



ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΔΙΑΤΜΗΜΑΤΙΚΟ ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΠΛΗΡΟΦΟΡΙΚΗ ΚΑΙ ΥΠΟΛΟΓΙΣΤΙΚΗ ΒΙΟΙΑΤΡΙΚΗ
ΚΑΤΕΥΘΥΝΣΗ

«ΠΛΗΡΟΦΟΡΙΚΗ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗΝ ΑΣΦΑΛΕΙΑ,
ΔΙΑΧΕΙΡΙΣΗ ΜΕΓΑΛΟΥ ΟΓΚΟΥ ΔΕΔΟΜΕΝΩΝ ΚΑΙ
ΠΡΟΣΟΜΟΙΩΣΗ»

Αξιολόγηση Ποιότητας Υπηρεσιών Δεδομένων Δικτύων Κινητών
Επικοινωνιών με την Χρήση Εφαρμογής Έξυπνου Κινητού
Mobile Communication Networks Data Services Quality Evaluation
Using Smart Phone Application

Διπλωματική Εργασία

Νικόλαος Μιχαηλίδης

Επιβλέποντες

Δρ. Σταμούλης Γεώργιος

Δρ. Κορίνθιος Ιωάννης

Λαμία, 2017

«Υπεύθυνη Δήλωση μη λογοκλοπής και ανάληψης προσωπικής ευθύνης»

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, και γνωρίζοντας τις συνέπειες της λογοκλοπής, δηλώνω υπεύθυνα και ενυπογράφως ότι η παρούσα εργασία με τίτλο «Εφαρμογή καταγραφής ποιότητας σήματος σε δίκτυα κινητών επικοινωνιών» αποτελεί προϊόν αυστηρά προσωπικής εργασίας και όλες οι πηγές από τις οποίες χρησιμοποίησα δεδομένα, ιδέες, φράσεις, προτάσεις ή λέξεις, είτε επακριβώς (όπως υπάρχουν στο πρωτότυπο ή μεταφρασμένες) είτε με παράφραση, έχουν δηλωθεί κατάλληλα και ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής.

Ο δηλών

Νικόλαος Μιχαηλίδης

Ημερομηνία

Υπογραφή

Τριμελής Επιτροπή:

Σταμούλης Γεώργιος

Λουκόπουλος Αθανάσιος

Λαλλάς Ευθύμιος

Επιστημονικός Σύμβουλος:

Κορίνθιος Ιωάννης

Περίληψη

Η παρούσα διπλωματική εργασία έχει ως στόχο την υλοποίηση μιας εφαρμογής που αξιολογεί την ποιότητα υπηρεσιών δεδομένων δικτύων κινητών επικοινωνιών.

Η εφαρμογή υλοποιήθηκε σε περιβάλλον Android και είναι συμβατή με κινητές συσκευές που χρησιμοποιούν ως λειτουργικό σύστημα το Android έκδοσης 5 ή νεότερο. Η ανάπτυξη της εφαρμογής έγινε με την χρήση του λογισμικού Android Studio της Google.

Ο κύριος στόχος της εφαρμογής είναι η απεικόνιση σε πραγματικό χρόνο και σε χάρτη της διέλευσης/ταχύτητας μεταφοράς δεδομένων που επιτυγχάνεται σε ένα δίκτυο κινητής τηλεφωνίας από την πλευρά του χρήστη, και επίσης η καταγραφή/αποθήκευση και η αναπαραγωγή των δεδομένων αυτών κατά τη βούληση του χρήστη της εφαρμογής.

Τα στοιχεία που καταγράφονται είναι οι συντεταγμένες ο τύπος του δικτύου που χρησιμοποιείται (2G, 3G, 4G) και το πρωτόκολλο αυτού (π.χ. GPRS, EDGE, UMTS, HSPA, HSPAP, LTE), ο ρυθμός μεταφοράς δεδομένων σε kbps και η ισχύς του σήματος σε dBm.

Η δομή της διπλωματικής εργασίας περιγράφεται περιληπτικά στη συνέχεια. Στο Πρώτο Κεφάλαιο γίνεται αναφορά στα Δίκτυα Κινητών Επικοινωνιών. Αναπτύσσεται η έννοια της κινητής τηλεφωνίας, παρουσιάζεται η εξέλιξη των κυψελωτών συστημάτων και οι τεχνικές πολλαπλής πρόσβασης. Στο Δεύτερο Κεφάλαιο γίνεται μια εισαγωγή σε θέματα Ποιότητας Υπηρεσιών (Quality of Service, QoS), και επίσης ορίζονται οι αντικειμενικοί δείκτες ποιότητας υπηρεσιών. Στο Τρίτο Κεφάλαιο παρουσιάζεται η εφαρμογή Mobile Signal Tester που υλοποιήθηκε στα πλαίσια της διπλωματικής εργασίας σε περιβάλλον Android Studio. Στο Τέταρτο Κεφάλαιο παρατίθενται τα αποτελέσματα δοκιμών πεδίου τα οποία πραγματοποιήθηκαν με την χρήση της εφαρμογής τόσο σε Αστική Περιοχή όσο και σε Επαρχιακό Οδικό Δίκτυο. Τέλος, στο Πέμπτο Κεφάλαιο παρουσιάζονται τα συμπεράσματα που προκύπτουν από την καταγραφή των μετρήσεων της εφαρμογής Mobile Signal Tester και συζητούνται πιθανές επεκτάσεις της εφαρμογής που μπορούν να γίνουν μελλοντικά.

Περιεχόμενα

Περίληψη.....	4
Περιεχόμενα.....	5
1 Εισαγωγή στα Δίκτυα Κινητών Επικοινωνιών.....	7
1.1 Ιστορική αναδρομή.....	7
1.2 Το Δίκτυο της Κινητής Τηλεφωνίας.....	8
1.3 Κυψελωτό Δίκτυο Κινητής Τηλεφωνίας.....	10
1.3.1 Βασικές Ιδιότητες των Κυψελωτών Δικτύων.....	11
1.3.2 Τύποι Κυψελών.....	12
1.4 Τεχνικές Πολλαπλής Πρόσβασης.....	12
1.4.1 Πολλαπλή Πρόσβαση Συχνότητας (Frequency Division Multiple Access, FDMA).....	13
1.4.2 Πολλαπλή Πρόσβαση Χρόνου (Time Division Multiple Access, TDMA).....	13
1.4.3 Πολλαπλή Πρόσβαση Κώδικα (Code Division Multiple Access, CDMA).....	14
1.4.4 Πολλαπλή Πρόσβαση Χώρου (Space Division Multiple Access, SDMA).....	14
2 Εισαγωγή σε Θέματα Ποιότητας Υπηρεσιών (Quality of Service).....	16
2.1 Εισαγωγή.....	16
2.2 Τάξεις Ποιότητας Υπηρεσιών (QoS Classes).....	17
2.3 Μετρήσεις Ποιότητας Υπηρεσίας.....	19
2.3.1 Δείκτες Ανεξάρτητοι της υπηρεσίας.....	20
2.3.2 Δείκτες Υπηρεσίας Τηλεφωνίας.....	21
2.3.3 Δείκτες Ευρυζωνικών Υπηρεσιών Δεδομένων.....	22
3 Το Android OS και η εφαρμογή Mobile Signal Tester.....	24
3.1 Εισαγωγή.....	24
3.2 Αρχιτεκτονική του Android.....	25
3.3. Περιγραφή Εφαρμογής Mobile Signal Tester.....	27
3.3.1 Διάρθρωση του κώδικα.....	28
3.3.2 Οι κλάσεις της εφαρμογής.....	34
3.3.3 Λειτουργική Περιγραφή της Εφαρμογής.....	36
3.3.3.1 Αρχική σελίδα.....	36
3.3.3.2 Ρυθμίσεις.....	38

3.3.3.3 Εκτέλεση Τεστ.....	42
4 Αποτελέσματα Δοκιμών.....	48
4.1 Εισαγωγή.....	48
4.2 Περιορισμοί.....	48
4.3 Αστικό περιβάλλον.....	48
4.4 Εθνικό δίκτυο.....	52
4.5 Επαρχιακό δίκτυο.....	53
5 Συμπεράσματα και Μελλοντικές Επεκτάσεις.....	56
5.1 Συμπεράσματα.....	56
5.2 Μελλοντικές Επεκτάσεις.....	56
Βιβλιογραφία.....	58
Παράρτημα - Πηγαίος Κώδικας Εφαρμογής.....	60

1 Εισαγωγή στα Δίκτυα Κινητών Επικοινωνιών

1.1 Ιστορική αναδρομή

Οι κινητές επικοινωνίες έχουν τεράστια εξέλιξη τα τελευταία 30 περίπου χρόνια. Ιστορικά διαχωρίζουμε διάφορες γενιές συστημάτων κινητών επικοινωνιών ανάλογα με τα χαρακτηριστικά και τις δυνατότητες τους.

Η πρώτη γενιά δικτύων κινητών επικοινωνιών (1G) ήρθε στην αγορά το 1979 στην περίπτωση της Ιαπωνίας και το 1981 στις χώρες της Βορείου Ευρώπης. Οι ΗΠΑ περίμεναν περισσότερο, μέχρι το 1983, με το Ισραήλ να ακολουθεί το 1986 και την Αυστραλία το 1987. Όλα αυτά τα πρώτα δίκτυα, μερικά εκ των οποίων έμειναν ενεργά και μέχρι το 2008, ήταν αναλογικά, όχι ψηφιακά.

Η δεύτερη γενιά δικτύων κινητών επικοινωνιών (2G) πέρασε στην ψηφιακή εποχή με δύο βασικά συστήματα: GSM και CDMA. Το πρώτο το είδαμε στην Ευρώπη και στην Ελλάδα αλλά και παγκόσμια, το δεύτερο υλοποιήθηκε κυρίως στις ΗΠΑ. Η Φινλανδία έκανε την αρχή το 1991 και όλα τα υπόλοιπα πήραν το δρόμο τους. Χάρη στο πρότυπο GSM είδαμε την επίσημη πρώτη των μηνυμάτων SMS αλλά και τη δυνατότητα κατεβάσματος περιεχομένου (π.χ. ringtones), ακόμη και τις πρώτες δοκιμές για πληρωμές μέσω κινητού τηλεφώνου. Το 1999 η ιαπωνική NTT DoCoMo ήταν ο πρώτος πάροχος που φρόντισε να δώσει πρόσβαση στο Internet μέσω αυτής της γενιάς δικτύων.

Σύντομα ήταν ξεκάθαρο ότι ο κόσμος είχε απαίτηση για περισσότερη χρήση data και έγιναν προσπάθειες για την οριοθέτηση της τρίτης γενιάς δικτύων κινητών επικοινωνιών (3G) με μεγαλύτερες ταχύτητες μετάδοσης. Η τεχνολογία 3G ήταν το αποτέλεσμα των εργασιών έρευνας και ανάπτυξης που πραγματοποίησε η Διεθνής Ένωση Τηλεπικοινωνιών (ITU) στις αρχές της δεκαετίας του 1980. Χρειάστηκαν δεκαπέντε χρόνια για να αναπτυχθούν οι προδιαγραφές και τα πρότυπα του 3G. Οι τεχνικές προδιαγραφές τέθηκαν στη διάθεση του κοινού με την ονομασία IMT-2000. Το φάσμα επικοινωνίας που διατέθηκε ήταν μεταξύ 400 MHz και 3 GHz. Το πρώτο δίκτυο 3G ξεκίνησε από την NTT DoCoMo στην Ιαπωνία το 1998, με την επωνυμία FOMA. Διατέθηκε για πρώτη φορά το Μάιο του 2001 ως πρότυπη τεχνολογία W-CDMA. Η πρώτη εμπορική παρουσίαση της τεχνολογίας 3G πραγματοποιήθηκε επίσης

από την NTT DoCoMo στην Ιαπωνία την 1η Οκτωβρίου 2001, αν και αρχικά ήταν κάπως περιορισμένη. Το πρώτο ευρωπαϊκό προ-εμπορικό δίκτυο ήταν ένα δίκτυο UMTS στη Νήσο του Μαν από την Manx Telecom και το πρώτο εμπορικό δίκτυο (επίσης W-CDMA με βάση το UMTS) στην Ευρώπη άρχισε να λειτουργεί από την Telenor, τον Δεκέμβριο του 2001. Το πρώτο εμπορικό δίκτυο 3G των Ηνωμένων Πολιτειών ήταν από την Monet Mobile Networks, με τεχνολογία CDMA2000 1x EV-DO, και ακολουθήσε η Verizon Wireless τον Ιούλιο του 2002 επίσης σε CDMA2000 1x EV-DO. Το Mobility της AT&T ήταν επίσης ένα πραγματικό 3G δίκτυο UMTS, έχοντας ολοκληρώσει την αναβάθμιση του δικτύου 3G σε HSUPA.

Και πάλι όμως οι ανάγκες αυξήθηκαν και προέκυψαν δύο προδιαγραφές 4ης γενιάς, οι WiMAX και LTE/LTE Advanced. Η Τεχνολογία LTE επικράτησε σχετικά γρήγορα και εμφανίστηκε πρώτα σε σκανδιναβικές χώρες από την TeliaSonera. Φυσικά και έφερε πολλαπλάσιες ταχύτητες που σήμερα αγγίζουν ακόμη και τα 300 Mbps και εκτός της ταχύτητας περάσαμε σε δίκτυα επικοινωνιών βασισμένα σε IP. Από την γενιά 2.5G GPRS, τα κυψελοειδή συστήματα παρέχουν διπλές υποδομές: κόμβους μεταγωγής πακέτων για υπηρεσίες δεδομένων και κόμβους μεταγωγής κυκλωμάτων για φωνητικές κλήσεις. Στα συστήματα 4G, η υποδομή μεταγωγής κυκλωμάτων εγκαταλείπεται και παρέχεται μόνο ένα δίκτυο μεταγωγής πακέτων. Αυτό σημαίνει ότι στο 4G, οι παραδοσιακές φωνητικές κλήσεις αντικαθίστανται από IP τηλεφωνία.

Τα ασύρματα συστήματα 5ης γενιάς (5G) είναι τα προτεινόμενα επόμενα τηλεπικοινωνιακά πρότυπα πέρα από τα τρέχοντα πρότυπα 4G / IMT-Advanced. Ο σχεδιασμός 5G στοχεύει σε υψηλότερη χωρητικότητα από την τρέχουσα τεχνολογία 4G, επιτρέποντας μεγαλύτερη πυκνότητα χρηστών ευρυζωνικών κινητών τηλεφώνων και υποστηρίζοντας εξαιρετικά αξιόπιστες και μαζικές επικοινωνίες μεταξύ συσκευών.

1.2 Το Δίκτυο της Κινητής Τηλεφωνίας

Το ασύρματο δίκτυο κινητής τηλεφωνίας παρέχει την δυνατότητα χρήσης του κινητού τηλεφώνου. Τα δίκτυα αυτά αποτελούνται από σταθμούς βάσης για να καλύψουν με ηλεκτρομαγνητικό σήμα τις περιοχές που το ασύρματο δίκτυο είναι διαθέσιμο. Όταν χρησιμοποιούμε το κινητό μας τηλέφωνο για να επικοινωνήσουμε, τότε αυτό στέλνει και λαμβάνει ηλεκτρομαγνητικά σήματα προς και από έναν σταθμό βάσης, ο οποίος

στη συνέχεια επικοινωνεί ενσύρματα ή ασύρματα με κάποια κέντρα αναδιανέμοντας την πληροφορία, ώστε να φτάσει στον συνομιλητή μας.



Σε κάθε περιοχή στην οποία διαιρείται το ασύρματο δίκτυο, αντιστοιχεί και ένας σταθμός βάσης. Οι σταθμοί βάσης αποτελούνται από κεραιοσυστήματα εκπομπής και λήψης των ηλεκτρομαγνητικών σημάτων, καθώς και ηλεκτρονικό εξοπλισμό για την επεξεργασία των σημάτων αυτών. Τα κεραιοσυστήματα των σταθμών βάσης βρίσκονται τοποθετημένα πάνω σε μεταλλικούς πυλώνες ή ιστούς. Εντός των πόλεων, οι οροφές ψηλών κτιρίων αποτελούν τον συνήθη χώρο για εγκατάσταση κεραιοσυστημάτων.

Η επικοινωνία με τα κινητά τηλέφωνα γίνεται μέσω των κεραιών εκπομπής και λήψης των σταθμών βάσης. Οι κεραίες αυτές έχουν συνήθως μακρόστενο σχήμα, μήκος ένα με δύο μέτρα, πλάτος δέκα με είκοσι εκατοστά, πάχος μερικά εκατοστά και τοποθετούνται κατακόρυφα. Πέραν των κεραιών για την σύνδεση με τα κινητά τηλέφωνα, οι σταθμοί βάσης έχουν συνήθως και μία μικροκυματική κεραία που χρησιμοποιείται για την ασύρματη σύνδεση του σταθμού με το κέντρο για την λήψη και την προώθηση των τηλεφωνικών κλήσεων. Οι κεραίες αυτές είναι κυλινδρικές, με διάμετρο συνήθως έως εξήντα εκατοστά.



1.3 Κυψελωτό Δίκτυο Κινητής Τηλεφωνίας

Το δίκτυο κινητής τηλεφωνίας με κυψελοειδή μορφή ονομάζεται Κυψελωτό. Η δομή του έχει ως στόχο την μέγιστη δυνατή συνδρομητική χωρητικότητα και εκμετάλλευση του προσφερόμενου φάσματος ραδιοσυχνοτήτων. Παράλληλα δίνει δυνατότητα ραδιοκάλυψης σχετικά μεγάλων γεωγραφικών περιοχών, προσφέροντας στους συνδρομητές της κινητής τηλεφωνίας ποιότητα στην επικοινωνία με αποδεκτό κόστος

Βασικό δομικό στοιχείο του κυψελωτού δικτύου, είναι ο σταθμός βάσης (Base Station). Κάθε σταθμός βάσης εξυπηρετεί μια συγκεκριμένη γεωγραφική περιοχή, η οποία χωρίζεται σε μία ή περισσότερες κυψέλες (Cells). Το μέγεθος της κυψέλης σχεδιάζεται βάσει του αναμενόμενου αριθμού χρηστών (χωρητικότητα) αλλά και των ιδιαίτερων γεωγραφικών χαρακτηριστικών της περιοχής (κάλυψη). Επειδή κάθε σταθμός βάσης μπορεί να εξυπηρετήσει ταυτόχρονα περιορισμένο αριθμό τηλεφώνων εφόσον το διαθέσιμο εύρος ζώνης είναι περιορισμένο, οι κυψέλες είναι σχετικά μικρές εντός των πόλεων (της τάξης των εκατοντάδων μέτρων), και σχετικά μεγάλες στις αγροτικές περιοχές (της τάξης των χιλιομέτρων ως δεκάδων χιλιομέτρων).

Κάθε σταθμός βάσης διαχειρίζεται όλες τις κλήσεις των κινητών τηλεφώνων εντός της κυψέλης. Το σχήμα των κυψελών στην πράξη το τους είναι ουσιαστικά ακαθόριστο καθώς επηρεάζεται από τα χαρακτηριστικά του ανάγλυφου του εδάφους, όπως δένδρα, λόφοι και κτίρια, τα οποία μπορούν να εμποδίσουν ή να εξασθενήσουν τα ραδιοκύματα. Επίσης, οι κατάλληλες θέσεις για τοποθέτηση σταθμών βάσης δεν είναι πάντα διαθέσιμες, ενώ, σε περιοχές με υψηλή πυκνότητα χρηστών, όπως στα κέντρα των πόλεων, μικρότερες κυψέλες είναι απαραίτητες. Άρα, σε ένα σωστά σχεδιασμένο

δίκτυο ο αριθμός των κυψελών σε μια περιοχή εξαρτάται και από τον αριθμό των χρηστών καθώς και την τηλεπικοινωνιακή κίνηση στην περιοχή αυτή.

1.3.1 Βασικές Ιδιότητες των Κυψελωτών Δικτύων

Τρεις βασικές παράμετροι ενός κυψελωτού δικτύου είναι η επαναχρησιμοποίηση συχνοτήτων, η κυψελοειδής διάσπαση και η μεταπομπή του χρήστη από κυψέλη σε κυψέλη.

- Εκτεταμένη Επαναχρησιμοποίηση συχνοτήτων

Η επαναχρησιμοποίηση συχνοτήτων είναι η διάθεση ραδιοδιαύλων, οι οποίοι έχουν τις ίδιες συχνότητες φορέα χωρίς να δημιουργείται πρόβλημα παρεμβολών, καθώς η καταχώρηση ίδιων συχνοτήτων γίνεται σε διαφορετικές γεωγραφικές περιοχές κάλυψης με επαρκή απόσταση μεταξύ τους. Το μέγεθος κάθε περιοχή κυμαίνεται από λίγα χιλιόμετρα μέχρι 50 ή και παραπάνω χιλιόμετρα. Οι ίδιες συχνότητες μπορούν να χρησιμοποιηθούν από δύο κυψέλες, αν υπάρχει επαρκή απόσταση μεταξύ τους για αποφυγή παρεμβολών. Έτσι πετυχαίνουμε εκτεταμένη επαναχρησιμοποίηση συχνοτήτων που οδηγεί σε μεγαλύτερη χωρητικότητα του συστήματος

- Κυψελοειδής Δομή

Αν δεν επαρκούν οι συχνότητες της κάθε κυψέλης, σε ώρες αιχμής ενεργοποιείται η διαδικασία της κυψελοειδούς διάσπασης, με την οποία μία κυψέλη διασπάται σε κυψέλες μικρότερης ακτίνας. Έτσι, η κάθε μικρή κυψέλη έχει τη δυνατότητα να εξυπηρετεί τον ίδιο αριθμό συνδρομητών που εξυπηρετούσε πριν η αρχική κυψέλη, χωρίς να υπάρχει ανάγκη αύξησης του διατιθέμενου φάσματος συχνοτήτων. Έτσι επιτυγχάνεται η αύξηση της χωρητικότητας του συστήματος, ενώ παράλληλα ελαχιστοποιείται η συγκαναλική παρεμβολή από την επίδραση των γειτονικών κυψελών.

- Μεταπομπή (Handover)

Πρόκειται για την διαδικασία μεταφοράς ελέγχου του εξοπλισμού χρήστη από τον ένα σταθμό βάσης στον άλλο. Στόχος είναι να μην γίνεται αντιληπτή η

μεταγωγή από τον χρήστη. Όταν η ένταση του σήματος ενός κινητού ελαττώνεται, συνήθως γιατί πλησιάζει στα όρια της κυψέλης, το σύστημα μετάγει τη ζεύξη αυτόματα σε γειτονικό σταθμό βάσης, με ισχυρότερο σήμα που διαφορετικές συχνότητες. Η εναλλαγή διαρκεί μερικά χιλιοστά του δευτερολέπτου.

1.3.2 Τύποι Κυψελών

Ο αριθμός των χρηστών που μπορούν να φιλοξενηθούν σε ένα κυψελωτό σύστημα επηρεάζεται τον αριθμό των κυψελών. Γενικότερα, σε περιοχές με μεγάλη πυκνότητα πληθυσμού, εγκαθίστανται μικροί σταθμοί βάσης με χαμηλή ισχύ. Ωστόσο, ένας μεγαλύτερος αριθμός σταθμών βάσης μπορεί να αυξήσει το κόστος για τον πάροχο κινητών υπηρεσιών. Ανάλογα με το μέγεθος και τη λειτουργία τους οι κυψέλες χωρίζονται σε:

- **Μάκροκυψέλες (Macro cells):** Μεγάλες κυψέλες που συνήθως προορίζονται για περιοχές που είναι απομακρυσμένες ή αραιοκατοικημένες με ακτίνα 0.5km έως 10km ή και περισσότερο.
- **Μίκροκυψέλες (Micro cells):** Βρίσκονται συνήθως σε πυκνοκατοικημένες περιοχές με διάμετρο ως περίπου και 0.5km.
- **Πίκροκυψέλες (Pico cells):** Χρησιμοποιούνται για την κάλυψη συγκεκριμένων πολύ μικρών περιοχών, όπως τμήματα κτιρίων ή και σήραγγες μια μεγαλύτερη κυψέλη θα ήταν αδύνατο να καλύψει.
- **Ομπρέλα-κυψέλες (Umbrella cells):** Επικουρικές κυψέλες, συνήθως παράλληλα με το οδικό δίκτυο όπου οι χρήστες χρειάζονται συχνές μεταπομπές. Όσο αναπτύσσονται τα δίκτυα η χρήση των Umbrella cells μειώνεται καθώς στην πράξη οι κυψέλες αυτές παρεμβάλλουν τις υποκείμενες κυψέλες.

1.4 Τεχνικές πολλαπλής πρόσβασης

Σε αντίθεση με την κινητή τηλεφωνία, στην σταθερή τηλεφωνία η πρόσβαση του συνδρομητή στο δίκτυο είναι συνήθως εξασφαλισμένη με την χρήση μιας αφιερωμένης φυσικής σύνδεσης ανά χρήστη π.χ. ο κλασσικός βρόχος σύνδεση με το τοπικό τηλεφωνικό κέντρο. Επομένως, η ταυτόχρονη και πολλαπλή πρόσβαση σε τέτοιου

τύπου δίκτυα εξασφαλίζεται με τη χρήση διαφορετικών φυσικών συνδέσεων. Αυτή η αρχή λειτουργίας δεν μπορεί να εφαρμοστεί άμεσα στην περίπτωση των δικτύων κινητών επικοινωνιών όπου το μέσο φυσικής πρόσβασης, δηλ. το ασύρματο interface, δεν μπορεί να απονεμηθεί σε ένα χρήστη αλλά αντίθετα αποτελεί ένα κοινόχρηστο μέσο μετάδοσης δεδομένων. Η ταυτόχρονη πρόσβαση περισσότερων του ενός συνδρομητών μπορεί να διασφαλιστεί με τη χρήση τεχνικών που δημιουργούν, πάνω από το ίδιο φυσικό μέσο μετάδοσης, διακριτά φυσικά κανάλια για κάθε ένα χρήστη.

1.4.1 Πολλαπλή Πρόσβαση Συχνότητας (Frequency Division Multiple Access, FDMA)

Η πολλαπλή πρόσβαση συχνότητας (Frequency Division Multiple Access, FDMA) διακρίνει τα διαφορετικά φυσικά κανάλια ή/και τους διαφορετικούς χρήστες στο πεδίο των συχνοτήτων δηλ. με πολυπλεξία στο πεδίο των συχνοτήτων (Frequency Division Multiplexing, FDM). Η απονομή των φυσικών καναλιών/συχνοτήτων στους χρήστες μπορεί να είναι στατική (ραδιοφωνία / τηλεόραση) ή δυναμική (δηλ. ανάλογα με τη ζήτηση όπως συμβαίνει με την περίπτωση της κινητής τηλεφωνίας).

Επιπλέον η πολυπλεξία στο πεδίο των συχνοτήτων μπορεί να χρησιμοποιηθεί και για την διάκριση μεταξύ των δύο διαφορετικών κατευθύνσεων μεταφοράς δεδομένων μιας αμφίδρομης σύνδεσης π.χ. την ταυτόχρονη πρόσβαση του σταθμού βάσης προς τον χρήστη καθώς και το αντίστροφο αλλά σε δύο διαφορετικές συχνότητες.

1.4.2 Πολλαπλή Πρόσβαση Χρόνου (Time Division Multiple Access, TDMA)

Η πολλαπλή πρόσβαση χρόνου (Time Division Multiple Access, TDMA) διακρίνει τα διαφορετικά φυσικά κανάλια ή/και τους διαφορετικούς χρήστες στο πεδίο του χρόνου δηλ. με πολυπλεξία στο πεδίο του χρόνου (Time Division Multiplexing, TDM).

Η απονομή των φυσικών καναλιών στους χρήστες γίνεται με την απόδοση μιας συγκεκριμένης χρονοθυρίδας ή χρονοσχισμής (time slot).

Η χρήση των χρονοθυρίδων μπορεί να γίνεται με ένα σταθερό σχήμα δηλ. ο κάθε χρήστης να εκπέμπει με μια σταθερή περιοδικότητα και σε διαδοχικά χρονικά διαστήματα οι διαφορετικοί χρήστες. Μια τέτοια προσέγγιση εξασφαλίζει ένα σταθερό

ρυθμό μετάδοσης δεδομένων και γι' αυτό χαρακτηρίζει τα δίκτυα φωνής. Αντίθετα, στα δίκτυα δεδομένων, όπου η κίνηση είναι ασύμμετρη και παρουσιάζει διακυμάνσεις στον όγκο της μεταφερόμενης πληροφορίας, είναι προφανώς μη αποδοτική. Σε αυτές τις περιπτώσεις χρησιμοποιούνται πιο πολύπλοκοι αλγόριθμοι, οι οποίοι ελέγχουν την πρόσβαση του κάθε χρήστη στο πεδίο του χρόνου χωρίς να εγγυώνται ένα σταθερό ρυθμό μετάδοσης.

Η πολυπλεξία στο πεδίο του χρόνου μπορεί να χρησιμοποιηθεί και για την διάκριση μεταξύ των δύο διαφορετικών κατευθύνσεων μεταφοράς δεδομένων μιας αμφίδρομης σύνδεσης π.χ. για την ταυτόχρονη πρόσβαση του σταθμού βάσης προς τον χρήστη καθώς και το αντίστροφο με την χρήση δύο διαφορετικών χρονοσχημάτων.

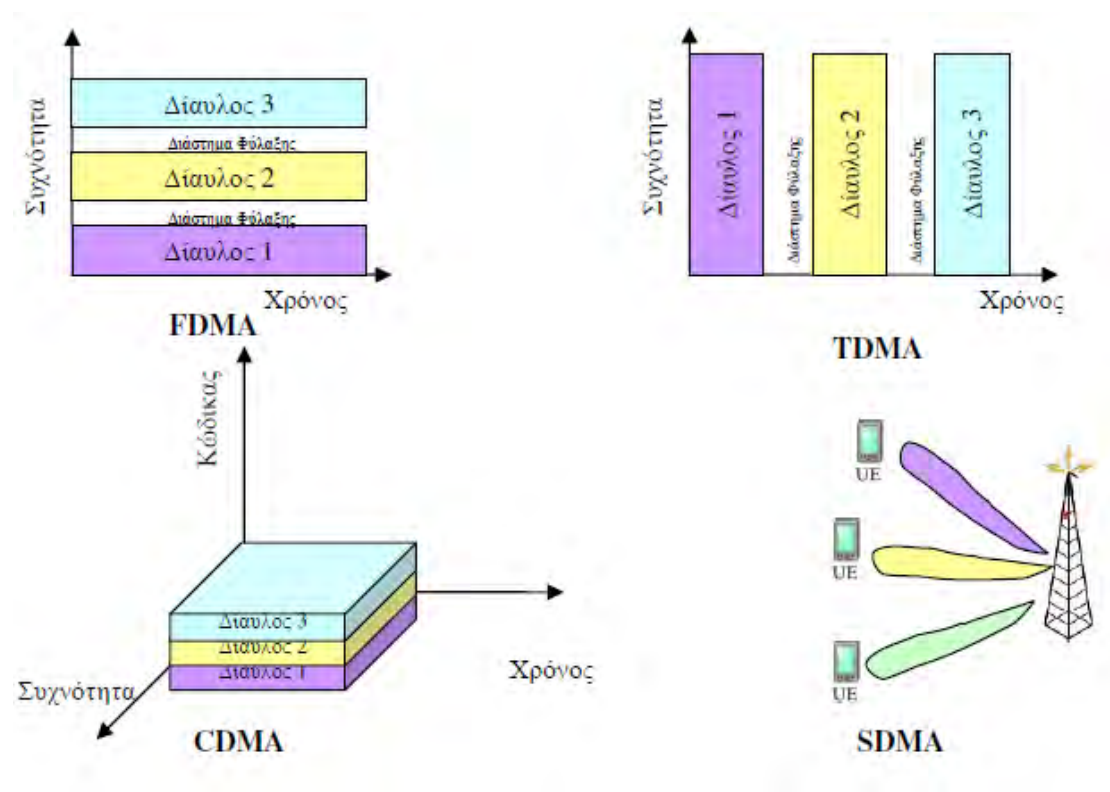
1.4.3 Πολλαπλή Πρόσβαση Κώδικα (Code Division Multiple Access, CDMA)

Η πολλαπλή πρόσβαση κώδικα (Code Division Multiple Access, CDMA) διακρίνει τα διαφορετικά φυσικά κανάλια ή/και τους διαφορετικούς χρήστες στο πεδίο των κωδικών δηλ. η πληροφορία του κάθε χρήστη κωδικοποιείται με διαφορετικούς κώδικες που είναι ορθογώνιοι μεταξύ τους. Η ορθογωνιότητα μεταξύ των διαφορετικών κωδικών σημαίνει ότι η συνάρτηση αυτοσυσχέτισης (autocorrelation) τους είναι υψηλή ενώ η συνάρτηση ετεροσυσχέτισης (cross-correlation) λαμβάνει χαμηλές ή μηδενικές τιμές. Η ανεξαρτησία των κωδικών και της κωδικοποιημένης πληροφορίας επιτρέπει στον δέκτη να μπορεί να διακρίνει τον κάθε χρήστη εφαρμόζοντας τον κωδικό του χρήστη, έστω και αν πολλοί διαφορετικοί χρήστες εκπέμπουν ταυτόχρονα στις ίδιες συχνότητες. Ένα φυσικό ανάλογο της λειτουργίας του CDMA είναι το παράδειγμα μιας αίθουσας αεροδρομίου, όπου άνθρωποι διαφορετικών εθνικοτήτων μιλούν ταυτόχρονα σε διαφορετικές γλώσσες. Εάν εισέλθει κάποιος στην αίθουσα μπορεί, ανάμεσα στις παράλληλες ομιλίες, να αντιληφθεί την ομιλία που διεξάγεται στη γλώσσα του. Η CDMA τεχνική χρησιμοποιείται στα δίκτυα κυψελωτής τηλεφωνίας 2ης γενιάς των ΗΠΑ καθώς και στα δίκτυα 3ης γενιάς WCDMA-UMTS και στα ασύρματα τοπικά δίκτυα 802.11.

1.4.4 Πολλαπλή Πρόσβαση Χώρου (Space Division Multiple Access, SDMA)

Η πολλαπλή πρόσβαση χώρου λειτουργεί απονέμοντας ένα χωροταξικά διακριτό χώρο στους χρήστες κινητών επικοινωνιών. Το πλέον τυπικό σενάριο αυτής της περίπτωσης

δεν ανήκει στο πεδίο των κινητών επικοινωνιών και αφορά στις εφαρμογές ραδιοφωνίας και τηλεόρασης. Σε αυτές τις περιπτώσεις ο χρήστης (τηλεοπτική / ραδιοφωνική εταιρεία) λειτουργεί με αποκλειστικότητα (συνήθως στην συχνότητα εκπομπής) σε μια περιοχή. Σε αρκετή απόσταση, όταν το σήμα ενός χρήστη έχει εξασθενήσει και δεν δημιουργούνται παρεμβολές, η αποκλειστικότητα της χρήσης του χώρου μπορεί να αποδοθεί σε άλλον χρήστη. Ένα σενάριο, το οποίο προσιδιάζει στα δίκτυα των κινητών επικοινωνιών είναι η περίπτωση των κυψελωτών δικτύων. Σε αυτά τα δίκτυα μια ομάδα χρηστών ανήκει και συνδέεται μέσω του σταθμού βάσης μιας κυψέλης διακρινόμενη από τους χρήστες των υπόλοιπων κυψελών. Στην πράξη η SDMA τεχνική δεν χρησιμοποιείται αυτόνομα αλλά σε συνδυασμό με τις τεχνικές FDMA, TDMA, CDMA.



2 Εισαγωγή σε Θέματα Ποιότητας Υπηρεσιών (Quality of Service)

2.1 Εισαγωγή

Για ένα ασύρματο δίκτυο η διατήρηση της απαιτούμενης Ποιότητας Υπηρεσιών (QoS) είναι δυσκολότερη από ό,τι σε ένα ενσύρματο δίκτυο. Αυτό συμβαίνει καθώς ένας από τους κυριότερους πόρους του συστήματος, οι ραδιοσυχνότητες είναι περιορισμένες. Τα νέα συστήματα διαμόρφωσης, κωδικοποίησης ή πρόσβασης επιτρέπουν την καλύτερη δυνατή αξιοποίηση των πόρων, παρόλα αυτά οι βελτιώσεις αυτές δυσκολεύονται να συμβαδίσουν με την αύξηση των χρηστών και με την εκρηκτική ανάπτυξη της ζήτησης για εφαρμογές μεγάλου εύρους ζώνης. Ένα ακόμα πρόβλημα είναι ότι οι χρήστες κινητών δικτύων δεν συνδέονται μέσω ενός σημείου σταθερής προσάρτησης λόγω της κινητικότητας τους. Άρα, το σύστημα πρέπει να ελαχιστοποιεί τις απρόβλεπτες αποσυνδέσεις κατά την μετακίνηση των χρηστών.

Με την εμφάνιση των δικτύων 3G και 4G η Ποιότητα Υπηρεσίας (QoS) πρέπει να παρέχεται τόσο στη φωνή όσο και στα δεδομένα. Και σε αυτά δίνεται προτεραιότητα στις υπηρεσίες φωνής, καθώς θεωρείται ως η κύρια υπηρεσία. Οι υπηρεσίες δεδομένων αποτελούνται από κείμενο και πολυμέσα και είναι λιγότερο ευαίσθητες στην καθυστέρηση μετάδοσης αλλά εξαρτώνται μεγαλύτερους ρυθμούς μετάδοσης και μικρότερο ρυθμό απωλειών. Οι μεγαλύτερες προκλήσεις, στην Ποιότητα Υπηρεσιών σε ασύρματα κυψελωτά δίκτυα, είναι ο ποικίλος ρυθμός των χαρακτηριστικών του κάθε καναλιού, ο καταμερισμός του εύρους ζώνης, τα επίπεδα ανοχής σε σφάλματα και η υποστήριξη ανάμεσα σε ετερογενή δίκτυα.

Η εξασφάλιση της Ποιότητα Υπηρεσιών (QoS) σε κάθε επίπεδο (Φυσικό, MAC, IP, TCP και Εφαρμογής), έχει ενδεχομένως τους δικούς της ανεξάρτητους μηχανισμούς. Για να είναι το δίκτυο ευέλικτο και ανεκτικό σε θέματα Ποιότητας Υπηρεσιών (QoS), πρέπει να ικανοποιούνται τα κριτήρια κάθε επιπέδου. Μια ακόμη πρόκληση είναι η αποτελεσματική χρήση του φάσματος καθώς η διαθεσιμότητά του είναι περιορισμένη. Η κατανομή του εύρους ζώνης παίζει σημαντικό ρόλο και θα πρέπει να μοιράζεται με αποδοτικό τρόπο και επιπλέον το πλεονάζον εύρος ζώνης δεν θα πρέπει να ξοδεύεται ή να παραμένει αχρησιμοποίητο. Ορισμένα συστήματα, όπως το σύστημα επαναδιαπραγμάτευσης εξαλείφουν αυτό το θέμα με την κατανομή της εναπομένουσας εύρους ζώνης σε κλάσεις μικρότερης προτεραιότητας.

2.2 Τάξεις Ποιότητας Υπηρεσιών (QoS Classes)

Γενικά μπορεί να διαχωριστούν τέσσερις τάξεις κίνησης (Traffic Classes), όπως φαίνονται και στον παρακάτω πίνακα.

Τάξεις	Conversational	Streaming	Interactive	Background
Βασικά χαρακτηριστικά	-Διασφάλιση της χρονικής συσχέτισης μεταξύ των οντοτήτων μιας ροής (stream) -Βασισμένο στην αντίληψη του χρήστη -Σε πραγματικό χρόνο	-Διασφάλιση της χρονικής συσχέτισης μεταξύ των οντοτήτων μιας ροής (stream) -Σε πραγματικό χρόνο	-Καθορισμένα χρονικά όρια απόκρισης -Διασφάλιση της ακεραιότητας των δεδομένων	-Μη καθορισμένα χρονικά όρια απόκρισης -Διασφάλιση της ακεραιότητας των δεδομένων
Παραδείγματα εφαρμογών	φωνή, video-κλήση, video παιχνίδια	multimedia streaming	web browsing, network games	background download of e-mails
Σχετικές απαιτήσεις QoS	Χαμηλό jitter, χαμηλή καθυστέρηση	Χαμηλό jitter	Χαμηλή καθυστέρηση, χαμηλό BER	Χαμηλό BER

- Conversational Class

Η τάξη αυτή περιλαμβάνει την απλή τηλεφωνία. Αυτό που διαφαίνεται από τις εξελίξεις είναι ότι ακόμα και η τηλεφωνία θα διεξάγεται με την χρήση του πρωτοκόλλου IP και έτσι θα έχει την μορφή του Voice over IP (VoIP) ή ακόμα και της βίντεο-συνδιάσκεψης. Οι εφαρμογές πραγματικού χρόνου απαιτούν μικρούς χρόνους μετάδοσης, για να πραγματοποιηθεί μια συνομιλία ικανοποιητικά. Επίσης οι διάφορες ροές δεδομένων (όπως είναι για παράδειγμα η ροή του ήχου και της εικόνας) θα πρέπει να είναι χρονικά συσχετισμένες και να είναι συγχρονισμένες. Οι μεγάλοι χρόνοι καθυστέρησης δεν είναι αποδεκτοί σε αυτή την τάξη, καθώς κάτι τέτοιο θα σήμαινε δραματική υποβάθμιση της ποιότητας.

- Streaming Class

Και αυτή η τάξη αφορά σε ροές δεδομένων πραγματικού χρόνου, όπως είναι οι ροές ήχου ή βίντεο. Η ροή των δεδομένων έχει κατεύθυνση από έναν εξυπηρετητή προς ένα χρήστη που έχει ζητήσει την αντίστοιχη υπηρεσία. Έτσι η επικοινωνία είναι σχεδόν μονόδρομη, καθώς υπάρχει ελάχιστη αλληλεπίδραση μεταξύ των δύο τελικών άκρων. Στην πράξη ο εξυπηρετητής στέλνει τα δεδομένα και ο χρήστης προβαίνει σε ελάχιστες ενέργειες όπως είναι η απλή επιλογή της υπηρεσίας, η αναπαραγωγή της και το σταμάτημά της. Έτσι δεν έχουμε μεγάλη ζήτηση για χαμηλή καθυστέρηση, παρόλο που αυτή είναι

πάντοτε επιθυμητή. Για μια εφαρμογή όπως είναι η παρακολούθηση μιας ταινίας, μία ροή μπορεί να συμπεριλαμβάνει διάφορα στοιχεία, καθώς είναι δυνατή η χρήση διαφορετικών υπό-ροών ώστε να μεταφερθούν χωριστά ο ήχος, η εικόνα και οι υπότιτλοι. Για να επιτευχθεί όμως η ορθή αναπαραγωγή στον τελικό χρήστη, πρέπει να υπάρχει συγχρονισμός όλων αυτών των στοιχείων και εξαιτίας αυτού δημιουργείται η ανάγκη για χρονική συσχέτιση μεταξύ των ροών. Οι εφαρμογές streaming είναι ασύμμετρες σε μεγάλο βαθμό και γι' αυτό τυπικά αντέχουν περισσότερη καθυστέρηση απ' ότι άλλες συμμετρικές conversational υπηρεσίες. Αυτό επίσης σημαίνει ότι είναι πιο ανεκτικές στην παραμόρφωση χρονισμού (jitter) κατά την μετάδοση.

- **Interactive Class**

Οι εφαρμογές αυτής της τάξης απαιτούν αλληλεπίδραση μεταξύ του εξυπηρετητή και του χρήστη, όπως είναι η περιήγηση ή οι online συναλλαγές. Επίσης περιλαμβάνονται και εφαρμογές στις οποίες υπάρχει αλληλεπίδραση μεταξύ μηχανών, όπως για παράδειγμα η διαρκής ανανέωση στοιχείων από την βάση δεδομένων ενός απομακρυσμένου μηχανήματος. Οι κύριες ενέργειες βασίζονται στον μηχανισμό ερωτώ-αποκρίσεων, και έτσι οι χρόνοι μετάδοσης θα πρέπει να είναι σύντομοι. Όταν γίνεται μια αίτηση, ένα χρονόμετρο τίθεται σε λειτουργία και περιμένει την απάντηση. Έτσι όσο μικρότερος είναι ο χρόνος αυτός, τόσο καλύτερη είναι η προσφερόμενη ποιότητα. Σε αυτή την κατηγορία ανήκουν και τα απαιτητικά multiplayer computer games.

- **Background Class**

Πρόκειται για χρήση δεδομένων στο παρασκήνιο από εφαρμογές όπως είναι η αποστολή e-mail, SMS, η λήψη βάσεων δεδομένων (downloading) και άλλες, για τις οποίες δεν απαιτείται άμεση ενέργεια από τον χρήστη. Η καθυστέρηση μπορεί να είναι δευτερόλεπτα ή και λεπτά. Αυτή η κλάση χαρακτηρίζεται από το γεγονός ότι δεν υπάρχει κάποιο αυστηρό χρονικό διάστημα μέσα στο οποίο ο προορισμός να περιμένει τα δεδομένα.

2.3 Μετρήσεις Ποιότητας Υπηρεσίας

Σε αυτή την ενότητα παρουσιάζονται οι αντικειμενικοί και συγκρίσιμοι δείκτες επίδοσης (Key Performance Indicators, KPIs) παρεχόμενων προς τους χρήστες υπηρεσιών κινητών επικοινωνιών, ώστε αφενός οι πάροχοι να έχουν ένα μέτρο για την ποιότητα υπηρεσιών που παρέχουν και αφετέρου οι τελικοί χρήστες να διευκολύνονται στη σύγκριση της ποιότητας υπηρεσιών που παρέχονται από διαφορετικούς παρόχους καθώς και στην πιστοποίηση της ποιότητας υπηρεσιών που ήδη τους παρέχονται. Διακρίνουμε δύο κύριες μεθόδους της ποιότητας μιας προσφερόμενης υπηρεσίας: τους δείκτες ποιότητας προς μέτρηση και την καμπάνια μετρήσεων, που συμπεριλαμβάνει τον μετρητικό εξοπλισμό, τις δοκιμές και τα σενάρια μέτρησης.

Ως Δείκτης Ποιότητας (Quality Indicator) ορίζεται ένα σύνολο μεγεθών, το μέτρο των οποίων αποτιμάται για τη εξαγωγή στοιχεία της ποιότητας μιας παρεχόμενης υπηρεσίας κινητών επικοινωνιών. Ως Καμπάνια Μετρήσεων (Measurement Campaign) ορίζεται το σύνολο των ενεργειών που απαιτούνται για τη διεξαγωγή των μετρήσεων και την παρουσίαση των αποτελεσμάτων των δεικτών ποιότητας υπηρεσιών. Κατά την διάρκεια μιας καμπάνιας μέτρησης συλλέγεται ένας μεγάλος όγκος δεδομένων όπως είναι η ισχύς του σήματος, η ποιότητα του σήματος, παρεμβολές, γεγονότα κλήσεων (dropped/blocked calls) και τα στατιστικά στοιχεία αυτών (call statistics), πληροφορίες ποιότητας υπηρεσιών (QoS statistics) και γενικότερες πληροφορίες του δικτύου ασύρματης πρόσβασης όπως διαπομπές (handovers) και λοιπές παράμετροι (neighbor cells). Η μετρητική καμπάνια λαμβάνει χώρα σε διάφορα περιβάλλοντα τα οποία κατηγοριοποιούνται σε α) στατικές μετρήσεις (static/ hot spots) για εξωτερικούς και εσωτερικούς χώρους όπως πλατείες πόλεων, σημεία τουριστικού ενδιαφέροντος, αεροδρόμια, σταθμοί τρένων και μετρό, εμπορικά κέντρα, συγκροτήματα γραφείων και χώροι εκθέσεων και σε β) μετρήσεις εν κινήσει με όχημα (drive testing) για αστικές περιοχές, μητροπολιτικά κέντρα, πόλεις, εθνικές οδοί, επαρχία (επαρχιακοί δρόμοι).

Οι σημαντικοί δείκτες ποιότητας που μετρώνται και παρουσιάζονται για τις υπηρεσίες δικτύων κινητών επικοινωνιών είναι οι ακόλουθοι:

- Δείκτες υπηρεσίας τηλεφωνίας

- Πιθανότητα εμπλοκής κλήσης τηλεφωνίας
- Πιθανότητα διακοπής κλήσης τηλεφωνίας
- Ποιότητα φωνής
- Χρόνος αποκατάστασης κλήσης φωνής
- Δείκτες υπηρεσίας βιντεοτηλεφωνίας
 - Πιθανότητα εμπλοκής κλήσης βιντεοτηλεφωνίας
 - Πιθανότητα διακοπής κλήσης βιντεοτηλεφωνίας
 - Ποιότητα φωνής βιντεοτηλεφωνίας
 - Ποιότητα βίντεο βιντεοτηλεφωνίας
 - Χρόνος αποκατάστασης κλήσης βιντεοτηλεφωνίας
- Δείκτες ευρυζωνικών υπηρεσιών δεδομένων
 - Πιθανότητα αποτυχίας μεταφοράς δεδομένων http
 - Μέσος ρυθμός μεταφοράς δεδομένων http
 - Μέσος ρυθμός μεταφοράς δεδομένων ftp upload
 - Μέσος ρυθμός μεταφοράς δεδομένων ftp download
- Δείκτες ανεξάρτητοι της υπηρεσίας
 - Διαθεσιμότητα δικτύου – ραδιοκάλυψη

Η παρούσα εργασία είναι αφιερωμένη στην μεταφορά δεδομένων από και προς έναν ftp server. Ο χρήστης ορίζει τις παραμέτρους του ftp server από την εφαρμογή, ενώ κάθε δειγματοληψία που διενεργείται αφορά στο μέσο ρυθμό μεταφοράς δεδομένων μεταξύ συσκευής και ftp server.

2.3.1 Δείκτες Ανεξάρτητοι της υπηρεσίας

- Διαθεσιμότητα Δικτύου – Ραδιοκάλυψη

Η διαθεσιμότητα ενός δικτύου κινητών επικοινωνιών είναι ο κατεξοχήν παράγοντας για την προσφορά οποιασδήποτε υπηρεσίας μέσω του δικτύου αυτού και εξαρτάται άμεσα από τη ραδιοκάλυψη στο σημείο μέτρησης. Με τον όρο ραδιοκάλυψη μιας γεωγραφικής περιοχής εννοούμε ότι το επίπεδο της έντασης του πεδίου στην εν λόγω περιοχή είναι ίσο ή μεγαλύτερο από μία συγκεκριμένη τιμή (κατώφλι) η οποία εξασφαλίζει την επικοινωνία μεταξύ κινητού τελικού χρήστη και του δικτύου. Στην εφαρμογή, μία από τις παραμέτρους που καταγράφουμε είναι η ισχύς του σήματος, με

αποτέλεσμα ένα τεστ να μας δίνει και μια εικόνα από την διαθεσιμότητα του δικτύου στην περιοχή.

$$\text{Διαθεσιμότητα ραδιοδίκτυου (\%)} = \frac{\text{συνολικός αριθμός σημείων μέτρησης στα οποία ικανοποιούνται οι παράμετροι ραδιοκάλυψης}}{\text{συνολικός αριθμός σημείων μέτρησης}} * 100$$

2.3.2 Δείκτες Υπηρεσίας Τηλεφωνίας

- Πιθανότητα εμπλοκής κλήσης τηλεφωνίας

Ο αριθμός των κλήσεων όπου παρουσιάστηκε εμπλοκή, προς το συνολικό αριθμό των κλήσεων, ενώ υπάρχει διαθεσιμότητα του δικτύου (ικανοποιητική ραδιοκάλυψη), είναι ο δείκτης που αφορά στην εμπλοκή κλήσεων (blocked calls). Ποιοτικά ο εν λόγω δείκτης χαρακτηρίζει την προσβασιμότητα στην υπηρεσία φωνής ενός δικτύου.

$$\text{Πιθανότητα εμπλοκής κλήσεων (\%)} = \frac{\text{αριθμός κλήσεων όπου υπήρξε εμπλοκή}}{\text{συνολικός αριθμός κλήσεων}} * 100$$

- Πιθανότητα διακοπής κλήσης τηλεφωνίας

Η πιθανότητα διακοπής κλήσης είναι η πιθανότητα τερματισμού μιας επιτυχημένης προσπάθειας κλήσης για οποιοδήποτε λόγο εκτός από σκόπιμο τερματισμό του καλούντος ή του καλούμενου, ενώ υπάρχει διαθεσιμότητα του δικτύου (ικανοποιητική ραδιοκάλυψη).

$$\text{Πιθανότητα διακοπής κλήσης (\%)} = \frac{\text{τερματισμένες χωρίς πρόθεση κλήσεις}}{\text{συνολικές επιτυχημένες κλήσεις}} * 100$$

- Ποιότητα Φωνής

Ο υπολογισμός της ποιότητας φωνής γίνεται βάσει των ολοκληρωμένων κλήσεων, όπου ο τερματισμός τους έγινε από τον τελικό χρήστη. Η μέτρηση της «απ' άκρο σ' άκρο» ποιότητας γίνεται με χρήση της κλίμακας MOS_LQO, η οποία ποσοτικοποιεί την προσπάθεια που χρειάζεται για να γίνει κατανοητή μια ομιλία κατά τη διάρκεια της κλήσης. Ο δείκτης MOS παίρνει τιμές από 1 έως 5 με 1 να σημαίνει ότι δεν υπήρξε ουσιαστικά επικοινωνία και το 5 ότι η επικοινωνία ήταν άριστη.

$$\begin{aligned} \text{Ποιότητα φωνής σε επίπεδο κλήσης [received A side]} &= f(MOS_{LQO}) \\ \text{Ποιότητα φωνής σε επίπεδο κλήσης [received B side]} &= f(MOS_{LQO}) \end{aligned}$$

- Χρόνος αποκατάστασης κλήσης φωνής

Αφορά στο χρόνο από τη συμπλήρωση της πληροφορίας διεύθυνσης (δηλ. τον αριθμό τηλεφώνου) μέχρι τη λήψη ειδοποίησης εγκατάστασης κλήσης (call set-up notification).

$$\text{Χρόνος εγκατάστασης κλήσης [s]} = t_{\text{connection established}} - t_{\text{customer presses send button on UE}}$$

2.3.3 Δείκτες Ευρυζωνικών Υπηρεσιών Δεδομένων

- Πιθανότητα αποτυχίας μεταφοράς δεδομένων http

Το ποσοστό των μη ολοκληρωμένων προσπαθειών μεταφοράς δεδομένων σε επίπεδο HTTP ως προς τον συνολικό αριθμό προσπαθειών. Η γενικευμένη εξίσωση υπολογισμού του συγκεκριμένου δείκτη είναι η ακόλουθη:

$$\text{Πιθανότητα αποτυχίας μεταφοράς δεδομένων HTTP (\%)} = \frac{\text{μη ολοκληρωμένες προσπάθειες μεταφοράς δεδομένων HTTP}}{\text{επιτυχώς αρχικοποιημένες προσπάθειες μεταφοράς}}$$

- Μέσος ρυθμός μεταφοράς δεδομένων http

Ο μέσος ρυθμός μεταφοράς δεδομένων μετρημένος καθ' όλη την διάρκεια της κλήσης, μετά την επιτυχημένη εγκατάσταση σύνδεσης δεδομένων. Η γενικευμένη εξίσωση υπολογισμού του συγκεκριμένου δείκτη είναι η ακόλουθη:

$$\text{Μέσος ρυθμός δεδομένων HTTP [kbps]} = \frac{\text{όγκος δεδομένων που μεταφέρθηκαν}}{t_{\text{content received}} - t_{\text{dial-up connection initiated}}}$$

- Μέσος ρυθμός μεταφοράς δεδομένων ftp upload και Μέσος ρυθμός μεταφοράς δεδομένων ftp download

Ο δείκτης Μέσος ρυθμός μεταφοράς δεδομένων ftp upload και ftp download εκφράζει τον μέσο ρυθμό μεταφοράς δεδομένων σε προκαθορισμένο χρονικό διάστημα Δt_d με βάση το πρωτόκολλο ftp, μετά την επιτυχημένη αποκατάσταση σύνδεσης δεδομένων. Ορίζεται ως το πηλίκο του όγκου των δεδομένων που μεταφέρθηκαν προς το χρονικό διάστημα Δt_d και μετράται σε Kbit/s σε ακέραιες τιμές.

$$\text{Μέσος ρυθμός δεδομένων FTP [kbps]} = \frac{\text{όγκος δεδομένων που μεταφέρθηκαν}}{t_{\text{content received}} - t_{\text{dial-up connection initiated}}}$$

Πριν ένα αρχείο μεταφερθεί μέσω του FTP πρωτοκόλλου, μια σύνδεση καναλιού ελέγχου πρέπει να εγκατασταθεί προς τον FTP server. Το κανάλι ελέγχου χρησιμοποιείται στην ανταλλαγή FTP εντολών μεταξύ πελάτη (κινητής συσκευής Android) και του server. Εφαρμόζονται εντολές όπως σύνδεση στον λογαριασμό χρήστη (login), αλλαγή φακέλου και επιλογή λειτουργίας FTP. Στην περίπτωση της λειτουργίας FTP GET (download) ή FTP PUT (upload), το κανάλι δεδομένων εγκαθίσταται με σκοπό την μεταφορά των δεδομένων. Το κανάλι ελέγχου διατηρείται ανοιχτό κατά την διάρκεια μεταφοράς αρχείου με σκοπό την ανταλλαγή άλλων εντολών.

3 To Android OS και η εφαρμογή Mobile Signal Tester

3.1 Εισαγωγή

Code name	Version	API level
Nougat	7.0	API level 24
Marshmallow	6.0	API level 23
Lollipop	5.1	API level 22
Lollipop	5.0	API level 21
KitKat	4.4 - 4.4.4	API level 19
Jelly Bean	4.3.x	API level 18
Jelly Bean	4.2.x	API level 17
Jelly Bean	4.1.x	API level 16
Ice Cream Sandwich	4.0.3 - 4.0.4	API level 15, NDK 8
Ice Cream Sandwich	4.0.1 - 4.0.2	API level 14, NDK 7
Honeycomb	3.2.x	API level 13
Honeycomb	3.1	API level 12, NDK 6
Honeycomb	3.0	API level 11
Gingerbread	2.3.3 - 2.3.7	API level 10
Gingerbread	2.3 - 2.3.2	API level 9, NDK 5
Froyo	2.2.x	API level 8, NDK 4
Eclair	2.1	API level 7, NDK 3
Eclair	2.0.1	API level 6
Eclair	2.0	API level 5
Donut	1.6	API level 4, NDK 2
Cupcake	1.5	API level 3, NDK 1
(no code name)	1.1	API level 2
(no code name)	1.0	API level 1

Το Android είναι ένα λειτουργικό σύστημα για κινητά τηλέφωνα και όχι μόνο. Είναι βασισμένο στο λειτουργικό Linux και αναπτύσσεται από την Google. Πρόκειται για λογισμικό ανοιχτού κώδικα (Open Source Software, OSS), το οποίο σημαίνει ότι ο πηγαίος κώδικας διατίθεται ελεύθερα σε όσους ζητούν να τον εξετάσουν ή να τον τροποποιήσουν για να τον αξιοποιήσουν σε άλλες εφαρμογές. Επιτρέπει στους κατασκευαστές λογισμικού να γράφουν κώδικα με την χρήση της γλώσσας προγραμματισμού Java. Είναι κατά κύριο λόγο σχεδιασμένο για συσκευές με οθόνη αφής, όπως τα smartphones και τα tablets. Ωστόσο, χρησιμοποιείται ακόμη και σε κονσόλες παιχνιδιών, ψηφιακές φωτογραφικές μηχανές και ηλεκτρονικούς υπολογιστές. Με την εξέλιξή του τα τελευταία χρόνια, έχει χρησιμοποιηθεί και σε

τηλεοράσεις, ρολόγια, ακόμα και αυτοκίνητα. Το 2008 κυκλοφόρησε το πρώτο Android smartphone. Κάθε έκδοση του Android φέρει και μία κωδική ονομασία ενός γλυκού όπως Cupcake, Donut, ή Eclair, ενώ από τις εκδόσεις αυτές εκείνη που εισήγαγε την υποστήριξη πιο ανεπτυγμένων λειτουργιών ήταν σίγουρα η 2.0 στην οποία ενσωματώθηκε η υποστήριξη multitouch, HTML 5, text-to-speech και η δυνατότητα προχωρημένων αναζητήσεων.

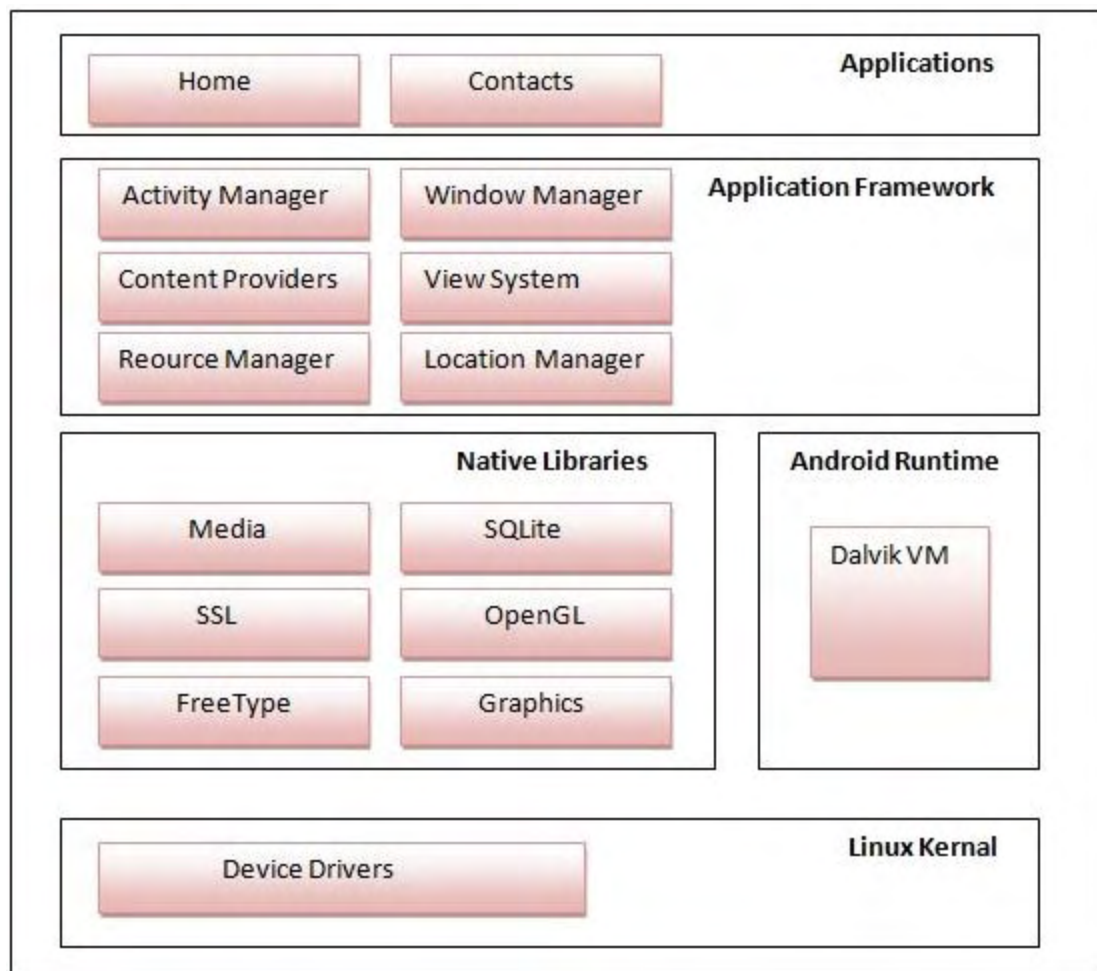
3.2 Αρχιτεκτονική του Android

Το Android χωρίζεται σε πέντε κατηγορίες:

1. Πυρήνας Linux (Linux kernel): Αυτός είναι ο πυρήνας στον οποίο βασίζεται το Android και βρίσκεται στο χαμηλότερο επίπεδο. Πρόκειται για τον πυρήνα Linux, για βασικές υπηρεσίες συστήματος όπως ασφάλεια, διαχείριση μνήμης, διαχείριση διεργασιών, στοίβα δικτύου, και οδηγούς συσκευών. Ο πυρήνας λειτουργεί επίσης ως ένα ενδιάμεσο επίπεδο αφαίρεσης μεταξύ της στοίβας λογισμικού και του υλικού.
2. Βιβλιοθήκες (Native libraries, middleware): Πάνω από τον πυρήνα του Linux, βρίσκονται οι βιβλιοθήκες, όπως το WebKit, OpenGL, FreeType, SQLite, Media, C βιβλιοθήκη χρόνου εκτέλεσης (libc) κ.λπ. Η βιβλιοθήκη WebKit είναι υπεύθυνη για την υποστήριξη του προγράμματος περιήγησης, η SQLite είναι για τη βάση δεδομένων, η FreeType για την υποστήριξη γραμματοσειρών, η Media για την αναπαραγωγή και εγγραφή ήχου και βίντεο.
3. Χρόνος Εκτέλεσης(Android Runtime): Στο Android runtime, υπάρχουν οι βασικές βιβλιοθήκες και η DVM (Dalvik Virtual Machine), η οποία είναι υπεύθυνη για την εκτέλεση εφαρμογής Android. Η DVM μοιάζει με την JVM, αλλά έχει βελτιστοποιηθεί για κινητές συσκευές. Καταναλώνει λιγότερη μνήμη και παρέχει γρήγορη απόδοση.
4. Πλαίσιο Εφαρμογών (Application Framework): Παρέχοντας μια ανοικτή πλατφόρμα ανάπτυξης, το Android προσφέρει στους προγραμματιστές τη δυνατότητα να κατασκευάσουν πλούσιες και καινοτόμες εφαρμογές. Οι προγραμματιστές δύνανται να εκμεταλλευτούν πλήρως το hardware της συσκευής, να έχουν πρόσβαση σε υπηρεσίες εντοπισμού θέσης, να τρέξουν υπηρεσίες στο background, να θέσουν χρονοδιακόπτες για εμφάνιση

ειδοποιήσεων και πολλά άλλα. Επίσης έχουν πλήρη πρόσβαση στο ίδιο πλαίσιο από APIs (Application Programming Interface) που έχουν οι βασικές εφαρμογές του Android. Η αρχιτεκτονική είναι διαμορφωμένη με τέτοιο τρόπο που κάθε εφαρμογή μπορεί να χρησιμοποιήσει τις δυνατότητες μιας άλλης και επίσης με τρόπο που δίνει την δυνατότητα στον χρήστη να αλλάξει τα συστατικά κάθε εφαρμογής. Κάτω από το πλαίσιο των εφαρμογών υπάρχει ένα σύστημα από υπηρεσίες και συστήματα που συγκροτούν τη βάση της αρχιτεκτονικής των Android apps και απαρτίζονται ως εξής:

- Activity Manager: Διαχειριστή δραστηριοτήτων ο οποίος διαχειρίζεται τον κύκλο ζωής (life cycle) των εφαρμογών.
 - Views: Ένα σύνολο από γραφικά στοιχεία για τη δημιουργία γραφικού περιβάλλοντος (User Interface - UI). συμπεριλαμβανομένων λιστών (lists), πλεγμάτων (grids), κουτιών κειμένου (text boxes), κουμπιών (buttons) και άλλων.
 - Notification Manager: Διαχειριστής ειδοποιήσεων ο οποίος επιτρέπει την προβολή ειδοποιήσεων στην μπάρα κατάστασης (status bar).
 - Content Manager: Διαχειριστής περιεχομένου ο οποίος επιτρέπει στις εφαρμογές την πρόσβαση σε δεδομένα άλλων εφαρμογών ή τον διαμοιρασμό των δικών τους δεδομένων με άλλες εφαρμογές.
 - Resource Manager: Διαχειριστής πόρων για την πρόσβαση στους πόρους, όπως strings, εικόνες, layout files. Υποστηρίζει συμβολοσειρές και γραφικά.
5. Εφαρμογές (Applications): Πάνω από το επίπεδο του πλαισίου εφαρμογών βρίσκονται οι εφαρμογές. Όλες οι εφαρμογές όπως, ηλεκτρονική αλληλογραφία, η αποστολή και λήψη SMS, η χρήση ημερολογίου, η απεικόνιση χαρτών και ο ταυτόχρονος εντοπισμός θέσης, εφαρμογές διαχείρισης επαφών, φυλλομετρητή για περιήγηση στο διαδίκτυο κ.α. κάνουν χρήση του πλαισίου εφαρμογών το οποίο χρησιμοποιεί τις βιβλιοθήκες και τον χρόνο εκτέλεσης των χαμηλότερων επιπέδων.



3.3 Περιγραφή Εφαρμογής Mobile Signal Tester

Η Εφαρμογή αποτελεί ένα εργαλείο που προσφέρει στο χειριστή τη δυνατότητα καταγραφής του ρυθμού μεταφοράς δεδομένων που μπορεί να επιτευχθεί από το δίκτυο. Για την υλοποίηση της εφαρμογής αυτής χρησιμοποιήθηκε η δυνατότητα του κινητού για τη μέτρηση της ισχύος του σήματος του δικτύου από το οποίο λαμβάνει υπηρεσίες, συνδυασμένη με την παροχή πληροφοριών εντοπισμού θέσης που δίνει το Android αλλά και η Google. Το πρόγραμμα που χρησιμοποιήθηκε για την ανάπτυξη της εφαρμογής είναι το Android Studio και η γλώσσα προγραμματισμού στην οποία υλοποιήθηκε η εφαρμογή είναι η Java. Η εφαρμογή έχει σαν βασικό σκοπό τη διενέργεια ελέγχων της δυνατότητας αποστολής και λήψης δεδομένων από την κινητή συσκευή στο δίκτυο, την οποία και προβάλλει σε χάρτη αλλά και την αποθήκευση των μετρήσεων για ενδεχόμενη μετ' έπειτα επεξεργασία.

Οι πληροφορίες που αποθηκεύονται σε κάθε καταγραφή είναι το γεωγραφικό μήκος και πλάτος (Latitude / Longitude), η ισχύς του σήματος σε dBm, ο τύπος του δικτύου (Network Type όπως CDMA, EDGE, GPRS, HSDPA, HSPA+, UMTS), ένα «timestamp», που υποδεικνύει την στιγμή της δειγματοληψίας, ένας κωδικός για την τρέχουσα φάση του τεστ (1-upload, 2-download, 3-upload complete, 4-download complete) και ο MO της ταχύτητας μεταφοράς σε kbps, ανάμεσα σε δύο δειγματοληψίες είτε ο MO της συνολικής μεταφοράς του αρχείου αν είμαστε στο τέλος του download ή του upload του αρχείου.

Το αρχείο .txt που δημιουργείται αποθηκεύεται στο κινητό και είναι προσβάσιμο μέσω της εφαρμογής. Όπως θα δούμε παρακάτω, υπάρχει επίσης η δυνατότητα να σταλεί το αρχείο .txt ενός τεστ στον ftp server.

3.3.1 Διάρθρωση του κώδικα

Το Android παρέχει ένα πλαίσιο εφαρμογών (framework) που επιτρέπει την ανάπτυξη καινοτόμων εφαρμογών για φορητές συσκευές σε περιβάλλον Java. Το Android υποστηρίζει την ανάπτυξη εφαρμογών με την χρήση διεπαφών (APIs, Application Programming Interface). Μια διεπαφή προγραμματισμού εφαρμογών (API) είναι ένα σύνολο ορισμών διεργασιών, πρωτοκόλλων, και εργαλείων για τη δημιουργία λογισμικού και εφαρμογών.

Στο παρακάτω διάγραμμα παρουσιάζεται η ροή διεργασιών καθώς και οι κλάσεις που χρησιμοποιήθηκαν για την αποτύπωση του αποτελέσματος. Η εφαρμογή χωρίζεται σε 3 Activities: α. την MainActivity, β. την TestActivity και γ. την SettingsActivity.

Η MainActivity, διαβάζει από το σύστημα αρχείων τα ολοκληρωμένα tests και τα εμφανίζει σε μορφή λίστας. Ο χρήστης μπορεί είτε να 'ανοίξει' ένα υπάρχον test, ή να ξεκινήσει ένα καινούριο. Και οι δύο αυτές ενέργειες τον οδηγούν στην TestActivity.

Η TestActivity, αποτελείται από 4 tabs:

- Το πρώτο tab περιέχει τη λίστα με τα samples του επιλεγμένου test.
- Το δεύτερο περιέχει το χάρτη στον οποίο εμφανίζονται οι θέσεις των samples σε μορφή μονοπατιού.

- Το τρίτο περιέχει το γράφημα χρόνου-dBm
- Το τέταρτο περιέχει το γράφημα χρόνου-ταχύτητας μετάδοσης

Κατά τη διάρκεια ενός καινούριου test, τα περιεχόμενα των tabs ανανεώνονται με τα samples που δημιουργούνται.

Η SettingsActivity είναι προσβάσιμη και από τις δύο παραπάνω Activities και σ' αυτή μπορούμε να τροποποιήσουμε τις διάφορες ρυθμίσεις της εφαρμογής όπως τη διεύθυνση του FTP server, το διάστημα μεταξύ των samples κλπ.

3.3.2 Οι κλάσεις της εφαρμογής

Ακολουθεί σύντομη περιγραφή των κλάσεων της εφαρμογής. Γενικά ο κώδικας της εφαρμογής που ακολουθεί στο παράρτημα έχει επαρκή σχόλια στα αγγλικά για την λειτουργία κάθε μεθόδου.

com.nikmix.mobilesignaltester.homeactivity.FileInfoParser.java

Αυτή η κλάση διαβάζει τις δύο πρώτες γραμμές του αρχείου ενός test οι οποίες περιέχουν συνοπτικές πληροφορίες του test. Χρησιμοποιείται από την HomeActivity για να εμφανίσει τα αποθηκευμένα tests σε μορφή λίστας.

com.nikmix.mobilesignaltester.homeactivity.HomeActivity.java

Η κλάση-Activity η οποία αρχικοποιεί τη λίστα των tests και μέσω της οποίας μεταφερόμαστε στην TestActivity είτε για να δούμε κάποιο υπάρχον test είτε για να ξεκινήσουμε ένα καινούριο.

com.nikmix.mobilesignaltester.homeactivity.TestPreview.java

Η κλάση αυτή αποθηκεύει τις πληροφορίες σύνοψης ενός αποθηκευμένου test ώστε να μπορεί αυτό να εμφανιστεί στη λίστα μέσω των TestPreviewRecyclerViewAdapter και TestsListFragment.

com.nikmix.mobilesignaltester.homeactivity.TestPreviewRecyclerViewAdapter.java

a

Αυτός ο Adapter χρησιμοποιείται από την TestsListFragment για να εμφανίζει τα αποθηκευμένα tests. Παράλληλα αναλαμβάνει να προσθέσει onClickListener και onLongClickListener στα TestPreviews της λίστας ώστε να μεταβαίνουμε στο αντίστοιχο test ή να μας δίνεται η δυνατότητα να το διαγράψουμε ή να το κάνουμε upload στον FTP server.

com.nikmix.mobilesignaltester.homeactivity.TestsListFragment.java

Αυτή η κλάση Fragment φορτώνει τα αποθηκευμένα tests και τα περνάει ως όρισμα στον TestPreviewRecyclerViewAdapter καθώς τον δημιουργεί, ενώ τον θέτει ως Adapter στην RecyclerView που περιέχει.

com.nikmix.mobilesignaltester.settings.AppCompatPreferenceActivity.java

Αυτή η κλάση υπερβαίνει και προωθεί τις απαραίτητες μεθόδους της PreferenceActivity σε έναν AppCompatActivity τον οποίο περιέχει. Υπερβαίνεται από τη SettingsActivity η οποία είναι η Activity με τις ρυθμίσεις της εφαρμογής.

com.nikmix.mobilesignaltester.settings.SettingsActivity.java

Η κλάση αυτή υλοποιεί μια Activity με την οποία τροποποιούνται οι ρυθμίσεις της εφαρμογής. Οι ρυθμίσεις περιγράφονται από τρία αρχεία .xml και χωρίζονται σε δύο κατηγορίες: ρυθμίσεις σχετικές με τα tests και ρυθμίσεις σχετικές με τον FTP server.

com.nikmix.mobilesignaltester.testactivity.core.Coordinates.java

Η κλάση αυτή περιέχει δύο μεταβλητές υποδιαστολής ώστε να αποθηκεύουν τις συντεταγμένες ενός SignalSample. Είναι Serializable για να μπορεί να σώζεται προσωρινά σε ένα Bundle κατά την παύση της TestActivity.

com.nikmix.mobilesignaltester.testactivity.core.SignalSample.java

Η κλάση που περιέχει όλες τις πληροφορίες ενός δείγματος ενός test και που χρησιμοποιείται κατά την προβολή, φόρτωση και αποθήκευσή τους.

com.nikmix.mobilesignaltester.testactivity.datatransfer.DataTransferTask.java

Η AsyncTask που υλοποιεί τη μεταφορά αρχείων μέσω ενός FTP server. Κατά τη διάρκεια εκτέλεσής της υπολογίζει την εκάστοτε ταχύτητα μεταφοράς του τρέχοντος αρχείου ενώ κατά διαστήματα 'ενημερώνει' την TestActivity ως προς την πρόοδο της μεταφοράς του.

com.nikmix.mobilesignaltester.testactivity.datatransfer.FtpServer.java

Το interface που υλοποιούν οι κλάσεις SimulatedFtpServer και RealFtpServer και το οποίο χρησιμοποιεί η DataTransferTask για να πραγματοποιήσει τη μεταφορά των αρχείων (πραγματικά ή εικονικά).

Χρησιμοποιεί το interface ChunkFinishedListener για να μπορέσει να υπολογιστεί η τρέχουσα ταχύτητα μεταφοράς του αρχείου καθώς και η πρόοδος μεταφοράς του, από την DataTransferTask.

com.nikmix.mobilesignaltester.testactivity.datatransfer.RealFtpServer.java

Η πρώτη κλάση που υλοποιεί το interface FtpServer και η οποία χρησιμοποιεί βιβλιοθήκες της Apache για την πραγματική μεταφορά των αρχείων από και προς τον FTP server.

com.nikmix.mobilesignaltester.testactivity.datatransfer.SimulatedFtpServer.java

Η δεύτερη κλάση που υλοποιεί το interface FtpServer, η οποία απλώς 'περιμένει' ανά τακτά χρονικά διαστήματα καλώντας τη μέθοδο του ChunkFinishedListener ώστε να φαίνεται στο χρήστη ότι κατεβαίνει/ανεβαίνει το αρχείο με συγκεκριμένη ταχύτητα (άλλη για το download, άλλη για το upload).

com.nikmix.mobilesignaltester.testactivity.graph.GraphFragment.java

Αυτό το Fragment υλοποιεί τις γραφικές παραστάσεις των dBm του σήματος και της ταχύτητας μεταφοράς των αρχείων. Χρησιμοποιεί μια εξωτερική βιβλιοθήκη γραφημάτων ενώ δημιουργούνται 2 instances του Fragment τα οποία προστίθενται σε διαφορετικά tabs της TestActivity.

com.nikmix.mobilesignaltester.testactivity.io.FileWriterParser.java

Η κλάση αυτή αναλαμβάνει να διαβάσει ή να αποθηκεύσει ένα test σε μορφή κειμένου. Η διαδικασία της αποθήκευσης είναι πιο πολύπλοκη γιατί προσπαθεί να προσδιορίσει τη μέση τοποθεσία του test με χρήση geocoding.

com.nikmix.mobilesignaltester.testactivity.location.LocationProvider.java

Η κλάση αυτή χρησιμοποιεί το service τοποθεσίας της συσκευής για να επιστρέψει την τελευταία γνωστή τοποθεσία.

com.nikmix.mobilesignaltester.testactivity.sampleslist.SignalSampleRecyclerViewAdapter.java

Αυτός ο Adapter χρησιμοποιείται από την SignalSamplesListFragment για να εμφανίζει τα samples ενός test σε μορφή λίστας. Παράλληλα αναλαμβάνει να προσθέσει onClickListener στα SignalSamples της λίστας ώστε πατώντας τα να μεταβαίνουμε στις συντεταγμένες τους στο tab του χάρτη.

com.nikmix.mobilesignaltester.testactivity.sampleslist.SignalSamplesListFragment.java

Αυτή η κλάση Fragment φορτώνει τα samples ενός αποθηκευμένου test και τα περνάει ως όρισμα στον SignalSampleRecyclerViewAdapter καθώς τον δημιουργεί, ενώ τον θέτει ως Adapter στην RecyclerView που περιέχει.

com.nikmix.mobilesignaltester.testactivity.samplestask.GetSamplesTask.java

Αυτή η AsyncTask αναλαμβάνει να δημιουργεί SignalSamples ανά τακτά χρονικά διαστήματα και επίσης ξεκινά και σταματά την DataTransferTask η οποία κάνει τη μεταφορά των αρχείων από και προς τον FTP server.

com.nikmix.mobilesignaltester.testactivity.signal.MobileDataSignalProvider.java

Αυτό το interface υλοποιείται από τις κλάσεις RealMobileDataSignalProvider και SimulatedMobileDataSignalProvider και σκοπός του είναι να παρέχει στην εφαρμογή την τρέχουσα τιμή σε dBm της ισχύος του σήματος της συσκευής καθώς και τον τύπο δικτύου δεδομένων στο οποίο είναι συνδεδεμένη.

com.nikmix.mobilesignaltester.testactivity.signal.RealMobileDataSignalProvider.java

Η κλάση αυτή υλοποιεί το παραπάνω interface ενώ χρησιμοποιεί το service του συστήματος που παρέχει τις τιμές της ισχύος σήματος και του τύπου δικτύου δεδομένων της συσκευής.

com.nikmix.mobilesignaltester.testactivity.signal.SimulatedMobileDataSignalProvider.java

Η κλάση αυτή επίσης υλοποιεί το interface MobileDataSignalProvider όμως οι τιμές που επιστρέφει είναι εικονικές/τυχαίες και χρησιμοποιείται σε περίπτωση που η συσκευή δεν υποστηρίζει δίκτυο δεδομένων.

com.nikmix.mobilesignaltester.testactivity.SignalSamplesListener.java

Αυτό το interface υλοποιείται από τα Fragments της TestActivity ώστε με την προσθήκη ενός SignalSample στη λίστα της να ανανεώνονται τα περιεχόμενα των Fragments. Τα περιεχόμενά τους ανανεώνονται και όταν γίνεται 'άδειασμα' της λίστας ώστε να 'καθαρίζουν' τα Fragments όταν ξεκινάει ένα νέο test.

com.nikmix.mobilesignaltester.testactivity.TestActivity.java

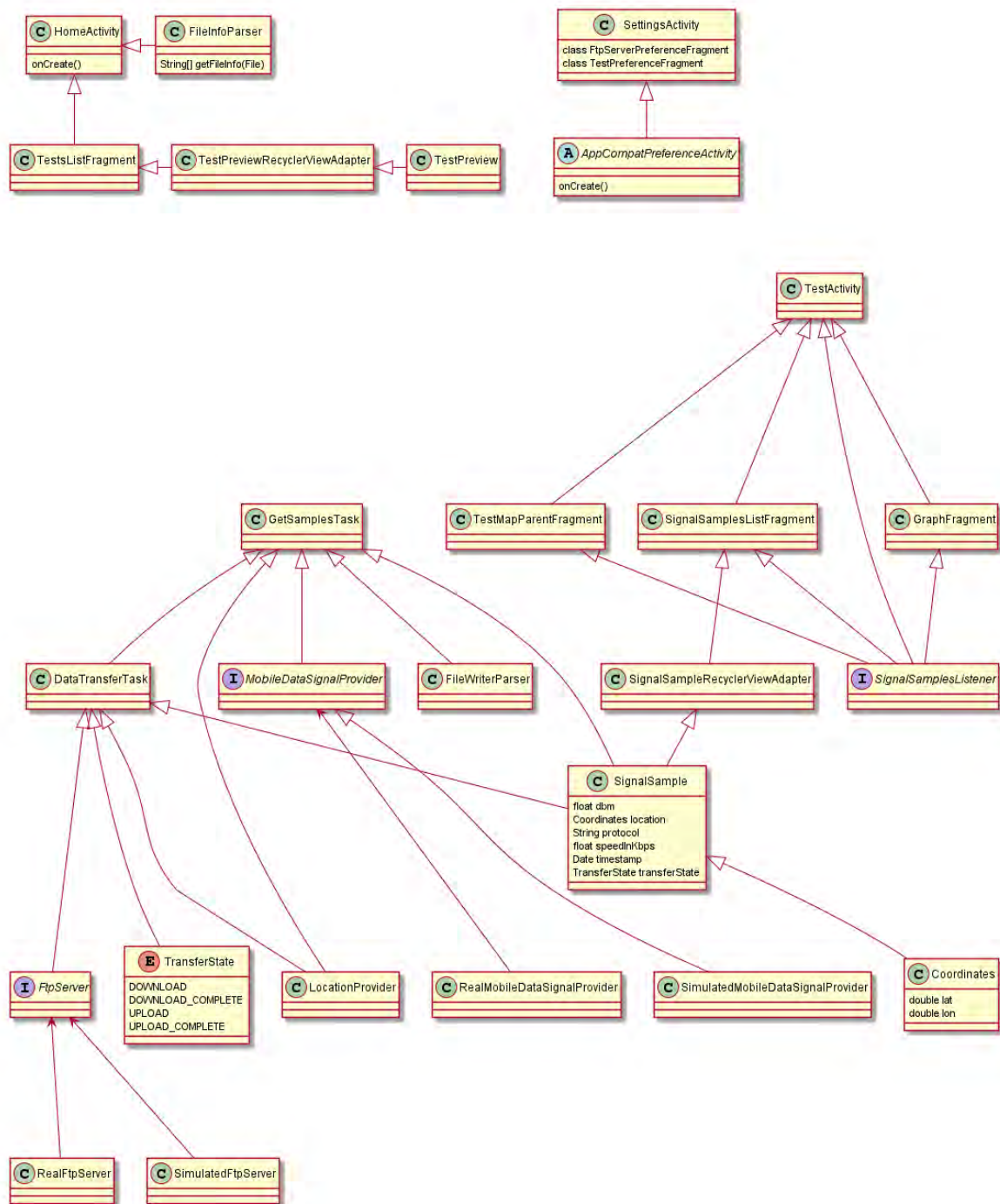
Αυτή η Activity περιέχει 4 tabs κάθε ένα απ' τα οποία περιέχει ένα Fragment (λίστα samples, χάρτης, γράφημα ισχύος σήματος, γράφημα ταχύτητας μεταφοράς). Περιέχει επίσης μία label στην οποία εμφανίζεται η πρόοδος της μεταφοράς των αρχείων κατά τη διάρκεια του

test.com.nikmix.mobilesignaltester.testactivity.TestMapParentFragment.java

Αυτό το Fragment περιέχει ένα SupportMapFragment. Διατηρεί μια μεταβλητή τύπου GoogleMap και αναλαμβάνει να εμφανίσει το μονοπάτι που ακολουθούν τα samples στο χάρτη, καθώς και markers που περιέχουν πληροφορίες όπως η ισχύς του σήματος, ο τύπος δικτύου και η ταχύτητα μεταφοράς εκείνη τη στιγμή.

Οι markers έχουν διαφορετικά χρώματα ανάλογα με το αν εκείνη τη στιγμή γίνεται download ή upload, ενώ έχουν διαφορετική απόχρωση ανάλογα με την ταχύτητα μεταφοράς εκείνη τη στιγμή.

Ακολουθεί το διάγραμμα κλάσεων:



3.3.3 Λειτουργική Περιγραφή της Εφαρμογής

Γενικά η εφαρμογή έχει τρία κύρια παράθυρα που ακολουθούν τα τρία κύρια Activity: Home Activity για την αρχική σελίδα, Test Activity για το παράθυρο διενέργειας και αναπαραγωγής των τεστ και Settings Activity για το μενού των επιλογών.

3.3.3.1 Αρχική σελίδα

Η αρχική σελίδα εμφανίζει τα τεστ που έχουν γίνει στο παρελθόν και δίνει την επιλογή για νέο τεστ (NEW TEST). Πάνω δεξιά υπάρχουν οι επιλογές (Settings).



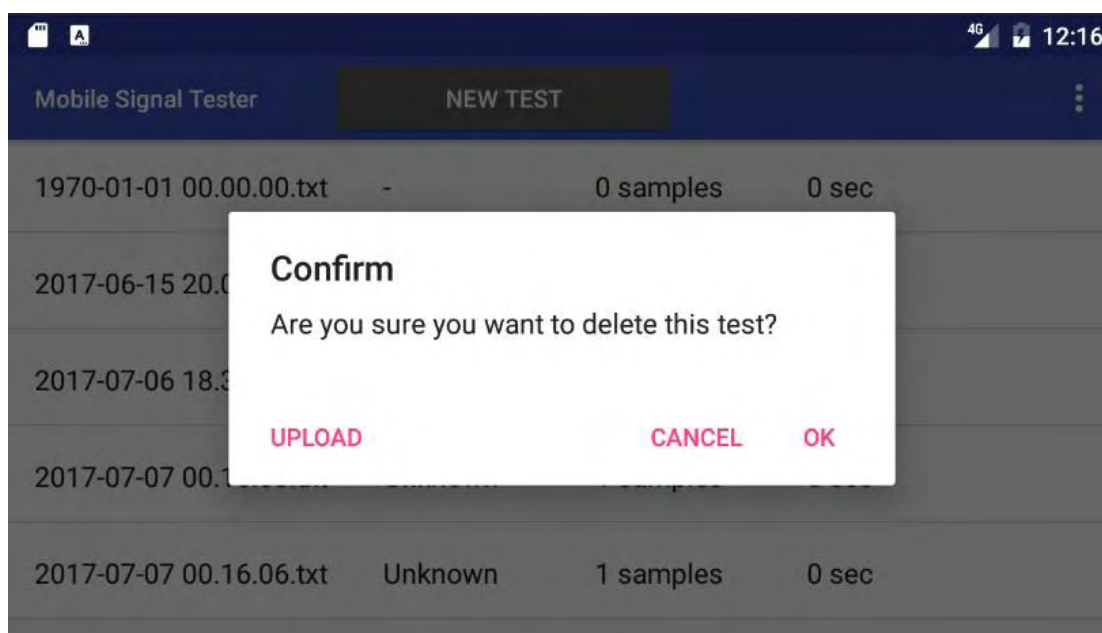
2017-07-05 22.15.51.txt	Ταύρος	62 samples	296 sec
2017-07-05 22.21.14.txt	Αθήνα	59 samples	296 sec
2017-07-05 22.27.31.txt	Μεταμόρφωση	60 samples	297 sec
2017-07-05 22.33.59.txt	Άγιος Στέφανος	67 samples	296 sec
2017-07-05 22.55.27.txt	Μακρυνίτη	61 samples	297 sec

Υπάρχει ένα τεστ ανά γραμμή με τις εξής πληροφορίες:

- Το όνομα του .txt αρχείου που έχει δημιουργηθεί. Το όνομα δίνεται αυτόματα από την ημέρα και ώρα του τεστ.
- Η περιοχή που διεξήχθη το τεστ. Βασίζεται στην πληροφορία που δίνουν οι συντεταγμένες. Επιλέγεται η περιοχή στην οποία έχουν γίνει οι περισσότερες δειγματοληψίες για το τεστ.
- Ο αριθμός των δειγματοληψιών (samples)
- Η διάρκεια του τεστ σε δευτερόλεπτα.

Η επιλογή για νέο τεστ αλλά και η επιλογή ενός παλιότερου τεστ, μας οδηγούν στην ίδια σελίδα των δοκιμών και των αποτελεσμάτων.

Πατώντας παρατεταμένα σε ένα από τα τεστ βγαίνει παράθυρο με επιλογή για την διαγραφή του τεστ(OK), αλλά και την επιλογή να σταλεί το .txt αρχείο του test στον ftp server(UPLOAD, κάτω αριστερά).



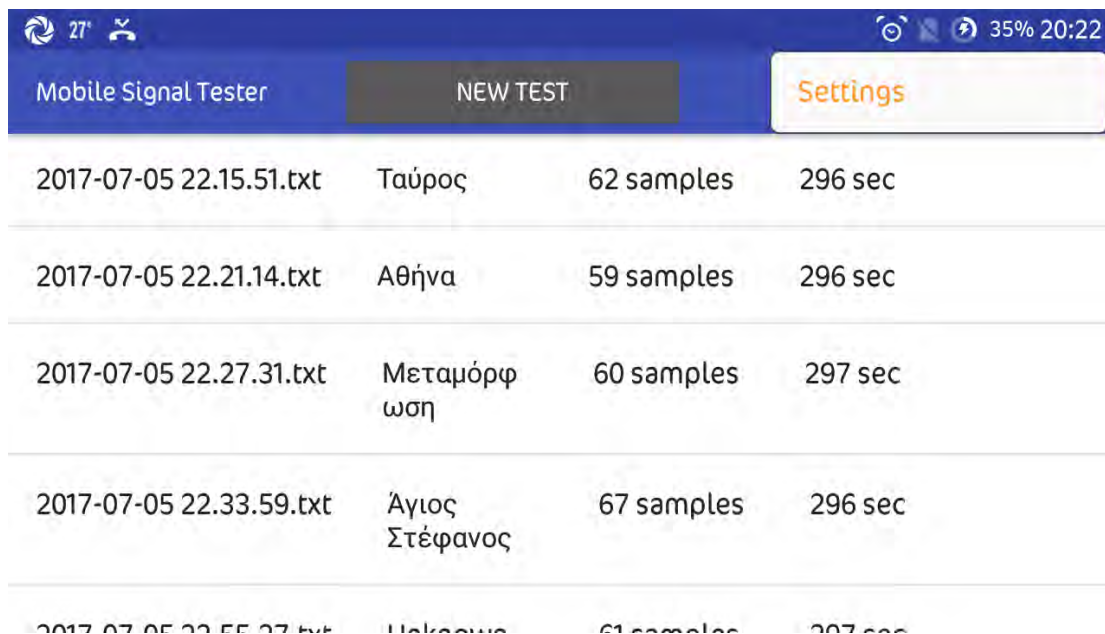
Ακολουθεί παράδειγμα .txt αρχείου που ανέβηκε στον ftp server:

```
1499376278914|Φιλαδέλφεια Χαλκηδόνα|236
60
1499376278914|38.0181802|23.7225709|HSPAP|-89.0|2|-2.0
1499376284958|38.0189892|23.7233832|HSPAP|-95.0|2|1810.1467
1499376290990|38.0197|23.7239194|HSPAP|-89.0|2|2411.1575
1499376292014|38.0197|23.7239194|HSPAP|-89.0|4|1936.7545
1499376297022|38.0205572|23.7244976|HSPAP|-89.0|2|1949.5477
1499376303044|38.0214694|23.7249644|HSPAP|-87.0|2|2528.841
1499376304638|38.0223732|23.7252541|HSPAP|-87.0|4|2009.6283
1499376309071|38.0223732|23.7252541|HSPAP|-87.0|2|2184.404
1499376315090|38.0244437|23.7258484|HSPAP|-69.0|2|2551.3438
1499376315706|38.0244437|23.7258484|HSPAP|-69.0|4|2294.2256
1499376321128|38.0254623|23.7263897|HSPAP|-53.0|2|2884.507
1499376325387|38.0254623|23.7263897|HSPAP|-53.0|4|2617.8462
1499376327160|38.0267206|23.7269893|HSPAP|-53.0|2|2377.249
1499376333200|38.0276161|23.7273478|HSPAP|-53.0|2|2485.8677
1499376335995|38.0286373|23.7278162|HSPAP|-53.0|4|2394.709
1499376339217|38.0286373|23.7278162|HSPAP|-53.0|2|2926.7595
1499376344569|38.0307017|23.7288932|HSPAP|-53.0|4|2959.238
```

1499376345360|38.0307017|23.7288932|HSPAP|-63.0|2|-2.0
 1499376351383|38.0316145|23.7295443|HSPAP|-63.0|2|3093.6558
 1499376354787|38.0326266|23.730173|HSPAP|-67.0|4|2486.2354
 1499376357413|38.0326266|23.730173|HSPAP|-59.0|2|1958.684
 1499376363441|38.0337493|23.7308868|HSPAP|-59.0|2|1853.3937
 1499376368935|38.034787|23.7316203|HSPAP|-51.0|4|1793.1543
 1499376369488|38.0358393|23.7323748|HSPAP|-51.0|2|-2.0
 1499376375534|38.0368783|23.7331049|HSPAP|-61.0|2|2080.455
 1499376381612|38.038224|23.7339434|HSPAP|-61.0|2|2384.1677
 1499376382575|38.038224|23.7339434|HSPAP|-61.0|4|1866.2312
 1499376387641|38.0391968|23.7341611|HSPAP|-55.0|2|2352.4006
 1499376392545|38.0403601|23.7343268|HSPAP|-55.0|4|2544.5884
 1499376393666|38.0403601|23.7343268|HSPAP|-55.0|2|-2.0
 1499376399685|38.0425221|23.734852|HSPAP|-55.0|2|-2.0
 1499376405693|38.0434509|23.7354558|HSPAP|-63.0|2|-2.0
 1499376411717|38.0442599|23.7363132|HSPAP|-69.0|2|-2.0
 1499376417742|38.0449488|23.7373|HSPA|-53.0|2|-2.0
 1499376423767|38.0457333|23.7382193|HSPAP|-71.0|2|-2.0
 1499376429788|38.0473595|23.7399216|HSPAP|-51.0|2|3005.525
 1499376432941|38.0473595|23.7399216|HSPAP|-51.0|4|628.239
 1499376435803|38.0480991|23.7407835|HSPAP|-51.0|2|3203.7544
 1499376441696|38.0488762|23.7415878|HSPAP|-51.0|4|2902.362
 1499376441836|38.0488762|23.7415878|HSPAP|-51.0|2|-2.0
 1499376447868|38.0496193|23.7424037|HSPAP|-51.0|2|2793.9973
 1499376451429|38.0504214|23.7432137|HSPAP|-51.0|4|2609.7673
 1499376453895|38.0504214|23.7432137|HSPAP|-51.0|2|2914.6108
 1499376459106|38.0512504|23.7440263|HSPAP|-51.0|4|3309.7908
 1499376459920|38.052166|23.7446815|HSPAP|-51.0|2|2904.9646
 1499376465936|38.0531352|23.7450719|HSPAP|-51.0|2|3535.5261
 1499376467217|38.0531352|23.7450719|HSPAP|-51.0|4|3132.4294
 1499376471957|38.0541491|23.7454884|HSPAP|-51.0|2|2442.6187
 1499376477969|38.0551299|23.7457827|HSPAP|-51.0|4|2359.5188
 1499376478031|38.0551299|23.7457827|HSPAP|-51.0|2|-2.0
 1499376484106|38.0561805|23.7462397|HSPAP|-51.0|2|3306.1885
 1499376485861|38.0572096|23.7467343|HSPAP|-51.0|4|3219.8943
 1499376490125|38.0583398|23.7473182|HSPAP|-51.0|2|3617.4614
 1499376492714|38.0583398|23.7473182|HSPAP|-61.0|4|3701.9622
 1499376496146|38.0594721|23.7480679|HSPAP|-61.0|2|3372.8591
 1499376500427|38.0606607|23.7488327|HSPAP|-61.0|4|3295.1743
 1499376502170|38.0606607|23.7488327|HSPAP|-71.0|2|2555.209
 1499376508192|38.0621281|23.7498118|HSPAP|-55.0|2|2612.2449
 1499376511306|38.0630761|23.7503537|HSPAP|-77.0|4|2334.7642
 1499376514214|38.0630761|23.7503537|HSPAP|-77.0|2|2033.3599

3.3.3.2 Ρυθμίσεις

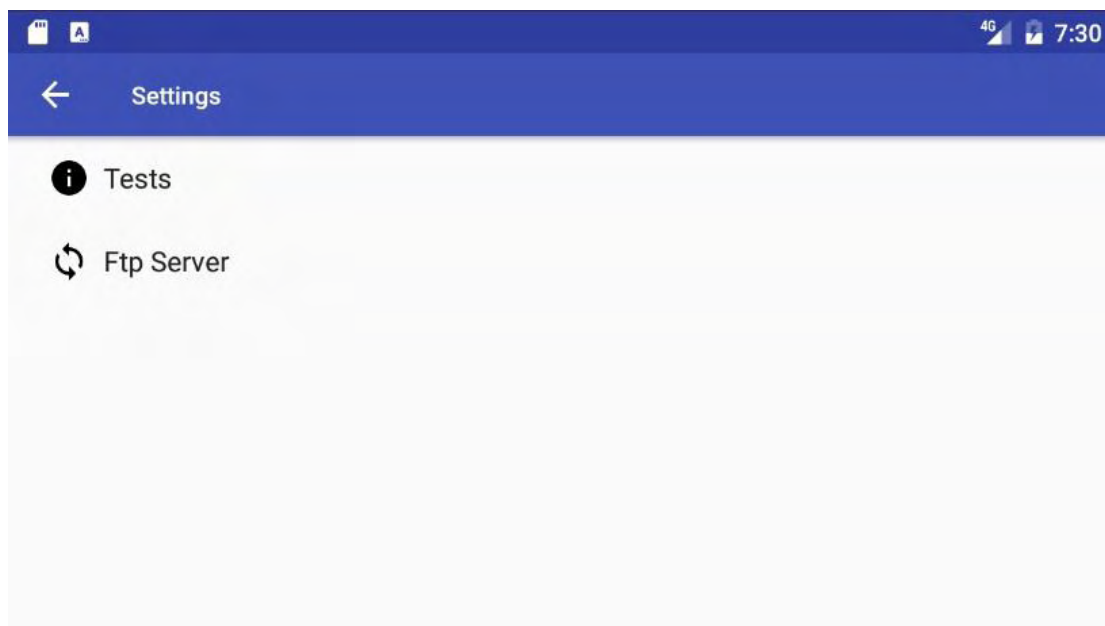
Η είσοδος στο μενού γίνεται από το σύμβολο (τρεις κάθετες τελείες) πάνω δεξιά.



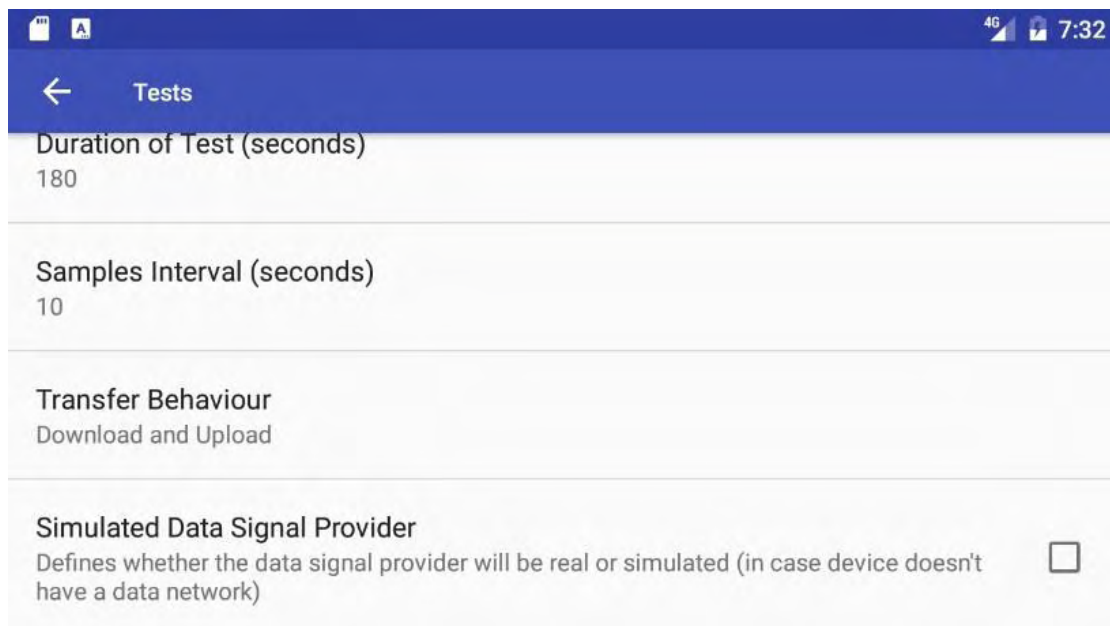
2017-07-05 22.15.51.txt	Ταύρος	62 samples	296 sec
2017-07-05 22.21.14.txt	Αθήνα	59 samples	296 sec
2017-07-05 22.27.31.txt	Μεταμόρφωση	60 samples	297 sec
2017-07-05 22.33.59.txt	Άγιος Στέφανος	67 samples	296 sec
2017-07-05 22.55.37.txt	Μακρυνί	61 samples	297 sec

Το μενού των ρυθμίσεων έχει δύο επιλογές.

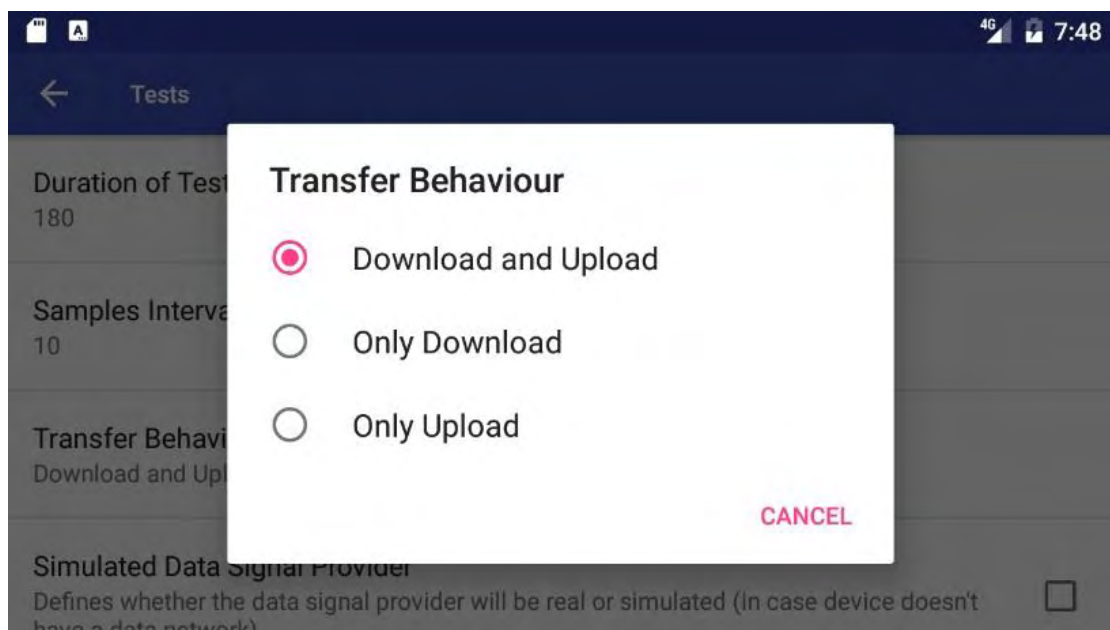
- Tests.
- Ftp Server.



Στην επιλογή Tests θα βρούμε την επιλογή για την διάρκεια του τεστ (Duration of Test), την επιλογή για το κάθε πότε θα γίνεται η δειγματοληψία (Samples Interval), την συμπεριφορά μεταφοράς δεδομένων (Transfer Behavior) και τον προσομοιωτή δικτύου κινητής τηλεφωνίας



Η διάρκεια του τεστ και των Intervals είναι σε δευτερόλεπτα. Στην συμπεριφορά της μεταφοράς έχουμε τρεις επιλογές: α. δοκιμή εναλλάξ download και upload, β. μόνο Download και γ. μόνο Upload.



Στην επιλογή Ftp Server θα βρούμε τον προσομοιωτή Ftp Server, την διεύθυνση IP του server, την πόρτα που χρησιμοποιεί και τον συνδυασμό username/password για να συνδεθεί η εφαρμογή ως client σε αυτόν. Επίσης μπορούμε να ορίσουμε και το αρχείο το οποίο θα κατεβάζει η εφαρμογή στο πλαίσιο των δοκιμών. Αυτό το αρχείο πρέπει να υπάρχει ήδη στον Server.

Ftp Server

Simulated Ftp Server
The simulated server doesn't use mobile data ☐

Ftp Server Url
89.210.86.166

Port
21

Username
nikmix

Ftp Server

Port
21

Username
nikmix

Password
R@nd0m_P@\$w0rd

Remote File Path
test_download.avi

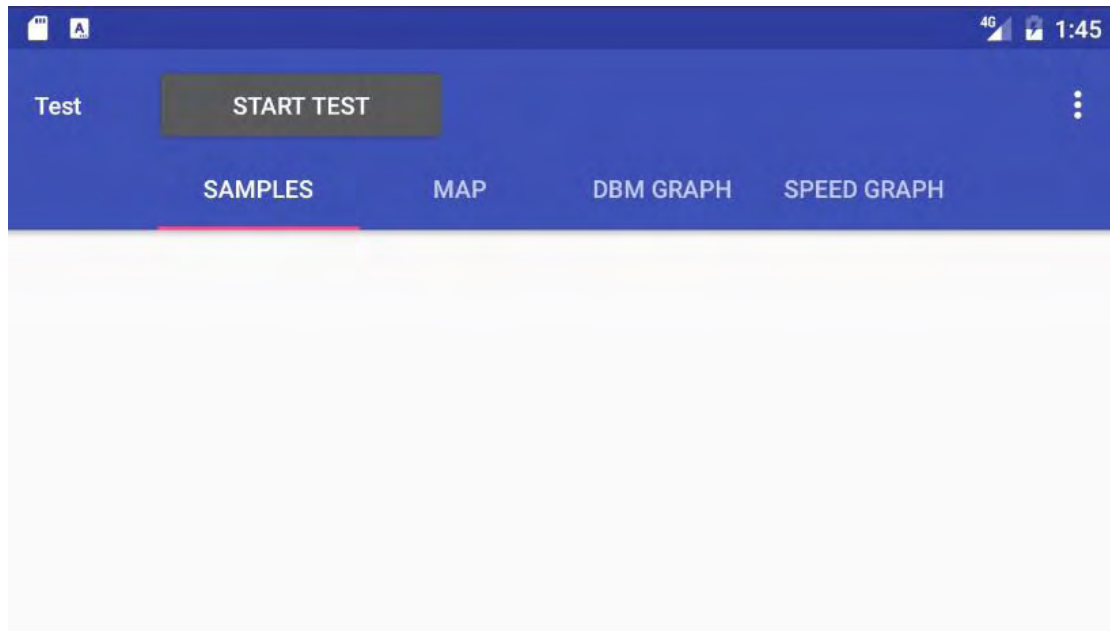
Οι διαδικασίες «Simulated» χρησίμευσαν κατά την ανάπτυξη του κώδικα για debugging. Πλέον δεν είναι απαραίτητες και καλό είναι να παραμένουν πάντα απενεργοποιημένες. Πάντως, αν κάποιος θελήσει να δοκιμάσει την εφαρμογή χωρίς να έχει εγκαταστήσει κάποιον ftp server, μπορεί να ενεργοποιήσει τον simulated, όπως επίσης να κάποιος θέλει να δει την εφαρμογή να τρέχει χωρίς να καταναλώσει δεδομένα, μπορεί να ενεργοποιήσει τον simulated data signal provider.

Κάτι που επίσης πρέπει να σημειωθεί εδώ είναι ότι σε αντίθεση με το αρχείο ελέγχου download, το οποίο μπορούμε να αλλάξουμε ευκολά στο server, το αρχείο ελέγχου

upload είναι ενσωματωμένο με την εφαρμογή. Έχει μέγεθος 3.1MB και αν θελήσουμε να το αλλάξουμε θα πρέπει να ξαναδημιουργήσουμε το αρχείο .apk.

3.3.3.3 Εκτέλεση Τεστ

Η σελίδα που εμφανίζεται όταν επιλέγουμε το νέο τεστ είναι ίδια με αυτή της αναπαραγωγής παλαιότερων τεστ.

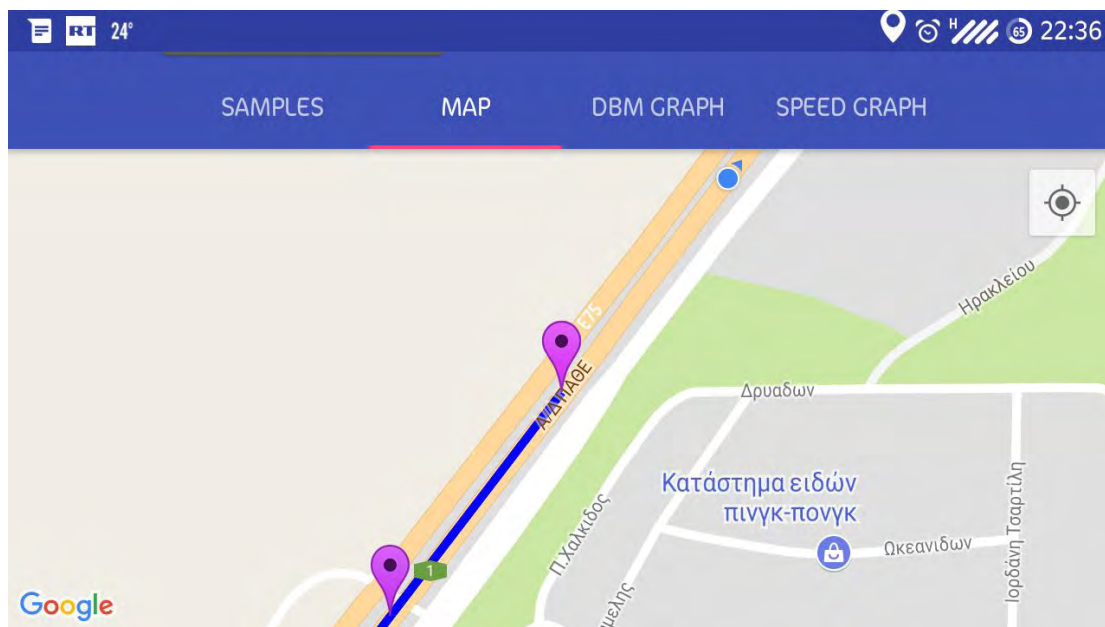


Αν επιλέξουμε παλαιότερο τεστ τότε βλέπουμε απευθείας τα samples της δοκιμής, που αποτελούν ανάγνωση του .txt αρχείου του τεστ. Στο .txt αρχείο βασίζεται και η απεικόνιση στον χάρτη, αλλά και τα γραφήματα ισχύς σήματα και ταχύτητας.

<div> <div>24°</div> <div> <div>Test</div> <div>START TEST</div> <div></div> </div> </div>				
<div> <div>SAMPLES</div> <div>MAP</div> <div>DBM GRAPH</div> <div>SPEED GRAPH</div> </div>				
2017-07-05 23.47.28	HSPAP	-63.0 dBm	Download	1406.7 kbps
2017-07-05 23.47.34	HSPAP	-63.0 dBm	Download	1698.3 kbps
2017-07-05 23.47.40	HSPAP	-63.0 dBm	Download	1807.1 kbps
2017-07-05 23.47.46	HSPAP	-63.0 dBm	Download	1741.4 kbps

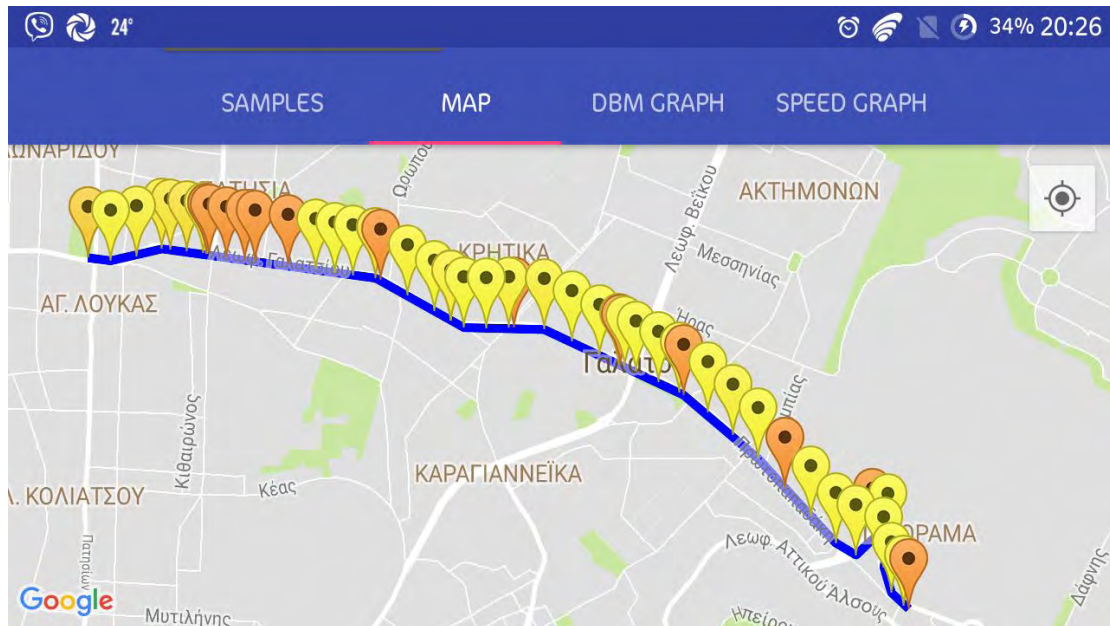
Στα νέα τεστ ο χάρτης ανανεώνεται σε πραγματικό χρόνο, όπως και τα γραφήματα. Μόλις πραγματοποιείται μια νέα δειγματοληψία, εμφανίζεται στην λίστα των samples στο τέλος, ένα νέο marker εμφανίζεται στον χάρτη και ο χάρτης μετακινείται ώστε το marker να είναι στο κέντρο.

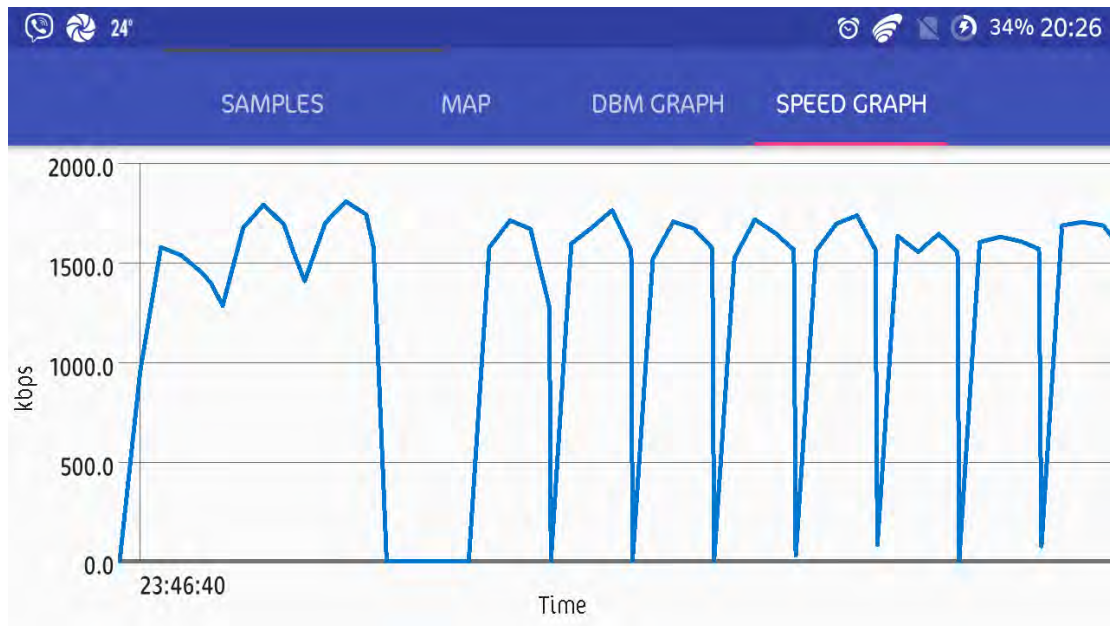
Ακολουθεί στιγμιότυπο οθόνης κινητού στον χάρτη κατά την διάρκεια του τεστ.



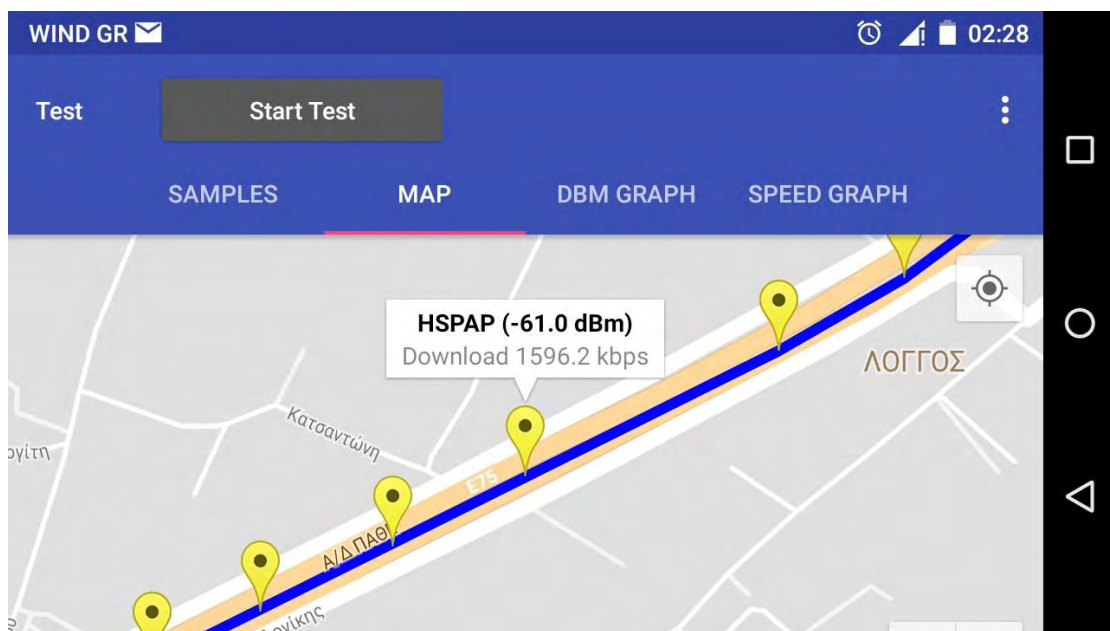
Στο πάνω μέρος επί της εθνικής οδού διακρίνεται η τρέχουσα θέση της κινητής συσκευής. Με τον ορισμό νέου marker, ο χάρτης θα κεντράρει σε αυτό, όπως φαίνεται και στο στιγμιότυπο με το τελευταίο marker.

Από το οριζόντιο μενού σε ένα υπάρχον τεστ μπορούμε να μεταβούμε στον χάρτη και τα γραφήματα.





Επίσης υπάρχει η δυνατότητα μόλις επιλέξεις ένα sample η εφαρμογή να σε πάει αυτόματα σε αυτό στον χάρτη, δείχνοντάς το στο κέντρο με τις λοιπές πληροφορίες σε ορθογώνιο παράθυρο πάνω του.



Όσο για τα χρώματα των marker και για να μην υπάρχει πρόβλημα διακρίσεις μεταξύ Upload και Download, έχουν χρησιμοποιηθεί οι συνδυασμοί πορτοκαλί-κίτρινο-πράσινο για το download, με πορτοκαλί την μηδενική ταχύτητα και πράσινο το μέγιστο, και ροζ-μοβ-μπλε για το upload με ροζ για μηδενική ταχύτητα και μπλε για την μέγιστη.

Τα χρώματα βασίζονται στην μεταβλητή hue η οποία παίρνει τιμή διαιρώντας την ταχύτητα που επιστρέφει το sample με την μέγιστη ταχύτητα up/down που έχουμε ορίσει στο πρόγραμμα. Έτσι πετυχαίνουμε μεγάλο αριθμό αποχρώσεων. Αυτή τη στιγμή έχουμε ορίσει ως μέγιστη τιμή για το up τα 1600kbps. Αυτό σημαίνει ότι μέτρηση 1600 η μεγαλύτερη θα χρωματιστεί με καθαρό μπλε. Αντίστοιχα καθαρό πράσινο στο down θα πάρουν οι μετρήσεις με ταχύτητες 5000kbps και πάνω.

Οι μέγιστες ταχύτητες ίσως είναι ικανοποιητικές για ένα δίκτυο 3G, αλλά για ένα δίκτυο 4G ίσως είναι ξεπερασμένες με αποτέλεσμα όλες οι μετρήσεις να παίρνουν τον χρωματισμό της μέγιστης τιμής. Ωστόσο, μπορούμε στον κώδικα να αλλάξουμε εύκολα την μέγιστη τιμή. Ακολουθεί το τμήμα του κώδικα που κάνει ακριβώς αυτή την εργασία. Πρόκειται για τις μεταβλητές maxSpeed.

```
private BitmapDescriptor getColoredIcon(SignalSample signalSample) {
    //If it's a download, we use an orange icon if the speed is zero
    //and a green one if the speed is above 5000 kbps. If the speed
    //is between, we use an intermediate color depending on the speed.
    if (signalSample.transferState == SignalSample.TransferState.DOWNLOAD ||
        signalSample.transferState == SignalSample.TransferState.DOWNLOAD_COMPLETE)
    {
        float hueMin = BitmapDescriptorFactory.HUE_ORANGE;
        float hueMax = BitmapDescriptorFactory.HUE_GREEN;
        float maxSpeed = 5000;
        float speedInKbps = Math.max(0, Math.min(signalSample.speedInKbps, maxSpeed));
        return BitmapDescriptorFactory.defaultMarker(hueMin + (hueMax - hueMin) *
            speedInKbps / maxSpeed);
    } else {
        //If it's an upload, we use a rose icon if the speed is zero
        //and a blue one if the speed is above 1600 kbps. If the speed
        //is between, we use an intermediate color depending on the speed.
        float hueMin = BitmapDescriptorFactory.HUE_BLUE;
        float hueMax = BitmapDescriptorFactory.HUE_ROSE;
        float maxSpeed = 1600;
        float speedInKbps = Math.max(0, Math.min(signalSample.speedInKbps, maxSpeed));
        return BitmapDescriptorFactory.defaultMarker(hueMin + (hueMax - hueMin) * (1 -
            (speedInKbps / maxSpeed)));
    }
}
```

Σε αυτό το σημείο του κώδικα μπορούμε να αλλάξουμε και τις αποχρώσεις αν το επιθυμούμε.

0 upload 1600



0 download 5000



4 Αποτελέσματα Δοκίμων

4.1 Εισαγωγή

Η εφαρμογή Mobile Signal Tester είναι σε θέση να εκτελεί απεριόριστα up και down test για την μέτρηση των δυνατοτήτων του δικτύου στην μεταφορά δεδομένων. Ο χρήστης επιλέγει την γεωγραφική τοποθεσία, η οποία αποτυπώνεται πάνω σε ενσωματωμένο στην εφαρμογή Google Map. Η διαδρομή καταγραφής αποτυπώνεται με μια σειρά από marker με τις αντίστοιχες αποχρώσεις. Στα παραδείγματα που ακολουθούν παρουσιάζονται μετρήσεις που καταγράφηκαν τόσο σε εθνικό, όσο και σε επαρχιακό οδικό δίκτυο.

Όπως έχει αναφερθεί, το αρχείο για την δοκιμή upload είναι περίπου 3MB. Για τις παρούσες δοκιμές, το αρχείο download που χρησιμοποιήθηκε είναι περίπου 5MB.

4.2 Περιορισμοί

Τα test που ακολουθούν έχουν γίνει χρησιμοποιώντας ως ftp server τον FileZilla Server ο οποίος είναι εγκατεστημένος στον υπολογιστή μου. Έτσι υπάρχουν οι περιορισμοί της adsl σύνδεσης. Το Up της γραμμής μου, το οποίο χρησιμοποιείται όταν η εφαρμογή κάνει download test, είναι 2000kbps, καθώς εφαρμόζει πρότυπο Annex M (ADSL2+ M). Έτσι σε όλα τα test το μέγιστο download που βλέπουμε, δεν ξεπερνά τα 2000kbps.

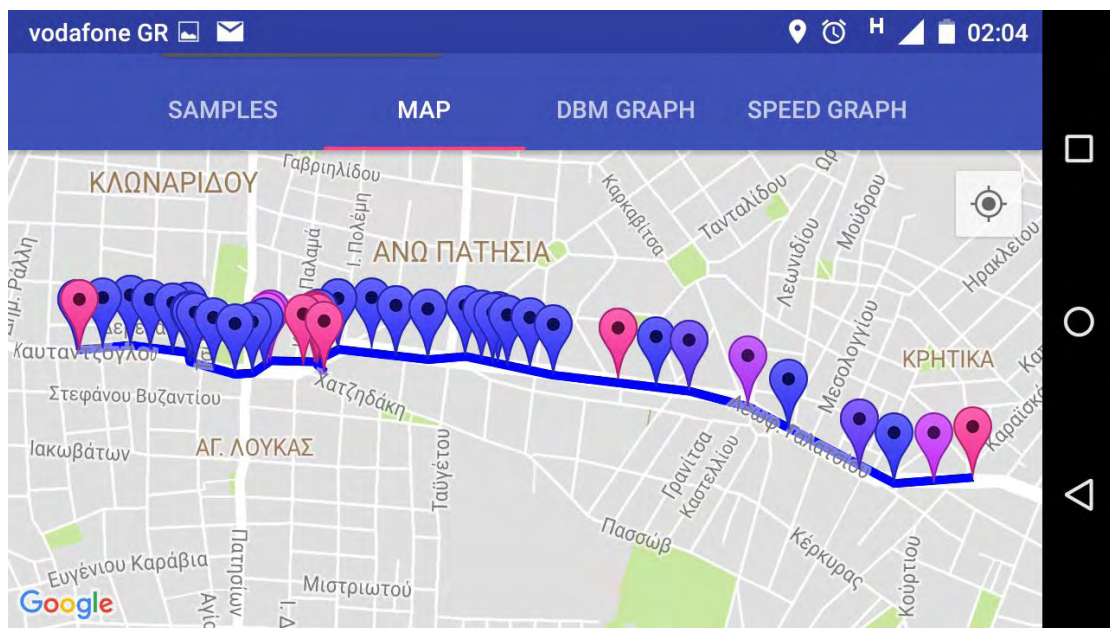
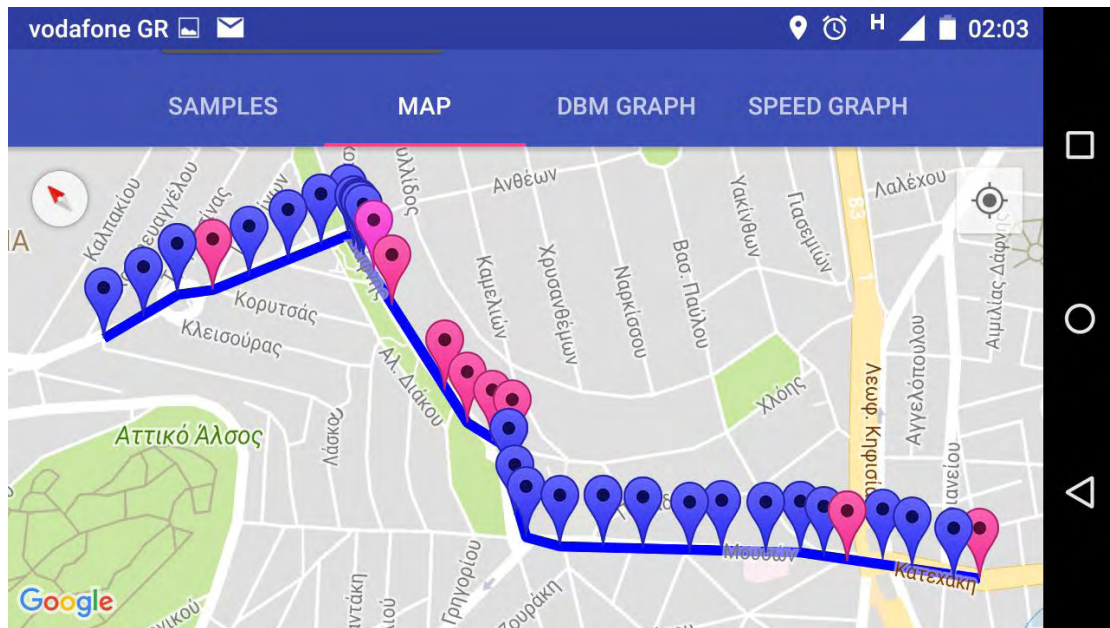
Ένας δεύτερος περιορισμός είναι ότι οι κινητές συσκευές που χρησιμοποιήθηκαν δεν υποστηρίζουν τεχνολογία 4G με αποτέλεσμα να μην παίρνουμε μετρήσεις από το 4G δίκτυο του παρόχου. Αντίθετα το πιο σύγχρονο πρωτόκολλο που βλέπουμε στις δοκιμές είναι το HSPAP(H+, 3G).

4.3 Αστικό περιβάλλον

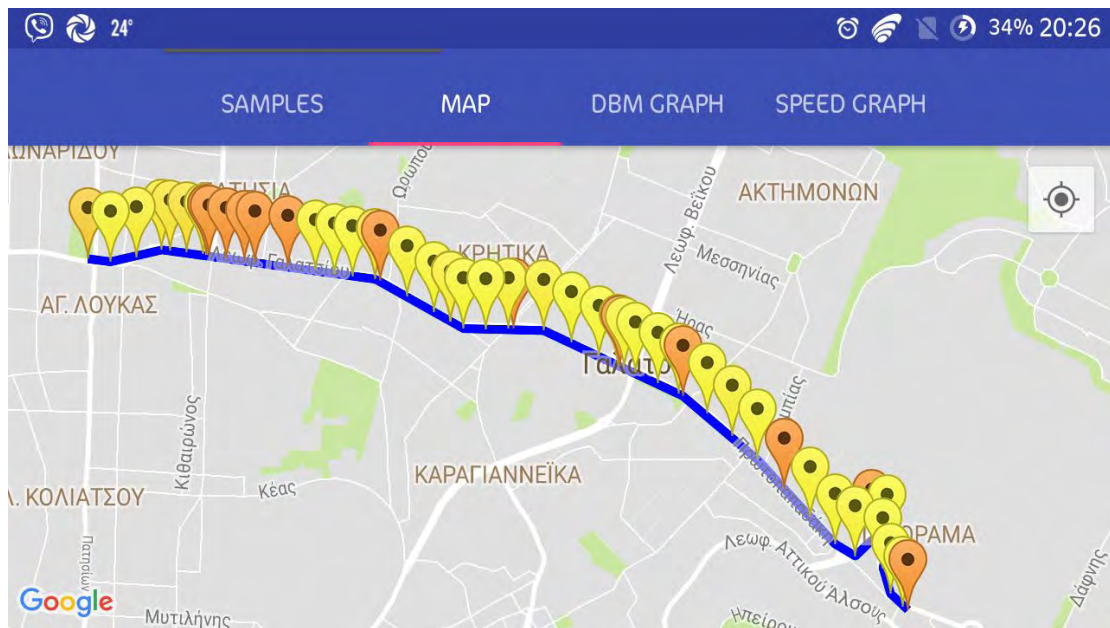
Στα όρια μιας αστικής περιοχής παρατηρούνται διάφορες διακυμάνσεις στην ταχύτητα μεταφοράς και αυτό μπορεί να οφείλεται στο φαινόμενο των πολλαπλών οδεύσεων. Τα κτίρια, τα αυτοκίνητα, τα δέντρα που υπάρχουν προκαλούν διαδοχικές ανακλάσεις και διάχυση στα ηλεκτρομαγνητικά κύματα που έρχονται από τις κεραίες κινητών

επικοινωνιών. Υπάρχει εναλλαγή στον τύπο δικτύου από EDGE ως και HSPA+ με μέγιστη τιμή στο up (που έχουμε ασφαλή συμπεράσματα) τα 3600kbps.

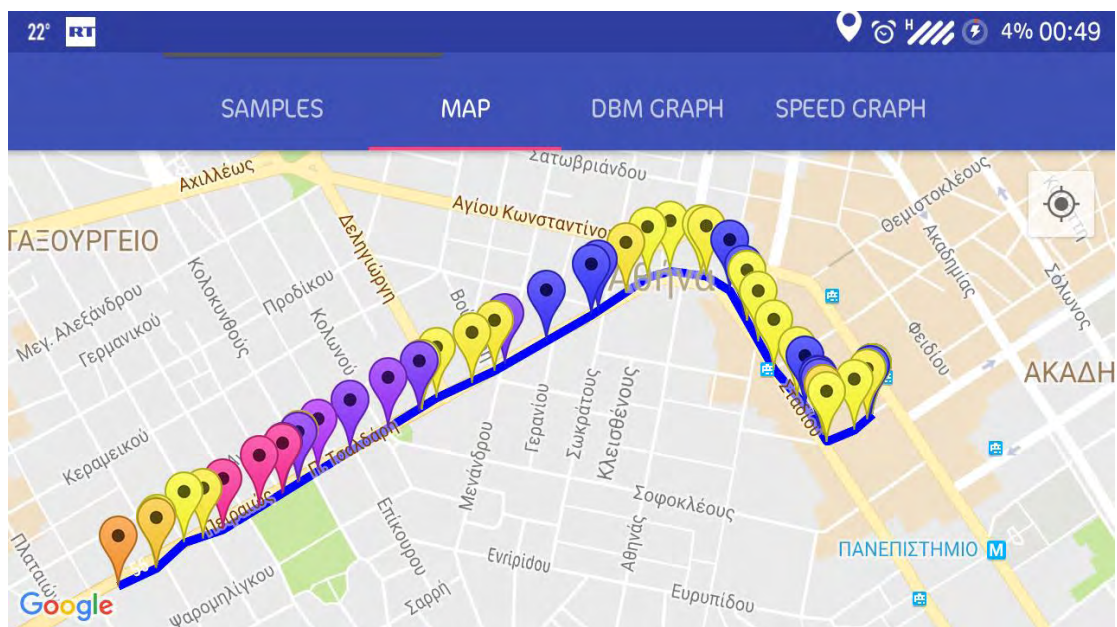
Δοκιμές μόνο upload. Σε αυτοκίνητο. Duration: 240sec, Sample interval:6sec

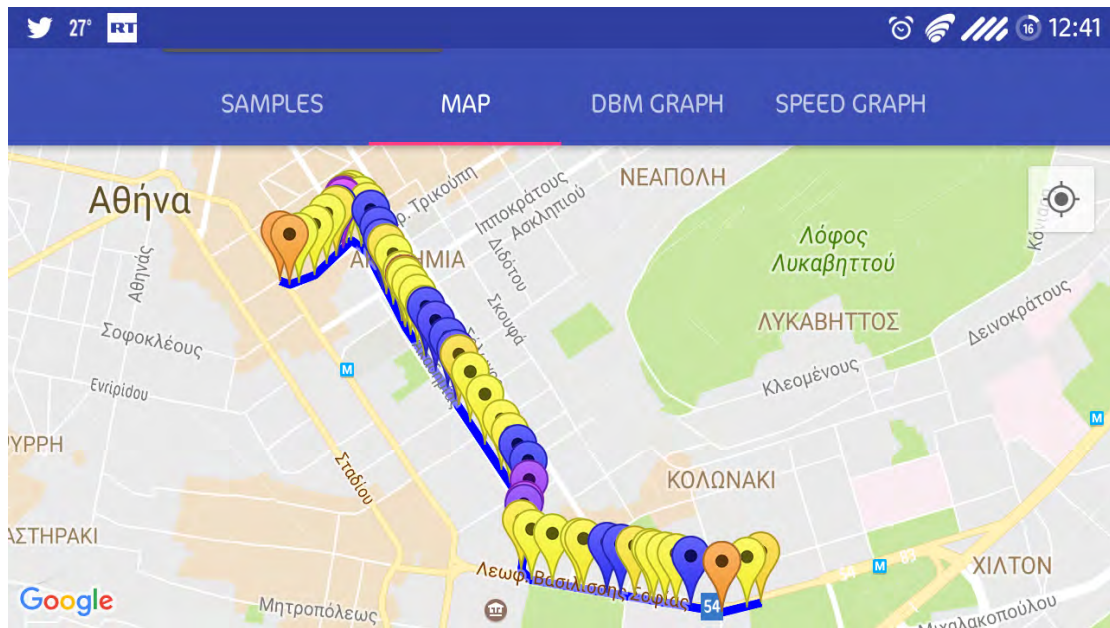


Δοκιμή μόνο download. Σε αυτοκίνητο. Duration: 300sec, Sample interval:6sec

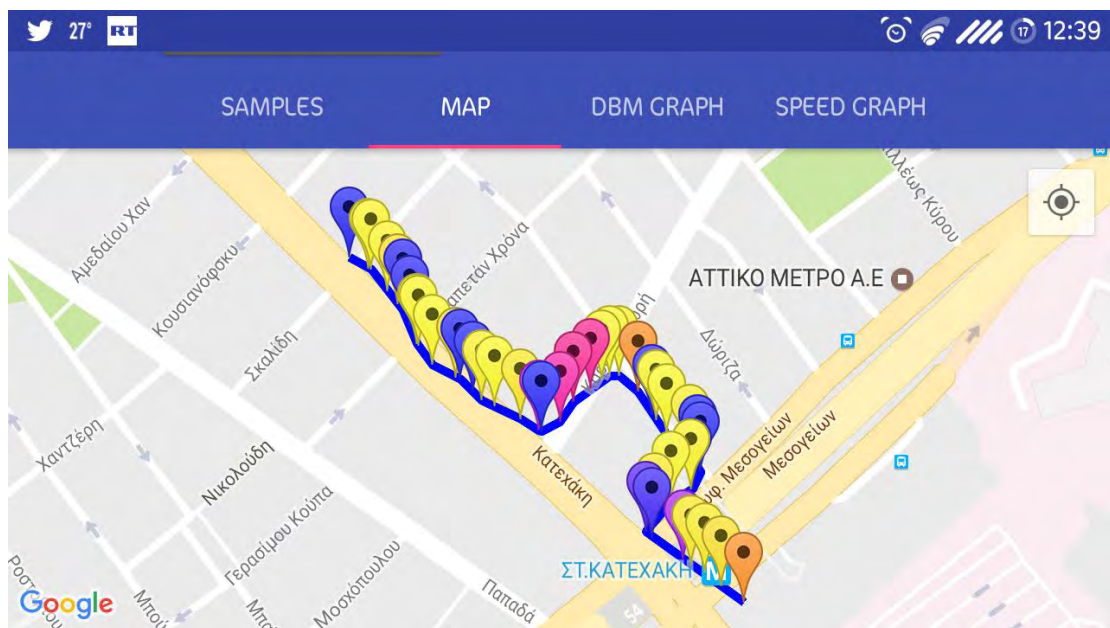


Δοκιμές upload και download. Σε λεωφορείο. Duration: 300sec, Sample interval:6sec





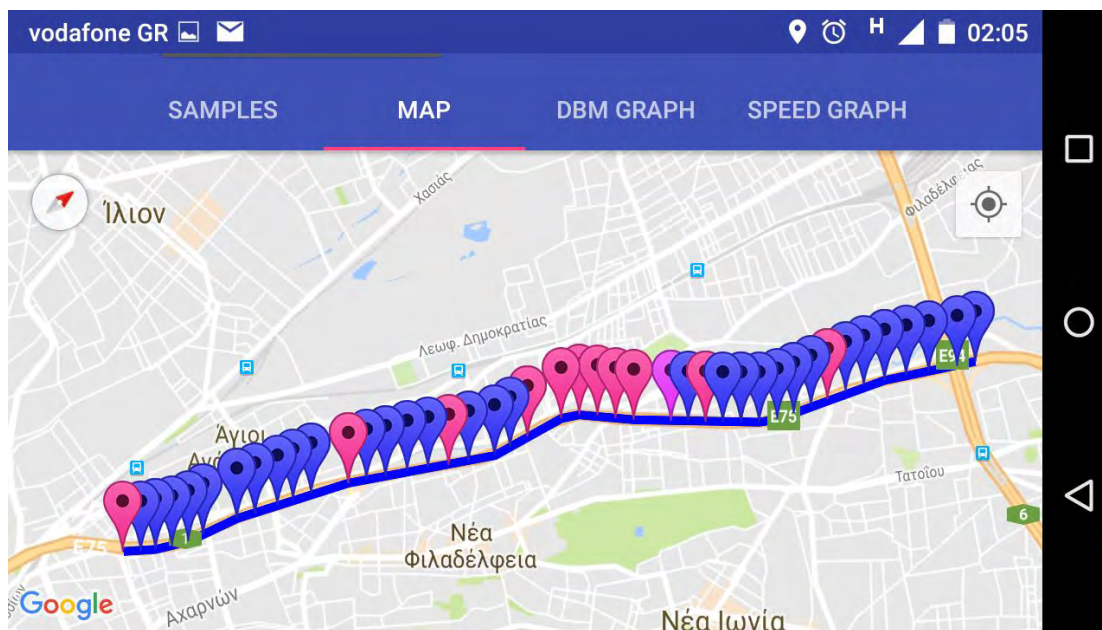
Δοκιμή upload και download. Με τα πόδια. Duration: 300sec, Sample interval:10sec



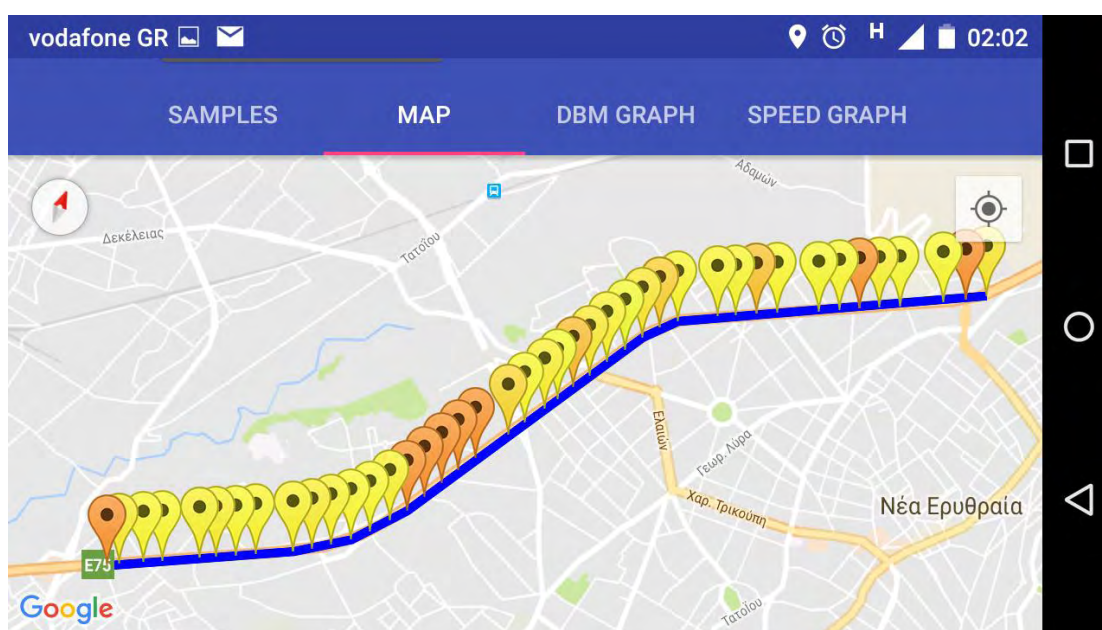
4.4 Εθνικό δίκτυο

Η μετρήσεις της ταχύτητας μεταφοράς ήταν σχετικά σταθερή, με μεταβολές στο πρωτόκολλο δικτύου μεταξύ H και H+. Υπήρχαν κατ' εξαίρεση σημεία με μικρότερες δυνατότητες μεταφοράς δεδομένων που αποτυπώθηκαν στα τεστ.

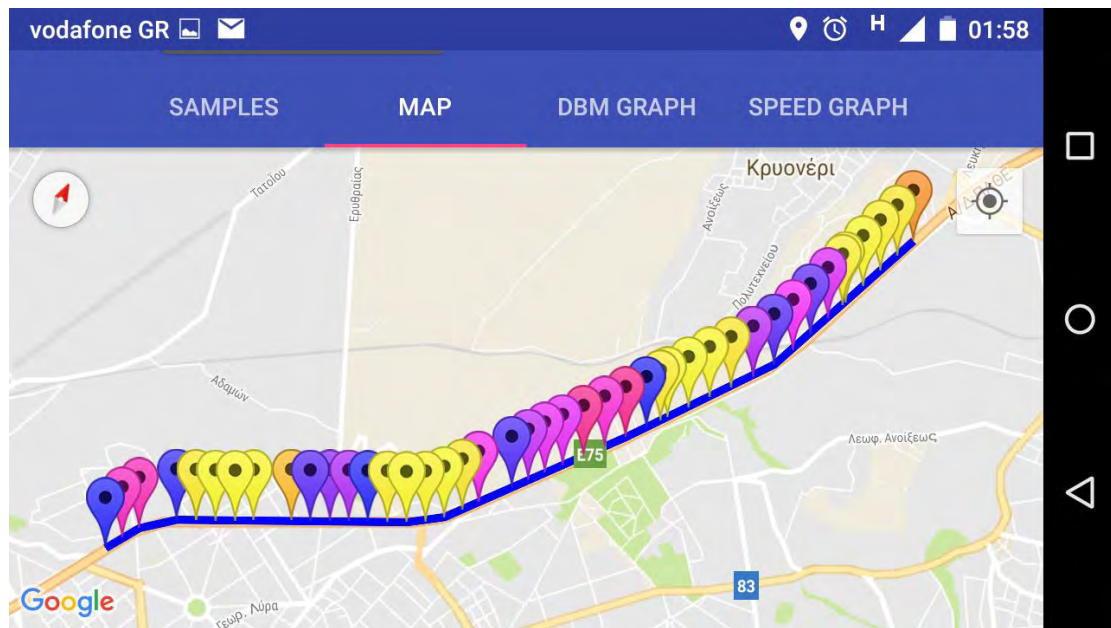
Δοκιμή μόνο upload. Με αυτοκίνητο. Duration: 240sec, Sample interval:6sec



Δοκιμή μόνο download. Με αυτοκίνητο. Duration: 240sec, Sample interval:6sec



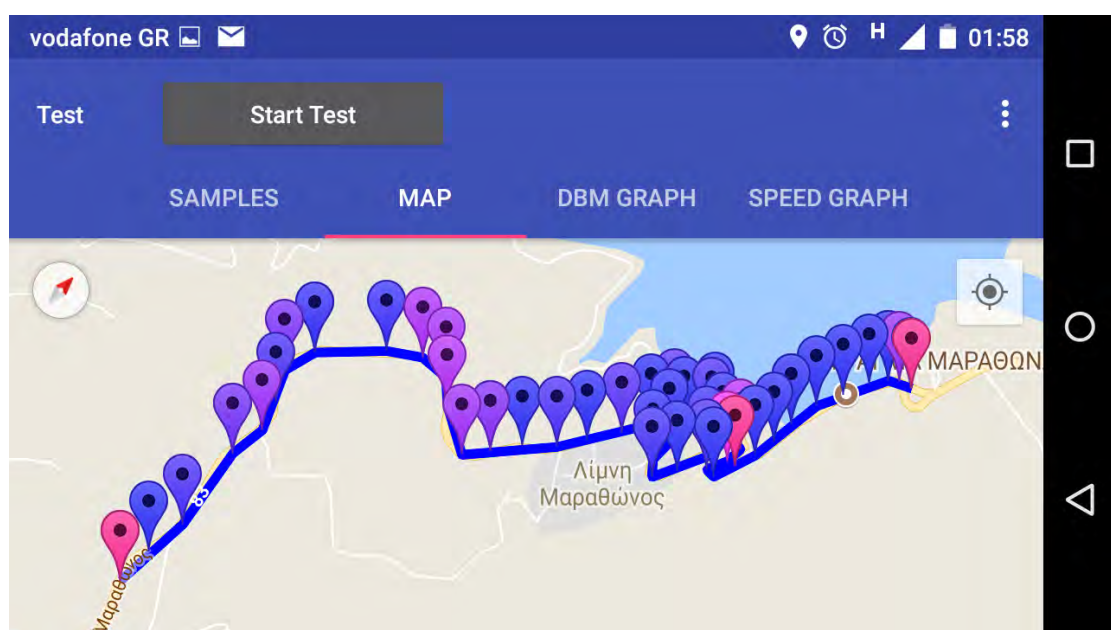
Δοκιμή upload και download. Με αυτοκίνητο. Duration: 240sec, Sample interval:6sec



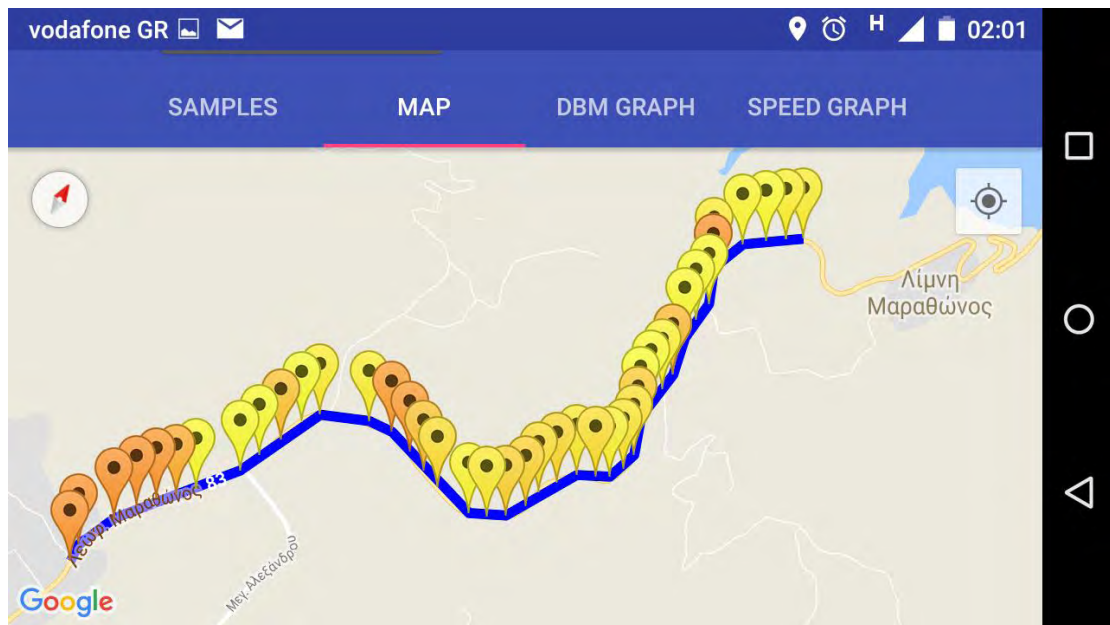
4.5 Επαρχιακό δίκτυο

Παρουσιάστηκαν αρκετές διακυμάνσεις σε ταχύτητα μεταφοράς και σε τύπο δικτύου. Οι διακυμάνσεις αυτές μπορεί να οφείλονται στην μορφολογία του εδάφους και στη διαφορετική ισχύ εκπομπής των κεραιών λόγω της απόστασης.

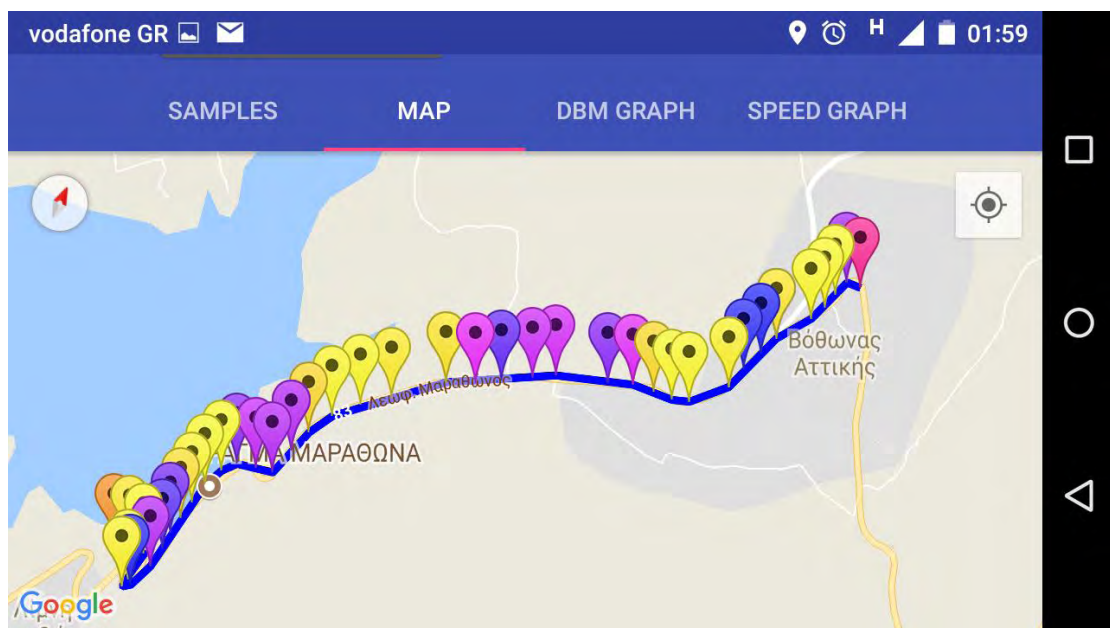
Δοκιμή μόνο upload. Με αυτοκίνητο. Duration: 240sec, Sample interval:6sec



Δοκιμή μόνο download. Με αυτοκίνητο. Duration: 240sec, Sample interval:6sec

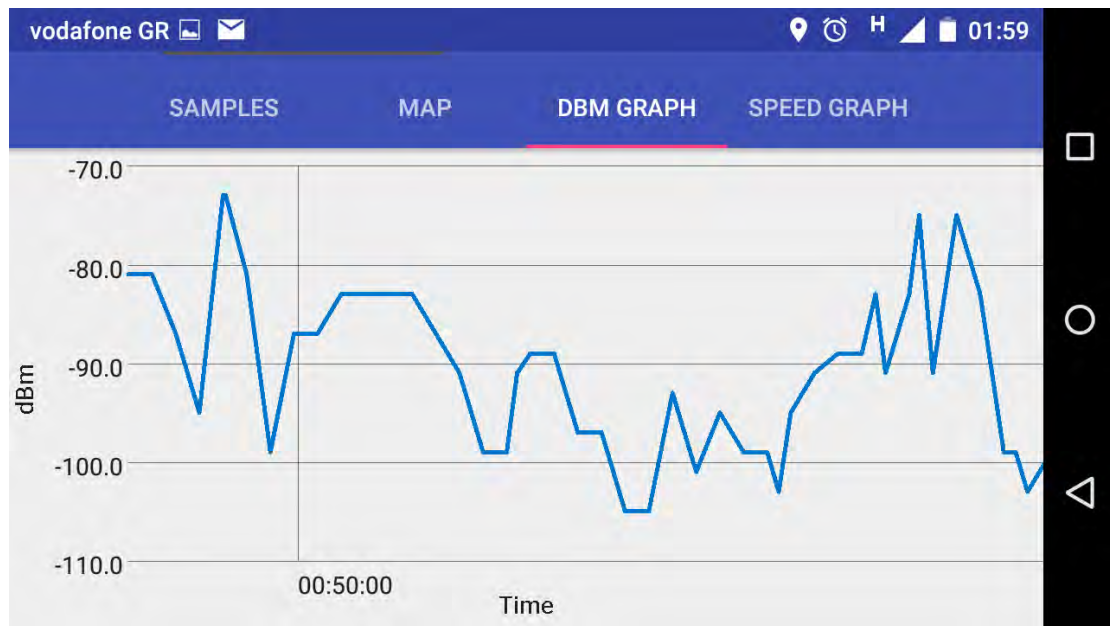


Δοκιμή upload και download. Με αυτοκίνητο. Duration: 240sec, Sample interval:6sec

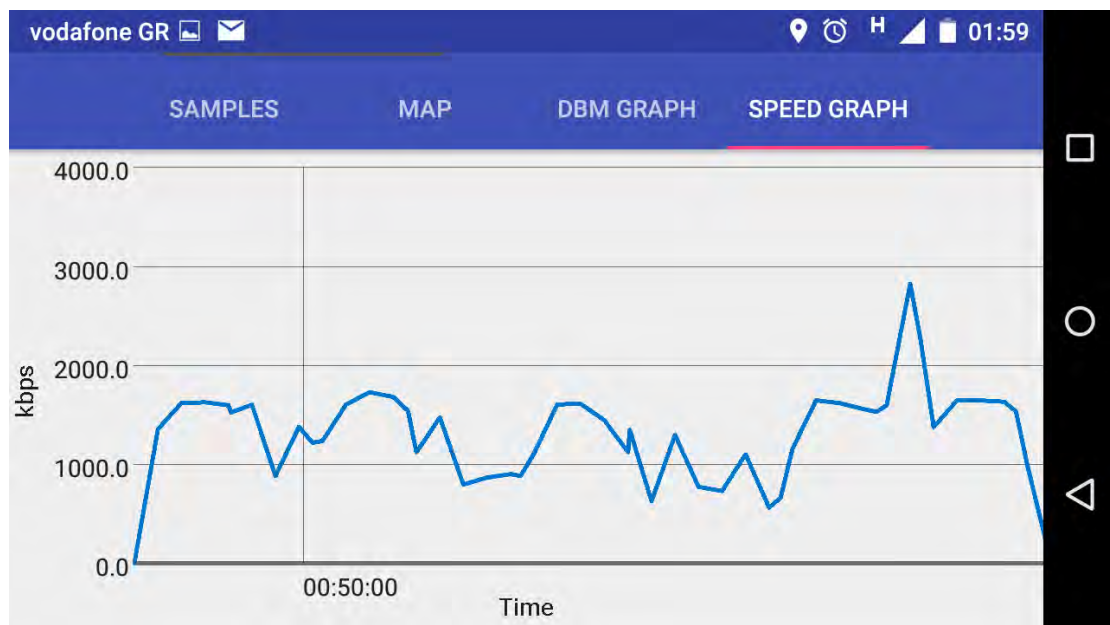


Ακολουθούν τα γραφήματα για την τελευταία διαδρομή

Ισχύς σήματος



Ταχύτητα μεταφοράς



5 Συμπεράσματα και Μελλοντικές Επεκτάσεις

5.1 Συμπεράσματα

Η εφαρμογή Mobile Signal Tester που δημιουργήθηκε σε περιβάλλον Android παρέχει στους χρήστες άμεσα πληροφορίες για την ποιότητα μεταφοράς δεδομένων σε πραγματικό χρόνο. Το πετυχαίνει διενεργώντας τεστ ταχύτητας μεταφοράς δεδομένων σε χρονικό διάστημα και περιοχή που ορίζει ο χρήστης. Έτσι, μπορεί να δώσει λύση σχετικά με την επιλογή του παρόχου κινητής τηλεφωνίας ο οποίος και θα προσφέρει την καλύτερη δυνατή εμπειρία στις καθημερινές ανάγκες στις περιοχές που τους ενδιαφέρουν.

Η συλλογή δεδομένων μπορεί να βοηθήσει και τους παρόχους κινητών επικοινωνιών, στον έλεγχο και τη συντήρηση των δικτύων, καθώς και στην αντιμετώπιση προβλημάτων στις υπάρχουσες εγκαταστάσεις. Ακόμη, βασιζόμενοι στα αποτελέσματα των μετρήσεων, οι πάροχοι μπορούν να αποφασίσουν ποια σημεία του δικτύου χρήζουν αναβάθμισης ή επέκτασης.

5.2 Μελλοντικές Επεκτάσεις

Υπάρχουν αρκετές σκέψεις για την προσθήκη περισσότερων δυνατοτήτων στην εφαρμογή. Ορισμένες από αυτές είναι:

- Καθορισμός αρχείου Upload από τον χρήστη. Το αρχείο που «κατεβαίνει» στο κινητό βρίσκεται στον ftp server και έτσι μπορούμε να το αντικαταστήσουμε με όποιο αρχείο, οποιουδήποτε μεγέθους θέλουμε. Υπάρχει ήδη στις ρυθμίσεις η επιλογή «Remote File Path» που στην ουσία είναι το όνομα του αρχείου που θα ζητήσει η εφαρμογή από τον ftp server. Δεν συμβαίνει όμως το ίδιο για το αρχείο που «ανεβαίνει» στον server. Αυτή τη στιγμή είναι ενσωματωμένο στην εφαρμογή (test_upload.mp3, 3.1MB) και δεν υπάρχει δυνατότητα να αλλάξει. Σε μελλοντική έκδοση μπορεί να δίνεται η επιλογή στον χρήστη να διαλέγει ο ίδιος ένα αρχείο για upload από αυτά που βρίσκονται στην κινητή του συσκευή.
- Επιλογή δικτύου για αποκλειστικό έλεγχο (π.χ. μόνο LTE)
- Επιλογή αποχρώσεων και μέγιστων τιμών μεταφοράς για τα όρια των αποχρώσεων. Αν θέλουμε να κάνουμε έλεγχο συγκεκριμένου δικτύου, καλό

είναι οι αποχρώσεις των μετρήσεων να προσαρμόζονται ανάλογα με τις δυνατότητες του κάθε δικτύου για σωστότερη απεικόνιση. Αυτό θα μπορεί να γίνεται είτε αυτόματα είτε να δίνεται η επιλογή στον χρήστη. Π.χ. να ορίζει ο χρήστης ότι το χρώμα για το μέγιστο Upload (καθαρό μπλε) θα δίνεται σε τιμές Upload από 10000kbps και πάνω.

- Εισαγωγή .txt αρχείων από μετρήσεις που έγιναν σε άλλα κινητά
- Σύγκριση μετρήσεων της ίδιας περιοχής. Μπορούμε να συγκινούμε τιμές σε διάφορες ώρες της ημέρας ακολουθώντας την ίδια πάντα διαδρομή, Για να διαπιστώσουμε αν υπάρχει πρόβλημα με congestion τις ώρες αιχμής και πόσο σοβαρό είναι. Επίσης, μπορούμε να κάνουμε στην ίδια διαδρομή παράλληλο έλεγχο με δύο κινητές συσκευές για σύγκριση του δικτύου δύο παρόχων στην συγκεκριμένη περιοχή. Η σύγκριση για τον χρήστη θα είναι είτε ο παραλληλισμός της απεικόνισης των δύο μετρήσεων είτε καλύτερα μπορούμε να παρουσιάζουμε σε ένα χάρτη την διαφορά των μετρήσεων ανά sample. Δηλαδή, αν ο ένας πάροχος σε ένα σημείο μας δώσει download 6000kbps και ο άλλος 4000kbps, στον τελικό χάρτη που θα δημιουργείται για το συγκεκριμένο σημείο θα απεικονίζεται η διαφορά τους 2000kbps. Έτσι θα έχουμε μια εικόνα για μια διαδρομή σε ποια σημεία υπερτερεί ο κάθε πάροχος και κατά πόσο.

Βιβλιογραφία

- [1] WIRELESS WANs. ΑΣΥΡΜΑΤΑ ΔΙΚΤΥΑ ΕΥΡΕΙΑΣ ΠΕΡΙΟΧΗΣ. [Online]. http://conta.uom.gr/conta/ekpaideysh/metaptyxiaka/technologies_diktywn/teaching_m/WirelessNetworks-Web/Chapter21.html#_Toc872042
- [2] Μιχαήλ Κωνσταντίνος, «Εφαρμογή καταγραφής ποιότητας σήματος σε δίκτυα κινητών επικοινωνιών», διπλωματική εργασία, Λαμία, 2016
- [3] Δρ. Ιωάννης Κορίνθιος, Σημειώσεις Μαθήματος «Ειδικά Θέματα Δικτύων Κινητών Επικοινωνιών», Λαμία, Εαρινό Εξάμηνο 2015-2016
- [4] Cellular Concepts and Basics. [Online]. http://www.radio-electronics.com/info/cellulartelecomms/cellular_concepts/mobile-basics-concepts.php
- [5] R.R.Karhe, M. A. Aher C. S. Patil, "Development of Mobile Technology: A Survey," International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, November 2012.
- [6] Anand Vardhan Bhalla Mudit Ratana Bhalla, "Generations of Mobile Wireless Technology:A Survey," International Journal of Computer Applications, August 2010.
- [7] Principles of Mobile Communication. [Online]. https://books.google.gr/books?hl=el&lr&id=rSz1-HtBQQkC&oi=fnd&pg=PR1&dq=qos+mobile+signal+measurement&ots=Gz0pV523Qa&sig=G1h_tdnabg5iC3J37jlnMcn_yh4&redir_esc=y#v=onepage&q&f=false
- [8] Dr. Anwar M. Mousa, "Prospective of Fifth Generation Mobile Communications," International Journal of Next-Generation Networks (IJNGN), September 2012.
- [9] Chun-Ting Chou, "Adaptive Quality-of-Service Provisioning in Wireless and Mobile Networks," 2005.
- [10] Dushyanth Balasubramanian, "QoS in Cellular Networks," September 2006.
- [11] Θεώνη Κωστάκη, «Σχήματα Χρέωσης για Υπηρεσίες που υποστηρίζονται από Κυψελωτά Δίκτυα Τρίτης Γενιάς» ΑΝΩΤΑΤΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ

[12] Χαράλαμπος Ν. Πήτα, "Μετρήσεις Χαρακτηρισμού και Μοντέλα Πρόβλεψης Ποιότητας Εμπειρίας (QoE) και Ποιότητας Υπηρεσιών (QoS) σε Σύγχρονα Ευρυζωνικά Δίκτυα Κινητών Επικοινωνιών," Εθνικό Μετσόβιο Πολυτεχνείο Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών Τομέας Συστημάτων Μετάδοσης Πληροφορίας και Τεχνολογίας Υλικών , Αθήνα, Διδακτορική Διατριβή 2012.

[13] Εθνική Επιτροπή Τηλεπικοινωνιών και Ταχυδρομείων (ΕΕΤΤ), Δημόσια Διαβούλευση αναφορικά με τον καθορισμό των προς μέτρηση παραμέτρων ποιότητας των παρεχόμενων προς το κοινό υπηρεσιών ηλεκτρονικών επικοινωνιών, τον προσδιορισμό του περιεχομένου και της μορφής των προς δημοσίευση πληροφοριών καθώς και του τρόπου, 2007.

[14] Android Open Source Project. [Online].
<https://source.android.com/source/build-numbers.html>

[15] Κρουσταλάκης Νικόλαος, "Εφαρμογή Android για λήψη ανακοινώσεων ("Android application for receiving announcements")," Τ.Ε.Ι Κρήτης Σχολή Τεχνολογικών Εφαρμογών Τμήμα Εφαρμοσμένης Πληροφορικής και Πολυμέσων, Ηράκλειο, Πτυχιακή Εργασία 2015.

Παράρτημα - Πηγαίος Κώδικας

```
package com.nikmix.mobilesignaltester.homeactivity;

import android.content.Intent;
import android.os.Bundle;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;

import com.nikmix.mobilesignaltester.R;
import com.nikmix.mobilesignaltester.settings.SettingsActivity;
import com.nikmix.mobilesignaltester.testactivity.TestActivity;
import com.nikmix.mobilesignaltester.testactivity.location.LocationProvider;
import com.nikmix.mobilesignaltester.testactivity.sampletask.GetSamplesTask;
import com.nikmix.mobilesignaltester.testactivity.signal.RealMobileDataSignalProvider;

public class HomeActivity extends AppCompatActivity implements
TestsListFragment.OnListFragmentInteractionListener {
    public static final String DATE_FORMAT = "yyyy-MM-dd HH.mm.ss";
    private static final String DEBUG_TAG = "nikmix_debug";

    public static final String ARG_EXISTING_TEST_FILE_NAME =
"ARG_EXISTING_TEST_FILE_NAME";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        LocationProvider.instantiate(getApplicationContext());
        RealMobileDataSignalProvider.instantiate(getApplicationContext());

        setContentView(R.layout.home_activity_activity_home);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        Button newTestButton = (Button) findViewById(R.id.new_test_button);
        newTestButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //Takes us to TestActivity without setting the testFileName argument
                //which results in an empty TestActivity (without any loaded samples)
                Intent intent = new Intent(HomeActivity.this, TestActivity.class);
                //intent.putExtra(ARG_EXISTING_TEST_FILE_NAME, null);
                startActivity(intent);
            }
        });
    }
}
```

```

@Override
public void onResume() {
    super.onResume();
    Button newTestButton = (Button) findViewById(R.id.new_test_button);
    if (GetSamplesTask.Instance != null) { //A test activity is already running
        newTestButton.setText(R.string.go_to_test);
    } else {
        newTestButton.setText(R.string.new_test);
    }
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_home, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
        Intent intent = new Intent(this, SettingsActivity.class);
        startActivity(intent); //Opens the SettingsActivity
        return true;
    }

    return super.onOptionsItemSelected(item);
}

//Function called each time we press a TestPreview list-item
@Override
public void onListFragmentInteraction(TestPreview item) {
    if (GetSamplesTask.Instance == null) {
        Intent intent = new Intent(this, TestActivity.class);
        //Adds the existing test file-name as argument for the TestActivity
        intent.putExtra(ARG_EXISTING_TEST_FILE_NAME, item.fileName);
        startActivity(intent); //Takes you to the TestActivity
    } else {
        Snackbar.make(findViewById(R.id.home_activity_view), R.string.test_in_progress,
        Snackbar.LENGTH_LONG).setAction("", null).show();
    }
}

//General function for debugging that adds a specific tag in order to

```

```

//filter the output inside the Android Monitor through the (nikmix_debug|Exception) filter
public static void println(String text) {
    Log.println(Log.DEBUG, DEBUG_TAG, text);
}

package com.nikmix.mobilesignaltester.homeactivity;

import android.content.Context;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v7.widget.GridLayoutManager;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com.nikmix.mobilesignaltester.R;

import java.io.File;
import java.io.FilenameFilter;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Comparator;
import java.util.List;

/**
 * A fragment representing a list of Items.
 * <p/>
 * Activities containing this fragment MUST implement the {@link
 * OnListFragmentInteractionListener}
 * interface.
 */
public class TestsListFragment extends Fragment {

    // TODO: Customize parameter argument names
    private static final String ARG_COLUMN_COUNT = "column-count";
    // TODO: Customize parameters
    private int mColumnCount = 1;
    private OnListFragmentInteractionListener mListener;

    /**
     * Mandatory empty constructor for the fragment manager to instantiate the
     * fragment (e.g. upon screen orientation changes).
     */
    public TestsListFragment() {
    }

    // TODO: Customize parameter initialization

```

```

@SuppressWarnings("unused")
public static TestsListFragment newInstance(int columnCount) {
    TestsListFragment fragment = new TestsListFragment();
    Bundle args = new Bundle();
    args.putInt(ARG_COLUMN_COUNT, columnCount);
    fragment.setArguments(args);
    return fragment;
}

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    if (getArguments() != null) {
        mColumnCount = getArguments().getInt(ARG_COLUMN_COUNT);
    }
}

@Override
public void onResume() {
    super.onResume();
    //Here, so that it doesn't have to be refreshed through the TestActivity
    //when a test is added.
    ((RecyclerView) getView()).setAdapter(new
TestPreviewRecyclerViewAdapter(loadTestPreviews(), mListener));
}

//Android Studio generated function
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
    Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.home_activity_fragment_tests_list, container,
false);
    // Set the adapter
    if (view instanceof RecyclerView) {
        Context context = view.getContext();
        RecyclerView recyclerView = (RecyclerView) view;
        if (mColumnCount <= 1) {
            LinearLayoutManager layoutManager = new LinearLayoutManager(context);
            recyclerView.setLayoutManager(layoutManager);
        } else {
            recyclerView.setLayoutManager(new GridLayoutManager(context,
mColumnCount));
        }
    }
    return view;
}

//Android Studio generated function
@Override
public void onAttach(Context context) {

```

```

super.onAttach(context);
if (context instanceof OnListFragmentInteractionListener) {
    mListener = (OnListFragmentInteractionListener) context;
} else {
    throw new RuntimeException(context.toString()
        + " must implement OnListFragmentInteractionListener");
}
}

@Override
public void onDetach() {
    super.onDetach();
    mListener = null;
}

//This function loads all existing test files from the
//application directory and orders them by name (which is also by date).
private List<TestPreview> loadTestPreviews() {
    File dir = getContext().getFilesDir();
    File[] subFiles = dir.listFiles(new FilenameFilter() {
        @Override
        public boolean accept(File dir, String name) {
            return name.endsWith(".txt");
        }
    });
    List<TestPreview> testPreviews = new ArrayList<>();

    if (subFiles != null) {
        Arrays.sort(subFiles, new Comparator<File>() {
            @Override
            public int compare(File o1, File o2) {
                return o1.getName().compareTo(o2.getName());
            }
        });
        for (File file : subFiles) {
            String[] fileInfo = FileInfoParser.getFileInfo(file);
            testPreviews.add(new TestPreview(file.getName(), fileInfo[0],
                Integer.parseInt(fileInfo[1]), Integer.parseInt(fileInfo[2])));
        }
    }
    return testPreviews;
}

/**
 * This interface must be implemented by activities that contain this
 * fragment to allow an interaction in this fragment to be communicated
 * to the activity and potentially other fragments contained in that
 * activity.
 * <p/>
 * See the Android Training lesson <a href=
 * "http://developer.android.com/training/basics/fragments/communicating.html"

```



```

    * >Communicating with Other Fragments</a> for more information.
    */
    public interface OnListFragmentInteractionListener {
        // TODO: Update argument type and name
        void onListFragmentInteraction(TestPreview item);
    }
}

package com.nikmix.mobilesignaltester.settings;

import android.annotation.TargetApi;
import android.content.Context;
import android.content.res.Configuration;
import android.os.Build;
import android.os.Bundle;
import android.preference.ListPreference;
import android.preference.Preference;
import android.preference.PreferenceActivity;
import android.preference.PreferenceFragment;
import android.preference.PreferenceManager;
import android.support.v7.app.ActionBar;
import android.view.MenuItem;

import com.nikmix.mobilesignaltester.R;

import java.util.List;

/**
 * A {@link PreferenceActivity} that presents a set of application settings. On
 * handset devices, settings are presented as a single list. On tablets,
 * settings are split by category, with category headers shown to the left of
 * the list of settings.
 * <p>
 * See <a href="http://developer.android.com/design/patterns/settings.html">
 * Android Design: Settings</a> for design guidelines and the <a
 * href="http://developer.android.com/guide/topics/ui/settings.html">Settings
 * API Guide</a> for more information on developing a Settings UI.
 */
public class SettingsActivity extends AppCompatActivity {
    public static final int TRANSFER_CHUNK_SIZE = 128 * 1024; //128 Kb

    public static final String KEY_SAMPLES_INTERVAL_SECONDS =
"samples_interval_seconds";
    public static final String KEY_TEST_DURATION_SECONDS = "test_duration_seconds";
    public static final String KEY_TRANSFER_BEHAVIOUR = "transfer_behaviour";
    public static final String KEY_IS_DATA_SIGNAL_PROVIDER_SIMULATED =
"is_data_signal_provider_simulated";

```

```

public static final String KEY_IS_FTP_SERVER_SIMULATED = "is_ftp_server_simulated";
public static final String KEY_FTP_SERVER_URL = "ftp_server_url";
public static final String KEY_FTP_SERVER_PORT = "ftp_server_port";
public static final String KEY_FTP_SERVER_USERNAME = "ftp_server_username";
public static final String KEY_FTP_SERVER_PASSWORD = "ftp_server_password";
public static final String KEY_REMOTE_FILE_PATH = "remote_file_path";

/**
 * A preference value change listener that updates the preference's summary
 * to reflect its new value.
 */
private static Preference.OnPreferenceChangeListener
sBindPreferenceSummaryToValueListener =
    new Preference.OnPreferenceChangeListener() {
        @Override
        public boolean onPreferenceChange(Preference preference, Object value) {
            String stringValue = value.toString();

            if (preference instanceof ListPreference) {
                // For list preferences, look up the correct display value in
                // the preference's 'entries' list.
                ListPreference listPreference = (ListPreference) preference;
                int index = listPreference.findIndexOfValue(stringValue);

                // Set the summary to reflect the new value.
                preference.setSummary(
                    index >= 0
                        ? listPreference.getEntries()[index]
                        : null);

            } else {
                // For all other preferences, set the summary to the value's
                // simple string representation.
                preference.setSummary(stringValue);
            }
            return true;
        }
    };

/**
 * Helper method to determine if the device has an extra-large screen. For
 * example, 10" tablets are extra-large.
 */
private static boolean isXLargeTablet(Context context) {
    return (context.getResources().getConfiguration().screenLayout
        & Configuration.SCREENLAYOUT_SIZE_MASK) >=
Configuration.SCREENLAYOUT_SIZE_XLARGE;
}

/**
 * Binds a preference's summary to its value. More specifically, when the

```

```

* preference's value is changed, its summary (line of text below the
* preference title) is updated to reflect the value. The summary is also
* immediately updated upon calling this method. The exact display format is
* dependent on the type of preference.
*
* @see #sBindPreferenceSummaryToValueListener
*/
private static void bindPreferenceSummaryToValue(Preference preference) {
    // Set the listener to watch for value changes.
    preference.setOnPreferenceChangeListener(sBindPreferenceSummaryToValueListener);

    // Trigger the listener immediately with the preference's
    // current value.
    sBindPreferenceSummaryToValueListener.onPreferenceChange(preference,
        PreferenceManager
            .getDefaultSharedPreferences(preference.getContext())
            .getString(preference.getKey(), ""));
}

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setupActionBar();
}

/**
 * Set up the {@link android.app.ActionBar}, if the API is available.
 */
private void setupActionBar() {
    ActionBar actionBar = getSupportActionBar();
    if (actionBar != null) {
        // Show the Up button in the action bar.
        actionBar.setDisplayHomeAsUpEnabled(true);
    }
}

/**
 * {@inheritDoc}
 */
@Override
public boolean onIsMultiPane() {
    return isXLargeTablet(this);
}

/**
 * {@inheritDoc}
 */
@Override
@TargetApi(Build.VERSION_CODES.HONEYCOMB)
public void onBuildHeaders(List<Header> target) {
    loadHeadersFromResource(R.xml.pref_headers, target);
}

```

```

    }

    /**
     * This method stops fragment injection in malicious applications.
     * Make sure to deny any unknown fragments here.
     */
    protected boolean isValidFragment(String fragmentName) {
        return PreferenceFragment.class.getName().equals(fragmentName)
            || TestsPreferenceFragment.class.getName().equals(fragmentName)
            || FtpServerPreferenceFragment.class.getName().equals(fragmentName);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            // Respond to the action bar's Up/Home button
            case android.R.id.home:
                super.onBackPressed();
                return true;
        }
        return super.onOptionsItemSelected(item);
    }

    /**
     * This fragment shows general preferences only. It is used when the
     * activity is showing a two-pane settings UI.
     */
    @TargetApi(Build.VERSION_CODES.HONEYCOMB)
    public static class TestsPreferenceFragment extends PreferenceFragment {
        @Override
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            addPreferencesFromResource(R.xml.pref_tests);
            setHasOptionsMenu(true);

            // Bind the summaries of EditText/List/Dialog/Ringtone preferences
            // to their values. When their values change, their summaries are
            // updated to reflect the new value, per the Android Design
            // guidelines.
            bindPreferenceSummaryToValue(findPreference(KEY_TEST_DURATION_SECONDS));

            bindPreferenceSummaryToValue(findPreference(KEY_SAMPLES_INTERVAL_SECONDS));
            bindPreferenceSummaryToValue(findPreference(KEY_TRANSFER_BEHAVIOUR));
        }

        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
            int id = item.getItemId();
            if (id == android.R.id.home) {
                //startActivity(new Intent(getActivity(), SettingsActivity.class));
                getActivity().onBackPressed();
            }
        }
    }

```

```

        return true;
    }
    return super.onOptionsItemSelected(item);
}
}

/**
 * This fragment shows data and sync preferences only. It is used when the
 * activity is showing a two-pane settings UI.
 */
@TargetApi(Build.VERSION_CODES.HONEYCOMB)
public static class FtpServerPreferenceFragment extends PreferenceFragment {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.pref_ftp_server);
        setHasOptionsMenu(true);

        // Bind the summaries of EditText/List/Dialog/Ringtone preferences
        // to their values. When their values change, their summaries are
        // updated to reflect the new value, per the Android Design
        // guidelines.
        bindPreferenceSummaryToValue(findPreference(KEY_FTP_SERVER_URL));
        bindPreferenceSummaryToValue(findPreference(KEY_FTP_SERVER_PORT));
        bindPreferenceSummaryToValue(findPreference(KEY_FTP_SERVER_USERNAME));
        bindPreferenceSummaryToValue(findPreference(KEY_FTP_SERVER_PASSWORD));
        bindPreferenceSummaryToValue(findPreference(KEY_REMOTE_FILE_PATH));
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        if (id == android.R.id.home) {
            //startActivity(new Intent(getActivity(), SettingsActivity.class));
            getActivity().onBackPressed();
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
}
}

```

```
package com.nikmix.mobilesignaltester.testactivity.datatransfer;
```

```
import android.os.AsyncTask;
```

```
import com.nikmix.mobilesignaltester.homeactivity.HomeActivity;
```

```
import com.nikmix.mobilesignaltester.testactivity.TestActivity;
```

```
import com.nikmix.mobilesignaltester.testactivity.location.LocationProvider;
```

```

import com.nikmix.mobilesignaltester.testactivity.core.SignalSample;
import com.nikmix.mobilesignaltester.testactivity.sampletask.GetSamplesTask;
import com.nikmix.mobilesignaltester.testactivity.signal.MobileDataSignalProvider;

import java.io.Closeable;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.lang.ref.WeakReference;
import java.util.Date;

/**
 * Created by nikmix on 23/4/2017.
 * This class represents a task that downloads or uploads a file to and from an FTP server.
 * It creates a SignalSample whenever a download or an upload completes and also
 * sends an event that updates the progress label of TestActivity whenever a 'chunk'
 * of the file is completed (through the ChunkFinishedListener).
 */
public class DataTransferTask extends AsyncTask<DataTransferTask.TaskProperties,
DataTransferTask.TaskProgress, Boolean> {
    private static final int RETRY_INTERVAL_MILLIS = 3000; //Waiting period before retrying
the transfer after connection is lost

    /**
     * This enum represents the settings parameter that determines whether the
     * test will only download the file, only upload it or both.
     */
    public static enum TransferBehaviour {
        DOWNLOAD_AND_UPLOAD("B"),
        ONLY_DOWNLOAD("D"),
        ONLY_UPLOAD("U");

        private String id;

        TransferBehaviour(String id) {
            this.id = id;
        }

        public static TransferBehaviour getById(String id) {
            for (TransferBehaviour transferBehaviour : TransferBehaviour.values()) {
                if (transferBehaviour.id.equals(id)) {
                    return transferBehaviour;
                }
            }
            return null;
        }
    }

    /**
     * A class that contains the arguments of the DataTransferTask

```

```

*/
public static class TaskProperties {
    public final TransferBehaviour transferBehaviour;
    public final String downloadFilePath; //The remote path to the FTP server file to be
downloaded
    public final String uploadFilePath; //The local path to the file that will get uploaded to
the FTP server
    //It's a 'generator' that creates the OutputStream in case of download
    //not an OutputStream because it has to be recreated inside the task when
    //the last transfer finishes.
    public final OutputStreamGenerator outputStreamGenerator;
    //It's a 'generator' that creates the InputStream in case of upload. It's
    //not an InputStream because it has to be recreated inside the task when
    //the last transfer finishes.
    public final InputStreamGenerator inputStreamGenerator;

    public TaskProperties(TransferBehaviour transferBehaviour, String downloadFilePath,
String uploadFilePath,
        OutputStreamGenerator outputStreamGenerator, InputStreamGenerator
inputStreamGenerator) {
        this.transferBehaviour = transferBehaviour;
        this.downloadFilePath = downloadFilePath;
        this.uploadFilePath = uploadFilePath;
        this.outputStreamGenerator = outputStreamGenerator;
        this.inputStreamGenerator = inputStreamGenerator;
    }
}

/**
 * A class that represents a progress event of the task.
 */
public static class TaskProgress {
    public final SignalSample.TransferState transferState; //Indicates whether the file is
downloading or uploading
    public final float kbps; //The speed of the last chunk that was transferred.
    public final int currentProgress; //The bytes transferred so far
    public final int fileSize; //The total bytes to be transferred

    public TaskProgress(SignalSample.TransferState transferState, float kbps, int
currentProgress, int fileSize) {
        this.transferState = transferState;
        this.kbps = kbps;
        this.currentProgress = currentProgress;
        this.fileSize = fileSize;
    }
}

/**
 * Interface used for generating a new OutputStream in each download iteration
 */
public interface OutputStreamGenerator {

```

```

    public OutputStream getOutputStream(String filePath);
}

/**
 * Interface used for generating a new InputStream in each upload iteration
 */
public interface InputStreamGenerator {
    public InputStream getInputStream(String filePath);
}

/**
 *
 */
private class DefaultChunkFinishedListener implements FtpServer.ChunkFinishedListener {
    private int currentTransferProgress; //This value is read when the download/upload
progress finishes
    private int lastPublishedChunkCurrentProgress;
    private long lastPublishedChunkTimeMillis;

    public DefaultChunkFinishedListener() {
        lastPublishedChunkTimeMillis = System.currentTimeMillis();
    }

    @Override
    public boolean onChunkFinished(int currentProgress, int fileSize) {
        long currentTimeMillis = System.currentTimeMillis();
        lostConnection = false; //Means that if the connection had been lost it has been
restored
        int timeElapsed = (int) (currentTimeMillis - lastPublishedChunkTimeMillis);
        /*
         * We use timeElapsed > 500 so that more frequent calls of onChunkFinished
         * get ignored so that the ui isn't updated too frequently.
         */
        if (timeElapsed > 500) {
            /*
             * We publish the transfer progress so that onProgressUpdate
             * will either update the progress label in TestActivity or
             * create a SignalSample if the transfer was completed.
             */
            publishProgress(new TaskProgress(
                isDownloading ? SignalSample.TransferState.DOWNLOAD :
                SignalSample.TransferState.UPLOAD,
                calculateKbps(currentProgress - this.lastPublishedChunkCurrentProgress,
timeElapsed),
                currentProgress, fileSize
            ));
            lastPublishedChunkTimeMillis = currentTimeMillis;
            this.lastPublishedChunkCurrentProgress = currentProgress;
        }
        this.currentTransferProgress = currentProgress;
        if (isCancelled()) {

```



```

        return false; //This value is used by the FTP server to stop the transfer progress
    }
    return true; //Returning true means that the transfer process should continue.
}

/**
 * Normally, when read below, it returns the total file size of the transfer
 */
public int getCurrentTransferProgress() {
    return currentTransferProgress;
}
}

private final FtpServer ftpServer; //The FTP server object doing the transfer.
private final MobileDataSignalProvider mobileDataSignalProvider; //Returns the network
type and the dbm of the mobile device

private WeakReference<TestActivity> testActivityWeakReference; //The TestActivity to be
notified for each new SignalSample

//The task that reads the samples periodically. It is also used to store
//new SignalSamples in case the parent TestActivity has been destroyed.
private GetSamplesTask getSamplesTask;

private boolean isDownloading;
private boolean lostConnection; //Indicates if the connection is at a 'lost-connection' state

/**
 * It stores the last speed result assigned by the onProgressUpdate method.
 * If that result hasn't changed since the last 'read' from the GetSamplesTask,
 * the getLastSpeedResult method returns 'Unknown' speed (-2). Otherwise it
 * returns the speed during the interval after the last SignalSample was created.
 */
private float lastSpeedResult;
private boolean wasSpeedUpdatedAfterLastMeasurement; //Indicates whether the
lastSpeedResult was updated after the last 'read' by getLastSpeedResult

/**
 * Indicates the timestamp of the last call of getLastSpeedResult
 */
private long lastSpeedMeasurementMillis;

/**
 * Indicates the bytes transferred after the last call of getLastSpeedResult
 */
private int bytesTransferredAfterLastMeasurement;

/**
 * Is used to help us calculate the bytesTransferredAfterLastMeasurement variable's value.
 * It stores the progress of the last chunk so that we add the difference with the current
chunk
 * to bytesTransferredAfterLastMeasurement.
 */

```

```

private int previousProgress;

public DataTransferTask(WeakReference<TestActivity> testActivityWeakReference,
    GetSamplesTask getSamplesTask,
    FtpServer ftpServer, MobileDataSignalProvider mobileDataSignalProvider) {
    this.testActivityWeakReference = testActivityWeakReference;
    this.getSamplesTask = getSamplesTask;
    this.ftpServer = ftpServer;
    this.mobileDataSignalProvider = mobileDataSignalProvider;
}

@Override
protected Boolean doInBackground(TaskProperties... taskProperties) {
    TaskProperties taskPropertiesObject = taskProperties[0];
    isDownloading = taskPropertiesObject.transferBehaviour !=
TransferBehaviour.ONLY_UPLOAD;
    restartMeasurement();
    previousProgress = 0;
    while (true) {
        //The first progressUpdate indicating the beginning of the transfer.
        publishProgress(new TaskProgress(isDownloading ?
SignalSample.TransferState.DOWNLOAD :
SignalSample.TransferState.UPLOAD, -2, 0, -1));
        Closeable stream = null;

        String filePath = isDownloading ? taskPropertiesObject.downloadFilePath :
taskPropertiesObject.uploadFilePath;

        /*
        * Here we calculate the remote file's name without its path.
        */
        int lastIndexOfSeparator = filePath.lastIndexOf(File.separator);
        String fileName;
        if (lastIndexOfSeparator == -1) {
            fileName = filePath;
        } else {
            if (lastIndexOfSeparator == filePath.length() - 1) {
                fileName = "";
            } else {
                fileName = filePath.substring(lastIndexOfSeparator + 1);
            }
        }
        try {
            DefaultChunkFinishedListener chunkFinishedListener = new
DefaultChunkFinishedListener();
            long startTimeMillis = System.currentTimeMillis();

            boolean successful;
            if (isDownloading) {
                OutputStream outputStream;

```

```

        stream = outputStream =
taskPropertiesObject.outputStreamGenerator.getOutputStream(fileName);
        successful = ftpServer.downloadFile(outputStream, filePath,
chunkFinishedListener);
    } else {
        InputStream inputStream;
        stream = inputStream =
taskPropertiesObject.inputStreamGenerator.getInputStream(fileName);
        successful = ftpServer.uploadFile(inputStream, filePath, chunkFinishedListener);
    }
    if (!isCancelled() && !getSamplesTask.isFinished() && successful) {
        //Here the download/upload has finished so we create a SignalSample indicating
that.
        publishProgress(new TaskProgress(isDownloading ?
SignalSample.TransferState.DOWNLOAD_COMPLETE :
SignalSample.TransferState.UPLOAD_COMPLETE,
calculateKbps(chunkFinishedListener.getCurrentTransferProgress()), (int)
(System.currentTimeMillis() - startTimeMillis)),
        chunkFinishedListener.getCurrentTransferProgress(),
        chunkFinishedListener.getCurrentTransferProgress()));
    }

} catch (IOException e) {
    //Means the connection was lost - the transfer failed.
    //After this code, we restart the same process (upload or download), unless
cancelled
    e.printStackTrace();
    if (!lostConnection) {
        if (!isCancelled() && !getSamplesTask.isFinished()) {
            publishProgress(new TaskProgress(isDownloading ?
SignalSample.TransferState.DOWNLOAD :
SignalSample.TransferState.UPLOAD, -2, -1, -1));
        }
        lostConnection = true;
    }
    try {
        Thread.sleep(RETRY_INTERVAL_MILLIS);
    } catch (InterruptedException e1) {
        e1.printStackTrace();
    }
} finally {
    try {
        if (stream != null) {
            stream.close();
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
if (isCancelled()) {
    //HomeActivity.println("Cancelled");

```

```

        return false;
    }
    //If the transfer behaviour is download-and-upload, we alter the next transfer
    if (taskPropertiesObject.transferBehaviour ==
TransferBehaviour.DOWNLOAD_AND_UPLOAD) {
        //lastSpeedResult = -2; //Unknown
        isDownloading = !isDownloading;
    }
}
}

/**
 * Here we handle the progress-update events
 * <p>
 * Typically, all System.currentTimeMillis() calls should be made in the
 * doInBackground method, but we assume that the delay between publishProgress
 * and onProgressUpdate is negligible.
 */
@Override
protected void onProgressUpdate(TaskProgress... taskProgresses) {
    //Ignore any progress updates after cancellation or after the GetSamplesTask has
    finished successfully
    if (isCancelled() || getSamplesTask.isFinished()) {
        return;
    }
    TaskProgress taskProgress = taskProgresses[0];
    TestActivity testActivity = testActivityWeakReference.get();
    //HomeActivity.println("onProgressUpdate " + testActivityWeakReference);

    if (taskProgress.currentProgress == -1) { //Connection dropped
        //add connection dropped 'sample'
        SignalSample signalSample = new SignalSample(new
Date(System.currentTimeMillis()),
        LocationProvider.getInstance().getLastKnownLocation(),
        mobileDataSignalProvider.getProtocol().getName(),
        mobileDataSignalProvider.getDbm(),
        taskProgress.transferState, -1); //Speed of -1 displays 'Connection Lost' in the
samples list
        /*
        * If the Activity has been destroyed, the sample will be added to the
        * GetSamplesTask which works as a temporary storage for when the TestActivity
        * has been destroyed.
        */
        if (testActivity != null) {
            testActivity.addSignalSample(signalSample); //Here, taskProgress.kbps = -2 so that
next measuring will be 'Unknown'
        } else {
            getSamplesTask.addSignalSample(signalSample);
        }

        lastSpeedResult = taskProgress.kbps; //We set the lastSpeedResult to -2

```

```

//so that the next measurement doesn't have a previous invalid value.
restartMeasurement();
previousProgress = 0; //Reset the progress for the next file transfer.
} else {
    long now = System.currentTimeMillis();
    /*
     * Here we create a download/upload-complete sample that has a speed value
     * of the average speed of the whole downloaded/uploaded file, not just during
     * the time after the last sample.
     */
    boolean transferComplete = taskProgress.transferState ==
SignalSample.TransferState.DOWNLOAD_COMPLETE ||
    taskProgress.transferState == SignalSample.TransferState.UPLOAD_COMPLETE;
    if (transferComplete) {
        SignalSample signalSample = new SignalSample(new Date(now),
            LocationProvider.getInstance().getLastKnownLocation(),
            mobileDataSignalProvider.getProtocol().getName(),
            mobileDataSignalProvider.getDbm(),
            taskProgress.transferState, taskProgress.kbps);
        if (testActivity != null) {
            testActivity.addSignalSample(signalSample);
        } else {
            getSamplesTask.addSignalSample(signalSample);
        }
        restartMeasurement();
        previousProgress = 0; //Reset the progress for the next file transfer.
    }
    if (testActivity != null) {
        //Here we update the transfer-progress label of TestActivity
        testActivity.updateTransferProgress(taskProgress, isDownloading);
    }
    if (transferComplete) { //So that if GetSamplesTask measures right after transfer-
complete sample, the speed will be 'Unknown'
        lastSpeedResult = -2; //Unknown
    } else if (taskProgress.kbps > -2) { //Not unknown
        //Here we add to the transferred bytes the value of the last chunk
        bytesTransferredAfterLastMeasurement += (taskProgress.currentProgress -
previousProgress);
        //We store the current progress to use it to calculate the next chunk's size
        previousProgress = taskProgress.currentProgress;
        //We calculate the speed of the transfer since the last measurement
        lastSpeedResult = calculateKbps(bytesTransferredAfterLastMeasurement,
            (int) (now - lastSpeedMeasurementMillis));
    } else {
        lastSpeedResult = taskProgress.kbps; //-2 = 'Unknown'
        if (taskProgress.fileSize == -1) { //If it's the first time we update the progress for that
transfer
            restartMeasurement();
            previousProgress = 0; //Will probably already be 0.
        }
    }
}
}

```

```

        //lastSpeedResult = taskProgress.kbps;
    }
    wasSpeedUpdatedAfterLastMeasurement = true; //This value is used to indicate
    //whether the next measurement value will have a valid or an
    //unknown speed.
}

/**
 * This method is never called 'natively'
 * We only call it by onCancelled
 */
@Override
protected void onPostExecute(Boolean result) {
    TestActivity testActivity = testActivityWeakReference.get();
    if (testActivity != null) {
        testActivity.clearTransferProgress(); //We clear the progress label of TestActivity
    }
}

@Override
protected void onCancelled(Boolean result) {
    onPostExecute(result);
    //HomeActivity.println("DataTransferTask.onCancelled()");
}

public boolean isDownloading() {
    return isDownloading;
}

public float getLastSpeedResult() {
    restartMeasurement();
    if (wasSpeedUpdatedAfterLastMeasurement) { //We have new information since the
last measurement
        wasSpeedUpdatedAfterLastMeasurement = false;
        return lastSpeedResult;
    } else {
        return -2; //Unknown
    }
}

/**
 * Used when a new TestActivity is created while the test is still running.
 * It's called after a TestActivity has been destroyed
 */
public void setTestActivityWeakReference(WeakReference<TestActivity>
testActivityWeakReference) {
    this.testActivityWeakReference = testActivityWeakReference;
}

/**
 * Called when a new sample is created and when a new download/upload starts

```

```

    * so as to dismiss the previously stored transfer data.
    */
    private void restartMeasurement() {
        lastSpeedMeasurementMillis = System.currentTimeMillis();
        bytesTransferredAfterLastMeasurement = 0;
    }

    /**
     * Calculates the transfer speed in kbps
     */
    private float calculateKbps(int currentProgress, int timeElapsedMillis) {
        return (currentProgress * 8f / 1024) / (Math.max(1, timeElapsedMillis) / 1000f);
    }
}

package com.nikmix.mobilesignaltester.testactivity.datatransfer;

import com.nikmix.mobilesignaltester.homeactivity.HomeActivity;

import org.apache.commons.net.ftp.FTP;
import org.apache.commons.net.ftp.FTPClient;
import org.apache.commons.net.io.CopyStreamEvent;
import org.apache.commons.net.io.CopyStreamListener;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

/**
 * Created by nikmix on 4/5/2017.
 * A real implementation of an FTP server that uses the Apache library
 */

public class RealFtpServer implements FtpServer {
    private final String url;
    private final int port;
    private final String username;
    private final String password;
    private final int chunkSize;

    public RealFtpServer(String url, int port, String username, String password, int chunkSize) {
        this.url = url;
        this.port = port;
        this.username = username;
        this.password = password;
        this.chunkSize = chunkSize;
    }

    @Override

```

```

    public boolean downloadFile(OutputStream outputStream, String remoteFilePath, final
ChunkFinishedListener chunkFinishedListener) throws IOException {
        FTPClient ftpClient = createFtpClient(chunkFinishedListener);
        ftpClient.setBufferSize(chunkSize); //The bytes between calls to the CopyStreamListener
        ftpClient.setReceiveBufferSize(chunkSize);
        ftpClient.setReceiveDataSocketBufferSize(chunkSize);
        try {
            ftpClient.connect(url, port);
            ftpClient.login(username, password);
            ftpClient.enterLocalPassiveMode();
            ftpClient.setFileType(FTP.BINARY_FILE_TYPE);

            boolean success = ftpClient.retrieveFile(remoteFilePath, outputStream); //Method for
downloading the file

            if (success) {
                //HomeActivity.println("File has been downloaded successfully.");
            } else {
                HomeActivity.println("File failed to download");
            }
            return success;
        } finally {
            outputStream.close();
            try {
                if (ftpClient.isConnected()) { //We disconnect from the server
                    ftpClient.logout();
                    ftpClient.disconnect();
                }
            } catch (IOException ex) {
                ex.printStackTrace();
            }
        }
    }
}

```

```

@Override
    public boolean uploadFile(InputStream inputStream, String remoteFilePath, final
ChunkFinishedListener chunkFinishedListener) throws IOException {
        FTPClient ftpClient = createFtpClient(chunkFinishedListener);
        ftpClient.setBufferSize(chunkSize); //The bytes between calls to the CopyStreamListener
        ftpClient.setSendBufferSize(chunkSize);
        ftpClient.setSendDataSocketBufferSize(chunkSize);
        try {
            ftpClient.connect(url, port);
            ftpClient.login(username, password);
            ftpClient.enterLocalPassiveMode();
            ftpClient.setFileType(FTP.BINARY_FILE_TYPE);
            boolean success = ftpClient.storeFile(remoteFilePath, inputStream); //Method for
uploading the file

            if (success) {
                //HomeActivity.println("File has been uploaded successfully.");
            }
        }
    }
}

```



```

    } else {
        HomeActivity.println("File failed to upload");
    }
    return success;
} finally {
    inputStream.close();
    try {
        if (ftpClient.isConnected()) { //We disconnect from the server
            ftpClient.logout();
            ftpClient.disconnect();
        }
    } catch (IOException ex) {
        ex.printStackTrace();
    }
}
}

private FTPClient createFtpClient(final ChunkFinishedListener chunkFinishedListener) {
    final FTPClient ftpClient = new FTPClient();
    ftpClient.setCopyStreamListener(new CopyStreamListener() {
        @Override
        public void bytesTransferred(CopyStreamEvent event) {
            bytesTransferred(event.getTotalBytesTransferred(), event.getBytesTransferred(),
event.getStreamSize());
        }

        /**
         *
         * @param totalBytesTransferred
         * @param bytesTransferred
         * @param streamSize
         */
        @Override
        public void bytesTransferred(long totalBytesTransferred,
            int bytesTransferred, long streamSize) {
            //HomeActivity.println("bytesTransferred " + totalBytesTransferred + " " +
bytesTransferred + " " + streamSize);

            //Here we call the chunkFinishedListener to indicate that some bytes were
transferred.
            //Returns whether the user didn't cancel the test during the transfer
            boolean shouldCancel = !chunkFinishedListener.onChunkFinished((int)
totalBytesTransferred,
                (int) streamSize);
            if (shouldCancel) {
                try {
                    ftpClient.abort(); //Aborts the transfer
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}

```

```

        }
    });
    return ftpClient;
}
}

package com.nikmix.mobilesignaltester.testactivity.sampletask;

import android.os.AsyncTask;

import com.nikmix.mobilesignaltester.testactivity.TestActivity;
import com.nikmix.mobilesignaltester.testactivity.datatransfer.DataTransferTask;
import com.nikmix.mobilesignaltester.testactivity.datatransfer.FtpServer;
import com.nikmix.mobilesignaltester.testactivity.location.LocationProvider;
import com.nikmix.mobilesignaltester.testactivity.core.SignalSample;
import com.nikmix.mobilesignaltester.testactivity.signal.MobileDataSignalProvider;

import java.lang.ref.WeakReference;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

/**
 * Created by nikmix on 21/4/2017.
 * This class represents an AsyncTask that creates a SignalSample every x seconds.
 * It also handles the start and cancellation of a DataTransferTask that transfers
 * files to and from an FTP server.
 */
public class GetSamplesTask extends AsyncTask<GetSamplesTask.TaskProperties,
SignalSample, Boolean> {
    //A global instance so that the reference of the task is not lost
    //if the activity that started it gets destroyed
    public static GetSamplesTask Instance;

    /**
     * A class containing the execution arguments of the task
     */
    public static class TaskProperties {
        public final float intervalMillis; //Milliseconds between samples
        public final float durationMillis; //Test duration

        public TaskProperties(float intervalMillis, float durationMillis) {
            this.intervalMillis = intervalMillis;
            this.durationMillis = durationMillis;
        }
    }

    //The TestActivity where the test data is displayed

```

```

private WeakReference<TestActivity> testActivityWeakReference;
//The list that stores the SignalSamples when the TestActivity is destroyed
//so that they can be 'reattached' to the TestActivity once it's recreated.
private List<SignalSample> signalSamples;

//The object that provides the dBm and network type of the mobile device
private final MobileDataSignalProvider mobileDataSignalProvider;
//The 'child-task' that handles the transfer of the files.
private DataTransferTask dataTransferTask;

//The start time of the test
private long startTime;
//A boolean to indicate whether the task has finished or not
//(regardless if it finished successfully or was cancelled)
private boolean isFinished;

/**
 * Constructor holding the objects that get used throughout each execution
 *
 * @param testActivity      The activity to be notified when a sample is created
 * @param ftpServer         The FTP server used for the file-transfer
 * @param mobileDataSignalProvider The dBm and network type provider
 */
public GetSamplesTask(TestActivity testActivity, FtpServer ftpServer,
MobileDataSignalProvider mobileDataSignalProvider) {
    Instance = this;
    this.testActivityWeakReference = new WeakReference<>(testActivity);
    this.mobileDataSignalProvider = mobileDataSignalProvider;
    //We create the DataTransferTask to be used for file-transfer
    dataTransferTask = new DataTransferTask(testActivityWeakReference, this, ftpServer,
mobileDataSignalProvider);
}

/**
 * Method that executes both tasks simultaneously
 *
 * @param taskProperties      The execution arguments of the GetSamplesTask
 * @param dataTransferTaskProperties The execution arguments of the DataTransferTask
 */
public void execute(TaskProperties taskProperties, DataTransferTask.TaskProperties
dataTransferTaskProperties) {
    signalSamples = new ArrayList<>(); //Create a new list of samples
    isFinished = false;
    executeOnExecutor(THREAD_POOL_EXECUTOR, taskProperties);
    dataTransferTask.executeOnExecutor(THREAD_POOL_EXECUTOR,
dataTransferTaskProperties);
}

/**
 * The main task function
 */

```

```

* @return whether it was cancelled or finished successfully
*/
@Override
protected Boolean doInBackground(TaskProperties... taskProperties) {
    startTime = System.currentTimeMillis();
    TaskProperties taskPropertiesObject = taskProperties[0];

    out:
    //Label so that we can break to it from an second level loop
    while (true) {
        /*
        * Create SignalSamples periodically and wait for intervalMillis milliseconds
        * before the next creation
        */
        publishProgress(new SignalSample(new Date(System.currentTimeMillis()),
            LocationProvider.getInstance().getLastKnownLocation(),
            mobileDataSignalProvider.getProtocol().getName(),
            mobileDataSignalProvider.getDbm(),
            dataTransferTask.isDownloading() ? SignalSample.TransferState.DOWNLOAD :
                SignalSample.TransferState.UPLOAD,
            dataTransferTask.getLastSpeedResult()));
        /*
        * Here, instead of waiting for intervalMillis, we wait multiple times
        * for 500 milliseconds in order to check for cancellation so that we don't
        * have to wait the whole interval for the task to finish.
        */
        int i = 0;
        do {
            if (isCancelled()) {
                //HomeActivity.println("Cancelled");
                return false;
            }
            try {
                Thread.sleep(500);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            /*
            * We finish the test if sufficient time has elapsed
            */
            if (System.currentTimeMillis() - startTime > taskPropertiesObject.durationMillis) {
                break out;
            }
            ++i;
        } while (i < taskPropertiesObject.intervalMillis / 500);
    }
    return true;
}

/**
* Used for adding the new SignalSample to either the TestActivity list

```

```

* or the signalSamples list of this task in case the activity has been
* destroyed.
*
* @param signalSamples the created SignalSample
*/
@Override
protected void onProgressUpdate(SignalSample... signalSamples) {
    TestActivity testActivity = testActivityWeakReference.get();
    if (testActivity != null) {
        //Added to TestActivity which in turn notifies the tab fragments to update their
content.
        testActivity.addSignalSample(signalSamples[0]);
    } else {
        //Added to this AsyncTask's signalSamples list.
        addSignalSample(signalSamples[0]);
    }
}

/**
* Called once the task finishes, we notify the TestActivity to write the test
* to a file.
*
* @param result the result returned from doInBackground
*/
@Override
protected void onPostExecute(Boolean result) {
    TestActivity testActivity = testActivityWeakReference.get();
    if (testActivity != null) {
        testActivity.setTestFinished(true);
    }
    //We cancel the DataTransferTask
    dataTransferTask.cancel(false);
    isFinished = true; //Set the isFinished variable
}

/**
* Called if the activity was cancelled, does the same as onPostExecute
*
* @param result the result returned from doInBackground
*/
@Override
protected void onCancelled(Boolean result) {
    //HomeActivity.println("GetSamplesTask.onCancelled(" + result + ")");
    onPostExecute(result);
}

/**
* Used when the TestActivity gets destroyed or recreated.
*
* @param testActivity If testActivity is null, it means that
* the TestActivity was destroyed during the test.

```

```

*           Otherwise, it means it was recreated so we
*           attach the signalSamples list contents to
*           it.
*/
public void setTestActivity(TestActivity testActivity) {
    if (testActivity != null) {
        //We reattach the signalSamples contents to TestActivity and
        //clear them from the GetSamplesTask
        testActivity.getSignalSamples().addAll(signalSamples);
        signalSamples.clear();
    } else {
        //We copy the TestActivity's samples to the signalSamples list
        //so that they don't get lost during the TestActivity's destruction
        signalSamples.addAll(testActivityWeakReference.get().getSignalSamples());
    }
    testActivityWeakReference = new WeakReference<>(testActivity);
    //We set the DataTransferTask's TestActivity to the new TestActivity (through the
    WeakReference object)
    dataTransferTask.setTestActivityWeakReference(testActivityWeakReference);
}

//Used by TestActivity to check if it has already been attached to the GetSamplesTask
//during the call of the onStart method of TestActivity
public TestActivity getTestActivity() {
    return testActivityWeakReference.get();
}

//Used by DataTransferTask to add samples when connection is lost or when a download
//or upload finishes. It in turn adds the sample to either TestActivity or the
//signalSamples list.
public void addSignalSample(SignalSample signalSample) {
    signalSamples.add(signalSample);
}

//Returns the start-time of the test
public long getStartTime() {
    return startTime;
}

//Returns whether the task has finished execution (regardless of whether it was cancelled
or not)
public boolean isFinished() {
    return isFinished;
}
}

package com.nikmix.mobilesignaltester.testactivity;

```

```

import android.content.Intent;
import android.content.SharedPreferences;
import android.content.res.Resources;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.support.design.widget.Snackbar;
import android.support.design.widget.TabLayout;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentPagerAdapter;
import android.support.v4.view.ViewPager;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.text.format.DateFormat;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import com.nikmix.mobilesignaltester.R;
import com.nikmix.mobilesignaltester.homeactivity.HomeActivity;
import com.nikmix.mobilesignaltester.settings.SettingsActivity;
import com.nikmix.mobilesignaltester.testactivity.datatransfer.DataTransferTask;
import com.nikmix.mobilesignaltester.testactivity.datatransfer.RealFtpServer;
import com.nikmix.mobilesignaltester.testactivity.datatransfer.SimulatedFtpServer;
import com.nikmix.mobilesignaltester.testactivity.graph.GraphFragment;
import com.nikmix.mobilesignaltester.testactivity.io.FileWriterParser;
import com.nikmix.mobilesignaltester.testactivity.core.SignalSample;
import com.nikmix.mobilesignaltester.testactivity.sampleslist.SignalSamplesListFragment;
import com.nikmix.mobilesignaltester.testactivity.sampletask.GetSamplesTask;
import com.nikmix.mobilesignaltester.testactivity.signal.RealMobileDataSignalProvider;
import
com.nikmix.mobilesignaltester.testactivity.signal.SimulatedMobileDataSignalProvider;

import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.Locale;

/**
 * Activity where tests are displayed/executed.
 */
public class TestActivity extends AppCompatActivity implements
SignalSamplesListFragment.OnListFragmentInteractionListener {

```

```

    public static final String UPLOAD_FILE_PATH = "test_upload.mp3"; //The file to upload to
    the FTP server

    //Key used in onCreate in order to de-serialize the list of samples saved in
    //onSaveInstanceState (when pausing) using the same key.
    public static final String KEY_SAMPLES_LIST = "KEY_SAMPLES_LIST";

    /**
     * The {@link android.support.v4.view.PagerAdapter} that will provide
     * fragments for each of the sections. We use a
     * {@link FragmentPagerAdapter} derivative, which will keep every
     * loaded fragment in memory. If this becomes too memory intensive, it
     * may be best to switch to a
     * {@link android.support.v4.app.FragmentStatePagerAdapter}.
     */
    private SectionsPagerAdapter mSectionsPagerAdapter;
    //List of listeners used to notify tab fragments of changes in the list of samples below
    private List<SignalSamplesListener> signalSamplesListeners;

    /**
     * The {@link ViewPager} that will host the section contents.
     */
    private ViewPager mViewPager;
    //Button that starts/cancels a test
    private Button startTestButton;
    //The file name to be loaded if we're loading an existing test.
    //This variable gets its value from the arguments of the Intent that started this activity.
    private String testFileName;
    //The current list of SignalSamples of the test. (if a test is in progress, this list gets bigger in
    time)
    private List<SignalSample> signalSamples;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        //We initialize it before super.onCreate, because the fragments' onCreateView
        //gets called during super.onCreate, and it makes use of this variable.
        signalSamplesListeners = new ArrayList<>();
        super.onCreate(savedInstanceState);
        //HomeActivity.println("onCreate");

        setContentView(R.layout.test_activity_tabs);

        //Set the Toolbar as action bar
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        //When we press the startTestButton we either start a new test or
        //cancel the GetSamplesTask of the existing test.
        startTestButton = (Button) findViewById(R.id.start_test_button);
        startTestButton.setOnClickListener(new View.OnClickListener() {
            @Override

```



```

        public void onClick(View view) {
            if (GetSamplesTask.Instance != null) {
                GetSamplesTask.Instance.cancel(false);
                //We disable it until the GetSamplesTask finishes writing the
                //test into a file.
                startTestButton.setEnabled(false);
            } else {
                setTestFinished(false);
            }
        }
    };

    /*
     * If there's no running task to attach to activity to, we either load
     * the list of SignalSamples from the savedInstanceState bundle (if the
     * activity was paused and recreated) or reload a test from its text file
     * if the TestActivity is created from scratch.
     * If the testFileName is null, loadSignalSamples just initializes the
     * signalSamples list.
     */
    if (!attachToRunningTask(false)) { //The argument is false cause the fragments haven't
        initialized yet.
            if (savedInstanceState != null) {
                signalSamples = (List<SignalSample>)
                savedInstanceState.getSerializable(KEY_SAMPLES_LIST);
            } else {
                testFileName =
                getIntent().getStringExtra(HomeActivity.ARG_EXISTING_TEST_FILE_NAME);
                signalSamples = loadSignalSamples();
            }
        }
        // Create the adapter that wi
        // ll return a fragment for each of the four
        // primary sections of the activity.
        mSectionsPagerAdapter = new SectionsPagerAdapter(getSupportFragmentManager());

        // Set up the ViewPager with the sections adapter.
        mViewPager = (ViewPager) findViewById(R.id.container);
        mViewPager.setAdapter(mSectionsPagerAdapter);

        TabLayout tabLayout = (TabLayout) findViewById(R.id.tabs);
        tabLayout.setupWithViewPager(mViewPager);

        // FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
        // fab.setOnClickListener(new View.OnClickListener() {
        //     @Override
        //     public void onClick(View view) {
        //         Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
        //             .setAction("Action", null).show();
        //     }
        // });

```

```

}

@Override
public void onStart() {
    super.onStart();
    //HomeActivity.println("onStart");
    //Attempts to attach the activity to an existing, running GetSamplesTask.
    //If the task exists, the signalSamples list is refreshed by the
    //fragments by calling the SignalSamplesListeners refreshSamples method
    attachToRunningTask(true);
}

/**
 * Just for debug reasons
 */
@Override
public void onRestart() {
    super.onRestart();
    //HomeActivity.println("onRestart");
}

/**
 * Just for debug reasons
 */
@Override
public void onPause() {
    super.onPause();
    //HomeActivity.println("onPause");
}

/**
 * If a GetSamplesTask is already running, we 'detach' this activity from
 * it, which transfers the signalSamples list to the GetSamplesTask.
 */
@Override
public void onStop() {
    super.onStop();
    //HomeActivity.println("onStop");
    if (GetSamplesTask.Instance != null) {
        GetSamplesTask.Instance.setTestActivity(null);
    }
}

/**
 * Just for debug reasons
 */
@Override
public void onResume() {
    super.onResume();
    //HomeActivity.println("onResume");
}

```

```

/**
 * Just for debug reasons
 */
@Override
public void onDestroy() {
    //HomeActivity.println("onDestroy");
    super.onDestroy();
}

/**
 * If no GetSamplesTask is running, we save the existing signalSamples of the
 * (finished) test inside the bundle as a serialized object - list.
 *
 * @param bundle the Bundle to save the signalSamples list.
 */
@Override
public void onSaveInstanceState(Bundle bundle) {
    super.onSaveInstanceState(bundle);
    //HomeActivity.println("onSaveInstanceState");
    if (GetSamplesTask.Instance == null) {
        bundle.putSerializable(KEY_SAMPLES_LIST, (ArrayList<SignalSample>) signalSamples);
    }
}

/**
 * Method for creating the options menu from an xml file
 *
 * @param menu The menu object that will be the parent of the inflated menu
 */
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_test, menu);
    return true;
}

/**
 * This method is called when a menu item of the options menu is selected
 *
 * @param item The selected item
 * @return if the event has handled
 */
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement

```

```

        if (id == R.id.action_settings) {
            //We open the SettingsActivity
            Intent intent = new Intent(this, SettingsActivity.class);
            startActivity(intent);
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    /**
     * Is called whenever a list item of the SignalSamplesListFragment list is pressed
     *
     * @param item The SignalSample connected to the list-item pressed.
     */
    @Override
    public void onListFragmentInteraction(SignalSample item) {
        //Select map tab and zoom to point...
        mViewPager.setCurrentItem(1); //We change the tab to the TestMapParentFragment
        //We center on the location of the SignalSample on the map
        ((TestMapParentFragment)
        findFragment(TestMapParentFragment.class)).centerOnLocation(item.location);
    }

    /**
     * Adds a SignalSamplesListener to the activity when a tab fragment is created
     */
    public void addSignalSamplesListener(SignalSamplesListener signalSamplesListener) {
        signalSamplesListeners.add(signalSamplesListener);
    }

    /**
     * Removes the added SignalSamplesListener when a tab fragment is destroyed
     */
    public void removeSignalSamplesListener(SignalSamplesListener signalSamplesListener) {
        signalSamplesListeners.remove(signalSamplesListener);
    }

    /**
     * Adds a SignalSample to the signalSamples list and notifies the SignalSamplesListeners
     * so that the fragments-listeners update their contents.
     *
     * @param signalSample the added SignalSample
     */
    public void addSignalSample(SignalSample signalSample) {
        signalSamples.add(signalSample);
        for (SignalSamplesListener signalSamplesListener : signalSamplesListeners) {
            signalSamplesListener.sampleCreated(signalSample);
        }
    }
}

```

```

/**
 * Clears the signalSamples list and notifies the SignalSamplesListeners so
 * that the fragments-listeners update their contents.
 */
public void clearSamples() {
    signalSamples.clear();
    for (SignalSamplesListener signalSamplesListener : signalSamplesListeners) {
        signalSamplesListener.refreshSamples();
    }
}

/**
 * @return the SignalSamples list of the activity.
 */
public List<SignalSample> getSignalSamples() {
    return signalSamples;
}

/**
 * This method updates the transfer progress label that shows the progress
 * of downloads and uploads.
 *
 * @param taskProgress The TaskProgress that contains info about the latest
 * progress update.
 * @param isDownloading Whether it's a download or an upload
 */
public void updateTransferProgress(DataTransferTask.TaskProgress taskProgress, boolean
isDownloading) {
    TextView dataTransferTextView = (TextView) findViewById(R.id.transfer_progress);
    //The result string is displayed in the transfer-progress label.
    String result = (isDownloading ? getString(R.string.downloading) :
getString(R.string.uploading)) + " ";
    String progressString;
    String fileSizeString;
    if (taskProgress.fileSize > 0) { //If the file-size is available it shows the transferred
percentage.
        progressString = String.format(Locale.US, "%.1f", (taskProgress.currentProgress /
(float) taskProgress.fileSize) * 100) + "%";
        fileSizeString = " " + getString(R.string.out_of) +
        " " + (int) Math.ceil(taskProgress.fileSize / 1024f) + " " + getString(R.string.kb);
    } else { //Only shows the Kb transferred
        progressString = (int) Math.ceil(taskProgress.currentProgress / 1024f) + " " +
getString(R.string.kb);
        fileSizeString = "";
    }
    result += progressString + fileSizeString;
    dataTransferTextView.setText(result);
}

//Clears the transfer-progress label
public void clearTransferProgress() {

```

```

        TextView dataTransferTextView = (TextView) findViewById(R.id.transfer_progress);
        dataTransferTextView.setText("");
    }

    /**
     * If the value is false, it starts a new test. Otherwise, it saves the
     * current signalSamples values to a file and enables the startTestButton
     * so that we can start a new test.
     *
     * @param value Argument to either start or end a test.
     */
    public void setTestFinished(boolean value) {
        if (value) {
            //We take the start timestamp of the test from GetSamplesTask
            Date testStartDate = new Date(GetSamplesTask.Instance.getStartTime());
            //Message to indicate the test is being saved.
            Snackbar snackbar = Snackbar.make(findViewById(R.id.main_content),
            R.string.saving_test, Snackbar.LENGTH_INDEFINITE);
            snackbar.setAction("", null).show();
            //We disable the startTestButton until the file gets written
            startTestButton.setEnabled(false);
            FileWriterParser.writeFile(getBaseContext(), testStartDate, signalSamples,
            new File(getBaseContext().getFilesDir(),
            DateFormat.format(HomeActivity.DATE_FORMAT, testStartDate) + ".txt"));
            snackbar.dismiss();
            startTestButton.setEnabled(true); //We re-enable the startTestButton
            GetSamplesTask.Instance = null;
        } else {
            /**
             * Here we start a new GetSamplesTask which commences the test. We
             * use the SettingsActivity shared preferences as parameters.
             */
            SharedPreferences sharedPreferences =
            PreferenceManager.getDefaultSharedPreferences(TestActivity.this);
            clearSamples();
            new GetSamplesTask(this,
            sharedPreferences.getBoolean(SettingsActivity.KEY_IS_FTP_SERVER_SIMULATED,
            false) ?
            new SimulatedFtpServer(SettingsActivity.TRANSFER_CHUNK_SIZE) : new
            RealFtpServer(
            sharedPreferences.getString(SettingsActivity.KEY_FTP_SERVER_URL,
            getResources().getString(R.string.pref_ftp_server_url_default_value)),

            Integer.parseInt(sharedPreferences.getString(SettingsActivity.KEY_FTP_SERVER_PORT,
            getResources().getString(R.string.pref_ftp_server_port_default_value)),
            sharedPreferences.getString(SettingsActivity.KEY_FTP_SERVER_USERNAME,
            getResources().getString(R.string.pref_ftp_server_username_default_value)),
            sharedPreferences.getString(SettingsActivity.KEY_FTP_SERVER_PASSWORD,
            getResources().getString(R.string.pref_ftp_server_password_default_value)),
            SettingsActivity.TRANSFER_CHUNK_SIZE
            ),
            ),

```

```

sharedPreferences.getBoolean(SettingsActivity.KEY_IS_DATA_SIGNAL_PROVIDER_SIMULATE
D, false) ?
    new SimulatedMobileDataSignalProvider() :
    (RealMobileDataSignalProvider.getInstance() == null ? new
SimulatedMobileDataSignalProvider() : RealMobileDataSignalProvider.getInstance())
    ).execute(new GetSamplesTask.TaskProperties(

Integer.parseInt(sharedPreferences.getString(SettingsActivity.KEY_SAMPLES_INTERVAL_SEC
ONDS,

getResources().getString(R.string.pref_samples_interval_seconds_default_value))) * 1000,

Integer.parseInt(sharedPreferences.getString(SettingsActivity.KEY_TEST_DURATION_SECON
DS,

getResources().getString(R.string.pref_test_duration_seconds_default_value))) * 1000
    ),
    new DataTransferTask.TaskProperties(

DataTransferTask.TransferBehaviour.getByld(sharedPreferences.getString(SettingsActivity.K
EY_TRANSFER_BEHAVIOUR,

getResources().getString(R.string.pref_transfer_behaviour_default_value))),
    sharedPreferences.getString(SettingsActivity.KEY_REMOTE_FILE_PATH,

getResources().getString(R.string.pref_remote_file_path_default_value)),
    UPLOAD_FILE_PATH,
    new DataTransferTask.OutputStreamGenerator() { //We don't save the
downloaded file, just put it inside a ByteArrayOutputStream
        @Override
        public OutputStream getOutputStream(String filePath) {
            return new ByteArrayOutputStream();
        }
    }, new DataTransferTask.InputStreamGenerator() {
        @Override
        public InputStream getInputStream(String filePath) {
            try {
                return getResources().openRawResource(R.raw.test_upload);
            } catch (Resources.NotFoundException e) {
                HomeActivity.println("ByteArrayInputStream");
                e.printStackTrace();
                //If the file to be uploaded is not found, we upload a 2Mb 'empty' byte-
array
                return new ByteArrayInputStream(new byte[2 * 1024 * 1024]);
            }
        }
    }
    ));
setStartTestButtonText(!value); //Update the startTestButton's text

```

```

}

/**
 * @param refreshSamples defines whether we will call the refreshSamples
 *        method on the SignalSamplesListeners added to
 *        TestActivity
 * @return if there was a running GetSamplesTask when this was called.
 */
private boolean attachToRunningTask(boolean refreshSamples) {
    if (GetSamplesTask.Instance != null) {
        //If the running GetSamplesTask doesn't have an activity attached
        //to it, we copy the task's SignalSamples to the TestActivity
        //through the setTestActivity() method.
        if (GetSamplesTask.Instance.getTestActivity() == null) {
            if (signalSamples != null) {
                signalSamples.clear();
            } else {
                signalSamples = new ArrayList<>();
            }
            GetSamplesTask.Instance.setTestActivity(this); //if the pending samples aren't
            cleared they should be added
            if (refreshSamples) { //We call the SignalSamplesListeners so
                //that the tab fragments update their contents.
                for (SignalSamplesListener signalSamplesListener : signalSamplesListeners) {
                    signalSamplesListener.refreshSamples();
                }
            }
            boolean isFinished = GetSamplesTask.Instance.isFinished();
            //If the GetSamplesTask hasn't been cancelled and hasn't been finished,
            //we enable the startTestButton so that the user can stop the test.
            startTestButton.setEnabled(!GetSamplesTask.Instance.isCancelled() &&
!isFinished);
            setStartTestButtonText(!isFinished); //Update
            if (isFinished) {
                setTestFinished(true); //Here, we write the test into a text file.
            }
        }
        return true; //Could return true only inside the if but it works as is too..
    }
    return false; //No task is running
}

//Here we load the SignalSamples from an existing test, unless the
//testFileName is null, when we only initialize the list.
private List<SignalSample> loadSignalSamples() {
    List<SignalSample> loadedSamples;
    if (testFileName != null) {
        loadedSamples = FileWriterParser.readFile(new File(getBaseContext().getFilesDir(),
testFileName));
    } else {
        loadedSamples = new ArrayList<>();
    }
}

```



```

    }
    return loadedSamples;
}

//A method used for getting the instance of a fragment from the
//fragment manager by its class
private Fragment findFragment(Class fragmentClass) {
    for (Fragment fragment : getSupportFragmentManager().getFragments()) {
        if (fragmentClass.isInstance(fragment)) {
            return fragment;
        }
    }
    return null;
}

//Here we update the text of the startTestButton, depending on
//whether a test is running or not.
private void setStartTestButtonText(boolean isTestRunning) {
    startTestButton.setText(isTestRunning ? R.string.stop_test : R.string.new_test);
}

/**
 * A {@link FragmentPagerAdapter} that returns a fragment corresponding to
 * one of the sections/tabs/pages.
 */
public class SectionsPagerAdapter extends FragmentPagerAdapter {
    public SectionsPagerAdapter(FragmentManager fm) {
        super(fm);
    }

    @Override
    public Fragment getItem(int position) {
        //HomeActivity.println("getItem");
        //HomeActivity.println(getSupportFragmentManager().getFragments() + "");
        // getItem is called to instantiate the fragment for the given page.
        // Returns a new Fragment instance for each position
        if (position == 0) {
            return SignalSamplesListFragment.newInstance();
        } else if (position == 1) {
            TestMapParentFragment mapParentFragment = new TestMapParentFragment();

            // Obtain the SupportMapFragment and get notified when the map is ready to be
            used.
            //mapParentFragment = (SupportMapFragment) getSupportFragmentManager()
            //    .findFragmentById(R.id.map);
            return mapParentFragment;
        } else if (position == 2) {
            return GraphFragment.newInstance(true);
        } else if (position == 3) {
            return GraphFragment.newInstance(false);
        }
    }
}

```

```

        return null;
    }

    @Override
    public int getCount() {
        // Show 4 total pages.
        return 4;
    }

    /**
     * @param position The tab position - index
     * @return the title of each tab
     */
    @Override
    public CharSequence getPageTitle(int position) {
        switch (position) {
            case 0:
                return getString(R.string.samples);
            case 1:
                return getString(R.string.map);
            case 2:
                return getString(R.string.dbm_graph);
            case 3:
                return getString(R.string.speed_graph);
        }
        return null;
    }
}
}
}

```

```
package com.nikmix.mobilesignaltester.testactivity;
```

```

import android.graphics.Color;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentTransaction;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.BitmapDescriptor;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;

```

```

import com.google.android.gms.maps.model.Polyline;
import com.google.android.gms.maps.model.PolylineOptions;
import com.nikmix.mobilesignaltester.R;
import com.nikmix.mobilesignaltester.homeactivity.HomeActivity;
import com.nikmix.mobilesignaltester.testactivity.location.LocationProvider;
import com.nikmix.mobilesignaltester.testactivity.core.Coordinates;
import com.nikmix.mobilesignaltester.testactivity.core.SignalSample;

import java.util.ArrayList;
import java.util.List;
import java.util.Locale;

/**
 * Created by nikmix on 10/4/2017.
 * This fragment-class contains a SupportMapFragment which displays the
 * locations of the SignalSamples of the test (with color variations for different
 * transfer speeds) and also the path of the locations.
 */

public class TestMapParentFragment extends Fragment implements SignalSamplesListener {
    private static final float ZOOM = 17; //The zoom of the map so that
    //enough locations are shown inside the map view-port during a car-ride.

    private SupportMapFragment mapFragment; //The SupportMapFragment that displays
    the main map.

    private GoogleMap googleMap; //The google map object used
    private Polyline polyline; //A Polyline object containing the locations of the SignalSamples.
    It represents the path of the locations.

    //It's a runnable we use to store a map action request as in a queue
    //so that it will be executed once the map is ready in case the map
    //wasn't ready when the request was made.
    private Runnable onMapReadyAction;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        //HomeActivity.println("onCreateView");
        View rootView = inflater.inflate(R.layout.test_activity_fragment_test_map_parent,
        container, false);
        mapFragment = new SupportMapFragment();
        addMapFragment(mapFragment);

        /**
         * Here we request a new GoogleMap object and once it's obtained
         * we set it up with the existing SignalSamples (if any)
         */
        mapFragment.getMapAsync(new OnMapReadyCallback() {
            @Override
            public void onMapReady(GoogleMap googleMap) {

```

```

//HomeActivity.println("onMapReady");
TestMapParentFragment.this.googleMap = googleMap;
try {
    //Adds a center-on-location button to the map
    googleMap.setMyLocationEnabled(true);
} catch (SecurityException e) {
    e.printStackTrace();
}
//We center the map to our current location
centerOnLocation(LocationProvider.getInstance().getLastKnownLocation(), false);
addSamplesPath(); //We add the existing samples path to the map (with markers)

//If the onMapReadyAction isn't null, we execute it
if (onMapReadyAction != null) {
    onMapReadyAction.run();
    onMapReadyAction = null;
}
}
});
//We add this object as a SignalSamplesListener to TestActivity
//so that it gets updated during the test)
((TestActivity) getContext()).addSignalSamplesListener(this);
return rootView;
}

// @Override
// public void onPause() {
//     super.onPause();
//     FragmentTransaction fragmentTransaction =
// getFragmentManager().beginTransaction();
//     fragmentTransaction.remove(mapFragment);
//     fragmentTransaction.commit();
// }
//
// @Override
// public void onResume() {
//     super.onResume();
//     if (!getFragmentManager().getFragments().contains(mapFragment)) {
//         addMapFragment(mapFragment);
//     }
// }
// }

@Override
public void onDestroy() {
    //We remove this object from the SignalSamplesListeners list of TestActivity.
    ((TestActivity) getActivity()).removeSignalSamplesListener(this);
    super.onDestroy();
}

//A method for identifying if the GoogleMap is ready.
public boolean isMapReady() {

```

```

        return googleMap != null;
    }

    /**
     * A method that centers the map to the requested location using smooth animation
     *
     * @param location The target location
     */
    public void centerOnLocation(final Coordinates location) {
        centerOnLocation(location, true);
    }

    /**
     * A method that centers the map to the requested location
     *
     * @param location The target location
     * @param animate Whether we want the map to center instantaneously or smoothly
     * (with animation)
     */
    public void centerOnLocation(final Coordinates location, final boolean animate) {
        Runnable centerOnLocationAction = new Runnable() {
            @Override
            public void run() {
                if (location != null) {
                    if (animate) {
                        googleMap.animateCamera(CameraUpdateFactory.newLatLngZoom(new
LatLng(location.lat, location.lon), ZOOM));
                    } else {
                        googleMap.moveCamera(CameraUpdateFactory.newLatLngZoom(new
LatLng(location.lat, location.lon), ZOOM));
                    }
                }
            }
        };
        //If the map is ready, we execute the action now.
        //Otherwise we set it as the onMapReadyAction so that it gets executed
        //once the map is ready.
        if (!isMapReady()) {
            this.onMapReadyAction = centerOnLocationAction;
        } else {
            centerOnLocationAction.run();
        }
    }

    /**
     * This method adds a new LatLng object to the polyline and also
     * a marker with information about the SignalSample
     * (dBm, speed, if it's download or upload).
     * It also centers the map to the new SignalSample.
     *
     * @param signalSample The new SignalSample

```

```

*/
@Override
public void sampleCreated(SignalSample signalSample) {
    if (isMapReady()) {
        Coordinates location = signalSample.location;
        if (location != null) {
            List<LatLng> points = polyline.getPoints();
            points.add(new LatLng(location.lat, location.lon));
            polyline.setPoints(points);
            addMarker(signalSample); //Creates the marker that describes the sample
            centerOnLocation(location); //Centers to the new sample
        }
    }
}

/**
 * Refreshes the map
 */
@Override
public void refreshSamples() {
    if (isMapReady()) {
        googleMap.clear();
        addSamplesPath(); //Adds the samples of TestActivity to the map
        //using a polyline and markers. It also centers the map to the
        //first SignalSample of the test.
    }
}

/**
 * Adds the samples of TestActivity to the map using a polyline
 * and markers. It also centers the map to the first SignalSample
 * of the test.
 */
private void addSamplesPath() {
    TestActivity testActivity = (TestActivity) getActivity();
    List<SignalSample> signalSamples = testActivity.getSignalSamples();

    //We create a list containing all the non-null locations of the
    //SignalSamples list.
    List<LatLng> lineCoordinates = new ArrayList<>();
    for (SignalSample signalSample : signalSamples) {
        Coordinates location = signalSample.location;
        if (location != null) {
            LatLng coordinates = new LatLng(location.lat, location.lon);
            lineCoordinates.add(coordinates);
            //We also add a marker for each of those locations.
            addMarker(signalSample);
        }
    }
    //We center to the first SignalSample's location of the test.
    if (!signalSamples.isEmpty()) {

```

```

        Coordinates location = signalSamples.get(0).location;
        centerOnLocation(location, false);
    }

    createPolyline(); //Add a new Polyline to the map
    List<LatLng> points = polyline.getPoints();
    points.addAll(lineCoordinates);
    polyline.setPoints(points); //Add the locations list to it
}

/**
 * Creates an empty Polyline with blue color and adds it to the map.
 */
private void createPolyline() {
    PolylineOptions polylineOptions = new PolylineOptions();
    polylineOptions.color(Color.BLUE);
    //polylineOptions.visible(true);
    polyline = googleMap.addPolyline(polylineOptions);
}

/**
 * Here we create a marker from the SignalSample's information.
 *
 * @param signalSample The SignalSample used to create the marker..
 */
private void addMarker(SignalSample signalSample) {
    Coordinates location = signalSample.location;
    if (location != null) {
        MarkerOptions markerOptions = new MarkerOptions();
        markerOptions.position(new LatLng(location.lat, location.lon));
        //Here, we set a colored icon to the marker which color depends
        //on the sample's transfer speed and whether it's a download
        //or an upload.
        markerOptions.icon(getColoredIcon(signalSample));
        //We set the title of the marker as the network type of the device
        //plus the dBms when the sample was created
        markerOptions.title(signalSample.protocol + " (" + String.format(Locale.US, "%.1f",
signalSample.dbm) + " " +
            getResources().getString(R.string.dbm) + ")");
        //We set the snippet of the marker as the transfer-state of the sample
        //plus the transfer speed.
        markerOptions.snippet(SignalSample.getStringByTransferState(getResources(),
signalSample.transferState) + " " +
            getSpeedString(signalSample.speedInKbps));
        googleMap.addMarker(markerOptions);
    }
}

/**
 * Creates a BitmapDescriptor that contains the marker icon having it
 * colored depending on the transfer speed of the SignalSample and

```

```

* whether the transfer-state is a download or an upload.
*
* @param signalSample The sample to create the marker from.
* @return the BitmapDescriptor
*/
private BitmapDescriptor getColoredIcon(SignalSample signalSample) {
    //If it's a download, we use a orange icon if the speed is zero
    //and a green one if the speed is above 5000 kbps. If the speed
    //is between, we use an intermediate color depending on the speed.
    if (signalSample.transferState == SignalSample.TransferState.DOWNLOAD ||
        signalSample.transferState == SignalSample.TransferState.DOWNLOAD_COMPLETE)
    {
        float hueMin = BitmapDescriptorFactory.HUE_ORANGE;
        float hueMax = BitmapDescriptorFactory.HUE_GREEN;
        float maxSpeed = 5000;
        float speedInKbps = Math.max(0, Math.min(signalSample.speedInKbps, maxSpeed));
        return BitmapDescriptorFactory.defaultMarker(hueMin + (hueMax - hueMin) *
speedInKbps / maxSpeed);
    } else {
        //If it's an upload, we use a red icon if the speed is zero
        //and a blue one if the speed is above 1600 kbps. If the speed
        //is between, we use an intermediate color depending on the speed.
        float hueMin = BitmapDescriptorFactory.HUE_BLUE;
        float hueMax = BitmapDescriptorFactory.HUE_ROSE;
        float maxSpeed = 1600;
        float speedInKbps = Math.max(0, Math.min(signalSample.speedInKbps, maxSpeed));
        return BitmapDescriptorFactory.defaultMarker(hueMin + (hueMax - hueMin) * (1 -
(speedInKbps / maxSpeed)));
    }
}

//Here we add the SupportMapFragment to the parent fragment through
//the FragmentManager.
private void addMapFragment(SupportMapFragment supportMapFragment) {
    FragmentTransaction fragmentTransaction =
getFragmentManager().beginTransaction();
    fragmentTransaction.replace(R.id.support_map_fragment_container,
supportMapFragment);
    fragmentTransaction.commit();
}

/**
* We return a String showing the speed.
*
* @param speedInKbps If the parameter is set to -2 we show 'Unknown'.
*                    If it's -1 (Connection Lost), we show 0.
*                    Otherwise we show the speedInKbps value.
* @return the String describing the speed
*/
private String getSpeedString(float speedInKbps) {
    if (speedInKbps > -2) {

```



```
        return String.format(Locale.US, "%.1f",
            speedInKbps < 0 ? 0 : speedInKbps) + " " +
            getResources().getString(R.string.kbps);
    } else {
        return getResources().getString(R.string.unknown);
    }
}
}
```

