsLong(Waypoints.MINLATENCY));
               valuesHelp.put(Waypoints.MAXLATENCY, values.getAsLong(Waypoints.MAXLATENCY));
               valuesHelp.put(Waypoints.RECEIVED, values.getAsLong(Waypoints.RECEIVED));
               valuesHelp.put(Waypoints.LOST, values.getAsLong(Waypoints.LOST));
            }

            // CellID and Lac information
            valuesHelp.put(Waypoints.CELLID, values.getAsInteger(Waypoints.CELLID));
            valuesHelp.put(Waypoints.LAC, values.getAsInteger(Waypoints.LAC));

            // NetworkType
            valuesHelp.put(Waypoints.NETWORKTYPE,
values.getAsInteger(Waypoints.NETWORKTYPE));

            waypointId = this.mDbHelper.insertWaypoint(trackId, segmentId, loc, valuesHelp);
            insertedUri = ContentUris.withAppendedId(uri, waypointId);
            break;
         case WAYPOINT_MEDIA:
            pathSegments = uri.getPathSegments();
            trackId = Integer.parseInt(pathSegments.get(1));
            segmentId = Integer.parseInt(pathSegments.get(3));
            waypointId = Integer.parseInt(pathSegments.get(5));
            String mediaUri = values.getAsString(Media.URI);
            mediaId = this.mDbHelper.insertMedia(trackId, segmentId, waypointId, mediaUri);
            insertedUri = ContentUris.withAppendedId(Media.CONTENT_URI, mediaId);
            break;
         case SEGMENTS:
            pathSegments = uri.getPathSegments();
            trackId = Integer.parseInt(pathSegments.get(1));
            segmentId = this.mDbHelper.toNextSegment(trackId);
            insertedUri = ContentUris.withAppendedId(uri, segmentId);
            break;
         case TRACKS:
            String name = (values == null) ? "" : values.getAsString(Tracks.NAME);

            ContentValues valuesHelper = new ContentValues();
            String NMC = values.getAsString(Tracks.NMC);
            String ServerIP = values.getAsString(Tracks.SERVERIP);
            String Area = values.getAsString(Tracks.AREA);

            valuesHelper.put(Tracks.NMC, NMC);
            valuesHelper.put(Tracks.SERVERIP, ServerIP);
            valuesHelper.put(Tracks.AREA, Area);
            if (NMC == "DOWNLOAD") {
               int PacketSize = values.getAsInteger(Tracks.PACKETSIZE);
               valuesHelper.put(Tracks.PACKETSIZE, PacketSize);
            } else if (NMC == "PING") {
               int PingAttempts = values.getAsInteger(Tracks.PINGATTEMPTS);
               valuesHelper.put(Tracks.PINGATTEMPTS, PingAttempts);
            }

            trackId = this.mDbHelper.toNextTrack(name, valuesHelper);
            insertedUri = ContentUris.withAppendedId(uri, trackId);
            break;
         default:
            Log.e(GPStrackingProvider.TAG, "Unable to match the insert URI: "
                  + uri.toString());
            insertedUri = null;
            break;
         }
```

```
      return insertedUri;
   }

   @Override
   public boolean onCreate() {
      if (this.mDbHelper == null) {
         this.mDbHelper = new DatabaseHelper(getContext());
      }
      return true;
   }

   @Override
   public Cursor query(Uri uri, String[] projection, String selection,
         String[] selectionArgs, String sortOrder) {
      int match = GPStrackingProvider.sURIMatcher.match(uri);

      String tableName = null;
      String whereclause = null;
      String sortorder = null;
      List<String> pathSegments = uri.getPathSegments();
      switch (match) {
      case TRACKS:
         tableName = Tracks.TABLE;
         break;
      case TRACK_ID:
         tableName = Tracks.TABLE;
         whereclause = Tracks._ID + " = " + new Long(pathSegments.get(1)).longValue();
         break;
      case SEGMENTS:
         tableName = Segments.TABLE;
         whereclause = Segments.TRACK + " = " + new Long(pathSegments.get(1)).longValue();
         break;
      case SEGMENT_ID:
         tableName = Segments.TABLE;
         whereclause = Segments.TRACK + " = "
               + new Long(pathSegments.get(1)).longValue() + " and "
               + Segments._ID + " = "
               + new Long(pathSegments.get(3)).longValue();
         break;
      case WAYPOINTS:
         tableName = Waypoints.TABLE;
         whereclause = Waypoints.SEGMENT + " = " + new
Long(pathSegments.get(3)).longValue();
         break;
      case WAYPOINT_ID:
         tableName = Waypoints.TABLE;
         whereclause = Waypoints.SEGMENT + " = "
               + new Long(pathSegments.get(3)).longValue() + " and "
               + Waypoints._ID + " = "
               + new Long(pathSegments.get(5)).longValue();
         break;
      case TRACK_WAYPOINTS:
         tableName = Waypoints.TABLE + " INNER JOIN " + Segments.TABLE
               + " ON " + Segments.TABLE + "." + Segments._ID + "=="
               + Waypoints.SEGMENT;
         whereclause = Segments.TRACK + " = " + new Long(pathSegments.get(1)).longValue();
         break;
      case GPStrackingProvider.MEDIA:
         tableName = Media.TABLE;
         break;
      case GPStrackingProvider.MEDIA_ID:
         tableName = Media.TABLE;
```

```
   whereclause = Media._ID + " = " + new Long(pathSegments.get(1)).longValue();
   break;
case TRACK_MEDIA:
   tableName = Media.TABLE;
   whereclause = Media.TRACK + " = " + new Long(pathSegments.get(1)).longValue();
   break;
case SEGMENT_MEDIA:
   tableName = Media.TABLE;
   whereclause = Media.TRACK + " = "
         + new Long(pathSegments.get(1)).longValue() + " and "
         + Media.SEGMENT + " = "
         + new Long(pathSegments.get(3)).longValue();
   break;
case WAYPOINT_MEDIA:
   tableName = Media.TABLE;
   whereclause = Media.TRACK + " = "
         + new Long(pathSegments.get(1)).longValue() + " and "
         + Media.SEGMENT + " = "
         + new Long(pathSegments.get(3)).longValue() + " and "
         + Media.WAYPOINT + " = "
         + new Long(pathSegments.get(5)).longValue();
   break;
case SEARCH_SUGGEST_ID:
   tableName = Tracks.TABLE;
   if (selectionArgs[0] == null || selectionArgs[0].equals("")) {
      selection = null;
      selectionArgs = null;
      sortorder = Tracks.CREATION_TIME + " desc";
   } else {
      selectionArgs[0] = "%" + selectionArgs[0] + "%";
   }
   projection = SUGGEST_PROJECTION;
   break;
case LIVE_FOLDERS:
   tableName = Tracks.TABLE;
   projection = LIVE_PROJECTION;
   sortorder = Tracks.CREATION_TIME + " desc";
   break;
default:
   Log.e(GPStrackingProvider.TAG,
         "Unable to come to an action in the query uri: "
               + uri.toString());
   return null;
}

// SQLiteQueryBuilder is a helper class that creates the
// proper SQL syntax for us.
SQLiteQueryBuilder qBuilder = new SQLiteQueryBuilder();

// Set the table we're querying.
qBuilder.setTables(tableName);

// If the query ends in a specific record number, we're
// being asked for a specific record, so set the
// WHERE clause in our query.
if (whereclause != null) {
   qBuilder.appendWhere(whereclause);
}

// Make the query.
SQLiteDatabase mDb = this.mDbHelper.getWritableDatabase();
Cursor c = qBuilder.query(mDb, projection, selection, selectionArgs,
```

```
            null, null, sortorder);
      c.setNotificationUri(getContext().getContentResolver(), uri);
      return c;
   }

   @Override
   public int update(Uri uri, ContentValues givenValues, String selection,
         String[] selectionArgs) {
      int updates = -1;

      int match = GPStrackingProvider.sURIMatcher.match(uri);

      String tableName;
      String whereclause;
      ContentValues args = new ContentValues();
      Uri notifyUri;

      switch (match) {
      case TRACK_ID:
         tableName = Tracks.TABLE;
         long trackId = new Long(uri.getLastPathSegment()).longValue();
         whereclause = Tracks._ID + " = " + trackId;
         args.put(Tracks.NAME, givenValues.getAsString(Tracks.NAME));
         notifyUri = ContentUris.withAppendedId(Tracks.CONTENT_URI, trackId);
         break;
      default:
         Log.e(GPStrackingProvider.TAG,
               "Unable to come to an action in the query uri"
                     + uri.toString());
         return -1;
      }

      // Execute the query.
      SQLiteDatabase mDb = this.mDbHelper.getWritableDatabase();
      updates = mDb.update(tableName, args, whereclause, null);

      ContentResolver resolver = this.getContext().getContentResolver();
      resolver.notifyChange(notifyUri, null);

      return updates;
   }
}
```

GPSLoggerService.java

```
package nl.sogeti.android.gpstracker.logger;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.NetworkInterface;
import java.net.SocketException;
import java.net.UnknownHostException;
import java.util.Enumeration;
import java.util.LinkedList;
import java.util.Queue;
import java.util.Vector;
import java.util.concurrent.Semaphore;

import nl.sogeti.android.gpstracker.R;
```

```
import nl.sogeti.android.gpstracker.actions.IndoorsGeopointSingleton;
import nl.sogeti.android.gpstracker.db.GPStracking.Media;
import nl.sogeti.android.gpstracker.db.GPStracking.Tracks;
import nl.sogeti.android.gpstracker.db.GPStracking.Waypoints;
import nl.sogeti.android.gpstracker.util.Constants;
import nl.sogeti.android.gpstracker.viewer.LoggerMap;
import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.app.Service;
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.content.SharedPreferences.OnSharedPreferenceChangeListener;
import android.location.GpsSatellite;
import android.location.GpsStatus;
import android.location.GpsStatus.Listener;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.net.Uri;
import android.os.Bundle;
import android.os.Handler;
import android.os.IBinder;
import android.os.Looper;
import android.os.Message;
import android.os.PowerManager;
import android.os.RemoteException;
import android.preference.PreferenceManager;
import android.telephony.CellLocation;
import android.telephony.PhoneStateListener;
import android.telephony.TelephonyManager;
import android.telephony.gsm.GsmCellLocation;
import android.util.Log;
import android.widget.Toast;

/**
 * A system service as controlling the background logging of gps locations.
 *
 * @author Alexandros Ntakas
 */
public class GPSLoggerService extends Service {
   private static final int MAX_REASONABLE_SPEED = 60;
   private static final int MAX_REASONABLE_ALTITUDECHANGE = 200;
   private static final String TAG = "OGT.GPSLoggerService";
   private static final int LOGGING_GLOBAL = 3;
   private static final String SERVICESTATE_STATE = "SERVICESTATE_STATE";
   private static final String SERVICESTATE_PRECISION = "SERVICESTATE_PRECISION";
   private static final String SERVICESTATE_SEGMENTID = "SERVICESTATE_SEGMENTID";
   private static final String SERVICESTATE_TRACKID = "SERVICESTATE_TRACKID";

   private static final int ADDGPSSTATUSLISTENER = 0;
   private static final int REQUEST_FINEGPS_LOCATIONUPDATES = 1;
   private static final int REQUEST_NORMALGPS_LOCATIONUPDATES = 2;
   private static final int REQUEST_COARSEGPS_LOCATIONUPDATES = 3;
   private static final int REQUEST_GLOBALGPS_LOCATIONUPDATES = 4;
   private static final int REGISTERONSHAREDPREFERENCECHANGELISTENER = 5;
```

```
   private static final int STOPLOOPER = 6;

   private Download DL;
   private TelephonyManager tm;
   private CellID_LAC cidlac;

   private Context mContext;
   private LocationManager mLocationManager;
   private NotificationManager mNoticationManager;
   private PowerManager.WakeLock mWakeLock;
   private Handler mHandler;

   private boolean mSpeedSanityCheck;
   private long mTrackId = -1;
   private long mSegmentId = -1;
   private long mWaypointId = -1;
   private int mPrecision;
   private int mLoggingState = Constants.STOPPED;
   private boolean mStartNextSegment;

   private Location mPreviousLocation;
   private Notification mNotification;

   private Vector<Location> mWeakLocations;
   private Queue<Double> mAltitudes;

   /**
    * <code>mAcceptableAccuracy</code> indicates the maximum acceptable
    * accuracy of a waypoint in meters.
    */
   private float mMaxAcceptableAccuracy = 20;
   private int mSatellites = 0;

   /**
    * Listens to changes in preference to precision and sanity checks
    */
   private OnSharedPreferenceChangeListener mSharedPreferenceChangeListener = new
OnSharedPreferenceChangeListener() {
      public void onSharedPreferenceChanged(SharedPreferences sharedPreferences, String
key) {
      }
   };

   /**
    * Listens to location changes and provider availability
    */
   private LocationListener mLocationListener = new LocationListener() {
      public void onLocationChanged(Location location) {
         if (isLogging() && (StartTime > 0)
               && (System.currentTimeMillis() - StartTime > 10000)) {
            if (mStartNextSegment) {
               mStartNextSegment = false;
               startNewSegment();
            }
            storeLocation(location);
         }
      }

      public void onProviderDisabled(String provider) {
         if (mPrecision != LOGGING_GLOBAL
               && provider.equals(LocationManager.GPS_PROVIDER)) {
            disabledProviderNotification(R.string.service_gpsdisabled);
```

```
      } else if (mPrecision == LOGGING_GLOBAL
            && provider.equals(LocationManager.NETWORK_PROVIDER)) {
         disabledProviderNotification(R.string.service_datadisabled);
      }

   }

   public void onProviderEnabled(String provider) {
      if (mPrecision != LOGGING_GLOBAL
            && provider.equals(LocationManager.GPS_PROVIDER)) {
         enabledProviderNotification(R.string.service_gpsenabled);
         mStartNextSegment = true;
      } else if (mPrecision == LOGGING_GLOBAL
            && provider.equals(LocationManager.NETWORK_PROVIDER)) {
         enabledProviderNotification(R.string.service_dataenabled);
      }
   }

   public void onStatusChanged(String provider, int status, Bundle extras) {
      Log.w(TAG, String.format("Provider %s changed to status %d",
            provider, status));
   }
};

/**
 * Listens to GPS status changes
 */
private Listener mStatusListener = new GpsStatus.Listener() {
   public synchronized void onGpsStatusChanged(int event) {
      switch (event) {
      case GpsStatus.GPS_EVENT_SATELLITE_STATUS:
         GpsStatus status = mLocationManager.getGpsStatus(null);
         mSatellites = 0;
         Iterable<GpsSatellite> list = status.getSatellites();
         for (GpsSatellite satellite : list) {
            if (satellite.usedInFix()) {
               mSatellites++;
            }
         }
         updateNotification();
         break;
      default:
         break;
      }
   }
};

/**
 * Listens to Signal changes
 */
private PhoneStateListener onCellChange = new PhoneStateListener() {
   GsmCellLocation loc;
   int cellid;
   int lac;
   int networkType;
   String networkTypeString;

   public void onCellLocationChanged(CellLocation location) {
      loc = (GsmCellLocation) location;
      cellid = loc.getCid();
      lac = loc.getLac();
```

```
      networkType = tm.getNetworkType();
      switch (networkType) {
      case TelephonyManager.NETWORK_TYPE_UNKNOWN:
         networkTypeString = "UNKNOWN";
         break;
      case TelephonyManager.NETWORK_TYPE_GPRS:
         networkTypeString = "GPRS";
         break;
      case TelephonyManager.NETWORK_TYPE_EDGE:
         networkTypeString = "EDGE";
         break;
      case TelephonyManager.NETWORK_TYPE_UMTS:
         networkTypeString = "UMTS";
         cellid = transform(cellid, "UMTS");
         break;
      case TelephonyManager.NETWORK_TYPE_CDMA:
         networkTypeString = "CDMA";
         break;
      case TelephonyManager.NETWORK_TYPE_EVDO_0:
         networkTypeString = "EVDO_0";
         cellid = transform(cellid, "EVDO_0");
         break;
      case TelephonyManager.NETWORK_TYPE_EVDO_A:
         networkTypeString = "EVDO_A";
         cellid = transform(cellid, "EVDO_A");
         break;
      case TelephonyManager.NETWORK_TYPE_1xRTT:
         networkTypeString = "1xRTT";
         break;
      case TelephonyManager.NETWORK_TYPE_HSDPA:
         networkTypeString = "HSDPA";
         cellid = transform(cellid, "HSDPA");
         break;
      case TelephonyManager.NETWORK_TYPE_HSUPA:
         networkTypeString = "HSUPA";
         cellid = transform(cellid, "HSUPA");
         break;
      case TelephonyManager.NETWORK_TYPE_HSPA:
         networkTypeString = "HSPA";
         cellid = transform(cellid, "HSPA");
         break;
      case TelephonyManager.NETWORK_TYPE_IDEN:
         networkTypeString = "IDEN";
         break;
      }

      cidlac.setValues(cellid, lac);
   }

   public int transform(int cellid, String caller) {
      Log.e("CellIDTest", "transform" + " ...caller:" + caller);
      String cidhexS = Integer.toHexString(cellid);
      int length_cidhexS = cidhexS.length();
      String cidhexSlast4S = cidhexS.substring(length_cidhexS - 4);
      int cid = Integer.parseInt(cidhexSlast4S, 16);

      return cid;
   }
};

private IBinder mBinder = new IGPSLoggerServiceRemote.Stub() {
   public int loggingState() throws RemoteException {
```

```
      return mLoggingState;
   }

   public long startLogging() throws RemoteException {
      GPSLoggerService.this.startLogging();
      return mTrackId;
   }

   public void pauseLogging() throws RemoteException {
      GPSLoggerService.this.pauseLogging();
   }

   public long resumeLogging() throws RemoteException {
      GPSLoggerService.this.resumeLogging();
      return mSegmentId;
   }

   public void stopLogging() throws RemoteException {
      GPSLoggerService.this.stopLogging();
   }

   public Uri storeMediaUri(Uri mediaUri) throws RemoteException {
      GPSLoggerService.this.storeMediaUri(mediaUri);
      return null;
   }

   public boolean isMediaPrepared() throws RemoteException {
      return GPSLoggerService.this.isMediaPrepared();
   }
};

private Semaphore handlerGatekeeper;

private class GPSLoggerServiceThread extends Thread {
   public void run() {
      Looper.prepare();
      mHandler = new Handler() {
         public void handleMessage(Message msg) {
            _handleMessage(msg);
         }
      };
      handlerGatekeeper.release();
      Looper.loop();
   }
}

/**
 * Message handler method to do the work off-loaded by mHandler to
 * GPSLoggerServiceThread
 *
 * @param msg
 */
private void _handleMessage(Message msg) {
   switch (msg.what) {
   case ADDGPSSTATUSLISTENER:
      this.mLocationManager.addGpsStatusListener(mStatusListener);
      break;
   case REGISTERONSHAREDPREFERENCECHANGELISTENER:
      PreferenceManager.getDefaultSharedPreferences(this.mContext)
            .registerOnSharedPreferenceChangeListener(
mSharedPreferenceChangeListener);
      break;
```

```
   case REQUEST_FINEGPS_LOCATIONUPDATES:
      mMaxAcceptableAccuracy = 10f;
      mLocationManager.requestLocationUpdates(
            LocationManager.GPS_PROVIDER, 0, 0, this.mLocationListener);
      break;
   case REQUEST_NORMALGPS_LOCATIONUPDATES:
      mMaxAcceptableAccuracy = 20f;
      mLocationManager.requestLocationUpdates(
            LocationManager.GPS_PROVIDER, 0, 0, this.mLocationListener);
      break;
   case REQUEST_COARSEGPS_LOCATIONUPDATES:
      mMaxAcceptableAccuracy = 50f;
      mLocationManager.requestLocationUpdates(
            LocationManager.GPS_PROVIDER, 0, 0, this.mLocationListener);
      break;
   case REQUEST_GLOBALGPS_LOCATIONUPDATES:
      mMaxAcceptableAccuracy = 1000f;
      mLocationManager.requestLocationUpdates(
            LocationManager.NETWORK_PROVIDER, 0, 0, this.mLocationListener);
      if (!isNetworkConnected()) {
         disabledProviderNotification(R.string.service_connectiondisabled);
      }
      break;
   case STOPLOOPER:
      mLocationManager.removeGpsStatusListener(mStatusListener);
      mLocationManager.removeUpdates(mLocationListener);
      Looper.myLooper().quit();
      break;
   }
}

/**
 * Called by the system when the service is first created. Do not call this
 * method directly. Be sure to call super.onCreate().
 */
@Override
public void onCreate() {
   super.onCreate();

   handlerGatekeeper = new Semaphore(0);
   new GPSLoggerServiceThread().start();

   mWeakLocations = new Vector<Location>(3);
   mAltitudes = new LinkedList<Double>();
   mLoggingState = Constants.STOPPED;
   mStartNextSegment = false;
   mContext = getApplicationContext();
   mLocationManager = (LocationManager) this.mContext
         .getSystemService(Context.LOCATION_SERVICE);
   mNoticationManager = (NotificationManager) this.mContext
         .getSystemService(Context.NOTIFICATION_SERVICE);
   mNoticationManager.cancel(R.layout.map);

   SharedPreferences sharedPreferences = PreferenceManager
         .getDefaultSharedPreferences(this.mContext);

   mSpeedSanityCheck = sharedPreferences.getBoolean(
         Constants.SPEEDSANITYCHECK, false);
   boolean startImmidiatly = PreferenceManager
         .getDefaultSharedPreferences(this.mContext).getBoolean(
               Constants.LOGATSTARTUP, false);
```

```
   if (startImmidiatly && mLoggingState == Constants.STOPPED) {
      startLogging();
      ContentValues values = new ContentValues();
      values.put(Tracks.NAME, "Recorded at startup");
      getContentResolver().update(
            ContentUris.withAppendedId(Tracks.CONTENT_URI, mTrackId),
            values, null, null);
   }

   Message msg = Message.obtain();
   msg.what = REGISTERONSHAREDPREFERENCECHANGELISTENER;
   sendMessage(msg);

}

@Override
public void onDestroy() {
   super.onDestroy();
   stopLogging();
   Message msg = Message.obtain();
   msg.what = STOPLOOPER;
   sendMessage(msg);
}

private void crashProtectState() {
   SharedPreferences preferences = PreferenceManager
         .getDefaultSharedPreferences(mContext);
   Editor editor = preferences.edit();
   editor.putLong(SERVICESTATE_TRACKID, mTrackId);
   editor.putLong(SERVICESTATE_SEGMENTID, mSegmentId);
   editor.putInt(SERVICESTATE_PRECISION, mPrecision);
   editor.putInt(SERVICESTATE_STATE, mLoggingState);
   editor.commit();
}

private synchronized void crashRestoreState() {
   SharedPreferences preferences = PreferenceManager
         .getDefaultSharedPreferences(mContext);
   long previousState = preferences.getInt(SERVICESTATE_STATE, Constants.STOPPED);
   if (previousState == Constants.LOGGING
         || previousState == Constants.PAUSED) {
      Log.w(TAG, "Recovering from a crash or kill and restoring state.");
      setupNotification();

      mTrackId = preferences.getLong(SERVICESTATE_TRACKID, -1);
      mSegmentId = preferences.getLong(SERVICESTATE_SEGMENTID, -1);
      mPrecision = preferences.getInt(SERVICESTATE_PRECISION, -1);
      if (previousState == Constants.LOGGING) {
         mLoggingState = Constants.PAUSED;
         resumeLogging();
      } else if (previousState == Constants.PAUSED) {
         mLoggingState = Constants.LOGGING;
         pauseLogging();
      }
   }
}

@Override
public IBinder onBind(Intent intent) {
   return this.mBinder;
}
```

```
protected boolean isLogging() {
   return this.mLoggingState == Constants.LOGGING;
}

protected boolean isMediaPrepared() {
   return !(mTrackId < 0 || mSegmentId < 0 || mWaypointId < 0);
}

public synchronized void startLogging() {
   if (this.mLoggingState == Constants.STOPPED) {
      ServerIPChoice = new String(PreferenceManager
            .getDefaultSharedPreferences(this.mContext).getString(
                  Constants.SERVERIP, "192.168.1.1"));
      Log.e("!!!!", ServerIPChoice);
      NetworkingMeasurementChoice = new Integer(PreferenceManager
            .getDefaultSharedPreferences(this.mContext).getString(
                  Constants.NETWORKINGMEASUREMENT, "0")).intValue();
      PacketSizeChoice = new Integer(PreferenceManager
            .getDefaultSharedPreferences(this.mContext).getString(
                  Constants.PACKETSIZE, "0")).intValue();
      IndoorsSelection = new Boolean(PreferenceManager
            .getDefaultSharedPreferences(this.mContext).getBoolean(
                  Constants.INDOORSCHECK, false)).booleanValue();
      switch (PacketSizeChoice) {
      case Constants.BYTES200:
         msg1 = "200 ";
         PacketSize1 = 200;
         break;
      case Constants.BYTES500:
         msg1 = "500 ";
         PacketSize1 = 500;
         break;
      case Constants.BYTES1000:
         msg1 = "1000";
         PacketSize1 = 1000;
         break;
      case Constants.BYTES1400:
         msg1 = "1400";
         PacketSize1 = 1400;
         break;
      }
      PingAttemptsChoice = new Integer(PreferenceManager
            .getDefaultSharedPreferences(this.mContext).getString(
                  Constants.PINGATTEMPTS, "0")).intValue();
      switch (PingAttemptsChoice) {
      case Constants.PINGATTEMPTS4:
         PingAttempts1 = 4;
         break;
      case Constants.PINGATTEMPTS8:
         PingAttempts1 = 8;
         break;
      case Constants.PINGATTEMPTS16:
         PingAttempts1 = 16;
         break;
      case Constants.PINGATTEMPTS32:
         PingAttempts1 = 32;
         break;
      }
      StartTime = 0; // To avoid storeLocation to store a location before
                  // DL or Ping has started

      startNewTrack();
```

```
      requestLocationUpdates();
      Message msg = Message.obtain();
      msg.what = ADDGPSSTATUSLISTENER;
      sendMessage(msg);
      this.mLoggingState = Constants.LOGGING;
      updateWakeLock();
      setupNotification();
      crashProtectState();

      // Telephony Manager (listen for signal changes)
      cidlac = new CellID_LAC();
      tm = (TelephonyManager) getSystemService(Context.TELEPHONY_SERVICE);
      tm.listen(onCellChange, PhoneStateListener.LISTEN_CELL_LOCATION);

      if (NetworkingMeasurementChoice == Constants.DOWNLOADCHOICE) {
         // Download
         c1 = new Count();
         d = new Downloading();

         DL = new Download(msg1, PacketSize1, ServerIPChoice);
         new Thread(DL).start();
      } else if (NetworkingMeasurementChoice == Constants.PINGCHOICE) {
         c = new Count();
         p = new Pinging();
         snts = new StartNextTimeSegment();
         cts = new CurrentTimeSegment();
         st = new SendTime();
         r = new Results();

         if (IndoorsSelection == false) {
            // Ping Outdoors
            p.setTrue();
            new Thread(new PingSend(PingAttempts1, ServerIPChoice)).start();
            new Thread(new PingReceive()).start();
         } else if (IndoorsSelection == true) {
            // Ping Indoors
            p.setTrue();
            new Thread(new PingSendIndoors(PingAttempts1, ServerIPChoice)).start();
            new Thread(new PingReceive()).start();
         }
      }
   }
}

public synchronized void pauseLogging() {
   if (this.mLoggingState == Constants.LOGGING) {
      mLocationManager.removeGpsStatusListener(mStatusListener);
      mLocationManager.removeUpdates(mLocationListener);
      mLoggingState = Constants.PAUSED;
      mPreviousLocation = null;
      updateWakeLock();
      updateNotification();
      crashProtectState();
   }
}

public synchronized void resumeLogging() {
   if (this.mLoggingState == Constants.PAUSED) {
      if (mPrecision != LOGGING_GLOBAL) {
         mStartNextSegment = true;
      }
      requestLocationUpdates();
```

```
            Message msg = Message.obtain();
            msg.what = ADDGPSSTATUSLISTENER;
            sendMessage(msg);

            this.mLoggingState = Constants.LOGGING;
            updateWakeLock();
            updateNotification();
            crashProtectState();
         }
      }

      public synchronized void stopLogging() {
         if (this.mLoggingState == Constants.PAUSED
               || this.mLoggingState == Constants.LOGGING) {
            PreferenceManager.getDefaultSharedPreferences(this.mContext)
                  .unregisterOnSharedPreferenceChangeListener(
                        this.mSharedPreferenceChangeListener);

            mLocationManager.removeGpsStatusListener(mStatusListener);
            mLocationManager.removeUpdates(mLocationListener);

            mLoggingState = Constants.STOPPED;
            updateWakeLock();
            mNoticationManager.cancel(R.layout.map);
            crashProtectState();

            if (NetworkingMeasurementChoice == Constants.DOWNLOADCHOICE) {
               DL.stopDownload();
            } else if (NetworkingMeasurementChoice == Constants.PINGCHOICE) {
               p.setFalse();
            }

            // Stop listening for updates (signal and cell location changes)
            tm.listen(onCellChange, PhoneStateListener.LISTEN_NONE);
         }
      }

      private void setupNotification() {
         mNoticationManager.cancel(R.layout.map);

         int icon = R.drawable.ic_maps_indicator_current_position;
         CharSequence tickerText = getResources().getString(R.string.service_start);
         long when = System.currentTimeMillis();

         mNotification = new Notification(icon, tickerText, when);
         mNotification.flags |= Notification.FLAG_ONGOING_EVENT;

         updateNotification();
      }

      private void updateNotification() {
         CharSequence contentTitle = getResources().getString(R.string.app_name);

         String precision = getResources().getStringArray(
               R.array.precision_choices)[mPrecision];
         String state = getResources().getStringArray(R.array.state_choices)[mLoggingState -
1];
         CharSequence contentText;
         switch (mPrecision) {
         case (LOGGING_GLOBAL):
            contentText = getResources().getString(
```

```
            R.string.service_networkstatus, state, precision);
      break;
   default:
      contentText = getResources().getString(R.string.service_gpsstatus,
            state, precision, mSatellites);
      break;
   }
   Intent notificationIntent = new Intent(this, LoggerMap.class);
   notificationIntent.setData(ContentUris.withAppendedId(
         Tracks.CONTENT_URI, mTrackId));

   PendingIntent contentIntent = PendingIntent.getActivity(this, 0,
         notificationIntent, Intent.FLAG_ACTIVITY_NEW_TASK);
   mNotification.setLatestEventInfo(this, contentTitle, contentText,
         contentIntent);

   mNoticationManager.notify(R.layout.map, mNotification);
}

private void enabledProviderNotification(int resId) {
   mNoticationManager.cancel(R.id.icon);
   CharSequence text = mContext.getString(resId);
   Toast toast = Toast.makeText(mContext.getApplicationContext(), text,
         Toast.LENGTH_LONG);
   toast.show();
}

private void disabledProviderNotification(int resId) {
   int icon = R.drawable.ic_maps_indicator_current_position;
   CharSequence tickerText = getResources().getString(resId);
   long when = System.currentTimeMillis();
   Notification gpsNotification = new Notification(icon, tickerText, when);
   gpsNotification.flags |= Notification.FLAG_AUTO_CANCEL;

   CharSequence contentTitle = getResources().getString(R.string.app_name);
   CharSequence contentText = getResources().getString(resId);
   Intent notificationIntent = new Intent(this, LoggerMap.class);
   notificationIntent.setData(ContentUris.withAppendedId(
         Tracks.CONTENT_URI, mTrackId));
   PendingIntent contentIntent = PendingIntent.getActivity(this, 0,
         notificationIntent, Intent.FLAG_ACTIVITY_NEW_TASK);
   gpsNotification.setLatestEventInfo(this, contentTitle, contentText,
         contentIntent);

   mNoticationManager.notify(R.id.icon, gpsNotification);
}

private void requestLocationUpdates() {
   this.mLocationManager.removeUpdates(mLocationListener);
   if (IndoorsSelection == false) {
      Log.e("GPSLoggerService",
            "requestLocationUpdates: indoors == false ");
      Message msg = Message.obtain();
      msg.what = REQUEST_FINEGPS_LOCATIONUPDATES;
      sendMessage(msg);
      handlerGatekeeper.release();
   } else if (IndoorsSelection == true) {
      Log.e("GPSLoggerService",
            "requestLocationUpdates: indoors == true ");
   }

}
```

```
   private void sendMessage(Message msg) {
      try {
         handlerGatekeeper.acquire();
         mHandler.sendMessage(msg);
         handlerGatekeeper.release();
      } catch (InterruptedException e) {
         Log.e(TAG,
               "Interrupted while waiting for handler to be created for message: "
                     + msg);
      }
   }

   private void updateWakeLock() {
      if (this.mLoggingState == Constants.LOGGING) {
         PowerManager pm = (PowerManager) this.mContext
               .getSystemService(Context.POWER_SERVICE);
         this.mWakeLock = pm
               .newWakeLock(PowerManager.PARTIAL_WAKE_LOCK, TAG);
         this.mWakeLock.acquire();
      } else {
         if (this.mWakeLock != null) {
            this.mWakeLock.release();
            this.mWakeLock = null;
         }
      }
   }

   /**
    * Some GPS waypoints received are of too low quality for tracking use.
    * Here, we filter those out.
    *
    * @param proposedLocation
    * @return either the (cleaned) original or null when unacceptable
    */
   public Location locationFilter(Location proposedLocation) {
      // Do not log a waypoint which is more inaccurate then is configured to
      // be acceptable
      if (proposedLocation != null
            && proposedLocation.getAccuracy() > mMaxAcceptableAccuracy) {
         Log.w(TAG,
               String.format(
                     "A weak location was received, lots of inaccuracy... (%f is more then
max %f)",
                     proposedLocation.getAccuracy(),
                     mMaxAcceptableAccuracy));
         proposedLocation = addBadLocation(proposedLocation);
      }

      // Do not log a waypoint which might be on any side of the previous
      // waypoint
      if (proposedLocation != null
            && mPreviousLocation != null
            && proposedLocation.getAccuracy() > mPreviousLocation
                  .distanceTo(proposedLocation)) {
         Log.w(TAG,
               String.format(
                     "A weak location was received, not quite clear from the previous
waypoint... (%f more then max %f)",
                     proposedLocation.getAccuracy(),
                     mPreviousLocation.distanceTo(proposedLocation)));
         proposedLocation = addBadLocation(proposedLocation);
```

```
   }

   // Speed checks for NETWORK logging, check if the proposed location
   // could be reached from the previous one in sane speed
   if (mSpeedSanityCheck && proposedLocation != null
         && mPreviousLocation != null && mPrecision == LOGGING_GLOBAL) {
      // To avoid near instant teleportation on network location or
      // glitches cause continent hopping
      float meters = proposedLocation.distanceTo(mPreviousLocation);
      long seconds = (proposedLocation.getTime() - mPreviousLocation
            .getTime()) / 1000L;
      if (meters / seconds > MAX_REASONABLE_SPEED) {
         Log.w(TAG,
               "A strange location was recieved, a really high speed, prob wrong...");
         proposedLocation = addBadLocation(proposedLocation);
      }
   }

   // Remove speed if not sane
   if (mSpeedSanityCheck && proposedLocation != null
         && proposedLocation.getSpeed() > MAX_REASONABLE_SPEED) {
      Log.w(TAG, "A strange speed, a really high speed, prob wrong...");
      proposedLocation.removeSpeed();
   }

   // Remove altitude if not sane
   if (mSpeedSanityCheck && proposedLocation != null
         && proposedLocation.hasAltitude()) {
      if (!addSaneAltitude(proposedLocation.getAltitude())) {
         Log.w(TAG,
               "A strange altitude, a really big difference, prob wrong...");
         proposedLocation.removeAltitude();
      }
   }
   // Older bad locations will not be needed
   if (proposedLocation != null) {
      mWeakLocations.clear();
   }
   return proposedLocation;
}

/**
 * Store a bad location, when to many bad locations are stored the the
 * storage is cleared and the least bad one is returned
 *
 * @param location
 *            bad location
 * @return null when the bad location is stored or the least bad one if the
 *         storage was full
 */
private Location addBadLocation(Location location) {
   mWeakLocations.add(location);
   if (mWeakLocations.size() < 3) {
      location = null;
   } else {
      Location best = mWeakLocations.lastElement();
      for (Location whimp : mWeakLocations) {
         if (whimp.hasAccuracy() && best.hasAccuracy()
               && whimp.getAccuracy() < best.getAccuracy()) {
            best = whimp;
         } else {
            if (whimp.hasAccuracy() && !best.hasAccuracy()) {
```

```
                best = whimp;
             }
          }
       }
       mWeakLocations.clear();
       location = best;
    }
    return location;
}

/**
 * Builds a bit of knowledge about altitudes to expect and return if the
 * added value is deemed sane.
 *
 * @param altitude
 * @return whether the altitude is considered sane
 */
private boolean addSaneAltitude(double altitude) {
    boolean sane = true;
    double avg = 0;
    int elements = 0;
    // Even insane altitude shifts increases alter perception
    mAltitudes.add(altitude);
    if (mAltitudes.size() > 3) {
       mAltitudes.poll();
    }
    for (Double alt : mAltitudes) {
       avg += alt;
       elements++;
    }
    avg = avg / elements;
    sane = Math.abs(altitude - avg) < MAX_REASONABLE_ALTITUDECHANGE;

    return sane;
}

/**
 * Triggered by events that start a new track
 */
private void startNewTrack() {
    ContentValues valuesHelper = new ContentValues();
    if (NetworkingMeasurementChoice == Constants.DOWNLOADCHOICE) {
       valuesHelper.put(Tracks.NMC, "DOWNLOAD");
       valuesHelper.put(Tracks.PACKETSIZE, PacketSize1);
    } else if (NetworkingMeasurementChoice == Constants.PINGCHOICE) {
       valuesHelper.put(Tracks.NMC, "PING");
       valuesHelper.put(Tracks.PINGATTEMPTS, PingAttempts1);
    }
    valuesHelper.put(Tracks.SERVERIP, ServerIPChoice);
    if (IndoorsSelection == false) {
       valuesHelper.put(Tracks.AREA, "outdoors");
    } else {
       valuesHelper.put(Tracks.AREA, "indoors");
    }

    Uri newTrack = this.mContext.getContentResolver().insert(
          Tracks.CONTENT_URI, valuesHelper);
    mTrackId = new Long(newTrack.getLastPathSegment()).longValue();
    mStartNextSegment = true;
}

/**
```

```
 * Triggered by events that start a new segment
 */
private void startNewSegment() {
   this.mPreviousLocation = null;
   Uri newSegment = this.mContext.getContentResolver()
         .insert(Uri.withAppendedPath(Tracks.CONTENT_URI, mTrackId
               + "/segments"), null);
   mSegmentId = new Long(newSegment.getLastPathSegment()).longValue();
}

protected void storeMediaUri(Uri mediaUri) {
   if (isMediaPrepared()) {
      Uri mediaInsertUri = Uri.withAppendedPath(Tracks.CONTENT_URI,
            mTrackId + "/segments/" + mSegmentId + "/waypoints/"
                  + mWaypointId + "/media");
      ContentValues args = new ContentValues();
      args.put(Media.URI, mediaUri.toString());
      mContext.getContentResolver().insert(mediaInsertUri, args);
   } else {
      Log.e(TAG, "No logging done under which to store the track");
   }
}

/**
 * Use the ContentResolver mechanism to store a received location
 *
 * @param location
 */
public void storeLocation(Location location) {
   if (!isLogging()) {
      Log.e(TAG, String.format(
            "Not logging but storing location %s, prepare to fail",
            location.toString()));
   }

   mPreviousLocation = location;
   ContentValues args = new ContentValues();

   if (NetworkingMeasurementChoice == Constants.DOWNLOADCHOICE) {
      EndTime = System.currentTimeMillis();
      long packets = c1.GetCount();
      TimeDL = EndTime - StartTime; // msec
      long bitrate;
      if (TimeDL > 0) {
         bitrate = packets * PacketSize1 * 8 / TimeDL; // bitrate (kbps)
      } else {
         bitrate = 0;
         Log.e("GPSLoggerService", "ERROR: TimeDL == 0");
      }
      SpeedVariable = bitrate;
      c1.ZeroCount();
      StartTime = System.currentTimeMillis();
   } else if (NetworkingMeasurementChoice == Constants.PINGCHOICE) {
      long received = c.GetCount();
      long lost;
      EndTime = System.currentTimeMillis();

      if (received > 0) {
         latency = getAvgValue();
         minlatency = getMinValue();
         maxlatency = getMaxValue();
         lost = PingAttempts1 - received;
```

```
         } else {
            latency = 0;
            minlatency = 0;
            maxlatency = 0;
            lost = PingAttempts1;
            Log.e("GPSLoggerService",
                  "ERROR: Not a single ping was successful");
         }
         snts.setTrue();
         SpeedVariable = latency;

         args.put(Waypoints.MINLATENCY, minlatency);
         args.put(Waypoints.MAXLATENCY, maxlatency);
         args.put(Waypoints.RECEIVED, received);
         args.put(Waypoints.LOST, lost);
      }

      // Store CellID and Lac
      int cid = cidlac.getValues()[0];
      int lac = cidlac.getValues()[1];

      // Get NetworkType
      int networktype = tm.getNetworkType();

      args.put(Waypoints.CELLID, cid);
      args.put(Waypoints.LAC, lac);
      args.put(Waypoints.NETWORKTYPE, networktype);

      args.put(Waypoints.LATITUDE, new Double(location.getLatitude()));
      args.put(Waypoints.LONGITUDE, new Double(location.getLongitude()));
      args.put(Waypoints.SPEED, new Long(SpeedVariable));
      args.put(Waypoints.TIME, new Long(EndTime));

      if (location.hasAccuracy()) {
         args.put(Waypoints.ACCURACY, new Float(location.getAccuracy()));
      }
      if (location.hasAltitude()) {
         args.put(Waypoints.ALTITUDE, new Double(location.getAltitude()));

      }
      if (location.hasBearing()) {
         args.put(Waypoints.BEARING, new Float(location.getBearing()));
      }

      Uri waypointInsertUri = Uri.withAppendedPath(Tracks.CONTENT_URI,
            mTrackId + "/segments/" + mSegmentId + "/waypoints");
      Uri inserted = mContext.getContentResolver().insert(waypointInsertUri, args);
      mWaypointId = Long.parseLong(inserted.getLastPathSegment());

   }

   private boolean isNetworkConnected() {
      ConnectivityManager connMgr = (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
      NetworkInfo info = connMgr.getActiveNetworkInfo();

      return (info != null && info.isConnected());
   }

   public int NetworkingMeasurementChoice;
   public int PacketSizeChoice;
   public int PingAttemptsChoice;
```

```
public Boolean IndoorsSelection;
public String ServerIPChoice;

public int PacketSize1;
public String msg1;
public int PingAttempts1;

public long SpeedVariable;
public long StartTime;
public long EndTime;
public long TimeDL;

// DOWNLOAD !!! DOWNLOAD !!! DOWNLOAD !!! DOWNLOAD !!! DOWNLOAD !!!
public Count c1;
public Downloading d;

class Download extends Thread {
   private static final int SERVERPORT = 5001;
   private static final int CLIENTPORT = 5001;
   String msg;
   Boolean EstablishedConnection = false;
   DatagramSocket socket;
   int PacketSize;
   String ServerIP;
   InetAddress serverAddr;
   String clientAddr;

   public Download(String msg1, int PacketSize1, String ServerIPChoice) {
      msg = msg1;
      PacketSize = PacketSize1;
      ServerIP = ServerIPChoice;
   }

   public void run() {
      try {
         // Sending message to initiate, with packet selection info
         serverAddr = InetAddress.getByName(ServerIP);

         int TryToConnect = 0;
         while ((EstablishedConnection.equals(false)) && (TryToConnect < 5)) {
            socket = new DatagramSocket();
            byte[] buf = (msg).getBytes();
            DatagramPacket TodoPacket = new DatagramPacket(buf,
                  buf.length, serverAddr, SERVERPORT);
            socket.send(TodoPacket);
            Log.e("UDP_Download", "C: Sent TodoPacket. Waiting...");
            final String clientAddr = getLocalIpAddress();
            socket.close();

            // Receiving first packet to start Timer
            socket = new DatagramSocket(CLIENTPORT, InetAddress.getByName(clientAddr));
            byte[] buf1 = new byte[PacketSize];
            DatagramPacket first_packet = new DatagramPacket(buf1, buf1.length);
            socket.setSoTimeout(3000);
            try {
               socket.receive(first_packet);
               EstablishedConnection = true;
            } catch (IOException e) {
               // If no packet was received in 3 seconds timeout, send
               // to Server "FIN" packet
               // to stop sending (if Server has started sending data),
               // close socket and try again.
```

```
                socket.close();
                socket = new DatagramSocket();
                byte[] buf2 = ("FIN ").getBytes();
                DatagramPacket FinishPacket = new DatagramPacket(buf2,
                      buf2.length, serverAddr, SERVERPORT);
                for (int i = 0; i < 3; i++) {
                   socket.send(FinishPacket);
                }
                socket.close();
                TryToConnect++;
             }
          }

          if (EstablishedConnection == true) {
             d.setTrue();
             socket.setSoTimeout(0);

             // Call DLIndoorsTimer for Indoors Measurement
             if (IndoorsSelection == true) {
                double lat = IndoorsGeopointSingleton.getInstance(0, 0).latitude();
                double lon = IndoorsGeopointSingleton.getInstance(0, 0).longitude();
                DLIndoorsTimer dlITimer = new DLIndoorsTimer(lat, lon);
                new Thread(dlITimer).start();
             }

             StartTime = System.currentTimeMillis();

             // Receiving packets
             byte[] buf3 = new byte[PacketSize];
             DatagramPacket IncomingPacket = new DatagramPacket(buf3, buf3.length);
             while (d.getValue() == true) {
                socket.receive(IncomingPacket);
                c1.AddCount();
             }
             socket.close();
          } else {
             ServerUnavailable();
          }
       } catch (Exception e) {
          Log.e("UDP_Download", "S: ERROR", e);
       }
    }

    public void stopDownload() {
       stopDL stDL = new stopDL(serverAddr, clientAddr);
       new Thread(stDL).start();
    }
}

class stopDL extends Thread {
    private static final int SERVERPORT = 5001;
    private static final int CLIENTPORTSTOP = 5002;
    private InetAddress serverAddr;
    private String clientAddr;
    private DatagramSocket socketT;
    private DatagramSocket socketR;

    public stopDL(InetAddress serverAddr1, String clientAddr1) {
       serverAddr = serverAddr1;
       clientAddr = clientAddr1;
    }
```

```
      public void run() {
         if (d.getValue() == true) {
            d.setFalse();
            Boolean ReceivedACK = false;
            String clientAddrT = clientAddr;

            int trytostop = 0;
            try {
               clientAddrT = getLocalIpAddress();
               socketR = new DatagramSocket(CLIENTPORTSTOP,
InetAddress.getByName(clientAddrT));
               socketR.setSoTimeout(5000);
            } catch (SocketException e1) {
               e1.printStackTrace();
            } catch (UnknownHostException e1) {
               e1.printStackTrace();
            }
            byte[] buf5 = new byte[3];
            DatagramPacket ACKpacket = new DatagramPacket(buf5, buf5.length);

            while ((ReceivedACK == false) && (trytostop < 5)) {
               try {
                  if (socketT != null) {
                     socketT.close();
                  }
                  socketT = new DatagramSocket();
                  byte[] buf4 = ("FIN ").getBytes();
                  DatagramPacket FinPacket = new DatagramPacket(buf4,
                        buf4.length, serverAddr, SERVERPORT);
                  for (int i = 0; i < 3; i++) {
                     socketT.send(FinPacket);
                  }

                  try {
                     socketR.receive(ACKpacket);
                     String ACKpacketString = new String(
                           ACKpacket.getData());
                     if (ACKpacketString.equals("ACK")) {
                        ReceivedACK = true;
                        socketR.close();
                        ReceivedACKfromServer();
                        Log.e("UDP_Download", "Received ACK packet");
                     } else {
                        Log.e("UDP_Download",
                              "Received a packet but not ACK");
                     }
                  } catch (SocketException e) {
                     e.printStackTrace();
                     Log.e("UDP_Download", e.toString(), e);
                  }
               } catch (IOException e) {
                  e.printStackTrace();
                  Log.e("UDP_Download", "Other Error" + e.toString(), e);
               }
               trytostop++;
            }
            if (socketR != null)
               socketR.close();
            if (socketT != null) {
               socketT.close();
            }
```

```
         if (ReceivedACK == false) {
            NoACKReplyfromServer();
         }
      }
   }
}

class Downloading {
   private Boolean downloading = false;

   public synchronized Boolean getValue() {
      return downloading;
   }

   public synchronized void setTrue() {
      downloading = true;
   }

   public synchronized void setFalse() {
      downloading = false;
   }
}

// PING !!! PING !!! PING !!! PING !!! PING !!! PING !!! PING !!! PING !!!
public Count c;
public Pinging p;
public StartNextTimeSegment snts;
public CurrentTimeSegment cts;
public SendTime st;
public Results r;
public long StartTimePing;
public long latency;
public long minlatency;
public long maxlatency;

class PingSend extends Thread {
   private static final int SERVERPORT = 5000;
   private int PingAttempts;
   private String ServerIP;
   private InetAddress serverAddr;

   public PingSend(int PingAttempts1, String ServerIPChoice) {
      PingAttempts = PingAttempts1;
      ServerIP = ServerIPChoice;
   }

   public void run() {
      try {
         serverAddr = InetAddress.getByName(ServerIP);
      } catch (UnknownHostException e1) {
         e1.printStackTrace();
      }

      while (p.getValue() == true) {
         if (snts.getValue() == true) {
            snts.setFalse();
            c.ZeroCount();
            StartTime = System.currentTimeMillis();
            try {
               DatagramSocket socket = new DatagramSocket();
               for (int j = 0; j < 4; j++) {
                  long StartTimeSegment = System.currentTimeMillis();
```

```
                    cts.setValue(j);
                    for (int i = 0; i < (PingAttempts / 4); i++) {
                       String PacketString = "" + j + "-" + i;
                       byte[] buf = PacketString.getBytes();
                       DatagramPacket SendPacket = new DatagramPacket(
                             buf, buf.length, serverAddr, SERVERPORT);
                       socket.send(SendPacket);
                       st.AddElement(j, i, System.currentTimeMillis());
                       sleep(50);
                    }
                    sleep(StartTimeSegment + 2500 - System.currentTimeMillis());
                 }
                 socket.close();
              } catch (SocketException e) {
                 e.printStackTrace();
              } catch (IOException e) {
                 e.printStackTrace();
              } catch (InterruptedException e) {
                 e.printStackTrace();
              }

           }
        }
     }
}

class PingReceive extends Thread {
   private static final int CLIENTPORT = 5000;
   private String clientAddr;

   public void run() {
      clientAddr = getLocalIpAddress();

      try {
         DatagramSocket socket = new DatagramSocket(CLIENTPORT,
               InetAddress.getByName(clientAddr));
         socket.setSoTimeout(3000);
         while (p.getValue() == true) {
            byte[] buf = new byte[3];
            DatagramPacket ReceivePacket = new DatagramPacket(buf, buf.length);
            try {
               socket.receive(ReceivePacket);
               long ReceiveTime = System.currentTimeMillis();
               new Thread(new PingCalc(ReceivePacket, ReceiveTime)).start();
            } catch (IOException e) {
               e.printStackTrace();
            }
         }
         socket.close();
      } catch (SocketException e) {
         e.printStackTrace();
      } catch (UnknownHostException e) {
         e.printStackTrace();
      }
   }
}

class PingCalc extends Thread {
   DatagramPacket ReceivePacket;
   long ReceiveTime;

   public PingCalc(DatagramPacket ReceivePacket1, long ReceiveTime1) {
```

```
      ReceivePacket = ReceivePacket1;
      ReceiveTime = ReceiveTime1;
   }

   public void run() {
      byte[] PacketStringData = ReceivePacket.getData();
      String PacketString = new String(PacketStringData);
      String[] PacketStringParts = PacketString.split("-");
      String PacketTimeSegment = PacketStringParts[0];
      String PacketID = PacketStringParts[1];

      int j = Integer.parseInt(PacketTimeSegment);
      int i = Integer.parseInt(PacketID);
      if (j == cts.getValue()) {
         long PacketStartTime = st.ReturnElement(j, i);
         long TotalPingTime = ReceiveTime - PacketStartTime;

         if (TotalPingTime < 2500) {
            r.AddElement(c.GetCount(), TotalPingTime);
            c.AddCount();
         }
      }
   }
}

class PingSendIndoors extends Thread {
   private static final int SERVERPORT = 5000;
   private int PingAttempts;
   private String ServerIP;
   private InetAddress serverAddr;
   private Location fixedLocation = new Location("fixedloc");

   public PingSendIndoors(int PingAttempts1, String ServerIPChoice) {
      PingAttempts = PingAttempts1;
      ServerIP = ServerIPChoice;
   }

   public void run() {
      try {
         serverAddr = InetAddress.getByName(ServerIP);
      } catch (UnknownHostException e1) {
         e1.printStackTrace();
      }

      double lat = IndoorsGeopointSingleton.getInstance(0, 0).latitude();
      double lon = IndoorsGeopointSingleton.getInstance(0, 0).longitude();
      fixedLocation.setLatitude(lat);
      fixedLocation.setLongitude(lon);
      fixedLocation.removeAccuracy();
      fixedLocation.removeAltitude();
      fixedLocation.removeBearing();

      while (p.getValue() == true) {
         if (snts.getValue() == true) {
            snts.setFalse();
            c.ZeroCount();
            StartTime = System.currentTimeMillis();
            try {
               DatagramSocket socket = new DatagramSocket();
               for (int j = 0; j < 4; j++) {
                  long StartTimeSegment = System.currentTimeMillis();
                  cts.setValue(j);
```

```
                    for (int i = 0; i < (PingAttempts / 4); i++) {
                       String PacketString = "" + j + "-" + i;
                       byte[] buf = PacketString.getBytes();
                       DatagramPacket SendPacket = new DatagramPacket(
                             buf, buf.length, serverAddr, SERVERPORT);
                       socket.send(SendPacket);
                       st.AddElement(j, i, System.currentTimeMillis());
                       sleep(50);
                    }
                    sleep(StartTimeSegment + 2500 - System.currentTimeMillis());
                 }
                 socket.close();
              } catch (SocketException e) {
                 e.printStackTrace();
              } catch (IOException e) {
                 e.printStackTrace();
              } catch (InterruptedException e) {
                 e.printStackTrace();
              }

              if (p.getValue() == true) {
                 if (mStartNextSegment) {
                    mStartNextSegment = false;
                    startNewSegment();
                 }
                 storeLocation(fixedLocation);
              }
           }
        }
     }
}

class Pinging {
   private Boolean pinging = false;

   public synchronized Boolean getValue() {
      return pinging;
   }

   public synchronized void setTrue() {
      pinging = true;
   }

   public synchronized void setFalse() {
      pinging = false;
   }
}

class Count {
   private int count = 0;

   public synchronized void AddCount() {
      count++;
   }

   public synchronized int GetCount() {
      return count;
   }

   public synchronized void ZeroCount() {
      count = 0;
   }
```

```
}

class CurrentTimeSegment {
   private int CurrentTimeSegment;

   public synchronized void setValue(int j) {
      CurrentTimeSegment = j;
   }

   public synchronized int getValue() {
      return CurrentTimeSegment;
   }
}

class StartNextTimeSegment {
   private Boolean StartNextTimeSegment = true;

   public synchronized void setTrue() {
      StartNextTimeSegment = true;
   }

   public synchronized void setFalse() {
      StartNextTimeSegment = false;
   }

   public synchronized Boolean getValue() {
      return StartNextTimeSegment;
   }
}

class SendTime {
   long[][] SendTime = new long[4][8];

   public synchronized void AddElement(int j, int i, long packetSendTime) {
      SendTime[j][i] = packetSendTime;
   }

   public synchronized long ReturnElement(int j, int i) {
      return SendTime[j][i];
   }
}

class Results {
   long[] Results = new long[32];

   public synchronized void AddElement(int position, long LatencyTime) {
      Results[position] = LatencyTime;
   }

   public synchronized long ReturnElement(int position) {
      return Results[position];
   }
}

public String getLocalIpAddress() {
   try {
      for (Enumeration<NetworkInterface> en = NetworkInterface
            .getNetworkInterfaces(); en.hasMoreElements();) {
         NetworkInterface intf = en.nextElement();
         for (Enumeration<InetAddress> enumIpAddr = intf
               .getInetAddresses(); enumIpAddr.hasMoreElements();) {
            InetAddress inetAddress = enumIpAddr.nextElement();
```

```
            if (!inetAddress.isLoopbackAddress()) {
               return inetAddress.getHostAddress().toString();
            }
         }
      }
   } catch (SocketException ex) {
      Log.e("GPSLoggerService", "getLocalIpAddress: " + ex.toString(), ex);
   }
   return null;
}

public long getMaxValue() {
   long maxValue = r.ReturnElement(0);
   for (int i = 1; i < c.GetCount(); i++) {
      if (r.ReturnElement(i) > maxValue) {
         maxValue = r.ReturnElement(i);
      }
   }
   return maxValue;
}

public long getMinValue() {
   long minValue = r.ReturnElement(0);
   for (int i = 1; i < c.GetCount(); i++) {
      if (r.ReturnElement(i) < minValue) {
         minValue = r.ReturnElement(i);
      }
   }
   return minValue;
}

public long getAvgValue() {
   long sum = 0;
   long avgValue;
   for (int i = 0; i < c.GetCount(); i++) {
      sum = sum + r.ReturnElement(i);
   }
   avgValue = sum / c.GetCount();
   return avgValue;
}

// !!! INDOORSTIMER !!! INDOORSTIMER !!! INDOORSTIMER !!! INDOORSTIMER !!!
// INDOORSTIMER
class DLIndoorsTimer extends Thread {
   private double latitude;
   private double longitude;
   private Location fixedLocation = new Location("fixedloc");

   public DLIndoorsTimer(double latitude1, double longitude1) {
      latitude = latitude1;
      longitude = longitude1;
      fixedLocation.setLatitude(latitude);
      fixedLocation.setLongitude(longitude);
      fixedLocation.removeAccuracy();
      fixedLocation.removeAltitude();
      fixedLocation.removeBearing();
   }

   public void run() {
      while (d.getValue() == true) {
         try {
            Thread.sleep(10000);
```

```
               if (d.getValue() == true) {
                  if (mStartNextSegment) {
                     mStartNextSegment = false;
                     startNewSegment();
                  }
                  storeLocation(fixedLocation);
               }
            } catch (InterruptedException e) {
               e.printStackTrace();
            }
         }
      }
   }

   // !!! CELL ID, LAC, SIGNAL STRENGTH
   class CellID_LAC {
      int[] cellidlac = new int[2];

      public synchronized void setValues(int cellID1, int lac1) {
         cellidlac[0] = cellID1;
         cellidlac[1] = lac1;
      }

      public synchronized int[] getValues() {
         return cellidlac;
      }
   }

   // !!! Server Unavailable Method
   public void ServerUnavailable() {
      Looper.prepare();
      Toast.makeText(getApplicationContext(), "Server Unavailable!", 7000).show();
      Looper.loop();
   }

   // !!! Received "ACK" from Server
   public void ReceivedACKfromServer() {
      Looper.prepare();
      Toast.makeText(getApplicationContext(),
            "Received ACK response from Server... Connection terminated!",
            5000).show();
      Looper.loop();
   }

   // !!! No "ACK" reply from Server was received
   public void NoACKReplyfromServer() {
      Looper.prepare();
      Toast.makeText(
            getApplicationContext(),
            "No ACK reply was received... Connection may be still active! Please, contact
Server's Administrator!",
            10000).show();
      Looper.loop();
   }
}
```

Constants.java

```
package nl.sogeti.android.gpstracker.util;

import nl.sogeti.android.gpstracker.db.GPStracking;
```

```
import android.net.Uri;


/**
 * Various application wide constants
 *
 * @author Alexandros Ntakas
 */
public class Constants
{
   public static final String DISABLEBLANKING = "disableblanking";
   public static final String SATELLITE = "SATELLITE";
   public static final String TRAFFIC = "TRAFFIC";
   public static final String SPEED = "showspeed";
   public static final String ALTITUDE = "showaltitude";
   public static final String COMPASS = "COMPASS";
   public static final String LOCATION = "LOCATION";
   public static final String MAPPROVIDER = "mapprovider";
   public static final String TRACKCOLORING = "trackcoloring";
   public static final int UNKNOWN = -1;
   public static final int LOGGING = 1;
   public static final int PAUSED = 2;
   public static final int STOPPED = 3;
   public static final String SPEEDSANITYCHECK = "speedsanitycheck";
   public static final String PRECISION = "precision";

   public static final String NETWORKINGMEASUREMENT = "networkingmeasurement";
   public static final String PACKETSIZE = "packetsize";
   public static final String PINGATTEMPTS = "pingattempts";
   public static final String SERVERIP = "serverip";
   public static final String INDOORSCHECK = "indoorscheck";

   public static final int DOWNLOADCHOICE = 0;
   public static final int PINGCHOICE = 1;

   public static final int BYTES200 = 0;
   public static final int BYTES500 = 1;
   public static final int BYTES1000 = 2;
   public static final int BYTES1400 = 3;

   public static final int PINGATTEMPTS4 = 0;
   public static final int PINGATTEMPTS8 = 1;
   public static final int PINGATTEMPTS16 = 2;
   public static final int PINGATTEMPTS32 = 3;

   public static final String NMC = "networkingmeasurementchoice";
   public static final String gotLoc = "gotlocation";
   public static final String filePath = "filepath";
   public static final String contentType = "contenttype";

   public static final int NTUAServer = 0;
   public static final String NTUAServerIP = "147.102.7.158";

   public static final String LOGATSTARTUP = "logatstartup";
   public static final String STARTUPATBOOT = "startupatboot";
   public static final String SERVICENAME =
"nl.sogeti.android.gpstracker.intent.action.GPSLoggerService";
   public static final String UNITS = "units";
   public static final int UNITS_DEFAULT = 0;
   public static final int UNITS_IMPERIAL = 1;
   public static final int UNITS_METRIC = 2;
   public static final int UNITS_NAUTIC = 3;
```

```
   public static final String EXTERNAL_DIR = "/KampRate/";
   public static final String TMPICTUREFILE_PATH = EXTERNAL_DIR+"media_tmp";
   public static final Uri NAME_URI = Uri.parse( "content://" +
GPStracking.AUTHORITY+".string" );
   public static final int GOOGLE = 0;
   public static final int OSM = 1;
}
```

LoggerMap.java

```
package nl.sogeti.android.gpstracker.viewer;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Calendar;
import java.util.List;

import nl.sogeti.android.gpstracker.R;
import nl.sogeti.android.gpstracker.actions.ControlTracking;
import nl.sogeti.android.gpstracker.actions.NameTrack;
import nl.sogeti.android.gpstracker.actions.Statistics;
import nl.sogeti.android.gpstracker.db.GPStracking.Media;
import nl.sogeti.android.gpstracker.db.GPStracking.Segments;
import nl.sogeti.android.gpstracker.db.GPStracking.Tracks;
import nl.sogeti.android.gpstracker.db.GPStracking.Waypoints;
import nl.sogeti.android.gpstracker.logger.GPSLoggerServiceManager;
import nl.sogeti.android.gpstracker.logger.SettingsDialog;
import nl.sogeti.android.gpstracker.util.Constants;
import nl.sogeti.android.gpstracker.util.UnitsI18n;
import android.app.AlertDialog;
import android.app.AlertDialog.Builder;
import android.app.Dialog;
import android.content.ActivityNotFoundException;
import android.content.ContentResolver;
import android.content.ContentUris;
import android.content.Context;
import android.content.DialogInterface;
import android.content.DialogInterface.OnClickListener;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.content.SharedPreferences.OnSharedPreferenceChangeListener;
import android.content.res.Resources;
import android.database.ContentObserver;
import android.database.Cursor;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Point;
import android.location.Location;
import android.location.LocationManager;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.os.Handler;
import android.os.PowerManager;
import android.os.PowerManager.WakeLock;
import android.preference.PreferenceManager;
import android.telephony.PhoneStateListener;
import android.telephony.SignalStrength;
import android.telephony.TelephonyManager;
```

```
import android.util.Log;
import android.view.ContextMenu;
import android.view.KeyEvent;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.SubMenu;
import android.view.View;
import android.widget.BaseAdapter;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.CompoundButton.OnCheckedChangeListener;
import android.widget.EditText;
import android.widget.Gallery;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.maps.GeoPoint;
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapView;
import com.google.android.maps.MyLocationOverlay;

/**
 * Main activity showing a track and allowing logging control
 *
 * @author Alexandros Ntakas
 */
public class LoggerMap extends MapActivity {
   private static final int ZOOM_LEVEL = 16;
   // MENU'S
   private static final int MENU_SETTINGS = 1;
   private static final int MENU_TRACKING = 2;
   private static final int MENU_TRACKLIST = 3;
   private static final int MENU_STATS = 4;
   private static final int MENU_ABOUT = 5;
   private static final int MENU_LAYERS = 6;
   private static final int MENU_NOTE = 7;
   private static final int MENU_NAME = 8;
   private static final int MENU_PICTURE = 9;
   private static final int MENU_TEXT = 10;
   private static final int MENU_VOICE = 11;
   private static final int MENU_VIDEO = 12;
   private static final int MENU_SHARE = 13;
   private static final int DIALOG_NOTRACK = 24;
   private static final int DIALOG_INSTALL_ABOUT = 29;
   private static final int DIALOG_LAYERS = 31;
   private static final int DIALOG_TEXT = 32;
   private static final int DIALOG_NAME = 33;
   private static final int DIALOG_URIS = 34;
   private static final String TAG = "OGT.LoggerMap";
   private CheckBox mSatellite;
   private CheckBox mTraffic;
   private CheckBox mSpeed;
   private CheckBox mAltitude;
   private CheckBox mCompass;
   private CheckBox mLocation;
   private TextView[] mSpeedtexts = null;
   private TextView mLastGPSSpeedView = null;
   private TextView mLastGPSAltitudeView = null;
   private TextView mLastCellIDView = null;
   private TextView mLastLacView = null;
   private TextView mCurrentRSSITextView = null;
```

```
private TelephonyManager tm;
private EditText mNoteNameView;
private EditText mNoteTextView;

private double mAverageSpeed = 33.33d / 2d;
private long mTrackId = -1;
private long mLastSegment = -1;
@SuppressWarnings("unused")
private long mLastWaypoint = -1;
private UnitsI18n mUnits;
private WakeLock mWakeLock = null;
private SharedPreferences mSharedPreferences;
private GPSLoggerServiceManager mLoggerServiceManager;
private SegmentOverlay mLastSegmentOverlay;
private BaseAdapter mMediaAdapter;
private Gallery mGallery;

private MapView mMapView = null;
private MyLocationOverlay mMylocation;

private Resources resources;
private String bitrate_unit;
private String latency_unit;
private String speedText;
private String cellidText;
private String lacText;
private String[] BitrateBar = { "0", "200", "500", "2000", "7000", "Inf" };
private String[] LatencyBar = { "2500", "800", "300", "120", "60", "0" };

private final ContentObserver mTrackSegmentsObserver = new ContentObserver(
      new Handler()) {
   @Override
   public void onChange(boolean selfUpdate) {
      if (!selfUpdate) {
         LoggerMap.this.updateDataOverlays();
      } else {
         Log.w(TAG, "mTrackSegmentsObserver skipping change on "
               + mLastSegment);
      }
   }
};
private final ContentObserver mSegmentWaypointsObserver = new ContentObserver(
      new Handler()) {
   @Override
   public void onChange(boolean selfUpdate) {
      if (!selfUpdate) {
         LoggerMap.this.updateDisplayedSpeedViews();
         if (mLastSegmentOverlay != null) {
            moveActiveViewWindow();
            mLastSegmentOverlay.calculateTrack();
            mMapView.postInvalidate();
         }
      } else {
         Log.w(TAG, "mSegmentWaypointsObserver skipping change on "
               + mLastSegment);
      }
   }
};
private final ContentObserver mTrackMediasObserver = new ContentObserver(
      new Handler()) {
   @Override
   public void onChange(boolean selfUpdate) {
```

```
         if (!selfUpdate) {
            if (mLastSegmentOverlay != null) {
               mLastSegmentOverlay.calculateMedia();
               mMapView.postInvalidate();
            }
         } else {
            Log.w(TAG, "mTrackMediasObserver skipping change on "
                  + mLastSegment);
         }
      }
   };
   private final DialogInterface.OnClickListener mNoTrackDialogListener = new
DialogInterface.OnClickListener() {
      public void onClick(DialogInterface dialog, int which) {
         Intent tracklistIntent = new Intent(LoggerMap.this, TrackList.class);
         tracklistIntent.putExtra(Tracks._ID, LoggerMap.this.mTrackId);
         startActivityForResult(tracklistIntent, MENU_TRACKLIST);
      }
   };

   private final DialogInterface.OnClickListener mOiAboutDialogListener = new
DialogInterface.OnClickListener() {
      public void onClick(DialogInterface dialog, int which) {
         Uri oiDownload = Uri.parse("market://details?id=org.openintents.about");
         Intent oiAboutIntent = new Intent(Intent.ACTION_VIEW, oiDownload);
         try {
            startActivity(oiAboutIntent);
         } catch (ActivityNotFoundException e) {
            oiDownload = Uri
                  .parse("http://openintents.googlecode.com/files/AboutApp-1.0.0.apk");
            oiAboutIntent = new Intent(Intent.ACTION_VIEW, oiDownload);
            startActivity(oiAboutIntent);
         }
      }
   };

   private final OnCheckedChangeListener mCheckedChangeListener = new
OnCheckedChangeListener() {
      public void onCheckedChanged(CompoundButton buttonView,
            boolean isChecked) {
         int which = buttonView.getId();
         switch (which) {
         case R.id.layer_satellite:
            setSatelliteOverlay(isChecked);
            break;
         case R.id.layer_speed:
            setSpeedOverlay(isChecked);
            break;
         case R.id.layer_compass:
            setCompassOverlay(isChecked);
            break;
         case R.id.layer_location:
            setLocationOverlay(isChecked);
            break;
         default:
            break;
         }
      }
   };
   private final OnSharedPreferenceChangeListener mSharedPreferenceChangeListener = new
OnSharedPreferenceChangeListener() {
      public void onSharedPreferenceChanged(
```

```
            SharedPreferences sharedPreferences, String key) {
         if (key.equals(Constants.TRACKCOLORING)) {
            updateSpeedColoring();
         } else if (key.equals(Constants.DISABLEBLANKING)) {
            updateBlankingBehavior();
         } else if (key.equals(Constants.SPEED)) {
            updateSpeedDisplayVisibility();
         } else if (key.equals(Constants.COMPASS)) {
            updateCompassDisplayVisibility();
         } else if (key.equals(Constants.SATELLITE)) {
            LoggerMap.this.mMapView.setSatellite(sharedPreferences
                  .getBoolean(key, false));
         } else if (key.equals(Constants.LOCATION)) {
            updateLocationDisplayVisibility();
         }
      }
   };
   private final UnitsI18n.UnitsChangeListener mUnitsChangeListener = new
UnitsI18n.UnitsChangeListener() {
      public void onUnitsChange() {
         updateDisplayedSpeedViews();
         updateSpeedColoring();
      }
   };
   private final OnClickListener mNoteTextDialogListener = new
DialogInterface.OnClickListener() {

      public void onClick(DialogInterface dialog, int which) {
         String noteText = mNoteTextView.getText().toString();
         Calendar c = Calendar.getInstance();
         String newName = String.format(
               "Textnote_%tY-%tm-%td_%tH%tM%tS.txt", c, c, c, c, c, c);
         String sdcard = Environment.getExternalStorageDirectory().getAbsolutePath();
         File file = new File(sdcard + Constants.EXTERNAL_DIR + newName);
         FileWriter filewriter = null;
         try {
            file.getParentFile().mkdirs();
            file.createNewFile();
            filewriter = new FileWriter(file);
            filewriter.append(noteText);
            filewriter.flush();
         } catch (IOException e) {
            Log.e(TAG, "Note storing failed", e);
            CharSequence text = e.getLocalizedMessage();
            Toast toast = Toast.makeText(
                  LoggerMap.this.getApplicationContext(), text,
                  Toast.LENGTH_LONG);
            toast.show();
         } finally {
            if (filewriter != null) {
               try {
                  filewriter.close();
               } catch (IOException e) { /* */
               }
            }
         }

         LoggerMap.this.mLoggerServiceManager.storeMediaUri(Uri.fromFile(file));
      }

   };
```

```
   private final OnClickListener mNoteNameDialogListener = new
DialogInterface.OnClickListener() {

      public void onClick(DialogInterface dialog, int which) {
         String name = mNoteNameView.getText().toString();
         Uri media = Uri.withAppendedPath(Constants.NAME_URI, Uri.encode(name));
         LoggerMap.this.mLoggerServiceManager.storeMediaUri(media);
      }

   };
   private final OnClickListener mNoteSelectDialogListener = new
DialogInterface.OnClickListener() {

      public void onClick(DialogInterface dialog, int which) {
         Uri selected = (Uri) mGallery.getSelectedItem();
         SegmentOverlay.handleMedia(LoggerMap.this, selected);
      }

   };

   private PhoneStateListener onSignalChange = new PhoneStateListener() {
      public void onSignalStrengthsChanged(SignalStrength signalStrength) {
         if (signalStrength.isGsm()) {
            int GsmSignalStrength = signalStrength.getGsmSignalStrength();
            if (GsmSignalStrength != 99) {
               mCurrentRSSITextView.setText("RSSI: "  + (2 * GsmSignalStrength - 113) + "
dBm");
            } else {
               mCurrentRSSITextView.setText("RSSI: " + "unknown");
            }
         } else {
            mCurrentRSSITextView.setText("CDMA Signal: "
                  + signalStrength.getCdmaDbm() + " dBm");
         }
      }
   };

   /**
    * Called when the activity is first created.
    */
   @Override
   protected void onCreate(Bundle load) {
      super.onCreate(load);
      this.startService(new Intent(Constants.SERVICENAME));

      Object previousInstanceData = getLastNonConfigurationInstance();
      if (previousInstanceData != null
            && previousInstanceData instanceof GPSLoggerServiceManager) {
         mLoggerServiceManager = (GPSLoggerServiceManager) previousInstanceData;
      } else {
         mLoggerServiceManager = new GPSLoggerServiceManager((Context) this);
      }
      mLoggerServiceManager.startup();

      mUnits = new UnitsI18n(this, mUnitsChangeListener);

      mSharedPreferences = PreferenceManager.getDefaultSharedPreferences(this);
      mSharedPreferences
            .registerOnSharedPreferenceChangeListener(mSharedPreferenceChangeListener);

      setContentView(R.layout.map);
```

```
   mMapView = (MapView) findViewById(R.id.myMapView);

   mMylocation = new FixedMyLocationOverlay(this, mMapView);
   mMapView.setBuiltInZoomControls(true);
   mMapView.setClickable(true);
   mMapView.setStreetView(false);
   mMapView.setSatellite(mSharedPreferences.getBoolean(Constants.SATELLITE, false));
   mMapView.setTraffic(mSharedPreferences.getBoolean(Constants.TRAFFIC, false));

   TextView[] speeds = { (TextView) findViewById(R.id.speedview05),
         (TextView) findViewById(R.id.speedview04),
         (TextView) findViewById(R.id.speedview03),
         (TextView) findViewById(R.id.speedview02),
         (TextView) findViewById(R.id.speedview01),
         (TextView) findViewById(R.id.speedview00) };
   mSpeedtexts = speeds;
   mLastGPSSpeedView = (TextView) findViewById(R.id.currentSpeed);
   mLastGPSAltitudeView = (TextView) findViewById(R.id.currentAltitude);
   mLastCellIDView = (TextView) findViewById(R.id.currentCellID);
   mLastLacView = (TextView) findViewById(R.id.currentLac);

   onRestoreInstanceState(load);

   resources = this.getResources();
   bitrate_unit = resources.getString(R.string.bitrate_unitname);
   latency_unit = resources.getString(R.string.latency_unitname);

   tm = (TelephonyManager) getSystemService(Context.TELEPHONY_SERVICE);
   tm.listen(onSignalChange, PhoneStateListener.LISTEN_SIGNAL_STRENGTHS);
   mCurrentRSSITextView = (TextView) findViewById(R.id.currentRSSI);
}

protected void onPause() {
   super.onPause();
   if (this.mWakeLock != null && this.mWakeLock.isHeld()) {
      this.mWakeLock.release();
      Log.w(TAG, "onPause(): Released lock to keep screen on!");
   }
   if (mTrackId > 0) {
      ContentResolver resolver = this.getApplicationContext().getContentResolver();
      resolver.unregisterContentObserver(this.mTrackSegmentsObserver);
      resolver.unregisterContentObserver(this.mSegmentWaypointsObserver);
      resolver.unregisterContentObserver(this.mTrackMediasObserver);
   }
   mMylocation.disableMyLocation();
   mMylocation.disableCompass();
}

protected void onResume() {
   super.onResume();
   updateTitleBar();
   updateBlankingBehavior();
   updateSpeedColoring();
   updateSpeedDisplayVisibility();
   updateAltitudeDisplayVisibility();
   updateCompassDisplayVisibility();
   updateLocationDisplayVisibility();

   if (mTrackId >= 0) {
      ContentResolver resolver = this.getApplicationContext().getContentResolver();
      Uri trackUri = Uri.withAppendedPath(Tracks.CONTENT_URI, mTrackId
            + "/segments");
```

```
      Uri lastSegmentUri = Uri.withAppendedPath(Tracks.CONTENT_URI,
            mTrackId + "/segments/" + mLastSegment + "/waypoints");
      Uri mediaUri = ContentUris.withAppendedId(Media.CONTENT_URI,  mTrackId);

      resolver.unregisterContentObserver(this.mTrackSegmentsObserver);
      resolver.unregisterContentObserver(this.mSegmentWaypointsObserver);
      resolver.unregisterContentObserver(this.mTrackMediasObserver);
      resolver.registerContentObserver(trackUri, false, this.mTrackSegmentsObserver);
      resolver.registerContentObserver(lastSegmentUri, true,
            this.mSegmentWaypointsObserver);
      resolver.registerContentObserver(mediaUri, true, this.mTrackMediasObserver);
   }
   updateDataOverlays();
}

@Override
protected void onDestroy() {
   super.onDestroy();
   this.mLoggerServiceManager.shutdown();
   if (mWakeLock != null && mWakeLock.isHeld()) {
      mWakeLock.release();
      Log.w(TAG, "onDestroy(): Released lock to keep screen on!");
   }
   mSharedPreferences

.unregisterOnSharedPreferenceChangeListener(this.mSharedPreferenceChangeListener);

   tm.listen(onSignalChange, PhoneStateListener.LISTEN_NONE);

   if (mLoggerServiceManager.getLoggingState() == Constants.STOPPED) {
      stopService(new Intent(Constants.SERVICENAME));
   }
}

@Override
public void onNewIntent(Intent newIntent) {
   Uri data = newIntent.getData();
   if (data != null) {
      moveToTrack(Long.parseLong(data.getLastPathSegment()), true);
   }
}

@Override
protected void onRestoreInstanceState(Bundle load) {
   if (load != null) {
      super.onRestoreInstanceState(load);
   }
   Uri data = this.getIntent().getData();

   if (load != null && load.containsKey("track")) // 1st track from a
                                       // previous instance of
                                       // this activity
   {
      long loadTrackId = load.getLong("track");
      moveToTrack(loadTrackId, false);
   } else if (data != null) // 2nd track ordered to make
   {
      long loadTrackId = Long.parseLong(data.getLastPathSegment());
      moveToTrack(loadTrackId, true);
   } else {
      moveToLastTrack(); // 3rd just try the last track
   }
```

```
   if (load != null && load.containsKey("zoom")) {
      mMapView.getController().setZoom(load.getInt("zoom"));
   } else {
      mMapView.getController().setZoom(LoggerMap.ZOOM_LEVEL);
   }

   if (load != null && load.containsKey("e6lat")
         && load.containsKey("e6long")) {
      GeoPoint storedPoint = new GeoPoint(load.getInt("e6lat"), load.getInt("e6long"));
      this.mMapView.getController().animateTo(storedPoint);
   } else {
      GeoPoint lastPoint = getLastTrackPoint();
      this.mMapView.getController().animateTo(lastPoint);
   }
}

@Override
protected void onSaveInstanceState(Bundle save) {
   super.onSaveInstanceState(save);
   save.putLong("track", this.mTrackId);
   save.putInt("zoom", this.mMapView.getZoomLevel());
   GeoPoint point = this.mMapView.getMapCenter();
   save.putInt("e6lat", point.getLatitudeE6());
   save.putInt("e6long", point.getLongitudeE6());
}

@Override
public Object onRetainNonConfigurationInstance() {
   Object nonConfigurationInstance = this.mLoggerServiceManager;
   return nonConfigurationInstance;
}

@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
   boolean propagate = true;
   switch (keyCode) {
   case KeyEvent.KEYCODE_T:
      propagate = this.mMapView.getController().zoomIn();
      break;
   case KeyEvent.KEYCODE_G:
      propagate = this.mMapView.getController().zoomOut();
      break;
   case KeyEvent.KEYCODE_S:
      setSatelliteOverlay(!this.mMapView.isSatellite());
      propagate = false;
      break;
   case KeyEvent.KEYCODE_A:
      setTrafficOverlay(!this.mMapView.isTraffic());
      propagate = false;
      break;
   case KeyEvent.KEYCODE_F:
      moveToTrack(this.mTrackId - 1, true);
      propagate = false;
      break;
   case KeyEvent.KEYCODE_H:
      moveToTrack(this.mTrackId + 1, true);
      propagate = false;
      break;
   default:
      propagate = super.onKeyDown(keyCode, event);
      break;
```

```
    }
    return propagate;
}

private void setTrafficOverlay(boolean b) {
    Editor editor = mSharedPreferences.edit();
    editor.putBoolean(Constants.TRAFFIC, b);
    editor.commit();
}

private void setSatelliteOverlay(boolean b) {
    Editor editor = mSharedPreferences.edit();
    editor.putBoolean(Constants.SATELLITE, b);
    editor.commit();
}

private void setSpeedOverlay(boolean b) {
    Editor editor = mSharedPreferences.edit();
    editor.putBoolean(Constants.SPEED, b);
    editor.commit();
}

private void setAltitudeOverlay(boolean b) {
    Editor editor = mSharedPreferences.edit();
    editor.putBoolean(Constants.ALTITUDE, b);
    editor.commit();
}

private void setCompassOverlay(boolean b) {
    Editor editor = mSharedPreferences.edit();
    editor.putBoolean(Constants.COMPASS, b);
    editor.commit();
}

private void setLocationOverlay(boolean b) {
    Editor editor = mSharedPreferences.edit();
    editor.putBoolean(Constants.LOCATION, b);
    editor.commit();
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    boolean result = super.onCreateOptionsMenu(menu);

    menu.add(ContextMenu.NONE, MENU_TRACKING, ContextMenu.NONE,
          R.string.menu_tracking).setIcon(R.drawable.ic_menu_movie)
          .setAlphabeticShortcut('T');
    menu.add(ContextMenu.NONE, MENU_LAYERS, ContextMenu.NONE,
          R.string.menu_showLayers).setIcon(R.drawable.ic_menu_mapmode)
          .setAlphabeticShortcut('L');
    SubMenu notemenu = menu.addSubMenu(ContextMenu.NONE, MENU_NOTE,
          ContextMenu.NONE, R.string.menu_insertnote).setIcon(
          R.drawable.ic_menu_myplaces);

    menu.add(ContextMenu.NONE, MENU_STATS, ContextMenu.NONE,
          R.string.menu_statistics).setIcon(R.drawable.ic_menu_picture)
          .setAlphabeticShortcut('S');
    menu.add(ContextMenu.NONE, MENU_SHARE, ContextMenu.NONE,
          R.string.menu_shareTrack).setIcon(R.drawable.ic_menu_share)
          .setAlphabeticShortcut('I');
    // More
```

```
        menu.add(ContextMenu.NONE, MENU_TRACKLIST, ContextMenu.NONE,
              R.string.menu_tracklist).setIcon(R.drawable.ic_menu_show_list)
              .setAlphabeticShortcut('P');
        menu.add(ContextMenu.NONE, MENU_SETTINGS, ContextMenu.NONE,
              R.string.menu_settings).setIcon(R.drawable.ic_menu_preferences)
              .setAlphabeticShortcut('C');
        menu.add(ContextMenu.NONE, MENU_ABOUT, ContextMenu.NONE,
              R.string.menu_about).setIcon(R.drawable.ic_menu_info_details)
              .setAlphabeticShortcut('A');

        notemenu.add(ContextMenu.NONE, MENU_NAME, ContextMenu.NONE,
              R.string.menu_notename);
        notemenu.add(ContextMenu.NONE, MENU_TEXT, ContextMenu.NONE,
              R.string.menu_notetext);
        notemenu.add(ContextMenu.NONE, MENU_VOICE, ContextMenu.NONE,
              R.string.menu_notespeech);
        notemenu.add(ContextMenu.NONE, MENU_PICTURE, ContextMenu.NONE,
              R.string.menu_notepicture);
        notemenu.add(ContextMenu.NONE, MENU_VIDEO, ContextMenu.NONE,
              R.string.menu_notevideo);

        return result;
     }

     @Override
     public boolean onPrepareOptionsMenu(Menu menu) {
        MenuItem notemenu = menu.findItem(MENU_NOTE);
        notemenu.setEnabled(mLoggerServiceManager.isMediaPrepared());
        MenuItem sharemenu = menu.findItem(MENU_SHARE);
        sharemenu.setEnabled(mTrackId >= 0);
        return super.onPrepareOptionsMenu(menu);
     }

     @Override
     public boolean onOptionsItemSelected(MenuItem item) {
        boolean handled = false;

        Uri trackUri;
        switch (item.getItemId()) {
        case MENU_TRACKING:
           Intent controlIntent = new Intent(this, ControlTracking.class);
           Boolean IndoorsCheck = mSharedPreferences.getBoolean(Constants.INDOORSCHECK,
false);
           controlIntent.putExtra(Constants.INDOORSCHECK, IndoorsCheck);
           startActivityForResult(controlIntent, MENU_TRACKING);
           handled = true;
           break;
        case MENU_LAYERS:
           showDialog(DIALOG_LAYERS);
           handled = true;
           break;
        case MENU_SETTINGS:
           Intent settingsIntent = new Intent(this, SettingsDialog.class);
           startActivity(settingsIntent);
           handled = true;
           break;
        case MENU_TRACKLIST:
           Intent tracklistIntent = new Intent(this, TrackList.class);
           tracklistIntent.putExtra(Tracks._ID, this.mTrackId);
           startActivityForResult(tracklistIntent, MENU_TRACKLIST);
           break;
        case MENU_STATS:
```

```
            if (this.mTrackId >= 0) {
               Intent actionIntent = new Intent(this, Statistics.class);
               trackUri = ContentUris.withAppendedId(Tracks.CONTENT_URI, mTrackId);
               actionIntent.setData(trackUri);
               startActivity(actionIntent);
               handled = true;
               break;
            } else {
               showDialog(DIALOG_NOTRACK);
            }
            handled = true;
            break;
         case MENU_SHARE:
            Intent actionIntent = new Intent(Intent.ACTION_RUN);
            trackUri = ContentUris.withAppendedId(Tracks.CONTENT_URI, mTrackId);
            actionIntent.setDataAndType(trackUri, Tracks.CONTENT_ITEM_TYPE);
            actionIntent.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);
            startActivity(Intent.createChooser(actionIntent,
getString(R.string.share_track)));
            handled = true;
            break;
         case MENU_PICTURE:
            addPicture();
            handled = true;
            break;
         case MENU_VIDEO:
            addVideo();
            handled = true;
            break;
         case MENU_VOICE:
            addVoice();
            handled = true;
            break;
         case MENU_TEXT:
            showDialog(DIALOG_TEXT);
            handled = true;
            break;
         case MENU_NAME:
            showDialog(DIALOG_NAME);
            handled = true;
            break;
         case MENU_ABOUT:
            AlertDialog.Builder alertbox = new AlertDialog.Builder(this);
            alertbox.setTitle("About KampRate");
            alertbox.setMessage("Developed by" + "\n" + "Alexandros Ntakas"
                  + "\n" + "Electrical and Computer Engineer, NTUA" + "\n"
                  + "alex_ntakas@hotmail.gr" + "\n"
                  + "© 2011 - All rights reserved");
            alertbox.setNeutralButton("Ok",
                  new DialogInterface.OnClickListener() {
                     public void onClick(DialogInterface arg0, int arg1) {
                     }
                  });
            alertbox.show();
            break;
         default:
            handled = super.onOptionsItemSelected(item);
            break;
         }
         return handled;
      }
```

```
@Override
protected Dialog onCreateDialog(int id) {
   Dialog dialog = null;
   LayoutInflater factory = null;
   View view = null;
   Builder builder = null;
   switch (id) {
   case DIALOG_LAYERS:
      builder = new AlertDialog.Builder(this);
      factory = LayoutInflater.from(this);
      view = factory.inflate(R.layout.layerdialog, null);
      mSatellite = (CheckBox) view.findViewById(R.id.layer_satellite);
      mSatellite.setOnCheckedChangeListener(mCheckedChangeListener);
      mSpeed = (CheckBox) view.findViewById(R.id.layer_speed);
      mSpeed.setOnCheckedChangeListener(mCheckedChangeListener);
      mCompass = (CheckBox) view.findViewById(R.id.layer_compass);
      mCompass.setOnCheckedChangeListener(mCheckedChangeListener);
      mLocation = (CheckBox) view.findViewById(R.id.layer_location);
      mLocation.setOnCheckedChangeListener(mCheckedChangeListener);
      builder.setTitle(R.string.dialog_layer_title)
            .setIcon(android.R.drawable.ic_dialog_map)
            .setPositiveButton(R.string.btn_okay, null).setView(view);
      dialog = builder.create();
      return dialog;
   case DIALOG_NOTRACK:
      builder = new AlertDialog.Builder(this);
      builder.setTitle(R.string.dialog_notrack_title)
            .setMessage(R.string.dialog_notrack_message)
            .setIcon(android.R.drawable.ic_dialog_alert)
            .setPositiveButton(R.string.btn_selecttrack,
                  mNoTrackDialogListener)
            .setNegativeButton(R.string.btn_cancel, null);
      dialog = builder.create();
      return dialog;
   case DIALOG_INSTALL_ABOUT:
      builder = new AlertDialog.Builder(this);
      builder.setTitle(R.string.dialog_nooiabout)
            .setMessage(R.string.dialog_nooiabout_message)
            .setIcon(android.R.drawable.ic_dialog_alert)
            .setPositiveButton(R.string.btn_install,
                  mOiAboutDialogListener)
            .setNegativeButton(R.string.btn_cancel, null);
      dialog = builder.create();
      return dialog;
   case DIALOG_TEXT:
      builder = new AlertDialog.Builder(this);
      factory = LayoutInflater.from(this);
      view = factory.inflate(R.layout.notetextdialog, null);
      mNoteTextView = (EditText) view.findViewById(R.id.notetext);
      builder.setTitle(R.string.dialog_notetext_title)
            .setMessage(R.string.dialog_notetext_message)
            .setIcon(android.R.drawable.ic_dialog_map)
            .setPositiveButton(R.string.btn_okay,
                  mNoteTextDialogListener)
            .setNegativeButton(R.string.btn_cancel, null).setView(view);
      dialog = builder.create();
      return dialog;
   case DIALOG_NAME:
      builder = new AlertDialog.Builder(this);
      factory = LayoutInflater.from(this);
      view = factory.inflate(R.layout.notenamedialog, null);
      mNoteNameView = (EditText) view.findViewById(R.id.notename);
```

```
      builder.setTitle(R.string.dialog_notename_title)
            .setMessage(R.string.dialog_notename_message)
            .setIcon(android.R.drawable.ic_dialog_map)
            .setPositiveButton(R.string.btn_okay,
                  mNoteNameDialogListener)
            .setNegativeButton(R.string.btn_cancel, null).setView(view);
      dialog = builder.create();
      return dialog;
   case DIALOG_URIS:
      builder = new AlertDialog.Builder(this);
      factory = LayoutInflater.from(this);
      view = factory.inflate(R.layout.mediachooser, null);
      mGallery = (Gallery) view.findViewById(R.id.gallery);
      builder.setTitle(R.string.dialog_select_media_title)
            .setMessage(R.string.dialog_select_media_message)
            .setIcon(android.R.drawable.ic_dialog_alert)
            .setNegativeButton(R.string.btn_cancel, null)
            .setPositiveButton(R.string.btn_okay,
                  mNoteSelectDialogListener).setView(view);
      dialog = builder.create();
      return dialog;
   default:
      return super.onCreateDialog(id);
   }
}

@Override
protected void onPrepareDialog(int id, Dialog dialog) {
   switch (id) {
   case DIALOG_LAYERS:
      mSatellite.setChecked(mSharedPreferences.getBoolean(Constants.SATELLITE, false));
      mSpeed.setChecked(mSharedPreferences.getBoolean(Constants.SPEED, false));
      mCompass.setChecked(mSharedPreferences.getBoolean(Constants.COMPASS, false));
      mLocation.setChecked(mSharedPreferences.getBoolean(Constants.LOCATION, false));
      break;
   case DIALOG_URIS:
      mGallery.setAdapter(mMediaAdapter);
   default:
      break;
   }
   super.onPrepareDialog(id, dialog);
}

@Override
protected void onActivityResult(int requestCode, int resultCode,
      Intent intent) {
   super.onActivityResult(requestCode, resultCode, intent);
   if (resultCode != RESULT_CANCELED) {
      String sdcard = Environment.getExternalStorageDirectory().getAbsolutePath();
      File file;
      Uri uri;
      File newFile;
      String newName;
      Uri fileUri;
      android.net.Uri.Builder builder;
      Uri trackUri;
      long trackId;
      switch (requestCode) {
      case MENU_TRACKLIST:
         trackUri = intent.getData();
         trackId = Long.parseLong(trackUri.getLastPathSegment());
         moveToTrack(trackId, true);
```

```
            break;
         case MENU_ABOUT:
            break;
         case MENU_PICTURE:
            file = new File(sdcard + Constants.TMPICTUREFILE_PATH);
            Calendar c = Calendar.getInstance();
            newName = String.format("Picture_%tY-%tm-%td_%tH%tM%tS.jpg", c, c, c, c, c,
c);
            newFile = new File(sdcard + Constants.EXTERNAL_DIR + newName);
            file.getParentFile().mkdirs();
            boolean isRenamed = file.renameTo(newFile);
            if (isRenamed) {
               Bitmap bm = BitmapFactory.decodeFile(newFile.getAbsolutePath());
               if (bm != null) {
                  String height = Integer.toString(bm.getHeight());
                  String width = Integer.toString(bm.getWidth());
                  bm.recycle();
                  bm = null;
                  builder = new Uri.Builder();
                  fileUri = builder.scheme("file").appendEncodedPath("/")
                        .appendEncodedPath(newFile.getAbsolutePath())
                        .appendQueryParameter("width", width)
                        .appendQueryParameter("height", height).build();
                  this.mLoggerServiceManager.storeMediaUri(fileUri);
                  mLastSegmentOverlay.calculateMedia();
                  mMapView.postInvalidate();
               } else {
                  Log.e(TAG,
                        "Failed to read image from: "
                              + newFile.getAbsolutePath());
               }
            } else {
               Log.e(TAG,
                     "Failed to rename image: " + file.getAbsolutePath());
            }
            break;
         case MENU_VIDEO:
            file = new File(sdcard + Constants.TMPICTUREFILE_PATH);
            c = Calendar.getInstance();
            newName = String.format("Video_%tY%tm%td_%tH%tM%tS.3gp", c, c, c, c, c, c);
            newFile = new File(sdcard + Constants.EXTERNAL_DIR + newName);
            file.getParentFile().mkdirs();
            file.renameTo(newFile);
            builder = new Uri.Builder();
            fileUri =
builder.scheme("file").appendPath(newFile.getAbsolutePath()).build();
            this.mLoggerServiceManager.storeMediaUri(fileUri);
            mLastSegmentOverlay.calculateMedia();
            mMapView.postInvalidate();
            break;
         case MENU_VOICE:
            uri = Uri.parse(intent.getDataString());
            this.mLoggerServiceManager.storeMediaUri(uri);
            mLastSegmentOverlay.calculateMedia();
            mMapView.postInvalidate();
            break;
         case MENU_TRACKING:
            trackUri = intent.getData();
            trackId = Long.parseLong(trackUri.getLastPathSegment());
            moveToTrack(trackId, true);
            Intent namingIntent = new Intent(this, NameTrack.class);
            namingIntent.setData(ContentUris.withAppendedId(Tracks.CONTENT_URI, trackId));
```

```
            int NMChelper = new Integer(mSharedPreferences.getString(
                  Constants.NETWORKINGMEASUREMENT, "0")).intValue();
            namingIntent.putExtra(Constants.NETWORKINGMEASUREMENT, NMChelper);
            startActivity(namingIntent);
            break;
         default:
            Log.e(TAG, "Returned form unknow activity: " + requestCode);
            break;
         }
      } else {
         Log.w(TAG, "Received unexpected resultcode " + resultCode);
      }
   }

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

   private void updateTitleBar() {
      ContentResolver resolver = this.getApplicationContext().getContentResolver();
      Cursor trackCursor = null;
      try {
         trackCursor = resolver.query(ContentUris.withAppendedId(
               Tracks.CONTENT_URI, this.mTrackId),
               new String[] { Tracks.NAME }, null, null, null);
         if (trackCursor != null && trackCursor.moveToLast()) {
            String trackName = trackCursor.getString(0);
            this.setTitle(this.getString(R.string.app_name) + ": "  + trackName);
         }
      } finally {
         if (trackCursor != null) {
            trackCursor.close();
         }
      }
   }

   private void updateBlankingBehavior() {
      boolean disableblanking = mSharedPreferences.getBoolean(
            Constants.DISABLEBLANKING, false);
      if (disableblanking) {
         if (mWakeLock == null) {
            PowerManager pm = (PowerManager) this.getSystemService(Context.POWER_SERVICE);
            mWakeLock = pm.newWakeLock(PowerManager.SCREEN_DIM_WAKE_LOCK, TAG);
         }
         if (mLoggerServiceManager.getLoggingState() == Constants.LOGGING
               && !mWakeLock.isHeld()) {
            mWakeLock.acquire();
            Log.w(TAG, "Acquired lock to keep screen on!");
         }
      }
   }

   private void updateSpeedColoring() {
      int trackColoringMethod = new Integer(mSharedPreferences.getString(
            Constants.TRACKCOLORING, "2")).intValue();
      ContentResolver resolver = this.getApplicationContext().getContentResolver();
      Cursor waypointsCursor = null;
      try {
         waypointsCursor = resolver.query(
               Uri.withAppendedPath(Tracks.CONTENT_URI, this.mTrackId
                     + "/waypoints"), new String[] { "avg("
```

```
                + Waypoints.SPEED + ")" }, null, null, null);
      if (waypointsCursor != null && waypointsCursor.moveToLast()) {
         mAverageSpeed = waypointsCursor.getDouble(0);
      }
      if (mAverageSpeed < 2) {
         mAverageSpeed = 5.55d / 2;
      }
   } finally {
      if (waypointsCursor != null) {
         waypointsCursor.close();
      }
   }
   View speedbar = findViewById(R.id.speedbar);
   if (trackColoringMethod == SegmentOverlay.DRAW_MEASURED
         || trackColoringMethod == SegmentOverlay.DRAW_CALCULATED) {
      drawSpeedTexts(mAverageSpeed);
      speedbar.setVisibility(View.VISIBLE);
      for (int i = 0; i < mSpeedtexts.length; i++) {
         mSpeedtexts[i].setVisibility(View.VISIBLE);
      }
   } else {
      speedbar.setVisibility(View.INVISIBLE);
      for (int i = 0; i < mSpeedtexts.length; i++) {
         mSpeedtexts[i].setVisibility(View.INVISIBLE);
      }
   }
   List<?> overlays = mMapView.getOverlays();
   for (Object overlay : overlays) {
      if (overlay instanceof SegmentOverlay) {
         ((SegmentOverlay) overlay).setTrackColoringMethod(
               trackColoringMethod, mAverageSpeed);
      }
   }
}

private void updateSpeedDisplayVisibility() {
   boolean showspeed = mSharedPreferences.getBoolean(Constants.SPEED, false);
   if (showspeed) {
      mLastGPSSpeedView.setVisibility(View.VISIBLE);
      mLastGPSSpeedView.setText("");
      mLastCellIDView.setVisibility(View.VISIBLE);
      mLastCellIDView.setText("");
      mLastLacView.setVisibility(View.VISIBLE);
      mLastLacView.setText("");
   } else {
      mLastGPSSpeedView.setVisibility(View.INVISIBLE);
      mLastCellIDView.setVisibility(View.INVISIBLE);
      mLastLacView.setVisibility(View.INVISIBLE);
   }
}

private void updateAltitudeDisplayVisibility() {
   boolean showaltitude = mSharedPreferences.getBoolean(Constants.ALTITUDE, false);
   if (showaltitude) {
      mLastGPSAltitudeView.setVisibility(View.VISIBLE);
      mLastGPSAltitudeView.setText("");
   } else {
      mLastGPSAltitudeView.setVisibility(View.INVISIBLE);
   }
}

private void updateCompassDisplayVisibility() {
```

```
   boolean compass = mSharedPreferences.getBoolean(Constants.COMPASS, false);
   if (compass) {
      mMylocation.enableCompass();
   } else {
      mMylocation.disableCompass();
   }
}

private void updateLocationDisplayVisibility() {
   boolean location = mSharedPreferences.getBoolean(Constants.LOCATION, false);
   if (location) {
      mMylocation.enableMyLocation();
   } else {
      mMylocation.disableMyLocation();
   }
}

/**
 * Retrieves the numbers of the measured speed and altitude from the most
 * recent waypoint and updates UI components with this latest bit of
 * information.
 *
 */
private void updateDisplayedSpeedViews() {
   ContentResolver resolver = this.getApplicationContext().getContentResolver();
   Cursor waypointsCursor = null;
   try {
      Uri lastSegmentUri = Uri.withAppendedPath(Tracks.CONTENT_URI,
            this.mTrackId + "/segments/" + mLastSegment + "/waypoints");
      waypointsCursor = resolver.query(lastSegmentUri, new String[] {
            Waypoints.SPEED, Waypoints.ALTITUDE, Waypoints.CELLID,
            Waypoints.LAC }, null, null, null);
      if (waypointsCursor != null && waypointsCursor.moveToLast()) {
         // Speed number
         double speed = waypointsCursor.getDouble(0);

         int cellid = waypointsCursor.getInt(2);
         cellidText = "Cell ID: " + cellid;
         mLastCellIDView.setText(cellidText);

         int lac = waypointsCursor.getInt(3);
         lacText = "LAC: " + lac;
         mLastLacView.setText(lacText);

         Uri trackURI = ContentUris.withAppendedId(Tracks.CONTENT_URI,this.mTrackId);
         Cursor trackCursor = null;
         trackCursor = resolver.query(trackURI,
               new String[] { Tracks.NMC }, null, null, null);

         if (trackCursor != null) {
            trackCursor.moveToFirst();
            if (trackCursor.getString(0).equals("DOWNLOAD")) {
               speedText = String.format("%.0f %s", speed, bitrate_unit);
            } else if (trackCursor.getString(0).equals("PING")) {
               speedText = String.format("%.0f %s", speed, latency_unit);
            }
         } else {
            speedText = String.format("%.0f %s", speed, bitrate_unit);
         }

         mLastGPSSpeedView.setText(speedText);
```

```
            // Speed color bar
            if (speed > 2 * mAverageSpeed) {
               updateSpeedColoring();
               mMapView.postInvalidate();
            }

            // Altitude number
            double altitude = waypointsCursor.getDouble(1);
            altitude = mUnits.conversionFromMeterToHeight(altitude);
            String altitudeText = String.format("%.0f %s", altitude,
mUnits.getHeightUnit());
            mLastGPSAltitudeView.setText(altitudeText);
         }
      } finally {
         if (waypointsCursor != null) {
            waypointsCursor.close();
         }
      }
   }

   /**
    * For the current track identifier the route of that track is drawn by
    * adding a OverLay for each segments in the track
    *
    * @param trackId
    * @see SegmentOverlay
    */
   private void createDataOverlays() {
      mLastSegmentOverlay = null;
      mMapView.getOverlays().clear();
      mMapView.getOverlays().add(mMylocation);

      ContentResolver resolver = this.getApplicationContext().getContentResolver();
      Cursor segments = null;
      int trackColoringMethod = new Integer(mSharedPreferences.getString(
            Constants.TRACKCOLORING, "2")).intValue();

      String NMC = "DOWNLOAD";
      Uri trackURI = ContentUris.withAppendedId(Tracks.CONTENT_URI, this.mTrackId);
      Cursor trackCursor = null;
      trackCursor = resolver.query(trackURI, new String[] { Tracks.NMC },
            null, null, null);
      if (trackCursor != null) {
         trackCursor.moveToFirst();
         if (trackCursor.getString(0).equals("DOWNLOAD")) {
            NMC = "DOWNLOAD";
         } else if (trackCursor.getString(0).equals("PING")) {
            NMC = "PING";
         }
      }

      try {
         Uri segmentsUri = Uri.withAppendedPath(Tracks.CONTENT_URI,
               this.mTrackId + "/segments");
         segments = resolver.query(segmentsUri,
               new String[] { Segments._ID }, null, null, null);
         if (segments != null && segments.moveToFirst()) {
            do {
               long segmentsId = segments.getLong(0);
               Uri segmentUri = ContentUris.withAppendedId(segmentsUri, segmentsId);
               SegmentOverlay segmentOverlay = new SegmentOverlay(this,
                     segmentUri, trackColoringMethod, mAverageSpeed,
```

```
                this.mMapView, NMC);
            mMapView.getOverlays().add(segmentOverlay);
            mLastSegmentOverlay = segmentOverlay;
            if (segments.isFirst()) {
               segmentOverlay .addPlacement(SegmentOverlay.FIRST_SEGMENT);
            }
            if (segments.isLast()) {
               segmentOverlay .addPlacement(SegmentOverlay.LAST_SEGMENT);
               getLastTrackPoint();
            }
            mLastSegment = segmentsId;
         } while (segments.moveToNext());
      }
   } finally {
      if (segments != null) {
         segments.close();
      }
   }

   moveActiveViewWindow();

   Uri lastSegmentUri = Uri.withAppendedPath(Tracks.CONTENT_URI, mTrackId
         + "/segments/" + mLastSegment + "/waypoints");
   resolver.unregisterContentObserver(this.mSegmentWaypointsObserver);
   resolver.registerContentObserver(lastSegmentUri, false,
         this.mSegmentWaypointsObserver);
}

private void updateDataOverlays() {
   ContentResolver resolver = this.getApplicationContext().getContentResolver();
   Uri segmentsUri = Uri.withAppendedPath(Tracks.CONTENT_URI,
         this.mTrackId + "/segments");
   Cursor segmentsCursor = null;
   List<?> overlays = this.mMapView.getOverlays();
   int segmentOverlaysCount = 0;

   for (Object overlay : overlays) {
      if (overlay instanceof SegmentOverlay) {
         segmentOverlaysCount++;
      }
   }
   try {
      segmentsCursor = resolver.query(segmentsUri,
            new String[] { Segments._ID }, null, null, null);
      if (segmentsCursor != null
            && segmentsCursor.getCount() == segmentOverlaysCount) {
      } else {
         createDataOverlays();
      }
   } finally {
      if (segmentsCursor != null) {
         segmentsCursor.close();
      }
   }
   moveActiveViewWindow();
}

private void moveActiveViewWindow() {
   GeoPoint lastPoint = getLastTrackPoint();
   if (lastPoint != null
         && mLoggerServiceManager.getLoggingState() == Constants.LOGGING) {
      Point out = new Point();
```

```
      this.mMapView.getProjection().toPixels(lastPoint, out);
      int height = this.mMapView.getHeight();
      int width = this.mMapView.getWidth();
      if (out.x < 0 || out.y < 0 || out.y > height || out.x > width) {

         this.mMapView.clearAnimation();
         this.mMapView.getController().setCenter(lastPoint);
      } else if (out.x < width / 4 || out.y < height / 4
            || out.x > (width / 4) * 3 || out.y > (height / 4) * 3) {
         this.mMapView.clearAnimation();
         this.mMapView.getController().animateTo(lastPoint);
      }
   }
}

/**
 * @param avgSpeed
 *            avgSpeed in m/s
 */
private void drawSpeedTexts(double avgSpeed) {
   avgSpeed = mUnits.conversionFromMetersPerSecond(avgSpeed);

   ContentResolver resolver = this.getApplicationContext().getContentResolver();
   Uri trackURI = ContentUris.withAppendedId(Tracks.CONTENT_URI, this.mTrackId);
   Cursor trackCursor = null;
   trackCursor = resolver.query(trackURI, new String[] { Tracks.NMC },
         null, null, null);

   if (trackCursor != null) {
      trackCursor.moveToFirst();
      if (trackCursor.getString(0).equals("DOWNLOAD")) {
         for (int i = 0; i < mSpeedtexts.length; i++) {
            mSpeedtexts[i].setVisibility(View.VISIBLE);
            String speedText = String.format("%s %s", BitrateBar[i], bitrate_unit);
            mSpeedtexts[i].setText(speedText);
         }
      } else if (trackCursor.getString(0).equals("PING")) {
         for (int i = 0; i < mSpeedtexts.length; i++) {
            mSpeedtexts[i].setVisibility(View.VISIBLE);
            String speedText = String.format("%s %s", LatencyBar[i], latency_unit);
            mSpeedtexts[i].setText(speedText);
         }
      }
   } else {
      for (int i = 0; i < mSpeedtexts.length; i++) {
         mSpeedtexts[i].setVisibility(View.VISIBLE);
         String speedText = String.format("%s %s", BitrateBar[i], bitrate_unit);
         mSpeedtexts[i].setText(speedText);
      }
   }
}

/**
 * Alter this to set a new track as current.
 *
 * @param trackId
 * @param center
 *            center on the end of the track
 */
private void moveToTrack(long trackId, boolean center) {
   Cursor track = null;
   try {
```

```
      ContentResolver resolver = this.getApplicationContext().getContentResolver();
      Uri trackUri = ContentUris.withAppendedId(Tracks.CONTENT_URI, trackId);
      Uri mediaUri = ContentUris.withAppendedId(Media.CONTENT_URI, trackId);
      track = resolver.query(trackUri, new String[] { Tracks.NAME },
            null, null, null);
      if (track != null && track.moveToFirst()) {
         this.mTrackId = trackId;
         mLastSegment = -1;
         mLastWaypoint = -1;
         resolver.unregisterContentObserver(this.mTrackSegmentsObserver);
         resolver.unregisterContentObserver(this.mTrackMediasObserver);
         Uri tracksegmentsUri = Uri.withAppendedPath(Tracks.CONTENT_URI,
               trackId + "/segments");

         resolver.registerContentObserver(tracksegmentsUri, false,
               this.mTrackSegmentsObserver);
         resolver.registerContentObserver(mediaUri, false,
               this.mTrackMediasObserver);

         this.mMapView.getOverlays().clear();

         updateTitleBar();
         updateDataOverlays();
         updateSpeedColoring();

         if (center) {
            GeoPoint lastPoint = getLastTrackPoint();
            this.mMapView.getController().animateTo(lastPoint);
         }
      }
   } finally {
      if (track != null) {
         track.close();
      }
   }
}

/**
 * Get the last known position from the GPS provider and return that
 * information wrapped in a GeoPoint to which the Map can navigate.
 *
 * @see GeoPoint
 * @return
 */
private GeoPoint getLastKnowGeopointLocation() {
   int microLatitude = 0;
   int microLongitude = 0;
   LocationManager locationManager = (LocationManager) this
         .getApplication().getSystemService(Context.LOCATION_SERVICE);
   Location locationFine = locationManager
         .getLastKnownLocation(LocationManager.GPS_PROVIDER);
   if (locationFine != null) {
      microLatitude = (int) (locationFine.getLatitude() * 1E6d);
      microLongitude = (int) (locationFine.getLongitude() * 1E6d);
   }
   if (locationFine == null || microLatitude == 0 || microLongitude == 0) {
      Location locationCoarse = locationManager
            .getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
      if (locationCoarse != null) {
         microLatitude = (int) (locationCoarse.getLatitude() * 1E6d);
         microLongitude = (int) (locationCoarse.getLongitude() * 1E6d);
      }
```

```
      if (locationCoarse == null || microLatitude == 0 || microLongitude == 0) {
         microLatitude = 51985105;
         microLongitude = 5106132;
      }
   }
   GeoPoint geoPoint = new GeoPoint(microLatitude, microLongitude);
   return geoPoint;
}

/**
 * Retrieve the last point of the current track
 *
 * @param context
 */
private GeoPoint getLastTrackPoint() {
   Cursor waypoint = null;
   GeoPoint lastPoint = null;
   try {
      ContentResolver resolver = this.getContentResolver();
      waypoint = resolver.query(
            Uri.withAppendedPath(Tracks.CONTENT_URI, mTrackId
                  + "/waypoints"), new String[] {
                  Waypoints.LATITUDE,
                  Waypoints.LONGITUDE,
                  "max(" + Waypoints.TABLE + "." + Waypoints._ID
                        + ")" }, null, null, null);
      if (waypoint != null && waypoint.moveToLast()) {
         int microLatitude = (int) (waypoint.getDouble(0) * 1E6d);
         int microLongitude = (int) (waypoint.getDouble(1) * 1E6d);
         lastPoint = new GeoPoint(microLatitude, microLongitude);
      }
      if (lastPoint == null || lastPoint.getLatitudeE6() == 0
            || lastPoint.getLongitudeE6() == 0) {
         lastPoint = getLastKnowGeopointLocation();
      } else {
         mLastWaypoint = waypoint.getLong(2);
      }
   } finally {
      if (waypoint != null) {
         waypoint.close();
      }
   }
   return lastPoint;
}

private void moveToLastTrack() {
   int trackId = -1;
   Cursor track = null;
   try {
      ContentResolver resolver = this.getApplicationContext().getContentResolver();
      track = resolver
            .query(Tracks.CONTENT_URI, new String[] {
                  "max(" + Tracks._ID + ")", Tracks.NAME, }, null,
                  null, null);
      if (track != null && track.moveToLast()) {
         trackId = track.getInt(0);
         moveToTrack(trackId, true);
      }
   } finally {
      if (track != null) {
         track.close();
      }
```

```
      }
   }

   /***
    * Collecting additional data
    */
   private void addPicture() {
      Intent i = new Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
      File file = new File(Environment.getExternalStorageDirectory()
            .getAbsolutePath() + Constants.TMPICTUREFILE_PATH);
      i.putExtra(android.provider.MediaStore.EXTRA_OUTPUT, Uri.fromFile(file));
      i.putExtra(android.provider.MediaStore.EXTRA_VIDEO_QUALITY, 1);
      startActivityForResult(i, MENU_PICTURE);
   }

   /***
    * Collecting additional data
    */
   private void addVideo() {
      Intent i = new Intent(android.provider.MediaStore.ACTION_VIDEO_CAPTURE);
      File file = new File(Environment.getExternalStorageDirectory()
            .getAbsolutePath() + Constants.TMPICTUREFILE_PATH);
      i.putExtra(android.provider.MediaStore.EXTRA_OUTPUT, Uri.fromFile(file));
      i.putExtra(android.provider.MediaStore.EXTRA_VIDEO_QUALITY, 1);
      startActivityForResult(i, MENU_VIDEO);
   }

   private void addVoice() {
      Intent intent = new Intent(
            android.provider.MediaStore.Audio.Media.RECORD_SOUND_ACTION);
      startActivityForResult(intent, MENU_VOICE);
   }

   /**
    * Enables a SegmentOverlay to call back to the MapActivity to show a dialog
    * with choices of media
    *
    * @param mediaAdapter
    */
   public void showDialog(BaseAdapter mediaAdapter) {
      mMediaAdapter = mediaAdapter;
      showDialog(LoggerMap.DIALOG_URIS);
   }
}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    package="nl.sogeti.android.gpstracker"
   android:versionName="0.9.20" android:versionCode="28">

    <uses-sdk android:minSdkVersion="4" android:targetSdkVersion="4"/>
    <supports-screens
         android:largeScreens="true"
         android:normalScreens="true"
         android:smallScreens="true" />

   <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
   <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />


<application
   android:icon="@drawable/icon"
   android:label="@string/app_name"
   android:description="@string/app_name"
   android:allowClearUserData="true">
      <meta-data
         android:name="org.openintents.metadata.COMMENTS"
         android:value="@string/about_comments" />
      <meta-data
         android:name="org.openintents.metadata.COPYRIGHT"
         android:value="@string/about_copyright" />
      <meta-data
         android:name="org.openintents.metadata.AUTHORS"
         android:resource="@array/about_authors" />
      <meta-data
         android:name="org.openintents.metadata.DOCUMENTERS"
         android:resource="@array/about_documenters" />
      <meta-data
         android:name="org.openintents.metadata.TRANSLATORS"
         android:resource="@array/about_translators" />
      <meta-data
         android:name="org.openintents.metadata.ARTISTS"
         android:resource="@array/about_artists" />
      <meta-data
         android:name="org.openintents.metadata.WEBSITE_LABEL"
         android:value="@string/about_website_label" />
      <meta-data
         android:name="org.openintents.metadata.WEBSITE_URL"
         android:value="@string/about_website_url" />
      <meta-data
         android:name="org.openintents.metadata.LICENSE"
         android:resource="@raw/licence_short" />
      <meta-data
         android:name="org.openintents.metadata.EMAIL"
         android:value="@string/about_email" />
      <meta-data
           android:name="android.app.default_searchable"
           android:value=".viewer.TrackList" />
      <meta-data
           android:name="CLOUDMADE_KEY"
           android:value="534dfce474894e218f363a5473248ff6" />

   <uses-library android:name="com.google.android.maps" />

   <activity
         android:label="@string/app_name"
         android:name=".viewer.LoggerMap"
         android:launchMode="singleTask"
         android:theme="@style/Theme.NoBackground"
         android:screenOrientation="portrait">
      <intent-filter>
         <action android:name="android.intent.action.MAIN" />
         <category android:name="android.intent.category.LAUNCHER" />
```

```
         </intent-filter>
            <intent-filter android:label="Show track on map">
                <action android:name="android.intent.action.VIEW" />
                <category android:name="android.intent.category.DEFAULT" />
                <data android:scheme="content" />
                <data android:host="nl.sogeti.android.gpstracker" />
                <data
android:mimeType="vnd.android.cursor.item/vnd.nl.sogeti.android.track" />
            </intent-filter>
      </activity>

         <activity
            android:label="@string/menu_settings"
            android:name=".logger.SettingsDialog"/>

         <activity
            android:label="@string/indoors_select_location"
            android:name=".actions.IndoorsSelectLocation"
            android:screenOrientation="portrait"/>

         <activity
            android:label="@string/upload_to_server"
            android:name=".actions.UploadToServer"
            android:screenOrientation="portrait"/>

         <activity
            android:label="@string/dialog_routename_title"
            android:theme="@android:style/Theme.Translucent.NoTitleBar"
            android:name=".actions.NameTrack"/>

         <activity
            android:label="@string/dialog_tracking_title"
            android:theme="@android:style/Theme.Translucent.NoTitleBar"
            android:name=".actions.ControlTracking"/>

         <activity
            android:label="@string/track_list"
            android:name=".viewer.TrackList"
            android:launchMode="singleTop" >
            <intent-filter>
               <action android:name="android.intent.action.SEARCH" />
               <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
            <meta-data
               android:name="android.app.searchable"
               android:resource="@xml/searchable" />
         </activity>

         <activity
             android:name=".viewer.TracksLiveFolder"
             android:label="@string/track_list"
             android:icon="@drawable/icon" >
             <intent-filter>
                 <action android:name="android.intent.action.CREATE_LIVE_FOLDER" />
                 <category android:name="android.intent.category.DEFAULT" />
             </intent-filter>
         </activity>

      <service android:enabled="true" android:name=".logger.GPSLoggerService">
         <intent-filter android:label="GPS logger">
            <action
android:name="nl.sogeti.android.gpstracker.intent.action.GPSLoggerService"></action>
```

```
         </intent-filter>
      </service>

      <provider
            android:enabled="true"
            android:authorities="nl.sogeti.android.gpstracker"
            android:permission="android.permission.ACCESS_FINE_LOCATION"
         android:name=".db.GPStrackingProvider" >
            <path-permission android:path="/search_suggest_query"
                android:readPermission="android.permission.GLOBAL_SEARCH" />
            <path-permission android:path="/live_folders/tracks"
                android:readPermission="com.android.launcher.permission.WRITE_SETTINGS"
/>
      </provider>

        <activity
            android:label="@string/menu_statistics"
            android:name=".actions.Statistics"
            android:screenOrientation="portrait">
        </activity>

         <activity
            android:label="@string/share_track"
            android:name=".actions.ShareTrack" >
            <intent-filter>
               <action android:name="android.intent.action.RUN" />
               <category android:name="android.intent.category.DEFAULT" />
               <data
android:mimeType="vnd.android.cursor.item/vnd.nl.sogeti.android.track" />
            </intent-filter>
         </activity>

         <!--  This is used to auto start the application at boot -->
         <receiver android:name=".util.BootReceiver">
            <intent-filter>
            <action android:name="android.intent.action.BOOT_COMPLETED" />
           </intent-filter>
        </receiver>
    </application>
</manifest>
```

Η εφαρμογή KampRate περιλαμβάνει ακόμη αρκετές κλάσεις που δεν μπορούν να παρουσιαστούν όλες σε αυτό το έντυπο. Επιπλέον, υπάρχουν πολλά xml αρχεία που αφορούν το γραφικό περιβάλλον τις εφαρμογής, καθώς και δεδομένα άλλων λειτουργιών.

Server

Τα DLServer.java και PingServer.java που «τρέχουν» στην πλευρά του Server έχουν παρουσιαστεί με ανάλυση και σχολιασμό στο κεφάλαιο 2, όπου ο αναγνώστης μπορεί να ανατρέξει για να μελετήσει τον κώδικα. Στην ενότητα αυτή αρχικά θα παρουσιάσουμε τον κώδικα του εργαλείου NetPerfTool.

NetPerfTool-DL.html

```
<html>
<head>
<meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
<meta http-equiv="content-type" content="text/html; charset=UTF-8"/>
<title>Performance Evaluation Tool (Bitrate Measurements)</title>
<link
href="http://code.google.com/apis/maps/documentation/javascript/examples/default.css"
rel="stylesheet" type="text/css" />
<script type="text/javascript"
src="http://maps.google.com/maps/api/js?sensor=false"></script>
<script type="text/javascript" src="loadxmldoc.js"></script>

<script type="text/javascript">
var codeID = 1;
var Range = 1;

var geocoder; // = new google.maps.Geocoder();
xmlDoc=loadXMLDoc("C:/NetPerfTool/Download.xml");
var x=xmlDoc.getElementsByTagName('StartLat');

var map = null;
var bounds = null;
var Athens_Center= new google.maps.LatLng(37.979116, 23.717165);


function initialize() {
  geocoder = new google.maps.Geocoder();

  var myOptions = {
    zoom: 10,
    center: new google.maps.LatLng(38.0782, 23.800),
    mapTypeControl: true,
    mapTypeControlOptions: {style: google.maps.MapTypeControlStyle.DROPDOWN_MENU},
    navigationControl: true,
    mapTypeId: google.maps.MapTypeId.ROADMAP
  }

  map = new google.maps.Map(document.getElementById("map_canvas"), myOptions);
  bounds = new google.maps.LatLngBounds();
  var homeControlDiv = document.createElement('DIV');
  var homeControl = new HomeControl(homeControlDiv, map);
  homeControlDiv.index = 1;
  map.controls[google.maps.ControlPosition.TOP_RIGHT].push(homeControlDiv);

  setCode();
```

```
  for (i=0;i<x.length;i++){

    var Bitrate = xmlDoc.getElementsByTagName("Bitrate")[i].childNodes[0].nodeValue;

   if (Bitrate>=7000)
   {
   codeID = 2;
   }

   if (Bitrate>=2000 && Bitrate<7000)
   {
   codeID = 3;
   }

   if (Bitrate>=500 && Bitrate<2000)
   {
   codeID = 4;
   }

   if (Bitrate>=200 && Bitrate<500)
   {
   codeID = 5;
   }

   if (Bitrate<200)
   {
   codeID = 6;
   }

   if (Bitrate==0)
   {
   codeID = 7;
   }

    if (gob('codechoice').value == 1) {codeID =1};

    if (gob('codechoice').value == codeID) {

     var SLat = xmlDoc.getElementsByTagName("StartLat")[i].childNodes[0].nodeValue;
     var SLon= xmlDoc.getElementsByTagName("StartLon")[i].childNodes[0].nodeValue;
     var ELat = xmlDoc.getElementsByTagName("EndLat")[i].childNodes[0].nodeValue;
     var ELon = xmlDoc.getElementsByTagName("EndLon")[i].childNodes[0].nodeValue;
     var CellID = xmlDoc.getElementsByTagName("CellID")[i].childNodes[0].nodeValue;

     var startPoint = new google.maps.LatLng(SLat,SLon);
     bounds.extend(startPoint);
     var endPoint = new google.maps.LatLng(ELat,ELon);
     bounds.extend(endPoint);

     var polyline = new google.maps.Polyline({
                 path: [startPoint, endPoint],
                 strokeColor:
xmlDoc.getElementsByTagName("Color")[i].childNodes[0].nodeValue,
                 strokeOpacity: 1.0,
                 strokeWeight: 5,
                 map: map
     });
   };

  } // for i...
```

```
  google.maps.event.addListener(map, 'click', function() {
      infowindow.close();
  });

  map.fitBounds(bounds);

}


function showMarkers(){

  for (i=0;i<x.length;i++){

    var Bitrate = xmlDoc.getElementsByTagName("Bitrate")[i].childNodes[0].nodeValue;


   if (Bitrate>=7000)
   {
   codeID = 2;
   }

   if (Bitrate>=2000 && Bitrate<7000)
   {
   codeID = 3;
   }

   if (Bitrate>=500 && Bitrate<2000)
   {
   codeID = 4;
   }

   if (Bitrate>=200 && Bitrate<500)
   {
   codeID = 5;
   }

   if (Bitrate<200)
   {
   codeID = 6;
   }

   if (Bitrate==0)
   {
   codeID = 7;
   }

    if (gob('codechoice').value == 1) {codeID =1};

    if (gob('codechoice').value == codeID) {

        var SLat = xmlDoc.getElementsByTagName("StartLat")[i].childNodes[0].nodeValue/2;
     var SLon= xmlDoc.getElementsByTagName("StartLon")[i].childNodes[0].nodeValue/2;
     var ELat = xmlDoc.getElementsByTagName("EndLat")[i].childNodes[0].nodeValue/2;
     var ELon = xmlDoc.getElementsByTagName("EndLon")[i].childNodes[0].nodeValue/2;
     var CellID = xmlDoc.getElementsByTagName("CellID")[i].childNodes[0].nodeValue;
     var Tech = xmlDoc.getElementsByTagName("NetworkType")[i].childNodes[0].nodeValue;
     var LAC = xmlDoc.getElementsByTagName("Lac")[i].childNodes[0].nodeValue;

     var SLatM = SLat+ELat; // to set Marker in the Middle
     var SLonM = SLon+ELon; // to set Marker in the Middle
     var myLatlng = new google.maps.LatLng(SLatM,SLonM);
```

```
      var theTitle =  "(" + Tech + ")" + " | Bitrate: " + Bitrate + ", CellID: " + CellID
+ ", LAC: " + LAC;

     var image = new google.maps.MarkerImage('myMarker.gif',
     new google.maps.Size(7, 42),
     new google.maps.Point(0,0));

      var marker = new google.maps.Marker({
        position: myLatlng,
        map: map,
       icon: image,
        title: theTitle
       });
    }
  }
  return;
}


function setCode(){
    codeID = gob('codechoice').value;
    return;
}


function gob(e){
    if(typeof(e)=='object')return(e);
    if(document.getElementById)return(document.getElementById(e));
    return(eval(e))
}


function HomeControl(controlDiv, map) {

  // Set CSS styles for the DIV containing the control
  // Setting padding to 5 px will offset the control from the edge of the map
  controlDiv.style.padding = '5px';

  // Set CSS for the control border
  var controlUI = document.createElement('DIV');
  controlUI.style.backgroundColor = 'yellow';
  controlUI.style.borderStyle = 'solid';
  controlUI.style.borderWidth = '2px';
  controlUI.style.cursor = 'pointer';
  controlUI.style.textAlign = 'center';
  controlUI.title = 'Click to set the map to Home';
  controlDiv.appendChild(controlUI);

  // Set CSS for the control interior
  var controlText = document.createElement('DIV');
  controlText.style.fontFamily = 'Arial,sans-serif';
  controlText.style.fontSize = '12px';
  controlText.style.paddingLeft = '4px';
  controlText.style.paddingRight = '4px';
  controlText.innerHTML = '<b>Athens-Center</b>';
  controlUI.appendChild(controlText);

  // Setup the click event listeners: simply set the map to Athens_Center
  google.maps.event.addDomListener(controlUI, 'click', function() {
     map.setCenter(Athens_Center)
  });
```

```
}


function showAddress(address) {
    geocoder.geocode({'address': address}, function(results, status) {
        if (status == google.maps.GeocoderStatus.OK) {
            var pos = results[0].geometry.location;
            map.setCenter(pos);
        } else {
            alert("Geocode was not successful for the following reason: " + status);
        }
    });
}

</script>

<style type="text/css">
* {margin:0;padding:0;border:0;outline:0}

BODY {
    font-family: Helvetica,Sans Serif;
    font-size: 11pt;
}

h1 {
    font-size:30px;
    font-weight: bold;
    color: #000000;
    padding-top: 4px;
    padding-left: 0px;
    padding-bottom: 11px;
}
.topbutton {
    padding-top: 10px;
    padding-bottom: 10px;
    padding-left: 1px;
    float: left;
}

#buttonrow {
    position: absolute;
    top: 90px;
    left: 1px;
    width: 350px;
    height: 20px;
    background-color: #ffffff;
}

#presenter {
    opacity:0.8;
    filter:alpha(opacity=80);
    position: absolute;
    top: 150px;
    left: 10px;
    width: 220px;
    height: 30px;
    background-color: #808080;
}

</style>
```

```
</head>

<body style="margin:0px; padding:0px;" onload="initialize()">


<div id="header">
    <div id="title">
        <h1>Bitrate Performance Tool</h1>
    </div>
        <div class="choice">
        <form id="codes" style="float:left;text-align:right;padding-top:10px;"
action="./" method="post" onsubmit="return false">
&nbspMeasurements' Range (msec):
        <select id="codechoice" name="codechoice"
onchange="codeID=parseInt(this.options[this.selectedIndex].value);setCode();">
        <option selected="selected" value="1">Show All</option>
        <option value="2">>7000</option>
        <option value="3">2000-7000</option>
        <option value="4">500-2000</option>
        <option value="5">200-500</option>
        <option value="6">0-200</option>
        <option value="7">"no conneX"</option>
        </select>
        </form>
    </div>

     <div align="left">
      <div class="topbutton">
      <SCRIPT LANGUAGE="JavaScript">
      document.write('<form><input type=button value="Refresh"
onClick="history.go()"></form>')
      </script>
      </div>
     </div>


      <div id="buttonrow">
   <div class="topbutton">
        <form style="margin-right:5px" action="#"
onsubmit="showAddress(this.address.value); return false">
      <input type="text" style="border: 1px solid #000000;" size="37" name="address"
value="Athens" />
     <input type="submit" value="GoToArea" />
      </form>
   </div>
   </div>

<div align="left">
<input type="button" onclick="showMarkers();" value="Show Tooltips"/>
<p></p>
</div>


   <div id="presenter">
       <form style="margin-bottom:2px" action="#">
                
       <textarea id="Legend" cols="25" rows="6">
   L-e-g-e-n-d
Blue: >7000 kbps
Green: 2000-7000 kbps
Yellow: 500-2000 kbps
Orange: 200-500 kbps
```

```
Red: 0-200 kbps
   </textarea>
   </form>
   </div>

</div>

<div id="map_canvas" style="width:100%; height:100%"></div>
<script src="http://www.google-analytics.com/urchin.js" type="text/javascript">
</script>
<script type="text/javascript">
_uacct = "UA-162157-1";
urchinTracker();
</script>

</body>
</html>
```

NetPerfTool-Latency.html

```
<html>
<head>
<meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
<meta http-equiv="content-type" content="text/html; charset=UTF-8"/>
<title>Performance Evaluation Tool (Ping Measurements)</title>
<link
href="http://code.google.com/apis/maps/documentation/javascript/examples/default.css"
rel="stylesheet" type="text/css" />
<script type="text/javascript"
src="http://maps.google.com/maps/api/js?sensor=false"></script>
<script type="text/javascript" src="loadxmldoc.js"></script>

<script type="text/javascript">
var codeID = 1;
var Range = 1;

var geocoder; // = new google.maps.Geocoder();
xmlDoc=loadXMLDoc("C:/NetPerfTool/IMEI9838Tr201102070913-PING.xml");
var x=xmlDoc.getElementsByTagName('StartLat');

var map = null;
var bounds = null;
var Athens_Center= new google.maps.LatLng(37.979116, 23.717165);


function initialize() {
  geocoder = new google.maps.Geocoder();

  var myOptions = {
    zoom: 10,
    center: new google.maps.LatLng(38.0782, 23.800),
    mapTypeControl: true,
    mapTypeControlOptions: {style: google.maps.MapTypeControlStyle.DROPDOWN_MENU},
    navigationControl: true,
    mapTypeId: google.maps.MapTypeId.ROADMAP
  }

  map = new google.maps.Map(document.getElementById("map_canvas"), myOptions);
  bounds = new google.maps.LatLngBounds();
  var homeControlDiv = document.createElement('DIV');
  var homeControl = new HomeControl(homeControlDiv, map);
```

```
  homeControlDiv.index = 1;
  map.controls[google.maps.ControlPosition.TOP_RIGHT].push(homeControlDiv);

  setCode();

  for (i=0;i<x.length;i++){

    var ALat = xmlDoc.getElementsByTagName("AvgLatency")[i].childNodes[0].nodeValue;

   if (ALat>0 && ALat<60)
   {
   codeID = 2;
   }

   if (ALat>=60 && ALat<120)
   {
   codeID = 3;
   }

   if (ALat>=120 && ALat<300)
   {
   codeID = 4;
   }

   if (ALat>=300 && ALat<800)
   {
   codeID = 5;
   }

   if (ALat>=800)
   {
   codeID = 6;
   }

   if (ALat==0)
   {
   codeID = 7;
   }

    if (gob('codechoice').value == 1) {codeID =1};

    if (gob('codechoice').value == codeID) {

     var SLat = xmlDoc.getElementsByTagName("StartLat")[i].childNodes[0].nodeValue;
     var SLon= xmlDoc.getElementsByTagName("StartLon")[i].childNodes[0].nodeValue;
     var ELat = xmlDoc.getElementsByTagName("EndLat")[i].childNodes[0].nodeValue;
     var ELon = xmlDoc.getElementsByTagName("EndLon")[i].childNodes[0].nodeValue;
     var CellID = xmlDoc.getElementsByTagName("CellID")[i].childNodes[0].nodeValue;

     var startPoint = new google.maps.LatLng(SLat,SLon);
     bounds.extend(startPoint);
     var endPoint = new google.maps.LatLng(ELat,ELon);
     bounds.extend(endPoint);

     var polyline = new google.maps.Polyline({
                 path: [startPoint, endPoint],
                 strokeColor:
xmlDoc.getElementsByTagName("Color")[i].childNodes[0].nodeValue,
                 strokeOpacity: 1.0,
                 strokeWeight: 5,
                 map: map
     });
```

```
   };

  } // for i...


  google.maps.event.addListener(map, 'click', function() {
      infowindow.close();
  });

  map.fitBounds(bounds);

}


function showMarkers(){

  for (i=0;i<x.length;i++){

    var ALat = xmlDoc.getElementsByTagName("AvgLatency")[i].childNodes[0].nodeValue;

   if (ALat>0 && ALat<60)
   {
   codeID = 2;
   }

   if (ALat>=60 && ALat<120)
   {
   codeID = 3;
   }

   if (ALat>=120 && ALat<300)
   {
   codeID = 4;
   }

   if (ALat>=300 && ALat<800)
   {
   codeID = 5;
   }

   if (ALat>=800)
   {
   codeID = 6;
   }

   if (ALat==0)
   {
   codeID = 7;
   }

    if (gob('codechoice').value == 1) {codeID =1};

    if (gob('codechoice').value == codeID) {

        var SLat = xmlDoc.getElementsByTagName("StartLat")[i].childNodes[0].nodeValue/2;
     var SLon= xmlDoc.getElementsByTagName("StartLon")[i].childNodes[0].nodeValue/2;
     var ELat = xmlDoc.getElementsByTagName("EndLat")[i].childNodes[0].nodeValue/2;
     var ELon = xmlDoc.getElementsByTagName("EndLon")[i].childNodes[0].nodeValue/2;
     var CellID = xmlDoc.getElementsByTagName("CellID")[i].childNodes[0].nodeValue;
     var Tech = xmlDoc.getElementsByTagName("NetworkType")[i].childNodes[0].nodeValue;
     var LAC = xmlDoc.getElementsByTagName("Lac")[i].childNodes[0].nodeValue;
```

```
     var SLatM = SLat+ELat; // to set Marker in the Middle
     var SLonM = SLon+ELon; // to set Marker in the Middle
     var myLatlng = new google.maps.LatLng(SLatM,SLonM);

      var theTitle =  "(" + Tech + ")" + " | Latency: " + ALat + ", CellID: " + CellID +
", LAC: " + LAC;

     var image = new google.maps.MarkerImage('myMarker.gif',
     new google.maps.Size(7, 42),
     new google.maps.Point(0,0));

      var marker = new google.maps.Marker({
        position: myLatlng,
        map: map,
       icon: image,
        title: theTitle
      });
    }
  }
  return;
}


function setCode(){
    codeID = gob('codechoice').value;
    return;
}


function gob(e){
    if(typeof(e)=='object')return(e);
    if(document.getElementById)return(document.getElementById(e));
    return(eval(e))
}


function HomeControl(controlDiv, map) {

  // Set CSS styles for the DIV containing the control
  // Setting padding to 5 px will offset the control from the edge of the map
  controlDiv.style.padding = '5px';

  // Set CSS for the control border
  var controlUI = document.createElement('DIV');
  controlUI.style.backgroundColor = 'yellow';
  controlUI.style.borderStyle = 'solid';
  controlUI.style.borderWidth = '2px';
  controlUI.style.cursor = 'pointer';
  controlUI.style.textAlign = 'center';
  controlUI.title = 'Click to set the map to Home';
  controlDiv.appendChild(controlUI);

  // Set CSS for the control interior
  var controlText = document.createElement('DIV');
  controlText.style.fontFamily = 'Arial,sans-serif';
  controlText.style.fontSize = '12px';
  controlText.style.paddingLeft = '4px';
  controlText.style.paddingRight = '4px';
  controlText.innerHTML = '<b>Athens-Center</b>';
  controlUI.appendChild(controlText);

  // Setup the click event listeners: simply set the map to Athens_Center
```

```
  google.maps.event.addDomListener(controlUI, 'click', function() {
     map.setCenter(Athens_Center)
  });

}


function showAddress(address) {
    geocoder.geocode({'address': address}, function(results, status) {
        if (status == google.maps.GeocoderStatus.OK) {
            var pos = results[0].geometry.location;
            map.setCenter(pos);
        } else {
            alert("Geocode was not successful for the following reason: " + status);
        }
    });
}

</script>

<style type="text/css">
* {margin:0;padding:0;border:0;outline:0}

BODY {
    font-family: Helvetica,Sans Serif;
    font-size: 11pt;
}

h1 {
    font-size:30px;
    font-weight: bold;
    color: #000000;
    padding-top: 4px;
    padding-left: 0px;
    padding-bottom: 11px;
}
.topbutton {
    padding-top: 10px;
    padding-bottom: 10px;
    padding-left: 1px;
    float: left;
}

#buttonrow {
    position: absolute;
    top: 90px;
    left: 1px;
    width: 350px;
    height: 20px;
    background-color: #ffffff;
}

#presenter {
    opacity:0.8;
    filter:alpha(opacity=80);
    position: absolute;
    top: 150px;
    left: 10px;
    width: 220px;
    height: 30px;
    background-color: #808080;
}
```

```
</style>


</head>

<body style="margin:0px; padding:0px;" onload="initialize()">


<div id="header">
    <div id="title">
        <h1>Latency Performance Tool</h1>
    </div>
        <div class="choice">
        <form id="codes" style="float:left;text-align:right;padding-top:10px;"
action="./" method="post" onsubmit="return false">
&nbspMeasurements' Range (msec):
        <select id="codechoice" name="codechoice"
onchange="codeID=parseInt(this.options[this.selectedIndex].value);setCode();">
        <option selected="selected" value="1">0-2500</option>
        <option value="2">0-60</option>
        <option value="3">60-120</option>
        <option value="4">120-300</option>
        <option value="5">300-800</option>
        <option value="6">800-2500</option>
        <option value="7">"no conneX"</option>
        </select>
        </form>
    </div>

     <div align="left">
      <div class="topbutton">
      <SCRIPT LANGUAGE="JavaScript">
      document.write('<form><input type=button value="Refresh"
onClick="history.go()"></form>')
      </script>
      </div>
     </div>


      <div id="buttonrow">
   <div class="topbutton">
        <form style="margin-right:5px" action="#"
onsubmit="showAddress(this.address.value); return false">
      <input type="text" style="border: 1px solid #000000;" size="37" name="address"
value="Athens" />
     <input type="submit" value="GoToArea" />
      </form>
   </div>
   </div>

<div align="left">
<input type="button" onclick="showMarkers();" value="Show Tooltips"/>
<p></p>
</div>


   <div id="presenter">
       <form style="margin-bottom:2px" action="#">
                
       <textarea id="Legend" cols="25" rows="6">
   L-e-g-e-n-d
```

```
Blue: 0-60 msec
Green: 60-120 msec
Yellow: 120-300 msec
Orange: 300-800 msec
Red: 800-2500 msec
   </textarea>
   </form>
   </div>

</div>

<div id="map_canvas" style="width:100%; height:100%"></div>
<script src="http://www.google-analytics.com/urchin.js" type="text/javascript">
</script>
<script type="text/javascript">
_uacct = "UA-162157-1";
urchinTracker();
</script>

</body>
</html>
```

loadxmldoc.js

```
function loadXMLDoc(dname)
{
var xmlDoc;
// code for IE
if (window.ActiveXObject)
{
xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
}
// code for Mozilla, Firefox, Opera, etc.
else if (document.implementation && document.implementation.createDocument)
{
xmlDoc=document.implementation.createDocument("","",null);
}
else
{
alert('Your browser cannot handle this script');
}
xmlDoc.async=false;
xmlDoc.load(dname);
return(xmlDoc);
}
```

Τα php scripts, handle_upload.php και update.php, που είναι υπεύθυνα για την υποδοχή των xml αρχείων και την εισαγωγή τους στη βάση δεδομένων αντίστοιχα, παρουσιάζονται παρακάτω.

handle_upload.php

```
<?php
// Where the file is going to be placed
$target_path = "KampRate/Temp/";
```

```
// Check for the following:
// file type == xml || gpx || kmz
// file size < 2MB
$filename = basename($_FILES['uploadedfile']['name']);
$ext = substr($filename, strrpos($filename, '.') + 1);

if ((($ext == "xml") || ($ext == "gpx") || ($ext == "kmz")) &&
     ($_FILES["uploaded_file"]["size"] < 2000000)) {

   /* Add the original filename to our target path.
   Result is "KampRate/Temp/filename.extension" */
   $target_path = $target_path . basename($_FILES['uploadedfile']['name']);

   if(move_uploaded_file($_FILES['uploadedfile']['tmp_name'], $target_path)) {
       echo "The file ".  basename( $_FILES['uploadedfile']['name']).
       " has been uploaded";
   } else{
       echo "There was an error uploading the file";
   }
}
?>
```

update.php

```
<?php
$dirname = "C:/Program Files/Apache Software Foundation/Apache2.2/htdocs/KampRate/Temp";
$dir = opendir($dirname);

$username="root";
$password="";
$database="kamprate";

mysql_connect( localhost, $username, $password );
@mysql_select_db( $database ) or die( "Unable to select database" );


echo "The following xml files were inserted into database: <br/>";

while( ( $file = readdir($dir) ) != false ){
   if( ($file != ".") and ($file != "..") ){

      $NMC = explode( "-", $file );


      if ( $NMC[1] == "DL.xml" ) {
         $query = "LOAD XML LOCAL INFILE 'C:/Program Files/Apache Software
Foundation/Apache2.2/htdocs/KampRate/Temp/$file' INTO TABLE download ROWS IDENTIFIED BY
'<line>'";
         mysql_query($query);

         if ( copy( "C:/Program Files/Apache Software
Foundation/Apache2.2/htdocs/KampRate/Temp/$file", "C:/Program Files/Apache Software
Foundation/Apache2.2/htdocs/KampRate/DL/$file" ) ) {
            unlink( "C:/Program Files/Apache Software
Foundation/Apache2.2/htdocs/KampRate/Temp/$file" );
         }
      }
      else if ( $NMC[1] == "PING.xml" ) {
```

```
         $query = "LOAD XML LOCAL INFILE 'C:/Program Files/Apache Software
Foundation/Apache2.2/htdocs/KampRate/Temp/$file' INTO TABLE ping ROWS IDENTIFIED BY
'<line>'";
         mysql_query($query);

         if ( copy( "C:/Program Files/Apache Software
Foundation/Apache2.2/htdocs/KampRate/Temp/$file", "C:/Program Files/Apache Software
Foundation/Apache2.2/htdocs/KampRate/Ping/$file" ) ) {
            unlink( "C:/Program Files/Apache Software
Foundation/Apache2.2/htdocs/KampRate/Temp/$file" );
         }
      }

      echo( "$file <br />" );

   }
}

mysql_close();

?>
```