ΤΕΙ ΠΕΛΟΠΟΝΝΗΣΟΥ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΤΕ

Μελέτη και ανάπτυξη εφαρμογής VoIP και/ή VVoIP (Voice & Video over IP) με την χρήση του πρωτοκόλλου SIP

Πτυχιακή εργασία

του Φαλιέρη Ηλία
Α.Μ. 2011079

Επιβλέπων καθηγητής
Μπάρδης Γεώργιος

Σπάρτη
Νοέμβριος 2015

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ενυπογράφως ότι είμαι αποκλειστικός συγγραφέας της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται λεπτομερώς στην εργασία αυτή. Έχω αναφέρει πλήρως και με σαφείς αναφορές, όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία είτε βάση επιστημονικής παράφρασης. Αναλαμβάνω την προσωπική και ατομική ευθύνη ότι σε περίπτωση αποτυχίας στην υλοποίηση των ανωτέρω δηλωθέντων στοιχείων, είμαι υπόλογος έναντι λογοκλοπής, γεγονός που σημαίνει αποτυχία στην Πτυχιακή μου Εργασία και κατά συνέπεια αποτυχία απόκτησης του Τίτλου Σπουδών, πέραν των λοιπών συνεπειών του νόμου περί πνευματικών δικαιωμάτων. Δηλώνω, συνεπώς, ότι αυτή η Πτυχιακή Εργασία προετοιμάστηκε και ολοκληρώθηκε από εμένα προσωπικά και αποκλειστικά και ότι, αναλαμβάνω πλήρως όλες τις συνέπειες του νόμου στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δε μου ανήκει διότι είναι προϊόν λογοκλοπής άλλης πνευματικής ιδιοκτησίας.

Όνομα και Επώνυμο Συγγραφέα (Με Κεφαλαία):

……

Υπογραφή (Ολογράφως, χωρίς μονογραφή):

……

Ημερομηνία (Ημέρα – Μήνας – Έτος):

……

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον καθηγητή μου και επιβλέποντα κ. Μπάρδη Γεώργιο που πίστεψε σε εμένα, με ενθάρρυνε, με καθοδήγησε και στήριξε τις προσπάθειές μου, ανοίγοντάς μου νέους ορίζοντες και επιτρέποντάς μου να ασχοληθώ με πολύ ενδιαφέροντα θέματα της επιστήμης τής Πληροφορικής.

Ιδιαίτερες ευχαριστίες οφείλω στους γονείς μου για την ενθάρρυνση και την υποστήριξή τους προς εμένα καθ' όλη τη διάρκεια των σπουδών μου, διότι χωρίς αυτούς δεν θα είχα τη δυνατότητα να ξεκινήσω και να ολοκληρώσω τις σπουδές μου.

ΠΕΡΙΛΗΨΗ

Τα τελευταία χρόνια η χρήση του διαδικτύου εισβάλει όλο και περισσότερο σε πολλούς τομείς τής καθημερινότητας και όχι μόνο. Η αυξανόμενη χρήση του διαδικτύου είναι άρρηκτα συνδεδεμένη με την ταχεία ανάπτυξή του, επηρεαζόμενο το ένα από το άλλο. Λόγο της αυξανόμενης χρήσης του διαδικτύου αναπτύσσεται ραγδαία και λόγο της ραγδαίας ανάπτυξης του διαδικτύου αυξάνεται η χρήση του σε όλους τους τομείς. Ένας από τους κλάδους που έχουν γνωρίσει μεγάλη ανάπτυξη λόγο του διαδικτύου είναι οι τηλεπικοινωνίες.

Η παρούσα εργασία αναφέρεται σε τεχνολογίες VoIP / VVoIP οι οποίες αναμένεται να είναι οι αντικαταστάτες της κλασσικής τηλεφωνίας κυκλώματος PSTN που χρησιμοποιείται ακόμα και σήμερα κατά κόρον. Με τον όρο VoIP / VVoIP αναφερόμαστε σε ένα σύνολο τεχνολογιών μέσω των οποίων είναι δυνατή η επικοινωνία δύο ή περισσότερων χρηστών με μεταφορά ήχου και εικόνας μέσω IP δικτύων όπως είναι το διαδίκτυο. Η υλοποίηση των τεχνολογιών αυτών μπορεί να επιτευχθεί με μία σειρά από πρωτόκολλα με πιο γνωστό και χρησιμοποιούμενο το SIP.

Σκοπός τής παρούσας εργασίας είναι η μελέτη του πρωτοκόλλου SIP, καθώς επίσης η επέκταση μίας υφιστάμενης εφαρμογής διαχείρισης στόλου ταξί για συσκευές Android, ενσωματώνοντας δυνατότητες τηλεφωνίας VoIP.

Η παρούσα εργασία χωρίζεται σε δύο μέρη:

A. Ανάλυση της τεχνολογίας VoIP και των πρωτοκόλλων υλοποίησής της.

B. Περιγραφή τής υφιστάμενης εφαρμογής και παρουσίαση της VoIP εφαρμογής.

Το πρώτο μέρος περιλαμβάνει το πρώτο κεφάλαιο στο οποίο γίνεται μια ιστορική αναδρομή της τηλεφωνίας και την παρουσίαση της τεχνολογίας VoIP με τα πρωτόκολλα τα οποία υλοποιούν την τεχνολογία αυτή. Στο δεύτερο κεφάλαιο γίνεται μία αναλυτική περιγραφή τής λειτουργία του πρωτοκόλλου SIP και στο τρίτο κεφάλαιο γίνεται αναφορά στο λειτουργικό σύστημα Android, στην γλώσσα προγραμματισμού Java και στα εργαλεία που υπάρχουν για τους μηχανικούς λογισμικού που θέλουν να αναπτύξουν εφαρμογές για συσκευές Android.

Το δεύτερο μέρος αποτελείται από τα κεφάλαια 4 έως 7. Στο τέταρτο κεφάλαιο περιγράφεται η υφιστάμενη εφαρμογή διαχείρισης στόλου ταξί και γίνεται ανάλυση του προβλήματος που έπρεπε να λυθεί καθώς επίσης και του πλάνου επίλυσης που ακολουθήθηκε. Στο πέμπτο κεφάλαιο γίνεται ανάλυση στην πορεία που ακολουθήθηκε για την ανάπτυξη της εφαρμογής ενώ στο έκτο γίνεται αναλυτική αναφορά στις κυριότερες κλάσεις τής εφαρμογής. Τέλος παρουσιάζεται κι ένα παράδειγμα εκτέλεσης της εφαρμογής στο έβδομο και τελευταίο κεφάλαιο της εργασίας.

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΡΟΛΟΓΟΣ

Η πτυχιακή αυτή εργασία αποτελεί την κορύφωση των σπουδών μου στο ΤΕΙ Πελοποννήσου, Παράρτημα Σπάρτης, τμήμα Μηχανικών Πληροφορικής. Αποτελεί εργασία μελέτης νέων τεχνολογιών στον κλάδο των τηλεπικοινωνιών καθώς επίσης και ανάπτυξης εφαρμογής τηλεπικοινωνιών για φορητές συσκευές Android.

Με την στήριξη του καθηγητή μου και μέντορα κ. Μπάρδη Γεώργιου, και εδώ θα ήθελα να τον ευχαριστήσω για ακόμη μία φορά, μου δόθηκε η ευκαιρία να αναπτύξω μία εφαρμογή η οποία διανέμεται κανονικά στους τελικούς χρήστες μέσω της πλατφόρμας Play Store. Επίσης μου δόθηκε η ευκαιρία να συνεργαστώ με άλλους μηχανικούς λογισμικού εντασσόμενος σε μία ομάδα και δουλεύοντας σε πραγματικές συνθήκες εργασίας.

1. ΕΙΣΑΓΩΓΗ

1.1 Ιστορική αναδρομή τηλεφωνίας

Από τα αρχαία κιόλας χρόνια ο άνθρωπος είχε την ανάγκη να επικοινωνήσει σε μεγάλες αποστάσεις. Οι προσπάθειες ήταν πολλές και οι λύσεις που βρέθηκαν κατά καιρούς πρωτότυπες. Οι περισσότερες από αυτές ήταν οπτικές και με το πέρασμα του χρόνου έκαναν την εμφάνισή τους και οι επικοινωνίες μεγάλων αποστάσεων μέσω φωνής.

Με τον όρο τηλεφωνία αναφερόμαστε στην επικοινωνία φωνής μεταξύ μεγάλων αποστάσεων. Η πρώτη αναφορά στην δημιουργία τηλεφώνου μετάδοσης φωνής μέσω ηλεκτρικού ρεύματος ήταν μέσω της πρότασης του Innocenzo Manzetti για έναν φωνητικό τηλέγραφο το έτος 1844. Από τότε ξεκίνησε μία συνεχής έρευνα από πολλούς, έως το 1875 όπου ο Alexander Graham Bell κατάφερε να το δημιουργήσει. Στην πραγματικότητα ο Alexander Graham Bell ανακάλυψε τον ηλεκτρομαγνητικό μετατροπέα ήχου, τον πρόγονο του σημερινού μικροφώνου. Στην άκρη ενός χωνιού στερέωσε μια μεταλλική μεμβράνη και πίσω από αυτή ένα πηνίο τυλιγμένο σε μια μαγνητική ράβδο. Η μεταλλική μεμβράνη ταλαντωνόταν από τα ηχητικά κύματα της φωνής και δημιουργούσε ασθενή ηλεκτρική τάση στο πηνίο. Αντίστοιχη κατασκευή, στην άλλη άκρη του καλωδίου λειτουργούσε ως μεγάφωνο. Η μεταλλική μεμβράνη ταλαντωνόταν από τις μεταβολές του

ηλεκτρικού ρεύματος στο πηνίο κι αυτή με τη σειρά της παρήγαγε τα ηχητικά κύματα.

1.2 Τι είναι το VoIP

Το VoIP (Voice over Internet Protocol) ή τηλεφωνία μέσω διαδικτύου, χαρακτηρίζει μια ομάδα πρωτοκόλλων-τεχνολογιών (H.323, SIP), η οποία προσφέρει φωνητική συνομιλία σε πραγματικό χρόνο με σχετικά καλή ποιότητα πλέον και στην ουσία χωρίς κόστος. Οι συνομιλίες αυτές παραδοσιακά γίνονταν αποκλειστικά μέσω ηλεκτρονικού υπολογιστή που ήταν συνδεδεμένος με το διαδίκτυο και διέθετε μικρόφωνο, ακουστικά και το κατάλληλο λογισμικό. Η κλήση κατέληγε σε έναν άλλο, ανάλογα εξοπλισμένο, ηλεκτρονικό υπολογιστή χωρίς να υπάρχει κάποια επιπλέον χρέωση, εκτός από αυτή της πρόσβασης στο διαδίκτυο, αφού στη συγκεκριμένη επικοινωνία δεν μεσολαβεί κάποιος παραδοσιακός φορέας τηλεπικοινωνιών (π.χ. ΟΤΕ) παρά μόνο το διαδίκτυο.

Για την πραγματοποίηση κλήσεων μέσω VoIP, ο χρήστης χρειάζεται κατάλληλο εξοπλισμό ο οποίος χωρίζεται σε δύο βασικές κατηγορίες. Στην μία κατηγορία ανήκουν διάφορα λογισμικά τα οποία μπορούν να εγκατασταθούν είτε σε κάποιον ηλεκτρονικό υπολογιστή, σταθερό ή φορητό, είτε σε κάποια φορητή συσκευή όπως είναι το tablet ή ακόμα και ένα smartphone. Η δεύτερη επιλογή που έχει ο χρήστης για την πραγματοποίηση κλήσεων μέσω VoIP είναι η χρήση κάποιας τηλεφωνικής

συσκευής η οποία συνδέεται απευθείας στο διαδίκτυο και προορίζεται αποκλειστικά για κλήσεις VoIP.

1.3 Πρωτόκολλα υλοποίησης υπηρεσιών VoIP

Τα πιο συχνά χρησιμοποιούμενα πρωτόκολλα για την υλοποίηση τεχνολογιών VoIP είναι τα πρωτόκολλα σηματοδοσίας (signaling protocols). Σε δίκτυα VoIP κάνουμε χρήση αυτών των πρωτοκόλλων για να εντοπίσουμε την συσκευή στο άλλο άκρο της επικοινωνίας, και στη συνέχεια να διαχειριστεί την ανταλλαγή μηνυμάτων μεταξύ των συσκευών αποστολής και λήψης.

Τα δύο πιο γνωστά πρωτόκολλα σηματοδοσία είναι:

- Session Initiation Protocol (SIP), που ορίζεται από το Internet Engineering Task Force (IETF).
- H.323, που ορίζεται από τη Διεθνή Ένωση Τηλεπικοινωνιών (ITU).

Αυτά τα δύο πρωτόκολλα έχουν δημιουργηθεί για να κάνουν το ίδιο πράγμα και οι περισσότερες VoIP συσκευές ή λογισμικά χρησιμοποιούν το ένα ή το άλλο. Τα δύο αυτά πρωτόκολλα εργάζονται για να επιτευχθεί μια σύνδεση VoIP, με διαφορετικό τρόπο όμως το καθένα. Το SIP είναι ASCII-based ενώ το H.323 είναι Binary-based. Στην αρχή ήταν πιο δημοφιλές το H.323 και πολλοί είναι αυτοί που θεωρούν ότι είναι ανώτερο στην ικανότητά του να συνεργαστεί με το δημόσιο τηλεφωνικό δίκτυο μεταγωγής (PSTN) και για τη μετάδοση βίντεο. Στις μέρες όμως το SIP έχει γίνει αρκετά πιο

δημοφιλές και οδεύει στο να καθιερωθεί ως το μοναδικό πρωτόκολλο υλοποίησης τεχνολογιών VoIP. Αυτό οφείλεται στο γεγονός ότι όλο και περισσότεροι κατασκευαστές συσκευών VoIP αλλά και λογισμικών, υποστηρίζουν το συγκεκριμένο πρωτόκολλο έναντι του αρχικού H.323 [2].

1.4 Συμπεράσματα

Ο άνθρωπος ανέκαθεν είχε την ανάγκη της επικοινωνίας με άλλους ανθρώπους. Από την αρχαιότητα κιόλας είχε εμφανιστεί η ανάγκη της επικοινωνίας μεταξύ ανθρώπων που τους χώριζαν μεγάλες αποστάσεις. Όλη αυτή η ανάγκη του ανθρώπου για επικοινωνία, ώθησε τον τελευταίο αιώνα, την ανάπτυξη των επικοινωνιών και ιδιαίτερα της τηλεφωνίας σε μία τεράστια ανάπτυξη.

Με την εμφάνιση της επιστήμης των υπολογιστών, η επικοινωνία γνώρισε μεγάλη πρόοδο, με αποτέλεσμα να μεταβεί σε άλλα επίπεδα. Έτσι η επικοινωνία κατάφερε να εξελιχθεί κι από μία απρόσωπη συσκευή επικοινωνίας, κυρίως δύο ανθρώπων, μετατράπηκε σε ένα ολοκληρωμένο σύστημα επικοινωνιών που κάνει απαραίτητη την χρήση του όχι μόνο σε κοινωνικό επίπεδο αλλά και επιχειρησιακό - οικονομικό, πολιτικό.

Σήμερα η τηλεδιάσκεψη αποτελεί τον αντικαταστάτη του απλού τηλεφώνου παρέχοντας έναν αμεσότερο τρόπο επικοινωνίας και την δυνατότητα υπηρεσιών όπως:

- Μάθηση από απόσταση
- Τηλε-εργασία
- Τηλε-συνεργασία

Στις μέρες κι η τηλεδιάσκεψη τείνει να αποτελέσει παρελθόν, μιας κι έχει αρχίσει να κάνει την εμφάνισή της η τηλε-παρουσία. Η τηλε-παρουσία έκανε την πρώτη της εμφάνιση το 2007 όταν η εταιρία Cisco παρουσίασε ένα σύστημα το οποίο επιτρέπει σε έναν ομιλητή κυριολεκτικά να μεταφερθεί ψηφιακά σε μια απομακρυσμένη αίθουσα και να εμφανιστεί ως τρισδιάστατο ολόγραμμα.

2. ΠΡΩΤΟΚΟΛΛΟ SIP

2.1 Γενικά για το πρωτόκολλο SIP

Το SIP (Session Initiation Protocol) είναι πρωτόκολλο επικοινωνίας μέσω δικτύων υπολογιστών, που επιτρέπει την μεταφορά πολυμεσικών πληροφοριών είτε μέσω του διαδικτύου, είτε μέσω ενός τοπικού δικτύου. Για την εφαρμογή του απαιτείται η χρήση ενός υπολογιστή που να έχει τον ρόλο του εξυπηρετητή SIP (SIP server) και φυσικά η ύπαρξη πελατών SIP (SIP clients). Δημιουργήθηκε από την IETF (Internet Engineering Task Force), το όργανο που είναι υπεύθυνο για τη διαχείριση και την ανάπτυξη των μηχανισμών που συνθέτουν το διαδίκτυο. Αρχικά δημοσιεύτηκε το 1996 ως RFC 2543 [3] και αργότερα το 2002 εξελίχθηκε ως στο RFC 3261 [4]. Το SIP κατέχει σημαντική θέση στη διαδικτυακή τηλεφωνία, γιατί δεν δεσμεύει τον χρήστη σε κάποιο συγκεκριμένο πάροχο, όπως για παράδειγμα την εταιρεία Skype, εφόσον εάν έχει τις γνώσεις μπορεί να το αξιοποιήσει ατομικά, είτε μέσω των εκατοντάδων παρόχων Voip με υποστήριξη SIP.

Προκειμένου να γίνει απλή χρήση την τεχνολογίας που βασίζεται στο πρωτόκολλο SIP χρειάζεται συσκευή SIP (Τηλέφωνο Voip με υποστήριξη SIP) όπου ανεξάρτητα χρήσης υπολογιστή είναι δυνατόν να συνδεθεί απευθείας στην ADSL γραμμή μας ή άλλης γρήγορης παροχής διαδικτύου μέσω ενός κοινού δρομολογητή (router).

Κατόπιν είναι δυνατή η επιλογή παρόχου για το πέρασμα των τηλεφωνικών κλήσεων στο διαδίκτυο. Υπάρχουν και συσκευές sip που

επιτρέπουν την είσοδο απλών τηλεφωνικών συσκευών (PSTN δικτύου) με την ονομασία ATA.

Τα τελευταία οχτώ χρόνια, η Voice over IP (VoIP) κοινότητα ενέκρινε το SIP, ως το κύριο πρωτόκολλο σηματοδοσίας και συνεχίζει να εξελίσσεται και επεκτείνεται όπως ωριμάζει η τεχνολογία ενώ συγχρόνως κερδίζει συνεχώς έδαφος στην αγορά.

Το SIP διακρίνεται λίγο πολύ από αυτή την φιλοσοφία. Έχοντας αναπτυχθεί αποκλειστικά ως ένας μηχανισμός για τη θέσπιση συνεδριών, δεν γνωρίζει τις λεπτομέρειες της συνεδρίας παρά μόνο αρχίζει, τερματίζει και τροποποιεί συνεδρίες. Αυτό η απλότητα σημαίνει ότι το SIP διακρίνεται από:

- Την επεκτασιμότητά του
- Και προσαρμόζεται εύκολα σε διαφορετικές αρχιτεκτονικές και σενάρια.

Το SIP είναι ένα request-response (αίτημα-απόκριση) πρωτόκολλο που μοιάζει με δύο άλλα πρωτόκολλα του διαδικτύου ,τα HTTP και SMTP, με συνέπεια να ταιριάζει άνετα δίπλα σε άλλες διαδικτυακές εφαρμογές. Κάποιοι λένε ότι: "ό,τι έκανε το HTTP για το Web, το SIP θα το κάνει για της τηλεπικοινωνίες". Το SIP έχει τεράστια συμβολή στην βιομηχανία των τηλεπικοινωνιών, με αποτέλεσμα όλες οι παραδοσιακές εταιρίες τεχνολογίας να έχουν αποφασίσει την καθιέρωση του πρωτοκόλλου SIP για όλες τις μελλοντικές τους εφαρμογές. Οι κατασκευαστές VoIP και IM (Instant Messaging) εφαρμογών (π.χ. Facebook messenger) έχουν καθιερώσει επίσης την χρήση του πρωτοκόλλου SIP. Το γεγονός που οδήγησε το SIP

στην καθιέρωσή του ήταν τα σημαντικά πλεονεκτήματα που έχει έναντι των υπολοίπων πρωτοκόλλων σηματοδότησης. Τα πλεονεκτήματα αυτά είναι:

- Η σταθερότητα την οποία κέρδισε μετά από μερικά χρόνια χρήσης.
- Η ταχύτητα, η οποία οφείλεται στην εξαιρετική αποδοτικότητα του πρωτοκόλλου.
- Η ευελιξία του πρωτοκόλλου λόγω της επεκτασιμότητάς του, η οποία οφείλεται στο γεγονός ότι είναι βασισμένο σε κείμενο.
- Η ασφάλεια του πρωτοκόλλου που προσφέρει η δυνατότητα κρυπτογράφησης (SSL, S/MIME) και οι διαθεσιμότητα πιστοποιήσεων.
- Η καθιέρωση του πρωτοκόλλου σε ολόκληρη τη βιομηχανία των τηλεπικοινωνιών το καθιστά σε πλεονεκτική θέση έναντι των υπολοίπων πρωτοκόλλων λόγω της ευρείας του χρήσης.

Χρησιμοποιώντας το SIP, η τηλεφωνία γίνεται άλλη μια διαδικτυακή εφαρμογή και ενσωματώνεται εύκολα σε άλλες διαδικτυακές υπηρεσίες. Το SIP είναι μια απλή εργαλειοθήκη όπου οι πάροχοι υπηρεσιών μπορούν να χρησιμοποιήσουν για την οικοδόμηση υπηρεσιών φωνής και πολυμέσων. Τέλος για να καταστεί δυνατή η τηλεφωνική επικοινωνία, το SIP χρειάζεται να συνεργαστεί με άλλα πρωτόκολλα όπως:

- Για την εξασφάλιση μεταφοράς (RTP/RTCP).
- Για την συμφωνία των παραμέτρων της κλήσης(SDP).
- Για τον έλεγχο ταυτότητας των χρηστών (ακτίνα, διάμετρος).

- Να παρέχουν καταλόγους (LDAP).
- Να είναι σε θέση να εγγυάται ποιότητα φωνής (RSVP, YESSIR) και συνεργασίας με τη σημερινή τού τηλεφωνικού δικτύου.

2.2 SIP Network Elements

Κάθε VoIP δίκτυο, που υλοποιείται με χρήση του πρωτοκόλλου SIP χρησιμοποιεί ένα σύνολο από στοιχεία-οντότητες οι οποίες συνεργάζονται για να επιτευχθεί η επικοινωνία μεταξύ δύο σταθμών. Τα βασικά στοιχεία που απαρτίζουν ένα τέτοιο δίκτυο είναι οι user agents, οι proxy servers, οι registrars και οι redirect servers. Στη συνέχεια ακολουθεί περιγραφή αυτών των στοιχείων-οντοτήτων.

2.2.1 SIP User Agents (UAs)

Οι SIP User Agents (UAs) χωρίζονται σε δύο λογικές οντότητες οι οποίες συνυπάρχουν σε ένα SIP UA. Στους User Agent Clients (UAC) και στους User Agent Servers (UAS). Ένας UAC μπορεί να δημιουργεί αιτήσεις και να επεξεργάζεται τις απαντήσεις από κάποιον UAS. Αντίστοιχα ένας UAS μπορεί να δέχεται τις αιτήσεις κάποιου UAC και να δημιουργεί απαντήσεις. Επειδή ένας UA αποτελείται από έναν UAC και έναν UAS, συχνά αναφέρουμε ότι συμπεριφέρεται ως UAC ή ως UAS αναλόγως με το αν στέλνει ένα SIP Request ή ένα SIP Response αντίστοιχα.

2.2.2 SIP Proxy Servers

Οι SIP Proxy Servers είναι κι αυτοί ένα από τα βασικά στοιχεία που απαρτίζουν ένα δίκτυο VoIP. Είναι υπεύθυνοι για την δρομολόγηση των μηνυμάτων που ανταλλάζουν δύο SIP UAs μεταξύ τους. Οι SIP Proxy Servers, καλούνται να διεκπεραιώσουν πολλές βασικές λειτουργίες για την επικοινωνία δύο σταθμών. Όπως περιγράφεται στο RFC 3621 [6], το SIP κάνει χρήση των Proxy Servers για να ολοκληρώσουν διαδικασίες όπως τη δρομολόγηση των αιτημάτων στην τρέχουσα θέση του χρήστη, την αυθεντικοποίηση και την εξουσιοδότηση των χρηστών για χρήση των υπηρεσιών, την εφαρμογή των πολιτικών κλήσεων-δρομολόγησης του παρόχου και να παρέχουν χαρακτηριστικά στους χρήστες. Στο πλαίσιο ενός δικτύου SIP οι Proxy Servers στην πραγματικότητα διαχειρίζονται τις ρυθμίσεις των κλήσεων μεταξύ δύο συσκευών VoIP, συμπεριλαμβανομένου του ελέγχου δρομολόγησης κλήσεων και εκτελεί επίσης απαραίτητες λειτουργίες, όπως εγγραφή και η εξουσιοδότηση των χρηστών, ο έλεγχος πρόσβασης στο δίκτυο και σε μερικές περιπτώσεις χειρίζονται επίσης την ασφάλεια του δικτύου.

Υπάρχουν δύο κατηγορίες SIP Proxy Servers:

Stateless Proxy Servers: Είναι Proxy Servers οι οποίοι έχουν πολύ απλή λειτουργία κι αυτό είναι το μοναδικό τους πλεονέκτημα έναντι των υπολοίπων Proxy Servers. Η μόνη λειτουργία που κάνουν είναι η αναμετάδοση των μηνυμάτων χωρίς να υποστηρίζουν την λειτουργία forking για την διανομή ενός μηνύματος σε περισσότερους από έναν παραλήπτες.

Επιπλέον, οι Stateless Proxy Servers δεν κάνουν χρήση των SIP transactions, ένα βασικό συστατικό στοιχείο του πρωτοκόλλου SIP, το επηρεάζει την αξιοπιστία της παράδοσης των μηνυμάτων.

Statefull Proxy Servers: Είναι κι αυτοί Proxy Servers όπως και οι Stateless αλλά με πιο σύνθετη λειτουργία. Σε αντίθεση με τους Stateless οι Statefull Proxy Servers εκτός από την προώθηση των μηνυμάτων τα αποθηκεύουν κιόλας. Έτσι δημιουργείται μία εσωτερική κατάσταση η οποία διατηρείται έως το τέλος της επικοινωνίας για την απόκτηση μεγαλύτερης ασφάλειας.

2.2.3 SIP Registrar Servers

Οι Registrar Servers είναι η βασική οντότητα ενός δικτύου SIP η οποία γνωρίζει την τρέχουσα θέση των χρηστών με σκοπό την ολοκλήρωση της τοπολογίας του δικτύου. Ένας Registrar Server δέχεται αιτήσεις εγγραφής από τους χρήστες που ανήκουν στο domain το οποίο ελέγχει, εξάγει πληροφορίες για την τρέχουσα θέση τους, όπως είναι η διεύθυνση IP, το όνομα χρήστη και η πόρτα. Τέλος αυτές τις πληροφορίες τις αποθηκεύει σε μία βάση δεδομένων με σκοπό να μπορέσει ένας Proxy Server να τις ανακτήσεις όταν χρειαστεί να προωθήσει ένα μήνυμα σε κάποιον χρήστη.

2.2.4 SIP Redirect Servers

Οι Redirect Servers αναλαμβάνουν να εντοπίσουν τον τελικό χρήστη όταν έχει μετακινηθεί προσωρινά και ταυτόχρονα έχει αλλάξει η λογική του διεύθυνση. Έτσι όταν ένας Registrar Server αναζητήσει έναν χρήστη ο οποίος έχει μετακινηθεί, θα ρωτήσει των Redirect Server κι αυτός με τη σειρά του θα ενημερώσει τον Registrar Server με την νέα IP διεύθυνση του χρήστη.

2.3 Μηνύματα ανταλλαγής πρωτοκόλλου SIP

Η έναρξη, η εφαρμογή κι ο τερματισμός μιας συνόδου με χρήση του πρωτοκόλλου SIP επιτυγχάνεται με την ανταλλαγή συγκεκριμένων μηνυμάτων μεταξύ των χρηστών (SIP Client) μέσω του εξυπηρετητή (SIP Server). Λόγω της ομοιότητας του πρωτοκόλλου SIP με τα πρωτόκολλα HTTP και SMTP τα μηνύματα που χρησιμοποιεί έχουν κληρονομήσει χαρακτηριστικά από αυτά τα πρωτόκολλα.

Τα μηνύματα αυτά χωρίζονται σε δύο βασικές κατηγορίες. Στην μία ανήκουν τα μηνύματα που στέλνονται από τον χρήστη προς τον εξυπηρετητή κι ονομάζονται αιτήσεις SIP (SIP Request) και στην δεύτερη κατηγορία ανήκουν οι τα μηνύματα απάντησης από των εξυπηρετητή προς τον χρήστη και ονομάζονται απαντήσεις SIP (SIP Responses).

Τα μηνύματα και των δύο κατηγοριών ακολουθούν την βασική δομή του RFC 2822 αν και υπάρχουν κάποιες διαφορές όπως για παράδειγμα οι

επικεφαλίδες που χρησιμοποιεί το SIP οι οποίες δεν είναι έγκυρες με το RFC 2822 [5].

Όλα τα μηνύματα αποτελούνται από μία γραμμή έναρξης, η οποία μπορεί να είναι είτε γραμμή αίτησης είτε γραμμή κατάστασης (Παράδειγμα γραμμή 1). Στη συνέχεια υπάρχουν μία ή περισσότερες γραμμές επικεφαλίδας (Παράδειγμα γραμμές 2 - 15) οι οποίες ακολουθούνται από μία κενή γραμμή που δηλώνει το τέλος των γραμμών επικεφαλίδας (Παράδειγμα γραμμή 16). Τέλος υπάρχει κι ένα προαιρετικό σώμα του μηνύματος (Παράδειγμα γραμμή 17).

Υπάρχει σαφής διάκριση μεταξύ των πληροφοριών που υπάρχουν στην γραμμή έναρξης, την επικεφαλίδα και το σώμα του μηνύματος. Η γραμμή έναρξης, κάθε γραμμή επικεφαλίδας και η κενή γραμμή τερματίζονται με μια αλλαγή γραμμής και ένα χαρακτήρα επαναφοράς (CRLF).

```
1.    INVITE sip:2103003531@213.144.173.77 SIP/2.0
2.    Accept: application/conference-info+xml, application/sdp,
 message/sipfrag, multipart/mixed
3.    Via: SIP/2.0/UDP 192.168.1.179
4.    Max-Forwards: 70
5.    From: "1447" <sip:1447@213.144.173.77:5060> ;tag=4918dae610
6.    To: <sip:2103003531@213.144.173.77>
7.    Call-ID: c33357ff49256b6c
8.    CSeq: 457930698 INVITE
9.    Allow: ACK, BYE, CANCEL, INVITE, MESSAGE, NOTIFY, OPTIONS,
 REFER, SUBSCRIBE, UPDATE
10.   Allow-Events: refer
11.   Contact: <sip:1447@192.168.1.179:5060>
12.   User-Agent: Media5-fone/4.2.0.3751
13.   Content-Disposition: session
14.   Content-Type: application/sdp
15.   Content-Length: 442
16.
17.   v=0
 o=- 4067036469487983372 4067036469487983373 IN IP4 192.168.1.179
 s=m5
 c=IN IP4 192.168.1.179
 t=0 0
 a=sendrecv
 m=audio 10000 RTP/AVP 9 96 0 8 97 18 125
 a=rtpmap:9 G722/8000
 a=rtpmap:96 ISAC/16000
 a=rtpmap:0 PCMU/8000
 a=rtpmap:8 PCMA/8000
 a=rtpmap:97 iLBC/8000
 a=rtpmap:18 G729/8000
 a=rtpmap:125 telephone-event/8000
 a=fmtp:96 ibitrate=20000;maxrate=32000
 a=fmtp:97 mode=30
 a=fmtp:18 annexb=no
 a=fmtp:125 0-15
 a=sendrecv
```

2.3.1 Γραμμή έναρξης

Η γραμμή έναρξης όπως φαίνεται και στο παράδειγμα είναι η πρώτη γραμμή ενός μηνύματος SIP. Η γραμμή αυτή αποτελείται από 3 συστατικά. Στην αρχή δηλώνεται η μέθοδος SIP για ένα αίτημα SIP ή ο κωδικός κατάστασης για μία απάντηση SIP αντίστοιχα. Στη συνέχεια δηλώνεται η διεύθυνση του χρήστη στον οποίο απευθύνεται το μήνυμα και στο τέλος δηλώνεται η έκδοση του πρωτοκόλλου SIP.

Για παράδειγμα από την γραμμή έναρξης του παραδείγματος μπορούμε να δούμε ότι το μήνυμα αναφέρεται σε ένα INVITE request, το οποίο απευθύνεται στον χρήστη με διεύθυνση sip:2103003531@213.144.173.77 και είναι σύμφωνο με την έκδοση SIP/2.0 του πρωτοκόλλου.

2.3.2 Γραμμή επικεφαλίδας

Η γραμμή επικεφαλίδας ενός μηνύματος SIP καταχρηστικά αναφέρεται ως γραμμή αντί γραμμών αφού πάντα αποτελείται από περισσότερες από μία γραμμή. Υπάρχουν επικεφαλίδες που είναι υποχρεωτική η ύπαρξή τους σε κάθε μήνυμα SIP και άλλες οι οποίες είναι προαιρετικές [πίνακας 2.1]. Σε αυτή την ενότητα θα παρουσιάσουμε τις βασικότερες από τις επικεφαλίδες.

Via

Όπως φαίνεται και στο παράδειγμα η επικεφαλίδα VIA περιέχει πληροφορίες για την έκδοση του πρωτοκόλλου SIP που χρησιμοποιείται, το πρωτόκολλο μεταφοράς του μηνύματος που όπως φαίνεται στο παράδειγμα είναι το πρωτόκολλο UDP, καθώς επίσης και την διεύθυνση του αποστολέα του μηνύματος. Στην πλειοψηφία τους τα μηνύματα SIP περνούν από αρκετούς SIP Proxy Servers και κάθε Server προσθέτει τη δική του VIA επικεφαλίδα πριν προωθήσει το μήνυμα στον επόμενο σταθμό. Έτσι, η διεύθυνση του αποστολέα που εμφανίζεται στην επικεφαλίδα VIA δεν αναφέρεται στην διεύθυνση του αρχικού αποστολέα αλλά του τελικού/ενδιάμεσου.

General-Headers	Entity-Headers	Request-Headers	Response-Headers
Call-ID Contact CSeq Date Encryption Expires From Record-route	Content-Encoding Content-Length Content-Type	Accept Accept-Encoding Accept-Language Authorization Contact Hide Max-Forwards Organization Priority Proxy-Authorization Route Require Response-Key Subject User-Agent	Allow Proxy-Authenticate Retry-After Server Unsupported Warning WWW-Authenticate

Πίνακας 2.1: Επικεφαλίδες μηνυμάτων πρωτοκόλλου SIP

Max-Forwards

Η επικεφαλίδα αυτή είναι μία από τις υποχρεωτικές επικεφαλίδες που πρέπει να υπάρχουν σε κάθε μήνυμα SIP. Σε κάθε μήνυμα είναι απαραίτητη η δήλωση του Max-Forwards για να καταφέρουμε να αποφύγουμε το φαινόμενο routing loops, όταν η εγκατάσταση αποτελείται από πολλούς SIP Proxy Servers. Έτσι ο αριθμός 70 που φαίνεται στο παράδειγμα δηλώνει το μέγιστο πλήθος Proxy Servers που θα μπορέσουν να παρεμβληθούν πριν καταλήξει το μήνυμα στον τελικό χρήστη. Κάθε Proxy Server πριν προωθήσει κάποιο μήνυμα στον επόμενο σταθμό μειώνει την τιμή αυτή κατά ένα.

From

Είναι κι αυτή μία υποχρεωτική επικεφαλίδα για κάθε μήνυμα SIP. Δηλώνει και αυτή τα στοιχεία του αποστολέα του αιτήματος αλλά σε αντίθεση με την VIA αναφέρεται στον αρχικό αποστολέα του αιτήματος κι όχι στους ενδιάμεσους. Σε αυτή την επικεφαλίδα διακρίνονται τρεις παράμετροι. Στην πρώτη παράμετρο, η οποία είναι και προαιρετική, εμφανίζεται το Display Name του αποστολέα και εμφανίζεται μέσα σε double quotes. Ακολουθεί το SIP URI, το οποίο δηλώνει τη λογική οντότητα του αποστολέα. Στην περίπτωση που υπάρχει Display Name, το SIP URI περικλείεται από τους χαρακτήρες <>. Τέλος ακολουθεί η παράμετρος tag η οποία είναι κι αυτή υποχρεωτική, όπως και η προηγούμενη. Η παράμετρος tag δηλώνει ένα μοναδικό αλφαριθμητικό για κάθε μήνυμα και χρησιμεύει στην αντιστοίχιση του μηνύματος σε υπάρχοντες διαλόγους.

To

Η επικεφαλίδα To είναι κι αυτή μία υποχρεωτική επικεφαλίδα για κάθε μήνυμα SIP. Αυτή η επικεφαλίδα περιέχει πληροφορίες για την λογική οντότητα του παραλήπτη του μηνύματος. Ακολουθεί κι αυτή την ίδια σύνταξη με την επικεφαλίδα From, με τη διαφορά ότι η παράμετρος tag στην επικεφαλίδα To δεν είναι υποχρεωτική.

CSeq

Άλλη μία υποχρεωτική επικεφαλίδα, η οποία υπάρχει σε κάθε μήνυμα SIP. Αποτελείται από δύο παραμέτρους. Η πρώτη παράμετρος δηλώνει έναν αύξον αριθμό που δηλώνει τη θέση του μηνύματος μέσα σε μία ακολουθία μηνυμάτων. Αυτός ο αριθμός είναι μοναδικός για κάθε SIP Request ενώ κάθε SIP Response έχει τον ίδιο αύξοντα αριθμό με το αντίστοιχο SIP Request.

2.3.3 Σώμα μηνύματος

Το σώμα του μηνύματος SIP είναι το τελευταίο συστατικό ενός τέτοιου μηνύματος. Η θέση του σώματος είναι μετά τις επικεφαλίδες του μηνύματος κι αφού προηγηθεί μία κενή γραμμή. Η ύπαρξη σώματος δεν είναι υποχρεωτική σε κάθε μήνυμα SIP γι’ αυτό υπάρχει και η κενή γραμμή, για να δηλώσει την ύπαρξη και τη θέση του σώματος. Στο σώμα του μηνύματος υπάρχουν πληροφορίες οι οποίες περιγράφουν το μέσο επικοινωνίας για την επικοινωνία των δύο άκρων. Έτσι, περιέχει πληροφορίες για τη συνεδρία αλλά και την περιγραφή του μέσου επικοινωνίας.

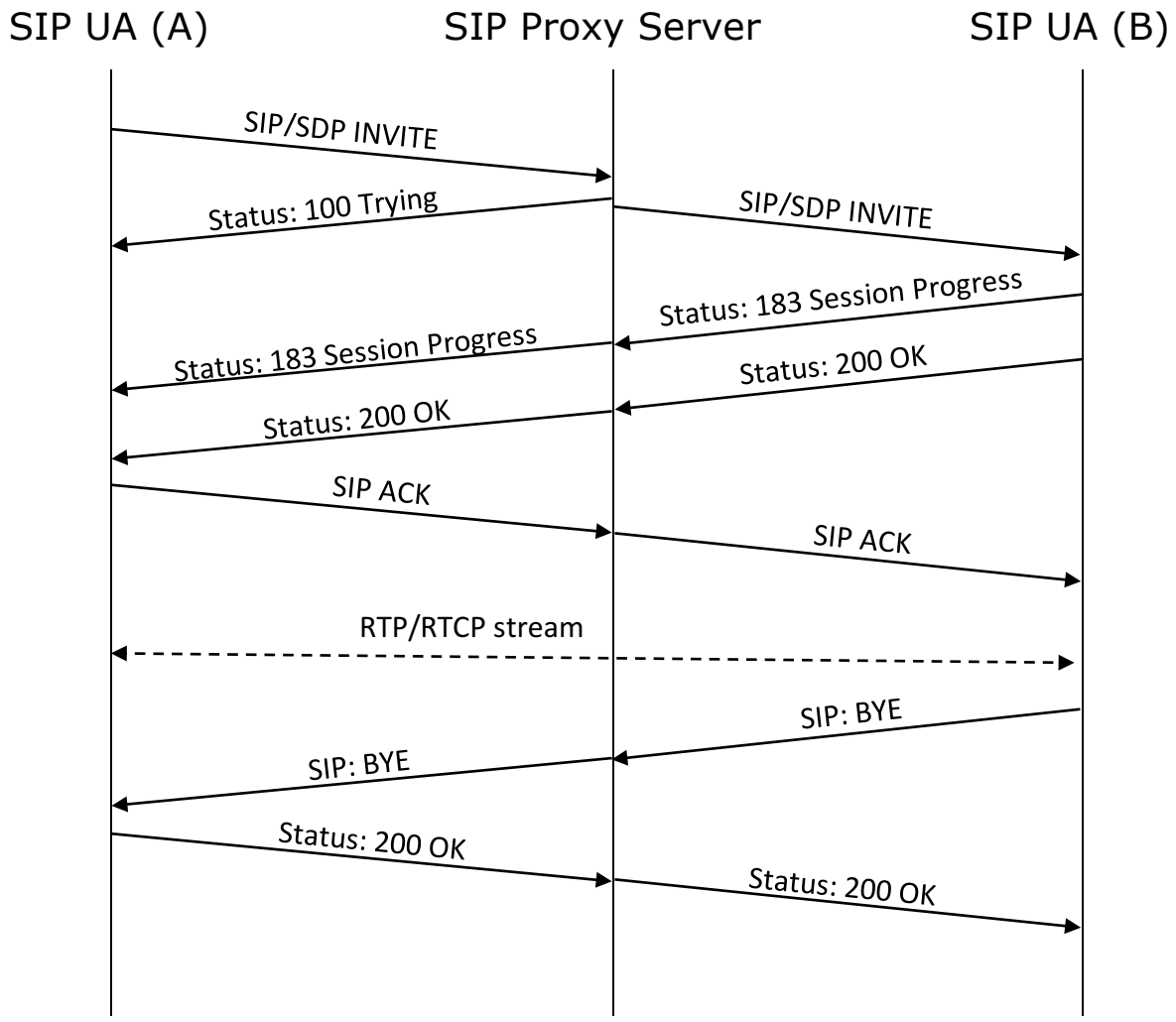
2.4 Διαδικασία υλοποίησης κλήσης με το πρωτόκολλο SIP

Για να πραγματοποιηθεί μια κλήση VoIP με χρήση του πρωτοκόλλου SIP είναι απαραίτητη η ύπαρξη δύο SIP User Agents και τουλάχιστον ενός SIP Proxy Server. Στο εξής θα αναφερόμαστε τους UAs ως πομπός και δέκτης, αυτός που πραγματοποιεί την κλήση κι αυτός που την δέχεται αντίστοιχα [σχήμα 2.1].

Ο πομπός στέλνει ένα μήνυμα INVITE στον Proxy Server, ο οποίος με τη σειρά του απαντάει με ένα μήνυμα 100 Trying. Στη συνέχεια ο Proxy Server προωθεί το μήνυμα INVITE του πομπού στον δέκτη. Ο δέκτης με τη σειρά του απαντάει με ένα μήνυμα 183 Session Progress, το οποίο ο Proxy Server προωθεί στον πομπό για να αναπαράγει τον γνωστό τόνο κλήσης[1] στο ακουστικό του. Όταν ο δέκτης αποδεχτεί την εισερχόμενη κλήση στέλνει ένα μήνυμα 200 OK στον Proxy Server, ο οποίος το προωθεί στον πομπό. Αυτός με τη σειρά του αφού παραλάβει το μήνυμα, στέλνει ένα μήνυμα επιβεβαίωσης (SIP ACK) στον δέκτη, μέσω του Proxy Server. Αφού το μήνυμα επιβεβαίωσης παραληφθεί από τον δέκτη, ξεκινάει η φωνητική επικοινωνία μεταξύ πομπού και δέκτη. Όταν ένας από τους δύο User Agents θελήσει να τερματίσει την κλήση, στέλνει ένα μήνυμα BYE στον άλλο User

[1] Ο τόνος αυτός μπορεί να διαφέρει ανάλογα με την χώρα του παρόχου τηλεφωνίας του πομπού. Οι Ευρωπαϊκές χώρες χρησιμοποιούν τόνους οι οποίοι ακολουθούν τις προτάσεις του Ευρωπαϊκού Ινστιτούτου Τηλεπικοινωνιακών Προτύπων (European Telecommunications Standards Institute - ETSI). Οι τόνοι αυτοί είναι συχνότητας 425 Hz. Τυπικά ο τόνος κλήσης έχει διάρκεια 1 δευτερόλεπτο ακολουθούμενος από μία παύση από 3 έως 5 δευτερόλεπτα.

Agent μέσω του Proxy Server κι αφού ο δεύτερος απαντήσει με ένα μήνυμα 200 OK η επικοινωνία τερματίζεται.



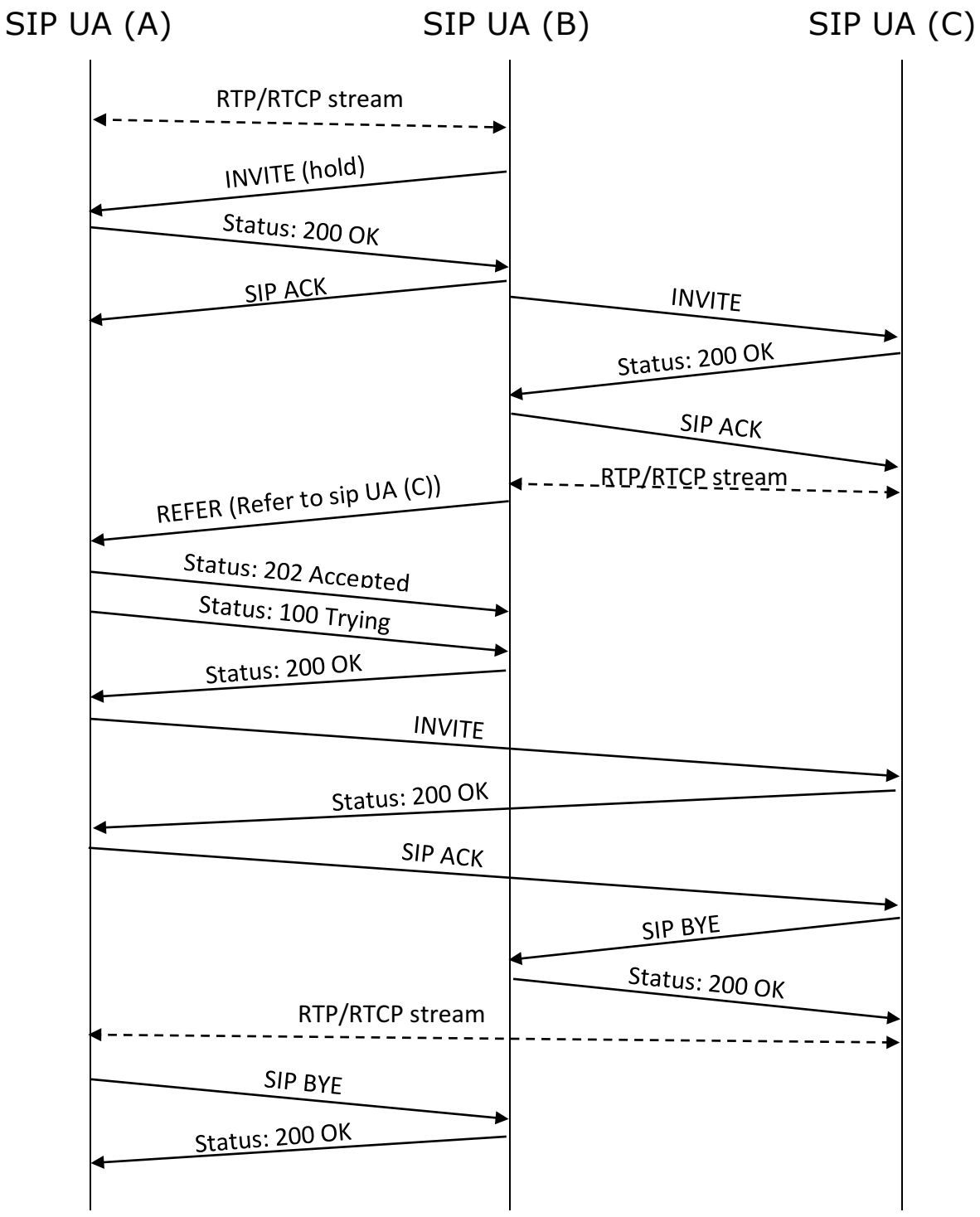
Σχήμα 2.1: Διάγραμμα ροής κλήσης VoIP με χρήστη του πρωτοκόλλου SIP

2.5 Διαδικασία υλοποίησης προώθησης κλήσης με το πρωτόκολλο SIP

Η τηλεφωνία μέσω διαδικτύου με χρήση του πρωτοκόλλου SIP υποστηρίζει πληθώρα υπηρεσιών, μερικές από τις οποίες είναι γνωστές στο κοινό και από την συμβατική τηλεφωνία. Μία από της υπηρεσίες αυτές είναι η προώθηση κλήσεων. Προώθηση μιας κλήσης ονομάζουμε την διαδικασία που ακολουθείται όταν ένας χρήστης έχει ήδη ξεκινήσει μία φωνητική επικοινωνία με έναν δεύτερο χρήστη, είτε μέσω εξερχόμενης είτε μέσω εισερχόμενης κλήσης και στη συνέχεια θέλει να συνδέσει τον συνομιλητή του με κάποιον τρίτο χρήστη κι αυτός να αποχωρήσει από την συνομιλία.

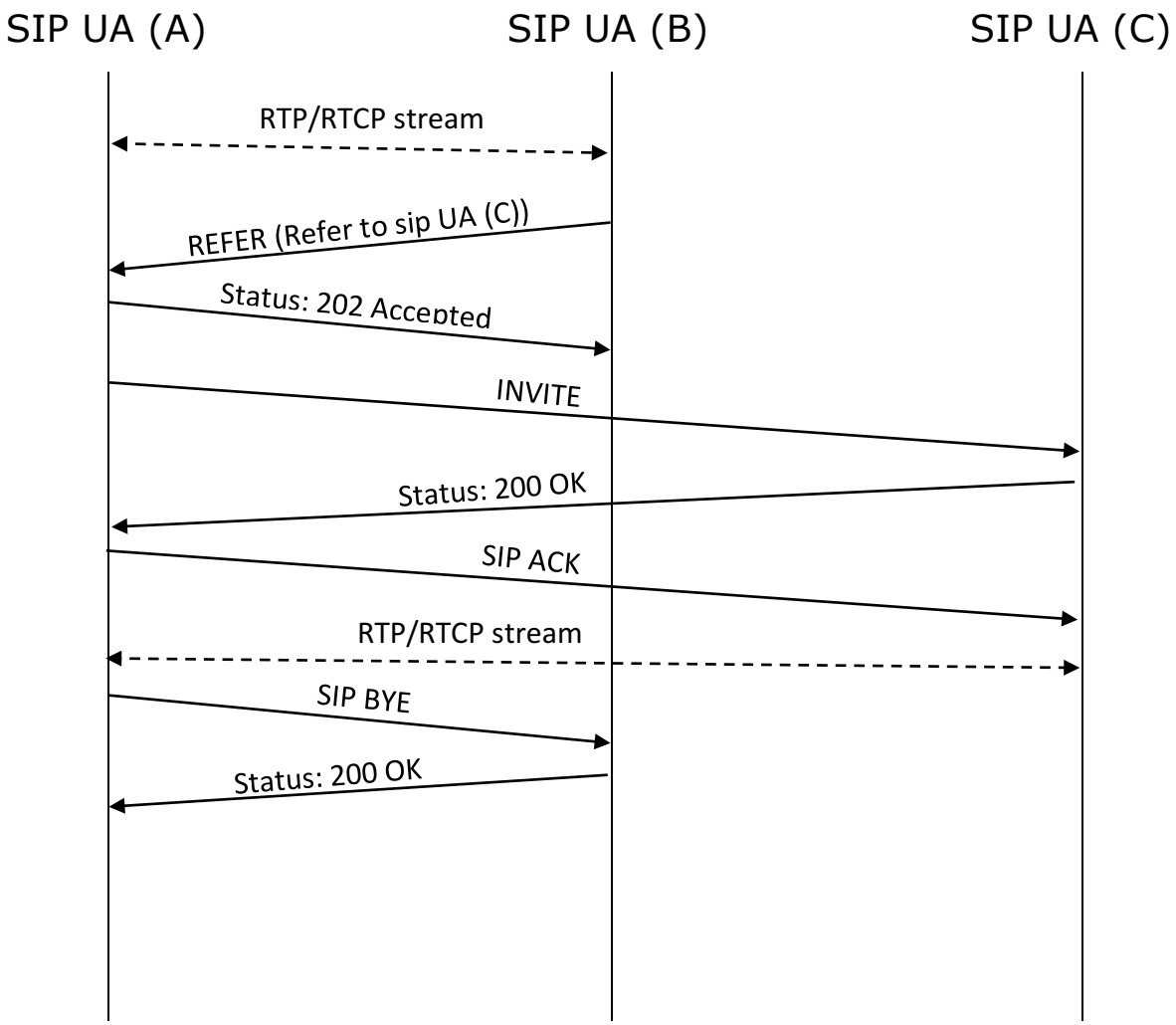
Το πρωτόκολλο SIP υποστηρίζει την υλοποίηση της προώθησης κλήσεων με δύο διαφορετικές υλοποιήσεις. Οι δύο αυτές υλοποιήσεις ονομάζονται Attended Transfer και Unattended/Blind Transfer αντίστοιχα. Η διαφορά στην υλοποίηση αυτών των δύο μεθόδων για την προώθηση μιας κλήσης επισημαίνεται στη διαχείριση που κάνει ο ενδιάμεσος χρήστης, αυτός δηλαδή που κάνει τη σύνδεση μεταξύ των άλλων δύο και αποχωρεί από την συνομιλία.

Στην περίπτωση της προώθησης μιας κλήσης εφαρμόζοντας την διαδικασία Attended Transfer [σχήμα 2.2], ο ενδιάμεσος χρήστης βάζει σε αναμονή τον συνομιλητή του. Στη συνέχεια καλεί τον τρίτο χρήστη κι αφού ολοκληρωθεί η μεταξύ τους σύνδεση και ξεκινήσει η φωνητική επικοινωνία, συνδέει τους δύο χρήστες μεταξύ τους, ολοκληρώνοντας μια φωνητική ομιλία του πρώτου και του τρίτου χρήστη, ενώ ο ενδιάμεσος αποχωρεί.



Σχήμα 2.2: Διάγραμμα ροής Attended Transfer με χρήση του πρωτοκόλλου SIP

Αντίθετα, στην δεύτερη περίπτωση προώθησης κλήσης, την ονομαζόμενη Unattended ή Blind Transfer [σχήμα 2.3] ο ενδιάμεσος χρήστης συνδέει τους άλλους δύο χωρίς να υλοποιήσει πρώτα σύνδεση με τον τρίτο. Πιο αναλυτικά, ενώ είναι σε εξέλιξη η φωνητική επικοινωνία με τον πρώτο χρήστη και χωρίς να τον βάλει σε αναμονή, καλεί τον τρίτο χρήστη και αποχωρεί από την σύνδεση πριν ακόμα υπάρξει κάποια σύνδεση μεταξύ των άλλων δύο. Έτσι, ο πρώτος χρήστης ακούει τον τόνο κλήσης ενώ το τηλέφωνο του τρίτου χτυπάει. Η υλοποίηση αυτής της μεθόδου είναι πολύ πιο απλή αλλά εγκυμονεί προβλήματα. Αυτά τα προβλήματα παρατηρούνται στην σύνδεση του τρίτου χρήστη με τον πρώτο. Για παράδειγμα μπορεί η σύνδεση του τρίτου χρήστη να μην είναι εφικτή κι έτσι η σύνδεση του πρώτου θα πέσει στο κενό. Επίσης μπορεί η σύνδεση του τρίτου χρήστη να είναι εφικτή αλλά αυτός να μην την αποδεχτεί. Έτσι και σε αυτή την περίπτωση η σύνδεση του πρώτου χρήστη θα πέσει στο κενό με τον το χρονικό όριο που θα έχει οριστεί για να χτυπάει η συσκευή του τρίτου χρήστη.



Σχήμα 2.2: Διάγραμμα ροής Unattended/Blind Transfer με χρήση του πρωτοκόλλου SIP

3. ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΓΙΑ ΣΥΣΚΕΥΕΣ ΜΕ ΛΕΙΤΟΥΡΓΙΚΟ ANDROID

3.1 Το λειτουργικό σύστημα Android

Το λειτουργικό σύστημα Android στην αρχή αναπτύχθηκε από την Google ενώ στη συνέχεια από την Open Handset Alliance. Προορίζεται για φορητές συσκευές και τρέχει τον πυρήνα του λειτουργικού συστήματος Linux. Η πρώτη παρουσίαση του Android έγινε τον Νοέμβριο του 2007 παράλληλα με την ανακοίνωση της ίδρυσης του οργανισμού Open Handset Alliance, ενώ η πρώτη έκδοση κυκλοφόρησε τον Σεπτέμβριο του 2009.

Η Open Handset Alliance είναι μία κοινοπραξία 48 τηλεπικοινωνιακών εταιριών, εταιριών λογισμικού αλλά και υλικού, οι οποίες έχουν στόχο την ανάπτυξη και εξέλιξη ανοιχτών προτύπων για φορητές συσκευές. Η Google έχει δημοσιεύσει το μεγαλύτερο μέρος του κώδικα του Android υπό την άδεια Apache License, μία ελεύθερη άδεια ανοιχτού λογισμικού.

Το Android δίνει τη δυνατότητα στους προγραμματιστές να γράφουν κώδικα για τις συσκευές με χρήση της γλώσσας προγραμματισμού Java. Η Google έχει αναπτύξει πληθώρα βιβλιοθηκών λογισμικού μέσω των οποίων, δίνεται η δυνατότητα στους προγραμματιστές να ελέγχουν τη συσκευή.

Κατά τη δημιουργία του Android υπήρχε ο στόχος εγκατάστασής του σε συσκευές που διαθέτουν οθόνες αφής όπως είναι τα κινητά τηλέφωνα και τα tablet, με διαφορετικό περιβάλλον χρήσης για ρολόγια χειρός (Android Wear), τηλεοράσεις (Android TV) και αυτοκίνητα (Android Auto). Στην

πορεία όμως, το Android εγκαταστάθηκε και σε άλλες συσκευές όπως παιχνιδομηχανές, φωτογραφικές μηχανές, και άλλες συσκευές.

Το λειτουργικό σύστημα Android είναι προσαρμοσμένο για κάθε οθόνη φορητής συσκευής, υποστηρίζοντας όλες τις αναλύσεις από VGA έως 4K. Για την προβολή περιεχομένου στην οθόνη, έχουν αναπτυχθεί βιβλιοθήκες για την υποστήριξη δισδιάστατων ψηφιακών γραφικών καθώς επίσης και τρισδιάστατων γραφικών βασισμένα στην OpenGL ES 3.0 ή νεότερη.

Οι χρήστες συσκευών που διαθέτουν Android είναι σε θέση να μεταφορτώσουν και να εγκαταστήσουν εφαρμογές από το Play Store. Το Play Store είναι ένας κατάλογος εφαρμογών αντίστοιχος του App Store για το λειτουργικό iOS. Στις αρχές τής δημιουργίας του Play Store επιτρεπόταν η διάθεση μέσω αυτού μόνο δωρεάν εφαρμογών. Ενώ αργότερα άρχισε να υποστηρίζεται και η διάθεση εφαρμογών επί πληρωμή ξεκινώντας το 2009 και καλύπτοντας μόνο τους χρήστες των Ηνωμένων Πολιτειών Αμερικής. Στην πορεία προστέθηκαν κι άλλες χώρες σταδιακά, φθάνοντας στο 2014 όπου άρχισαν να μπορούν κι οι προγραμματιστές από την Ελλάδα να διανέμουν εφαρμογές επί πληρωμή μέσω του Play Store.

Από το 2009 που κυκλοφόρησε η πρώτη έκδοση του Android (έκδοση 1.6) η Google παρουσιάζει και κυκλοφορεί περίπου δύο εκδόσεις Android σε ετήσια βάση. Η τελευταία έκδοση του Android που έχει κυκλοφορήσει είναι η έκδοση 6.0 Marshmallow, ενώ δεν έχει παρουσιαστεί ακόμα η επόμενη έκδοση από την Google.

3.2 Γνώσεις και λογισμικό για προγραμματισμό Android

Όπως αναφέρθηκε προηγουμένως, οι προγραμματιστές έχουν τη δυνατότητα να γράφουν κώδικα για τις συσκευές Android με χρήση της γλώσσας προγραμματισμού Java. Άρα θεωρητικά ένας γνώστης Java μπορεί να γράψει κώδικα για εφαρμογές Android. Στην ουσία ο κώδικας Java γράφεται για να υλοποιηθεί η λειτουργικότητα μίας Android εφαρμογής. Πέραν της λειτουργικότητας όμως για μία εφαρμογή πρέπει να υλοποιηθεί και η εμφάνισή της. Ο κώδικας για την υλοποίηση της εμφάνισης μίας εφαρμογής Android γράφεται σε γλώσσα XML. Επειδή όμως η XML δεν θεωρείται γλώσσα προγραμματισμού αλλά εμπλουτισμένο κείμενο, γι' αυτό το λόγο αναφέρεται μόνο η Java ως γλώσσα προγραμματισμού για την ανάπτυξη εφαρμογών Android.

Εκτός όμως από των κώδικα που πρέπει να γράψεις ένας μηχανικός λογισμικού, είναι απαραίτητη η τήρηση κάποιον κανόνων για να είναι δυνατή η παραγωγή εφαρμογής εκτελέσιμης από συσκευές Android. Οι κανόνες αυτοί έχουν να κάνουν κυρίως με την δομή που πρέπει να έχουν τα αρχεία ενός project.

Τα αρχεία ενός project χωρίζονται σε δύο βασικές κατηγορίες και βάσει αυτών δημιουργούνται δύο φάκελοι αντίστοιχα. Στον έναν φάκελο υπάρχουν τα αρχεία που είναι γραμμένα σε Java και αφορούν την λειτουργικότητα της εφαρμογής ενώ στον άλλο υπάρχουν αρχεία διάφορων τύπων που είναι απαραίτητα για την εμφάνιση της εφαρμογής στην οθόνη τής συσκευής αλλά κι όλα τα πολυμεσικά αρχεία που μπορεί να χρειάζεται η

εφαρμογή. Επιπλέον υπάρχει κι ένα αρχείο το οποίο δεν ανήκει σε καμία από τις δύο αυτές κατηγορίες. Αυτό το αρχείο είναι τύπου XML κι έχει υποχρεωτικά την ονομασία AndroidManifest. Σε αυτό το αρχείο υπάρχουν οι σημαντικότερες παράμετροι οι οποίες είναι απαραίτητες για την δημιουργία κάθε εφαρμογής Android. Μερικά από αυτά είναι η έκδοση της εφαρμογής, τα δικαιώματα που χρειάζεται η εφαρμογή για χρήση του υλικού της συσκευής, και άλλα. Αυτές οι γνώσεις, σε συνδυασμό με τις βιβλιοθήκες της Google, για τις οποίες υπάρχει το πολύ καλογραμμένο Documentation από την ίδια την Google, είναι αρκετές για κάποιον μηχανικό λογισμικού για να είναι σε θέση να γράψει κώδικα για συσκευές Android.

Εκτός όμως από τις γνώσεις προγραμματισμού, η εκμάθηση κάποιου ειδικού λογισμικού λειτουργεί βοηθητικά για τον μηχανικό λογισμικού. Η πρώτη κίνηση έγινε σε συνεργασία της Google με την ομάδα δημιουργίας και υποστήριξης του λογισμικού Eclipse. Το Eclipse είναι ένα λογισμικό, εργαλείο για πολλούς μηχανικούς λογισμικού λόγω της πληθώρας γλωσσών προγραμματισμού που υποστηρίζει. Η επιλογή αυτή έγινε από την Google αφενός λόγω της υποστήριξης που είχε το Eclipse για την γλώσσα προγραμματισμού Java, οπότε οι μηχανικού που χρησιμοποιούσαν αυτή την γλώσσα ήταν εξοικειωμένοι με το λογισμικό και αφετέρου λόγω ότι το Eclipse είναι ένα ελεύθερο λογισμικό. Έτσι μέσα από αυτή την συνεργασία προέκυψε μία νέα έκδοση του Eclipse, η έκδοση ADT (Android Developer Tools). Εκτός από πολλά μικρά εργαλεία που εμπεριέχονται σε αυτή την έκδοση για να διευκολύνουν την δουλειά του μηχανικού, το σημαντικότερο ίσως εργαλείο είναι αυτό που δίνει τη δυνατότητα δημιουργίας το πιο

συνηθισμένων γραφικών για την εφαρμογή τους σύροντας αντικείμενα πάνω στην οθόνη, γλιτώνοντάς τους από την συγγραφή αρκετών γραμμών κώδικα.

Μετά από μερικά χρόνια χρήσης του Eclipse από τους μηχανικούς λογισμικού για την συγγραφή κώδικα για εφαρμογές Android, η Google σταμάτησε την συνεργασία της με την ομάδα του Eclipse και ξεκίνησε συνεργασία με την ομάδα της IntelliJ. Αυτή η συνεργασία βασίστηκε στο ήδη υπάρχον λογισμικό τής IntelliJ το IntelliJ IDEA έχοντας ως αποτέλεσμα την δημιουργία ενός νέου λογισμικού, εργαλείο για τους μηχανικούς λογισμικού το Android Studio. Αυτό είναι το πλέον επίσημο λογισμικό που προτείνεται στου μηχανικούς λογισμικού επίσημα από την Google.

3.3 Η γλώσσα προγραμματισμού Java

Η Java είναι μία γλώσσα προγραμματισμού η οποία έκανε την εμφάνισή της για πρώτη φορά στα μέσα της δεκαετίας του ’90 από την εταιρία Sun Microsystems. Η Java δημιουργήθηκε με σκοπό να καλύψει τις ανάγκες που υπήρχαν για μία πλατφόρμα ανάπτυξης λογισμικού για μικρο-συσκευές. Η Java είναι γλώσσα που υποστηρίζει τον αντικειμενοστραφή προγραμματισμό έχοντας ως βασικό πλεονέκτημα έναντι των περισσότερων άλλων γλωσσών την ανεξαρτησία του λειτουργικού συστήματος και της πλατφόρμας.

Αυτή η ανεξαρτησία μεταξύ λειτουργικού συστήματος και πλατφόρμας έγινε εφικτό με την χρήση μίας εικονικής μηχανής (Virtual Machine). Για να

εκτελεστεί ένα πρόγραμμα γραμμένο σε Java σε οποιαδήποτε συσκευή θα πρέπει να είναι ήδη η εικονική μηχανή της Java. Η εικονική μηχανή διαβάζει τον κώδικα που έχει γραφτεί σε Java και τον μεταφράζει σε γλώσσα μηχανής αναλόγως του λειτουργικού που είναι εγκατεστημένο στην εκάστοτε συσκευή. Ένα μεγάλο πλεονέκτημα που προκύπτει από την χρήση αυτής της ενδιάμεσης εικονικής μηχανής είναι η ασφάλεια που προσφέρεται γιατί ο προγραμματιστής δεν μπορεί να γράψει κώδικα ο οποίος θα είναι καταστροφικός για την συσκευή αφού αυτός θα ανιχνευτεί από την εικονική μηχανή και δεν θα επιτρέψει την εκτέλεσή του.

Στην δημιουργία προγραμμάτων είναι κοινά τα σφάλματα που οφείλονται σε κακή χρήση της μνήμης. Εδώ είναι άλλη μία δικλίδα ασφαλείας που παρέχεται από την εικονική μηχανή τής Java. Η Java έχει υλοποιημένο και αυτοματοποιημένο τον συλλέκτη απορριμμάτων. Ο συλλέκτης απορριμμάτων είναι υπεύθυνος για την ελευθέρωση τμημάτων μνήμης από δεδομένα τα οποία δεν χρειάζονται. Έτσι ο προγραμματιστής δεν χρειάζεται να ανησυχεί για το πότε και αν θα ελευθερώσει ένα συγκεκριμένο τμήμα της μνήμης, ούτε και για σφάλματα δεικτών.

4. ΠΕΡΙΓΡΑΦΗ ΠΡΟΒΛΗΜΑΤΟΣ

Η εργασία αυτή υλοποιήθηκε για την επίλυση ενός προβλήματος στην λειτουργία μίας υφιστάμενης εφαρμογής, καθώς επίσης και για την απλοποίηση της παραμετροποίησής της αλλά και της χρήστης από τους τελικούς χρήστες.

Η υφιστάμενη εφαρμογή ήταν μία εφαρμογή για την διαχείριση στόλου ταξί. Η εφαρμογή αυτή είχε αναπτυχθεί με χρήση της γλώσσας JAVA για να μπορεί να εγκατασταθεί σε TABLET με λειτουργικό σύστημα ANDROID και κατ' επέκταση να αξιοποιηθεί από τους επαγγελματίες οδηγούς για την βελτιστοποίηση των προσφερόμενων υπηρεσιών προς τους πελάτες τους.

Η εφαρμογή που είχε αναπτυχθεί ήταν υπεύθυνη για την παρακολούθηση του στίγματος των αυτοκινήτων ταξί, με σκοπό να τα κατατάξει σε μία εκ των προκαθορισμένων ζωνών εξυπηρέτησης πελατών. Επίσης, η εφαρμογή αυτή καθόριζε και την σειρά προτεραιότητας του κάθε οδηγού σε σχέση με τον χρόνο αναμονής του στην αντίστοιχη ζώνη.

Με αυτόν τον τρόπο, το κεντρικό σύστημα ελέγχου είχε όλα τα απαραίτητα στοιχεία που χρειαζόταν για την ορθή δρομολόγηση των τηλεφωνικών κλήσεων από τους πελάτες στον κατάλληλο οδηγό για την γρηγορότερη εξυπηρέτησή τους. Με όλη αυτή τη λειτουργία τα οφέλη ήταν ορατά τόσο από την πλευρά των πελατών όσο και από την πλευρά των οδηγών. Ενώ οι πελάτες παρατήρησαν ελαχιστοποίηση στον χρόνο εξυπηρέτησής τους από τους επαγγελματίες οδηγούς ταξί, οι οδηγοί από την πλευρά τους ελαχιστοποίησαν τα λειτουργικά τους κόστη απαλλασσόμενοι

από την διατήρηση τηλεφωνικού κέντρου, καθώς επίσης ο κάθε οδηγός έχει πλέον πλήρη έλεγχο στον τρόπο με τον οποίο γίνεται η ανάθεση της κλήσης, πετυχαίνοντας την απόλυτη διαφάνεια και αξιοκρατία στην κατανομή των πελατών σε όλους τους επαγγελματίες οδηγούς.

4.1 Προβλήματα που έπρεπε να διορθωθούν

Όπως προαναφέρθηκε αυτή η εφαρμογή διαχειριζόταν την σειρά προτεραιότητας των οδηγών βάση της γεωγραφικής τους θέσης και του χρόνου αναμονής. Δεν είχε όμως καμία τεχνική δυνατότητα, από την εφαρμογή αυτή, για την διαχείριση των κλήσεων μεταξύ πελάτη και οδηγού.

Έτσι, ήταν απαραίτητη η εγκατάσταση, στην συσκευή του οδηγού, μιας εφαρμογής διαχείρισης κλήσεων VoIP του εμπορίου. Το κενό αυτό έως τώρα κάλυπτε η εφαρμογή Media5phone. Η εφαρμογή αυτή διανέμεται δωρεάν και μπορεί κάθε ιδιοκτήτης φορητής συσκευής που διαθέτει λειτουργικό σύστημα Android να την εγκαταστήσει μέσω του Play Store.

Το πρόβλημα που υπήρχε στην λειτουργικότητα του συστήματος που περιγράψαμε παρατηρήθηκε από τους επαγγελματίες οδηγούς ταξί, όταν παρουσιάστηκε η ανάγκη προώθησης κλήσεων από τους πελάτες προς άλλους οδηγούς. Όταν ένας οδηγός λάμβανε μία εισερχόμενη κλήση από κάποιον πελάτη αλλά δεν ήταν σε θέση να τον εξυπηρετήσει, έπρεπε να προωθήσει την κλήση στον καταλληλότερο οδηγό για την άμεση εξυπηρέτηση του πελάτη με τον ταχύτερο δυνατό τρόπο. Έτσι λοιπόν, η διαδικασία που έπρεπε να ακολουθηθεί από τον επαγγελματία οδηγό ήταν η

εξής. Έπρεπε να περάσει την εφαρμογή διαχείρισης κλήσεων VoIP στο παρασκήνιο της συσκευής του και να επαναφέρει στο προσκήνιο την εφαρμογή διαχείρισης στόλου ταξί. Έπειτα να ελέγξει ο ίδιος ποιος οδηγός από τους υπόλοιπους είναι ο ιδανικός για να εξυπηρετήσει την κλήση και να δει τον εσωτερικό αριθμό κλήσης του οδηγού. Στη συνέχεια αφού περάσει την εφαρμογή διαχείρισης στόλου ταξί στο παρασκήνιο και επαναφέρει την εφαρμογή διαχείρισης κλήσεων VoIP στο προσκήνιο, θα πρέπει να πληκτρολογήσει τον εσωτερικό αριθμό του οδηγού στον οποίο έχει επιλέξει να προωθήσει την κλήση και τελικά να ολοκληρωθεί η προώθηση της κλήσης κι ο πελάτης να επικοινωνήσει με τον αρμόδιο οδηγό.

Είναι προφανές το γεγονός ότι όλη αυτή η διαδικασία είναι αφενός χρονοβόρα, με αποτέλεσμα την αύξηση της αναμονής του πελάτη στο τηλέφωνο και αφετέρου πολύπλοκη για τους οδηγούς, αν αναλογιστούμε φυσικά και το γεγονός ότι βρίσκονται στον δρόμο και πολλές φορές οδηγούν την ώρα της κλήσης. Επίσης, εκτός από τα προβλήματα που προκαλούνται από αυτή τη διαδικασία κι είναι αναπόφευκτα, προκύπτει ένα κενό στην αξιοκρατία απόδοσης των κλήσεων που από την αρχή ήταν βασική ανάγκη για την υλοποίηση της εφαρμογής.

Σε κάθε εφαρμογή μπορεί να υπάρξει πλήρης διαφάνεια και αξιοκρατία όσο οι διαδικασίες είναι αυτοματοποιημένες. Όταν στην λειτουργία της εφαρμογής εμπλέκεται ο ανθρώπινος παράγοντας μπορούν να εμφανιστούν μη αξιοκρατικές διαδικασίες. Έτσι και στην δική μας περίπτωση. Όταν ο οδηγός ήθελε να προωθήσει μία κλήση θα μπορούσε κάλλιστα να την

προωθήσει σε όποιον οδηγό ήθελε κι όχι σε αυτόν που στην πραγματικότητα θα έπρεπε να εξυπηρετήσει την κλήση.

4.2 Πλάνο λύσης προβλημάτων

Μετά από μελέτη των προβλημάτων που προαναφέρθηκαν έπρεπε να καταλήξουμε σε μία λύση έτσι ώστε να απλοποιηθεί η λειτουργικότητα της εφαρμογής αλλά και να αυτοματοποιηθεί η διαδικασία της προώθησης κλήσεων και να αφαιρεθεί η ανθρώπινη παρέμβαση στο μέγιστο δυνατό. Για να επιτευχθεί ο στόχος μας καταλήξαμε ότι θα έπρεπε να απαλλαχθεί το σύστημά μας από την χρήση εξωτερικής εφαρμογής για την διαχείριση των τηλεφωνικών κλήσεων. Τέθηκε ως στόχος να δημιουργήσουμε δική μας εφαρμογή διαχείρισης τηλεφωνικών κλήσεων η οποία θα ενσωματωθεί στην υπάρχουσα εφαρμογή διαχείρισης στόλου ταξί. Με αυτό τον τρόπο θα καταφέρναμε να απλοποιηθεί η λειτουργικότητα της εφαρμογής στο μέγιστο αλλά και να αυτοματοποιήσουμε την διαδικασία της προώθησης κλήσεων αφού πλέον θα μπορούσαμε να ελέγξουμε πλήρως την λειτουργία της διαχείρισης κλήσεων.

Η απλοποίηση της λειτουργικότητας της εφαρμογής θα προερχόταν από δύο σημαντικές αλλαγές. Κυριότερη αλλαγή είναι αυτή που είχαμε επισημάνει κι από την αρχή τής μελέτης ως πρόβλημα. Πλέον θα υπάρχει μία και μόνο εφαρμογή η οποία θα είναι υπεύθυνη τόσο για την διαχείριση του στόλου των ταξί όσο και για την διαχείριση των τηλεφωνικών κλήσεων. Αυτό είναι αρκετό για να απαλλάξει τους τελικούς χρήστες – οδηγούς από

την διαδικασία της εναλλαγής μεταξύ δύο διαφορετικών εφαρμογών κάθε φορά που θα θέλουν να πραγματοποιήσουν μία προώθηση κλήσης ενός πελάτη σε κάποιον άλλο οδηγό. Επίσης, με την ενοποίηση των δύο αυτών εφαρμογών, μπορούσαμε πλέον να απλοποιήσουμε κάποιες επιμέρους διαδικασίες. Έτσι λοιπόν, με την δημιουργία μίας βάσης δεδομένων στο σύστημα, μπορούσαμε να αντιστοιχήσουμε τα στοιχεία εισόδου του χρήστη στην εφαρμογή, με τα στοιχεία σύνδεσης της τηλεφωνίας. Με αυτόν τον τρόπο απαλλάσσεται ο χρήστης από την απομνημόνευση πολλών στοιχείων εισόδου αλλά κι από την διαδικασία πολλαπλών ταυτοποιήσεων και αυθεντικοποιήσεων.

Με τις αλλαγές αυτές, εκτός από την απλοποίηση στην λειτουργικότητα της εφαρμογής για τον τελικό χρήστη, απλοποιείται σημαντικά και η διαδικασία εγκατάστασης και παραμετροποίησης των συσκευών χρήσης από τους τεχνικούς. Πλέον χρειάζεται να εγκατασταθεί μία και μόνο εφαρμογή στην οποία ο χρήστης συνδέεται κανονικά μόνο με το όνομα χρήστη και τον κωδικό πρόσβασης που του έχει κοινοποιηθεί από την υπηρεσία. Αυτό μας οδήγησε στην δημοσίευση της εφαρμογής στο κατάστημα της Google (Play Store), με αποτέλεσμα να μπορεί ο κάθε χρήστης να κατεβάσει μόνος του την εφαρμογή και να συνδεθεί, χωρίς να χρειάζεται η παρέμβαση κάποιου τεχνικού.

Όπως αναφέρθηκε και προηγουμένως, εκτός από την απλοποίηση της λειτουργικότητας στοχεύαμε και στην αυτοματοποίηση της διαδικασίας προώθησης κλήσεων για να επιτύχουμε πλήρη διαφάνεια και αξιοκρατία στην απόδοση των κλήσεων. Έτσι λοιπόν, με την ενσωμάτωση της

τηλεφωνίας μέσα στην εφαρμογή διαχείρισης των ταξί, μπορούσαμε να αξιοποιήσουμε όλες τις πληροφορίες που είχε η εφαρμογή για τους οδηγούς και να δημιουργήσουμε μία διαφορετική διαδικασία προώθησης των κλήσεων.

Η διαδικασία στην οποία καταλήξαμε ήταν να μην δίνεται στον χρήστη η δυνατότητα επιλογής συγκεκριμένου οδηγού για την προώθηση της κλήσης αλλά η επιλογή τής περιοχής – ζώνης στην οποία βρίσκεται ο πελάτης. Δηλώνοντας στην εφαρμογή την θέση του πελάτη, η εφαρμογή μπορούσε να δρομολογήσει την κλήση στον κατάλληλο οδηγό, με σκοπό την επίτευξη της όσο το δυνατόν γρηγορότερης εξυπηρέτησης του πελάτη.

5. ΣΤΑΔΙΑ ΥΛΟΠΟΙΗΣΗΣ

Μετά την μελέτη και την ανάλυση της υφιστάμενης εφαρμογής αλλά και των προβλημάτων προς επίλυση, αφού είχαμε καταλήξει στο τι τελικά θέλαμε να πετύχουμε και στον τρόπο με τον οποίο θα το πετυχαίναμε, αρχίσαμε την μελέτη για την υλοποίηση της ιδέας αυτής.

Ανακαλύψαμε ότι υπάρχει μία βιβλιοθήκη της Google για διαχείριση κλήσεων VoIP με χρήση του πρωτοκόλλου SIP αλλά και άλλες πολλές, αρκετές από τις οποίες και ανοιχτού κώδικα (Open Source). Καταλήξαμε στο να ξεκινήσουμε τις δοκιμές χρησιμοποιώντας την βιβλιοθήκη της Google λόγο της σιγουριάς την οποία μας έδινε αφού η δημιουργία της εφαρμογής προοριζόταν για συσκευές με λειτουργικό σύστημα Android το οποίο είναι δημιούργημα της Google.

5.1 Υλοποίηση εφαρμογής με τη βιβλιοθήκη της Google

Πριν ξεκινήσουμε την υλοποίηση της εφαρμογής χρειάστηκε να μελετήσουμε εις βάθος την βιβλιοθήκη της Google. Αφού διαβάσαμε για τον τρόπο λειτουργίας της αλλά και εφαρμογής της, καταλήξαμε στο γεγονός ότι η χρήση της για την υλοποίηση μιας απλής εφαρμογής τηλεφωνίας η οποία θα διαχειριζόταν μόνο τις βασικές λειτουργίες εισερχόμενων/εξερχόμενων κλήσεων ήταν απλή. Αυτές οι δύο λειτουργίες, σε συνδυασμό με την προώθηση κλήσεων ήταν αρκετές για να καλύψουν τις ανάγκες μας.

Στο documentation της βιβλιοθήκης είδαμε ότι υποστήριζε μόνο τους βασικούς codecs (κωδικοποιητές) φωνής. Αυτό έχει ως αποτέλεσμα, τα δεδομένα φωνής να μην συμπιέζονται αρκετά και να χρειάζεται διακίνηση μεγάλου όγκου δεδομένων. Όπως έχει αναφερθεί, η εφαρμογή απευθύνεται σε οδηγούς ταξί. Αυτό έχει ως αποτέλεσμα η διακίνηση των δεδομένων να γίνεται μέσω ασύρματων κυψελωτών δικτύων. Επειδή αυτά τα δίκτυα χρησιμοποιούνται από τις εταιρίες κινητής τηλεφωνίας και οι υπηρεσίες που παρέχουν στους τελικούς χρήστες χρεώνονται συναρτήσει του όγκου δεδομένων, είναι σημαντικό να μπορέσουμε να ελαχιστοποιήσουμε την διακίνηση δεδομένων. Εκτός από τις χρεώσεις των παρόχων, επειδή οι οδηγοί βρίσκονται εν κινήσει, η ένταση του σήματος που λαμβάνουν στις συσκευές τους δέχεται μεγάλες διακυμάνσεις. Αυτό έχει ως αποτέλεσμα την αύξηση των σφαλμάτων κατά την μετάδοση της πληροφορίας, η οποία αύξηση των σφαλμάτων είναι αντιληπτή κατά την διάρκεια μιας τηλεφωνικής κλήσης VoIP.

Παρ’ όλα αυτά και κυρίως λόγο της απλότητας στην χρήση τής βιβλιοθήκης αυτής αποφασίσαμε να την χρησιμοποιήσουμε και να κάνουμε κάποιες δοκιμές σε πραγματικές συνθήκες λειτουργίας πριν την οριστική απόρριψή της.

Δημιουργήσαμε μία εφαρμογή με τις πιο βασικές λειτουργίες που μπορεί να έχει για να μπορέσουμε να ολοκληρώσουμε κλήσεις εισερχόμενες αλλά και εξερχόμενες. Έπειτα από την δημιουργία της εφαρμογής ακολούθησε μια σειρά από πολυάριθμα πειράματα. Έγιναν δοκιμές σε πραγματικές συνθήκες λειτουργίας και με όλους τους codecs που

υποστηρίζονταν από την βιβλιοθήκη που είχαμε χρησιμοποιήσει. Η λειτουργικότητα της εφαρμογής ήταν απολύτως καλή αλλά η ποιότητα στην επικοινωνία των δύο άκρων, όπως ήταν αναμενόμενο άλλωστε, ήταν απογοητευτική. Όσο οι χρήστες ήταν ακίνητοι, η ποιότητα της μεταξύ τους επικοινωνίας ήταν μέτρια αλλά σίγουρα ήταν ανεκτή. Όταν όμως οι χρήστες βρισκόντουσαν σε κίνηση, η ποιότητα τής μεταξύ τους επικοινωνίας χειροτέρευε κι έφτανε σε επίπεδα μη αποδεκτά.

5.2 Αναφορά βιβλιοθηκών που μελετήθηκαν

Αφού τελικά απορρίφθηκε η χρήση τής βιβλιοθήκης από την Google άρχισε η αναζήτηση για μία νέα βιβλιοθήκη, η οποίο θα έπρεπε υποχρεωτικά να υποστηρίζει ποικιλία κωδικοποιήσεων φωνής για να αποφύγουμε τα προβλήματα που αντιμετωπίσαμε με την χρήση τής προηγούμενης βιβλιοθήκης. Ύστερα από μελέτη πληθώρας βιβλιοθηκών καταλήξαμε σε αυτές που μας φάνηκαν να ξεχωρίζουν από τις υπόλοιπες.

Ένα σημαντικό στοιχείο, το οποίο έπαιξε κύριο ρόλο για την ανάδειξη των συγκεκριμένων βιβλιοθηκών ήταν ότι αυτές οι βιβλιοθήκες είναι ανοιχτού κώδικα (open source). Η τρεις δημοφιλέστερες βιβλιοθήκες που αποφασίσαμε να ασχοληθούμε εκτενέστερα ήταν η Mjsip, η Pjsip και η Doubango. Αυτές οι βιβλιοθήκες μπορεί να μοιάζουν άγνωστες αλλά είναι πολύ συχνά χρησιμοποιούμενες από τους χρήστες φορητών συσκευών, μέσω πολύ δημοφιλών εφαρμογών που στηρίζονται σε αυτές. Την βιβλιοθήκη Mjsip [12] χρησιμοποιεί η εφαρμογή Sipdroid. Το Sipdroid είναι

μία εφαρμογή η οποία υπάρχει στο ηλεκτρονικό κατάστημα της Google και οποιοσδήποτε κάτοχος φορητής συσκευής με λειτουργικό σύστημα Android, μπορεί να την κατεβάσει δωρεάν στην συσκευή του. Η βιβλιοθήκη Mjsip, σε αντίθεση με την βιβλιοθήκη τής Google, υποστηρίζει αρκετούς κωδικοποιητές φωνής, μεταξύ των οποίων και ο κωδικοποιητής speex.

Αναφέρουμε τον συγκεκριμένο κωδικοποιητή γιατί τον συναντήσαμε σε αυτή την βιβλιοθήκη για πρώτη φορά και μας κίνησε το ενδιαφέρον. Ο καλύτερος κωδικοποιητής που γνωρίζαμε έως εκείνη τη στιγμή ήταν ο G729. Ήταν ο κωδικοποιητής που χρησιμοποιούταν από τους οδηγούς με την υπάρχουσα εγκατάσταση μέσω της εφαρμογής Media5fone. Όταν μελετήσαμε εις βάθος τον κωδικοποιητή speex ανακαλύψαμε για πρώτη φόρα έναν κωδικοποιητή που διέθετε χαρακτηριστικά εφάμιλλα αυτών του G729. Παρ' όλο που ο G729 ήταν δοκιμασμένος από εμάς και τον εμπιστευόμασταν, ο speex μας κέρδισε από την πρώτη στιγμή για έναν λόγο που, κατά την γνώμη μας, τον έκανε να υπερέχει του G729. Ο λόγος αυτός ήταν ότι ο speex είναι ένας κωδικοποιητής φωνής ο οποίος διανέμεται δωρεάν και είναι ανοιχτού κώδικα.

Η επόμενη βιβλιοθήκη στην λίστα μας προς μελέτη ήταν η Pjsip [13]. Κι αυτή μία βιβλιοθήκη που η πλειοψηφία δεν γνωρίζει, αν κι είναι η πιο πολυχρησιμοποιούμενη βιβλιοθήκη από τους χρήστες φορητών συσκευών μέσω της δημοφιλής, δωρεάν εφαρμογής CSipSimple, μία ακόμα εφαρμογή ανοιχτού κώδικα που διανέμεται δωρεάν στο ηλεκτρονικό κατάστημα της Google για τους χρήστες συσκευών με λειτουργικό σύστημα Android. Εκτός από την εφαρμογή αυτή, μάθαμε αργότερα ότι και η ήδη χρησιμοποιούμενη

από τους οδηγούς εφαρμογή Media5fone, κάνει χρήση τής βιβλιοθήκης Pjsip έχοντας όμως αγοράσει άδεια για χρήση τής βιβλιοθήκης σε εφαρμογή κλειστού κώδικα. Να σημειωθεί ότι η βιβλιοθήκη Pjsip καλύπτεται από την άδεια GPL version 2, παρ' όλα αυτά οι δημιουργοί της, διαθέτουν άλλη μία άδεια επί πληρωμή για την χρήση τής βιβλιοθήκης σε εφαρμογές κλειστού κώδικα. Χρήση αυτής της άδειας έκαναν και οι δημιουργοί της εφαρμογής Media5fone.

Όταν μελετήσαμε την βιβλιοθήκη Pjsip είχαμε ήδη σχεδόν καταλήξει στην χρήση της για την ανάπτυξη της εφαρμογής μας. Η Pjsip υποστηρίζει και αυτή τον κωδικοποιητή speex όπως και η Mjsip, οπότε δεν υπήρχε κάτι στο οποίο να υστερεί έναντι της Mjsip. Επίσης, πλεονέκτημα της συγκεκριμένης βιβλιοθήκης ήταν το γεγονός ότι ήταν δοκιμασμένη από εμάς ήδη, μέσω της εφαρμογής Media5fone. Όπως αναφέρθηκε νωρίτερα, η βιβλιοθήκη αυτή χρησιμοποιείται από την πολύ δημοφιλή εφαρμογή CSipSimple. Η εφαρμογή αυτή είναι ανοιχτού κώδικα και καλύπτεται από την άδεια GPL version 3. Αυτό σημαίνει ότι θα μπορούσαμε να χρησιμοποιήσουμε τον κώδικα αυτής της εφαρμογής στην δική μας, ανοιχτού κώδικα εφαρμογή, πράγμα το οποίο θα μας εξοικονομούσε πολύτιμο χρόνο.

Παρ' ότι η επιλογή μας ήταν σχεδόν βέβαιη, συνεχίσαμε και με την μελέτη της βιβλιοθήκης Doubango [14], τρίτη και τελευταία βιβλιοθήκη στην λίστα μας. Αναφερόμαστε στο Doubango με την όρο βιβλιοθήκη αν και οι δημιουργοί του δεν στάθηκαν απλά στην δημιουργία μιας βιβλιοθήκης αλλά ανέπτυξαν ολόκληρο framework. Αυτό σημαίνει ότι κάνει πολύ πιο

εύκολη τη χρήση του από μια απλή βιβλιοθήκη. Η Doubango επίσης υποστηρίζει των κωδικοποιητή speex και είναι και αυτή ανοιχτού κώδικα, καλυπτόμενη από την άδεια GPL version 3. Εάν δεν υπήρχε η ανοιχτού κώδικα εφαρμογή CSipSimple να έχει αξιοποιήσει την βιβλιοθήκη Pjsip πιθανότατα να μας είχε κερδίσει η Doubango αλλά όπως αναφέρθηκε νωρίτερα η Pjsip μας είχε ήδη κερδίσει και η επιλογή της ήταν οριστική.

5.3 Υλοποίηση εφαρμογής με βιβλιοθήκη Pjsip

Η βιβλιοθήκη Pjsip υποστηρίζει κι έχει ήδη αναπτύξει όλες τις βασικές και όχι μόνο λειτουργίες που μπορεί να χρειαστούν σε μία εφαρμογή διαχείρισης τηλεφωνικών κλήσεων μέσω της τεχνολογίας VoIP. Επιπροσθέτως να αναφέρουμε το γεγονός ότι υποστηρίζει επίσης βιντεοκλήσεις αλλά και ανταλλαγή γραπτών μηνυμάτων μέσω του πρωτοκόλλου SIP. Αν και είναι δύο εξίσου σημαντικές με την μετάδοση φωνής λειτουργίες δεν θα αναφερθούμε εκτενέστερα γιατί δεν μας αφορούσαν για την ανάπτυξη της εφαρμογής μας και δεν τις μελετήσαμε σε βάθος.

Στην υπάρχουσα εφαρμογή διαχείρισης στόλου ταξί προσθέσαμε στην αρχή μία κλάση η οποία κατά την αυθεντικοποίηση του χρήστη με την χρήση τού ονόματος χρήστη και του κωδικού πρόσβασης κάνει ένα Request στον Server ο οποίος με τη σειρά του επιστρέφει τα στοιχεία για την σύνδεση του χρήστη στην υπηρεσία τηλεφωνίας. Στη συνέχεια δημιουργήσαμε μία κλάση

η οποία καλείται από την κεντρική κλάση της εφαρμογής όταν ο χρήστης συνδεθεί.

Αυτή η κλάση είναι υπεύθυνη για την δημιουργία και διαχείριση της τηλεφωνίας. Όταν καλείται η κλάση αυτή δημιουργεί ένα αντικείμενο με τα στοιχεία που χρειάζονται για την τηλεφωνία VoIP. Τα προσωπικά στοιχεία (Όνομα χρήστη και Κωδικό πρόσβασης) του οδηγού τα έχουμε από την διαδικασία τής αυθεντικοποίησης ενώ τα υπόλοιπα είναι γνωστά. Τα γνωστά αυτά στοιχεία έχουν να κάνουν με την παραμετροποίηση των χαρακτηριστικών του χρήστη και με τον server που παρέχει τις υπηρεσίες τηλεφωνίας. Εκτενέστερη αναφορά για την κλάση αυτή θα γίνει σε επόμενη παράγραφο αυτής της εργασίας.

Ενώ αυτή η κλάση ήταν αρκετή για να πραγματοποιηθεί η σύνδεση με τον server του παρόχου των υπηρεσιών τηλεφωνίας, δημιουργήσαμε μία ακόμα κλάση μόνο και μόνο για να θέσουμε μία ακόμη δικλίδα ασφαλείας στην αδιάλειπτη λειτουργία τής τηλεφωνίας για να αποφύγουμε την περίπτωση όπου κάποιος οδηγός θα έχανε την κλήση από κάποιον εν δυνάμει πελάτη. Έτσι λοιπόν δημιουργήσαμε μία κλάση η οποία ενεργοποιεί μία ηχητική προειδοποίηση όταν για κάποιον λόγο η συσκευή του χρήστη χάσει την σύνδεση με την υπηρεσία παροχής τηλεφωνίας.

Τέλος έπρεπε να αναπτυχθεί και το κυριότερο τμήμα της εφαρμογής το οποίο ήταν κι ο κύριος λόγος για τον οποίο προχωρήσαμε στην δημιουργία της. Έπρεπε λοιπόν να τροποποιήσουμε την ήδη υπάρχουσα υλοποίηση, που υπήρχε στο CSipSimple, για την προώθηση των κλήσεων και να την κάνουμε να λειτουργεί με τέτοιο τρόπο ώστε να λύνει το

πρόβλημα που υπήρχε. Η υπάρχουσα υλοποίηση διέθετε μία επιλογή προώθησης κλήσης σε άλλον αριθμό τηλεφώνου εντός ενός μενού. Όταν ο χρήστης ακολουθούσε αυτή την επιλογή, εμφανιζόταν ένα πεδίο για την πληκτρολόγηση ενός αριθμού, καθώς επίσης και μία λίστα με τις επαφές τής συσκευής για τυχούσα επιλογή από αυτή.

Πρώτο βήμα για εμάς ήταν η επιλογή αυτή, της προώθησης κλήσης, να μην ήταν μέσα σε κάποιο μενού αλλά στην οθόνη εισερχόμενης κλήσης για να γίνει ευκολότερη η πρόσβαση των οδηγών-χρηστών σε αυτή. Έπειτα θέλαμε να αλλάξουμε τις δυνατότης που δίνονται στον χρήστη κατά την επιλογή προώθησης μιας κλήσης. Οπότε αντικαταστήσαμε την λίστα που υπήρχε με τις επαφές τής συσκευής, με μία λίστα από τις περιοχές/ζώνες στις οποίες υπήρχαν ελεύθερα ταξί για να εξυπηρετήσουν τον πελάτη. Κατά την επιλογή προώθησης μιας κλήσης από τον χρήστη, γίνεται ένα ερώτημα προς τον server τής εφαρμογής ο οποίος απαντάει με τις ζώνες εκείνες που έχουν ελεύθερα ταξί. Αφού εμφανιστεί η λίστα αυτή στον χρήστη, αυτός με τη σειρά του μπορεί να επιλέξει την ζώνη εκείνη που είναι η καταλληλότερη για την εξυπηρέτηση του πελάτη. Με την επιλογή αυτή του χρήστη, η κλήση αυτόματα προωθείται σ’ εκείνον τον οδηγό ο οποίος πρέπει να εξυπηρετήσει τον πελάτη. Με αυτόν τον τρόπο αποκλείεται από τον χρήστη-οδηγό να επιλέξει αυτός ποιος οδηγός θα εξυπηρετήσει τον πελάτη και ταυτόχρονα αποκλείεται η παράκαμψη της σειράς προτεραιότητας, κάνοντας το σύστημα ακόμα πιο αξιοκρατικό κατά την απόδοση των κλήσεων.

5.4 Δοκιμές

Αφού ολοκληρώθηκε η υλοποίηση όλων των απαραίτητων κλάσεων για την ενσωμάτωση της διαχείρισης των κλήσεων μέσα στην υφιστάμενη εφαρμογή ακολούθησαν οι απαραίτητες δοκιμές. Στην αρχή οι πρώτες δοκιμές έγιναν από εμάς τους ίδιους κι αφού διορθώθηκαν όσα προβλήματα μπορεί να παρουσιάστηκαν στην λειτουργία της εφαρμογής, στη συνέχεια η εφαρμογή εγκαταστάθηκε σε συγκεκριμένους χρήστες.

Η επιλογή των χρηστών έγινε σύμφωνα με την εξοικείωσή τους με την τεχνολογία. Έτσι η εφαρμογή εγκαταστάθηκε σε 5 χρήστες οι οποίοι εργάζονταν κανονικά με τη νέα εφαρμογή ενώ παράλληλα έκαναν όσα πειράματα μπορούσαν και φυσικά ενημερώνοντάς μας για οποιοδήποτε πρόβλημα παρουσιαζόταν ή οποιαδήποτε παρατήρηση είχαν να κάνουν τόσο στην λειτουργία τής εφαρμογής όσο και στην εμφάνισή της.

Στην συνέχεια, αφού διορθώθηκαν όσα προβλήματα είχαν παρουσιαστεί, η εφαρμογή δημοσιεύτηκε στο ηλεκτρονικό κατάστημα της Google. Όπως αναφέρθηκε νωρίτερα, ήταν κι αυτό ένας στόχος για να μπορούν εύκολα οι τελικοί χρήστες να εγκαθιστούν την εφαρμογή στις συσκευές τους, κάνοντας το έργο των τεχνικών ευκολότερο.

Έπειτα άρχισε η εφαρμογή σιγά σιγά να εγκαθίσταται σε μία μία περιοχή λειτουργίας τής υπηρεσίας σταδιακά. Κάθε περιοχή που μετέβαινε από την παλιά στη νέα εφαρμογή αναμέναμε ένα εύλογο διάστημα, να σιγουρευτούμε για την απροβλημάτιστη λειτουργία της κι έπειτα προχωρούσαμε στην μετάβαση της επόμενης περιοχής. Σε ένα διάστημα δύο

περίπου μηνών, και οι επτά συμβεβλημένοι στην υπηρεσία νομοί λειτουργούσαν με την νέα εφαρμογή.

6. ΠΕΙΓΡΑΦΗ ΒΑΣΙΚΩΝ ΚΛΑΣΕΩΝ

Όπως αναφέρθηκε σε προηγούμενο κεφάλαιο, για να ενσωματώσουμε την λειτουργία τής τηλεφωνίας στην υφιστάμενη εφαρμογή χρησιμοποιήθηκε η βιβλιοθήκη Pjsip. Πέρα από τις βασικές λειτουργίες τηλεφωνίας που καλύπτονται από την βιβλιοθήκη, χρειάστηκε η ανάπτυξη τριών βασικών λειτουργιών. Η πρώτη είναι η αντιστοιχία των προσωπικών στοιχείων του χρήστη με τα στοιχεία για την αυθεντικοποίηση της τηλεφωνίας, η δεύτερη είναι η δημιουργία του λογαριασμού τηλεφωνίας και η σύνδεσή του με τον server του παρόχου. Η τρίτη και τελευταία λειτουργία είναι η προώθηση κλήσεων. Στις επόμενες παραγράφους ακολουθεί ανάλυση των τριών βασικών κλάσεων για την υλοποίηση των προαναφερθέντων λειτουργιών.

6.1 Τροποποίηση της κλάσης LoginRequest

Στην υφιστάμενη εφαρμογή διαχείρισης στόλου ταξί υπήρχε η κλάση Login, η οποία ήταν υπεύθυνη να διεκπεραιώσει την διαδικασία ταυτοποίησης και αυθεντικοποίησης του χρήστη. Αυτό που είχαμε να προσθέσουμε σε αυτή την κλάση ήταν η αίτηση στον server για να αποκτήσουμε τα στοιχεία σύνδεσης με τον πάροχο της υπηρεσίας τηλεφωνίας. Έτσι λοιπόν μετά την επιτυχή αυθεντικοποίηση του χρήστη στο σύστημα διαχείρισης στόλου ταξί, γίνεται η κλήση στον server αποστέλλοντας το όνομα χρήστη και τον κωδικό πρόσβασης για να

απαντήσει αυτός με τη σειρά του στέλνοντας πίσω τα στοιχεία για την σύνδεση με την υπηρεσία τηλεφωνίας. Όπως φαίνεται και στον κώδικα που ακολουθεί, το request γίνεται με την μέθοδο post. Μετά την λήψη τής απάντησης από τον server καλούμε την μέθοδο initializeSip περνώντας ως παράμετρο την απάντηση του server σε μορφή JSON. Από την απάντηση ανακτάτε το όνομα χρήστη και ο κωδικός πρόσβασης για την αυθεντικοποίηση του χρήστη στην υπηρεσία τηλεφωνίας, καθώς επίσης η IP διεύθυνση του παρόχου τής τηλεφωνίας και ο τηλεφωνικός αριθμός που αντιστοιχεί στον χρήστη. Στη συνέχεια, αφού υπάρχουν όλα τα απαραίτητα στοιχεία, δημιουργείται ένα service τής κλάσης Connection από την κύρια κλάση τής εφαρμογής.

```
public class LoginRequest implements IRequest {
public static String ACTION = "gr.dga.mtaxi.login";
private Client c = new Client();
private String loginURL = "/driverlogin.json";
private Request req;
private String username;
private String password;
private Context context;
private String version;

public LoginRequest(Context context, String username, String
            password, String version) {
      this.context = context;
      String url=RequestManager.getBaseURL().concat(loginURL);
      req = new Request(url, Request.Method.POST);
      this.username = username;
      this.password = password;
      this.version = version;
      req.addBasicAuthentication(username, this.password);
      req.addParam("versioncode", version);
}

@Override
public void doRequest() {
      try {
            JSONObject loginResponse;
            Response res;
            if (RequestManager.DUMMY) {
                  String response =
                        Utils.dummyResponse("res/raw/login.json");
                  res = new Response(Response.Code.OK.getValue(),
                        "OK", response);
                  loginResponse = res.getJSONObject();
            } else {
                  res = c.execute(req);
                  loginResponse = res.getJSONObject();
            }
            Intent i = new Intent(ACTION);
            if (res.getResponseCode()==Response.Code.OK.getValue()) {
```

```
        Request request = new
                Request("http://sipsettings.mtaxi.gr",
                Request.Method.POST);
        Response response;
        request.addParam("username", this.username);
        request.addParam("password", this.password);
        JSONObject sipResponse;
        response = c.execute(request);
        sipResponse = response.getJSONObject();
        if (response.getResponseCode() ==
                Response.Code.OK.getValue()) {
                initializeUser(context, loginResponse);
                initializeZones(context, loginResponse);
                initializeSip(context, sipResponse);
                checkVersion(context, loginResponse, i);
                getLastCallDate(context);
                Intent i2 = new Intent(context,
                        RequestManager.class);
                i2.setAction(StatusRequest.ACTION);
                i2.putExtra("status", Status.Inactive.name());
                context.startService(i2);
                i.putExtra("responseCode",
                Response.Code.OK.toString());
                context.sendBroadcast(i);
        } else {
                i.putExtra("responseCode",
                        Response.Code.FAILURE.toString());
                context.sendBroadcast(i);
        }
} else if (res.getResponseCode() ==
                        Response.Code.UNAUTH.getValue()) {
        i.putExtra("responseCode",
        Response.Code.UNAUTH.toString());
        context.sendBroadcast(i);
} else if (res.getResponseCode() == Response.Code.NOT_FOUND
                .getValue()) {
        i.putExtra("responseCode",
                Response.Code.NOT_FOUND.toString());
        checkVersion(context, loginResponse, i);
        context.sendBroadcast(i);
} else {
```

```
                i.putExtra("responseCode",
                        Response.Code.FAILURE.toString());
                context.sendBroadcast(i);
            }
        } catch (Exception e) {
            e.printStackTrace();
            Intent i = new Intent(ACTION);
            i.putExtra("responseCode", Response.Code.FAILURE.toString());
            context.sendBroadcast(i);
        }
    }
    private void initializeUser(Context c, JSONObject obj) {
        try {
            String username = obj.getJSONObject("userHasVehicleDto")
                    .getJSONObject("userDto").getString("username");
            String password = this.password;
            String id =
                    obj.getJSONObject("userHasVehicleDto").getString("id");
            String region =
                    obj.getJSONObject("regionDto").getString("name");
            String regionlabel = obj.getJSONObject("regionDto").getString(
                    "label");
            String firstname = obj.getJSONObject("userHasVehicleDto")
                    .getJSONObject("userDto").getString("firstName");
            String lastname = obj.getJSONObject("userHasVehicleDto")
                    .getJSONObject("userDto").getString("lastName");
            User.initialize(context, username, password, firstname,
                    lastname, id, region, regionlabel);
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
    private void initializeSip(Context c, JSONObject obj) {
        try {
            MainActivity.username = obj.getString("sip_username");
            MainActivity.password = obj.getString("sip_password");
            MainActivity.server = obj.getString("server_ip");
            MainActivity.displayName = obj.getString("sip_number");
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
```

```
private void initializeZones(Context c, JSONObject obj) {
	try {
		JSONArray jsonZones =
			obj.getJSONObject("regionDto").getJSONArray(
			"zones");
		Zones zones = Zones.get(context);
		zones.clearZones();
		for (int i = 0; i < jsonZones.length(); i++) {
			JSONObject curr = (JSONObject) jsonZones.get(i);
			Zones.Type type = Zones.Type.Dummy;
			String nameType = "dummyZoneName";
			if (curr.has("compoundZoneName")) {
				type = Zones.Type.Compound;
				nameType = "compoundZoneName";
			} else if (curr.has("ringZoneName")) {
				type = Zones.Type.Ring;
				nameType = "ringZoneName";
			} else if (curr.has("boxZoneName")) {
				type = Zones.Type.Box;
				nameType = "boxZoneName";
			} else if (curr.has("triZoneName")) {
				type = Zones.Type.Tri;
				nameType = "triZoneName";
			}
			String name = curr.getString(nameType);
			String label = ((JSONObject) jsonZones.get(i))
				.getString("label");
			Zones.Zone zone = zones.new Zone(name, label, type, i);
				zones.add(zone);
		}
		zones.prZones();
	} catch (JSONException e) {
		e.printStackTrace();
	}
}
private void checkVersion(Context c, JSONObject obj, Intent i) {
	String versionname = null;
	String versioncode = null;
	String url = null;
	if (obj.has("updateToMtaxiVersion")) {
		try {
			versionname = obj.getJSONObject("updateToMtaxiVersion")
```

```
                        .getString("name");
                versioncode = obj.getJSONObject("updateToMtaxiVersion")
                        .getString("code");

                if (obj.getJSONObject("updateToMtaxiVersion").has("url"))
                        url =
            obj.getJSONObject("updateToMtaxiVersion").getString("url");
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
        if (versioncode != null && !versioncode.equals(version)) {
            i.putExtra("versionname", versionname);
            if (url != null)
                i.putExtra("url", url);
        }
    }
    private void getLastCallDate(Context c) {
        String[] projection = { CallLog.Calls.NUMBER, CallLog.Calls.TYPE,
                        CallLog.Calls.DATE, CallLog.Calls.DURATION };
        String selection = CallLog.Calls.TYPE + " = "
                        + CallLog.Calls.INCOMING_TYPE;
        Cursor managedCursor = c.getContentResolver().query(
                        CallLog.Calls.CONTENT_URI, projection, selection,
                        null, CallLog.Calls.DATE + " DESC");
        if (managedCursor.moveToFirst()) {
            long date = managedCursor.getLong(managedCursor
                    .getColumnIndex(CallLog.Calls.DATE));
            User.get(c).setLastCallDate(date);
            managedCursor.close();
        }
    }}
```

6.2 Η κλάση Connection

Όπως αναφέρθηκε σε προηγούμενο κεφάλαιο, μετά την απόκτηση των απαραίτητων στοιχείων για την σύνδεση στην υπηρεσία τηλεφωνίας δημιουργείται ένα service της κλάσης Connection. Αυτή η κλάση αναλαμβάνει την δημιουργία ενός προφίλ για τον χρήστη και την σύνδεσή του με τον server τού παρόχου υπηρεσιών τηλεφωνίας. Σε αυτή την παράγραφο θα αναφερθούν και θα αναλυθούν οι μέθοδοι της κλάσης αυτής.

Οι μέθοδοι της κλάσης χωρίζονται σε δύο βασικές κατηγορίες. Από την μία έχουμε τρεις μεθόδους οι οποίες έχουν κληρονομηθεί από την κλάση Service και αφορούν στάδια του κύκλου ζωής της κλάσης. Από την άλλη υπάρχει άλλη μία μέθοδος η οποία δεν κληρονομείται από κάποια άλλη κλάση κι είναι αυτή που δημιουργεί και συνδέει τον χρήστη με τον πάροχο τηλεφωνίας.

Η πρώτη μέθοδος που κληρονομείται από την κλάση Service είναι η μέθοδος onCreate. Αυτή η μέθοδος καλείται κατά την δημιουργία ενός αντικειμένου τής κλάσης αυτής και το μόνο που κάνει είναι να καλέσει την αντίστοιχη μέθοδο της κλάσης δημιουργού, δηλαδή της κλάσης στην οποία ανήκει το αντικείμενο το οποίο δημιούργησε το παρόν. Το σώμα της έχει μόνο αυτή την εντολή και τίποτα περισσότερο.

Στη συνέχεια υπάρχει η μέθοδος onDestroy και αυτή έχει κληρονομηθεί από την μητρική κλάση Service. Είναι μέθοδος που αφορά και αυτή με τη σειρά της τον κύκλο ζωής της κλάσης. Συγκεκριμένα η μέθοδος αυτή καλείται κατά την καταστροφή ενός αντικειμένου τής κλάσης αυτής.

Στο σώμα της γίνεται ένας έλεγχος για το αν υπάρχει κάποια ενεργή σύνδεση με τον Server του παρόχου υπηρεσιών τηλεφωνίας. Σε περίπτωση που υπάρχει, καλεί την μέθοδο stopService κι αυτή είναι μέθοδος της μητρικής κλάσης Service, περνώντας της ως όρισμα την ενεργή σύνδεση για την διακοπή εκτέλεσής της. Και τέλος, στο σώμα τής μεθόδου onDestroy καλείται η αντίστοιχη μέθοδος της κλάσης στην οποία ανήκει το αντικείμενο δημιουργός του παρόντος αντικειμένου.

Τρίτη και τελευταία μέθοδος από τις μεθόδους που κληρονομούνται από την κλάση Service είναι η μέθοδος onStartCommand. Όπως οι δύο μέθοδοι που αναλύθηκαν νωρίτερα έτσι και αυτή συσχετίζεται άμεσα με τον κύκλο ζωής της κλάσης. Αυτή η μέθοδος καλείται για πρώτη φορά αμέσως μετά την δημιουργία ενός αντικειμένου τής κλάσης αυτής και συγκεκριμένα μετά την εκτέλεση της μεθόδου onCreate. Εκτός όμως από την πρώτη εκτέλεση της μεθόδου κατά την δημιουργία ενός αντικειμένου, εκτελείται κάθε φορά που μπορεί ο κύκλος ζωής ενός αντικειμένου τής κλάσης αυτής, επανέρχεται στην κατάσταση «εκτέλεσης» (run) μετά από προσωρινή διακοπή της. Στο σώμα αυτής της μεθόδου γίνεται και η κλήση τής τέταρτης και τελευταία μεθόδου αυτής τής κλάσης. Έτσι κάθε φορά που δημιουργείται ένα αντικείμενο αυτής της κλάσης ή κάθε φορά που καλείται να μεταβεί σε κατάσταση «εκτέλεσης».

Όπως αναφέρθηκε νωρίτερα, η κλάση Connection εκτός από τις τρεις μεθόδους που κληρονομεί από την μητρική κλάση Service, διαθέτει μία ακόμα μέθοδο. Αυτή η μέθοδος είναι η connectSip. Αυτή η μέθοδος είναι

αρμόδια για την δημιουργία του προφίλ τού χρήστη και την σύνδεσή του με την υπηρεσία τηλεφωνίας.

Στις πρώτες γραμμές γίνεται ανάκτηση από την μνήμη τής συσκευής των αποθηκευμένων παραμέτρων της εφαρμογής. Έπειτα γίνεται ένας έλεγχος για το αν έχει ήδη δημιουργηθεί προφίλ σύνδεσης για τον χρήστη. Στην συνέχεια και σε περίπτωση που δεν υπάρχει ήδη κάποιο προφίλ για τον χρήστη, ακολουθούν οι γραμμές δημιουργίας του και παραμετροποίησής του. Στην αρχή δημιουργείται το προφίλ περνώντας τα απαραίτητα στοιχεία για την ταυτοποίηση και αυθεντικοποίηση του χρήστη καθώς επίσης και τα στοιχεία του παρόχου της υπηρεσίας τηλεφωνίας. Αυτά τα στοιχεία είναι τα εξής:

Account Id

Το account id είναι το αναγνωριστικό του χρήστη στο διαδίκτυο. Είναι κάτι αντίστοιχο της διεύθυνσης IP ενός web server. Αποτελείται από το username του χρήστη και την IP διεύθυνση του Proxy Server. Η γενική μορφή ενός SIP Account Id είναι "<sip:[username]@[proxy server IP]>". Όταν κάποιος κάνει κλήση σε μία διεύθυνση αυτού του τύπου αναγνωρίζεται άμεσα ότι αναφέρεται σε λογαριασμό SIP, σε ποιον Proxy Server είναι συνδεδεμένος ο χρήστης και ποιο είναι το username του, το μοναδικό αναγνωριστικό του για τον Proxy Server.

Registration Uri

Το Registration Uri δηλώνει τη διεύθυνση του Proxy Server στον οποίο συνδέεται ο χρήστης. Είναι ένα μέρος του account id, αποτελείται από την δήλωση χρήσης του πρωτοκόλλου SIP και την διεύθυνση IP του Proxy Server. Η γενική μορφή ενός Registration Uri είναι "sip:[proxy server IP]".

Username

Το username είναι το όνομα χρήστη και κατ' επέκταση η ταυτότητά του που χρησιμοποιείται για την . Είναι μοναδικό για κάθε συνδεδεμένο χρήστη στον ίδιο Proxy Server, οπότε το username σε συνδυασμό με την IP του Proxy Server κάνουν τον κάθε χρήστη μοναδικό στο διαδίκτυο.

Password

Το Password, όπως σε κάθε χρήση του στο διαδίκτυο είναι για την αυθεντικοποίηση του χρήστη. Έτσι και στη συγκεκριμένη εφαρμογή χρησιμοποιείται για την αυθεντικοποίηση του χρήστη κατά την σύνδεσή του με τον Proxy Server.

Proxy Server

Αυτό το πεδίο είναι σχεδόν ίδιο με το Registration Uri. Από χαρακτηριστικά έχει τα ίδια ακριβώς με το Registration Uri και ποιο συγκεκριμένα, σε περίπτωση που ο χρήστης εξυπηρετείται από έναν και μόνο Proxy Server δεν έχει απολύτως καμία διαφορά από το Registration Uri. Η διαφορά τους όμως είναι ότι αυτό το πεδίο σε αντίθεση με το

Registration Uri δέχεται ως όρισμα μία λίστα από διευθύνσεις περισσότερων του ενός Proxy Server.

Registration Timeout

Το Registration Timeout δέχεται ως όρισμα έναν ακέραιο αριθμό ο οποίος δηλώνει το χρονικό όριο για την απάντηση του Proxy Server σε κάθε αίτημα σύνδεσης που δέχεται από τον χρήστη.

Μετά την ολοκλήρωση των βασικών ρυθμίσεων του προφίλ τού χρήστη ακολουθεί ένα πακέτο εντολών παραμετροποίησης των κωδικοποιητών φωνής. Συγκεκριμένα δηλώνονται όλοι οι κωδικοποιητές φωνής που μπορούν να υποστηριχτούν από τον χρήστη τής εφαρμογής, καθώς επίσης και οι προτεραιότητά τους. Κατά την πραγματοποίηση μίας κλήσης, είτε αυτή είναι εισερχόμενη είτε εξερχόμενη, ο χρήστης επικοινωνεί με τον Proxy Server του παρόχου και ρωτάει για έναν έναν τους κωδικοποιητές φωνής από την λίστα προτεραιότητας έως ότου καταλήξουν σε ταύτιση κάποιου κωδικοποιητή ο οποίος υποστηρίζεται κι από τις δύο πλευρές (χρήστη – εξυπηρετητή).

Αφού ολοκληρωθεί κι η ρύθμιση της σειράς προτεραιότητας των κωδικοποιητών φωνής ακολουθούν οι τελευταίες ρυθμίσεις του προφίλ τού χρήστη. Σε αυτό το τελευταίο μπλοκ γίνονται ρυθμίσεις σύμφωνα με τις επιτρεπόμενος λειτουργίες του χρήστη αναλόγως του δικτύου χρήσης της συσκευής. Από την μία πλευρά υπάρχουν οι λειτουργίες των εξερχόμενων και των εισερχόμενων κλήσεων, ενώ από την άλλη υπάρχουν τα δίκτυα

δεδομένων που διαθέτουν οι φορητές συσκευές. Αυτά τα δίκτυα είναι από την μία το δίκτυο τεχνολογίας WiFi κι από την άλλη τα δεδομένα του παρόχου κινητής τηλεφωνίας χωρισμένα ανάλογα με το δίκτυο χρήσης (GPRS, EDGE, 3G). Έτσι δημιουργούνται όλοι οι συνδυασμοί ανά δύο δηλώνοντας για κάθε έναν από τους συνδυασμούς αν επιτρέπεται ή όχι.

Μετά την ολοκλήρωση της παραμετροποίησης του προφίλ τού χρήστη ακολουθεί το αίτημα σύνδεσης του χρήστη με τον Server του παρόχου τής υπηρεσίας τηλεφωνίας. Μετά από αυτό το αίτημα έχει ολοκληρωθεί η διαδικασία παραμετροποίησης και σύνδεσης του χρήστη με τον πάροχο και μαζί με την διαδικασία ολοκληρώνεται και η τελευταία μέθοδος της κλάσης Connection.

```
public class Connection extends Service {

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

	@Override
	public void onDestroy() {
		if (MainActivity.it != null)
			stopService(MainActivity.it); //Disconnect SIP Service
		super.onDestroy();
	}

	@Override
	public int onStartCommand(Intent intent, int flags, int startId) {
		connectSip();
		return super.onStartCommand(intent, flags, startId);
	}

	public void connectSip() {
		SharedPreferences prefs = PreferenceManager
				.getDefaultSharedPreferences(this);
		boolean alreadySetup = prefs.getBoolean(
				MainActivity.SAMPLE_ALREADY_SETUP, false);
		if (!alreadySetup) {
			SipConfigManager.setPreferenceStringValue(this,
					SipConfigManager.LOG_LEVEL, "4");
		}

		// Get current account if any
		Cursor c = getContentResolver().query(
				SipProfile.ACCOUNT_URI,
				new String[] { SipProfile.FIELD_ID,
					SipProfile.FIELD_ACC_ID,
					SipProfile.FIELD_REG_URI }, null, null,
					SipProfile.FIELD_PRIORITY + " ASC");
		if (c != null) {
			try {
				if (c.moveToFirst()) {
					SipProfile foundProfile = new SipProfile(c);
```

```
                    MainActivity.existingProfileId =
                    foundProfile.id;
              }
        } catch (Exception e) {
              Log.e(MainActivity.THIS_FILE,
              "Some problem occured while accessing cursor", e);
        } finally {
              c.close();
        }
}

// Build SipProfile
MainActivity.builtProfile = new SipProfile();
MainActivity.builtProfile.display_name =
      MainActivity.displayName;
MainActivity.builtProfile.id = MainActivity.existingProfileId;
MainActivity.builtProfile.acc_id = "<sip:" +
      MainActivity.username + "@" + MainActivity.server + ">";
MainActivity.builtProfile.reg_uri = "sip:"+MainActivity.server;
MainActivity.builtProfile.realm = "*";
MainActivity.builtProfile.username = MainActivity.username;
MainActivity.builtProfile.data = MainActivity.password;
MainActivity.builtProfile.proxies = new String[] { "sip:"
            + MainActivity.server };
MainActivity.builtProfile.reg_timeout = 5;

// Set codecs priority
PreferencesWrapper preferencesWrapper = new
            PreferencesWrapper(this);
preferencesWrapper.setCodecPriority("PCMU/8000/1",
            SipConfigManager.CODEC_NB, "60");
preferencesWrapper.setCodecPriority("PCMA/8000/1",
            SipConfigManager.CODEC_NB, "50");
preferencesWrapper.setCodecPriority("PCMU/8000/1",
            SipConfigManager.CODEC_WB, "60");
preferencesWrapper.setCodecPriority("PCMA/8000/1",
            SipConfigManager.CODEC_WB, "50");
preferencesWrapper.setCodecPriority("G729/8000/1",
            SipConfigManager.CODEC_NB, "100");
preferencesWrapper.setCodecPriority("G729/8000/1",
            SipConfigManager.CODEC_WB, "100");
preferencesWrapper.setCodecPriority("speex/8000/1",
```

```
                SipConfigManager.CODEC_NB, "90");
        preferencesWrapper.setCodecPriority("speex/8000/1",
                SipConfigManager.CODEC_WB, "90");
        // Set network preferences
        SipConfigManager.setPreferenceBooleanValue(this,
                SipConfigManager.USE_WIFI_IN, true);
        SipConfigManager.setPreferenceBooleanValue(this,
                SipConfigManager.USE_WIFI_OUT, true);
        SipConfigManager.setPreferenceBooleanValue(this,
                SipConfigManager.USE_3G_IN, true);
        SipConfigManager.setPreferenceBooleanValue(this,
                SipConfigManager.USE_3G_OUT, true);
        SipConfigManager.setPreferenceBooleanValue(this,
                SipConfigManager.USE_EDGE_IN, true);
        SipConfigManager.setPreferenceBooleanValue(this,
                SipConfigManager.USE_EDGE_OUT, true);
        SipConfigManager.setPreferenceBooleanValue(this,
                SipConfigManager.USE_GPRS_IN, true);
        SipConfigManager.setPreferenceBooleanValue(this,
                SipConfigManager.USE_GPRS_OUT, true);
        // Start Sip Service
        ContentValues builtValues = MainActivity.builtProfile
                .getDbContentValues();
        if (MainActivity.existingProfileId != SipProfile.INVALID_ID) {
              getContentResolver().update(
            ContentUris.withAppendedId(SipProfile.ACCOUNT_ID_URI_BASE,
            MainActivity.existingProfileId), builtValues, null, null);
        } else {
              Uri savedUri =
                    getContentResolver().insert(SipProfile.ACCOUNT_URI,
                          builtValues);
              if (savedUri != null) {
                    MainActivity.existingProfileId =
                          ContentUris.parseId(savedUri);
              }
        }
        if (true) {
          MainActivity.it = new Intent (SipManager.INTENT_SIP_SERVICE);
          this.startService(MainActivity.it);
        }
      }
}
```

6.3 Τροποποίηση της κλάσης EditSipUri

Τελευταία κλάση με την οποία θα ασχοληθούμε στην εργασία και την οποία θα αναλύσουμε σε αυτή την παράγραφο είναι η κλάση EditSipUri. Αυτή η κλάση υπήρχε υλοποιημένη στην εφαρμογή CSipSimple και χρειάστηκε λίγες γραμμές κώδικα για την διαμόρφωσή της με σκοπό να εξυπηρετήσει τις ανάγκες τής υφιστάμενης εφαρμογής διαχείρισης στόλου ταξί.

Κατά την σύνδεση του οδηγού στο σύστημα διαχείρισης στόλου ταξί, αποθηκεύεται στην συσκευή του χρήστη, σε μορφή αρχείου κειμένου, η περιοχή εργασίας του οδηγού. Κατά την επιλογή του χρήστη να προωθήσει μία κλήση σε άλλη ζώνη, γίνεται η ανάκτηση της περιοχής εργασίας του οδηγού από το αρχείο που είχε αποθηκευτεί κατά την σύνδεση του οδηγού. Αυτές είναι κι οι πρώτες γραμμές που προστέθηκαν στην κλάση EditSipUri. Στην συνέχει γίνεται μία κλήση στον server τής υπηρεσίες διαχείρισης στόλου ταξί, στην διεύθυνση `http://ms.mtaxi.gr:8069/v1/Cities/"[aerea]"/AreaBuffers/_STATE-ActiveForCall/queue.json`. Ο server με τη σειρά του απαντάει με μία λίστα στην οποία περιέχονται οι οδηγοί κατά σειρά προτεραιότητας, χωρισμένοι σε ομάδες αναλόγως τη ζώνη στην οποία βρίσκονται την δεδομένη στιγμή.

Έπειτα γίνεται μία ανάλυση στην απάντηση του server και φτιάχνεται μία λίστα με όσες ζώνες έχουν διαθέσιμους οδηγούς κι άλλη μία λίστα με τον τηλεφωνικό αριθμό του πρώτου οδηγού από κάθε ζώνη.

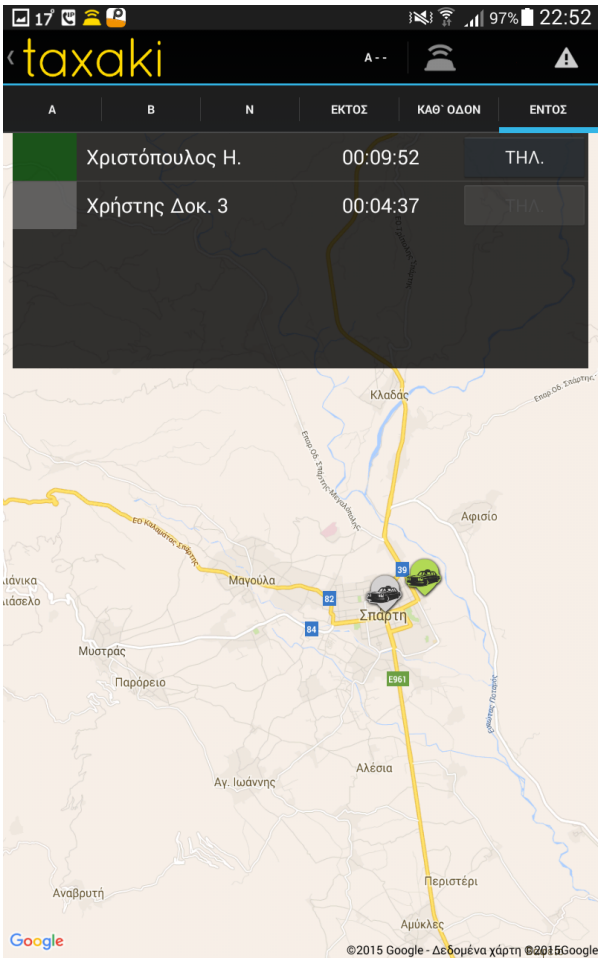
```
new Thread(new Runnable() {
    @Override
    public void run() {
        //Read Region from file
        String aBuffer = "";
        try {
            File myFile = new
File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DO
WNLOADS)                                                    +
"/mTAXIRegion.txt");
            FileInputStream fIn = new FileInputStream(myFile);
            BufferedReader myReader = new BufferedReader(
                                        new InputStreamReader(fIn));
            String aDataRow = "";
            while ((aDataRow = myReader.readLine()) != null) {
                aBuffer += aDataRow;
            }
            myReader.close();
        } catch (Exception e) {

        }
        response = Utils.getRequest("http://ms.mtaxi.gr:8069/v1/Cities/"
                                                    + aBuffer
                                                    +
"/AreaBuffers/_STATE-ActiveForCall/queue.json",
                                            "Basic DpjMTENTc2NOWE2Y2");
        InputStream instream;
        while (response == null);
        try {
            instream = response.getEntity().getContent();
            resultJSON = Utils.convertStreamToString(instream);
            json = new JSONObject(resultJSON);
            instream.close();
            JSONObject dto = json.getJSONObject("dto");
            JSONObject priority = dto.getJSONObject("priority");
            JSONArray queue = priority.getJSONArray("queue");
            for (int i = 0; i < queue.length(); i++) {
                JSONObject q = queue.getJSONObject(i);
                if (zones == null)
                    zones = new String[queue.length()];
                if (phones == null)
                    phones = new String[queue.length()];
```

```
                        zones[i] = q.getString("currentZoneName");
                        phones[i] = q.getString("phoneNumber");
            }
            // Clear from zones duplicates
            if (zones != null) {
                for (int i = 0; i < zones.length; i++) {
                    boolean FLAG = false;
                            if (finalZones == null) {
                                finalZones = new ArrayList<String>();
                                finalZones.add(zones[i]);
                                finalPhones = new ArrayList<String>();
                                finalPhones.add(phones[i]);
                            }
                            for (int j = 0; j < finalZones.size(); j++) {
                                if
(zones[i].equalsIgnoreCase(finalZones.get(j))) {
                                    FLAG = true;
                                    break;
                                }
                            }
                            if (!FLAG) {
                                finalZones.add(zones[i]);
                                finalPhones.add(phones[i]);
                            }
                        }
            }
            // Sort finalZones and finalPhones
            if (finalZones != null && finalPhones != null)
                        concurrentSort(finalZones, finalZones,
finalPhones);
            hasEnd = true;
        } catch (IllegalStateException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (JSONException e) {
            e.printStackTrace();
        }
        handler.sendEmptyMessage(0);
    }
}).start();
```

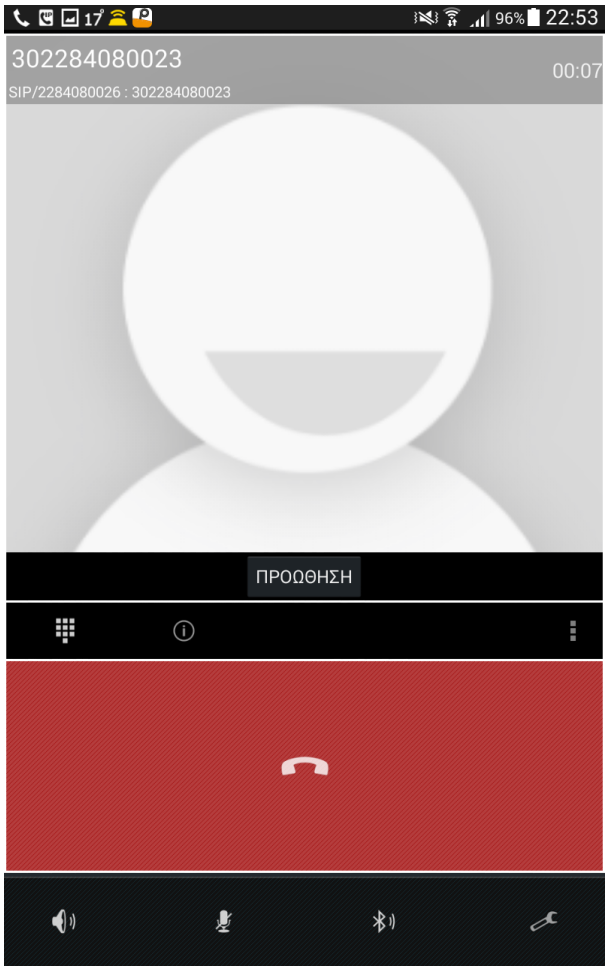
7. ΠΑΡΑΔΕΙΓΜΑ ΕΚΤΕΛΕΣΗΣ

Κατά την εκτέλεση της εφαρμογής το πρώτο πράγμα που έχει να κάνει ο χρήστης – οδηγός είναι να συνδεθεί στην υπηρεσία διαχείρισης στόλου ταξί χρησιμοποιώντας το όνομα χρήστη και τον κωδικό πρόσβασης που έχει προμηθευτεί από την υπηρεσία. Αφού γίνει η ταυτοποίηση και η αυθεντικοποίηση του χρήστη, του δίνεται η δυνατότητα να δει στον χάρτη τη θέση των συναδέλφων του οδηγών όπως φαίνεται στην εικόνα 7.1.



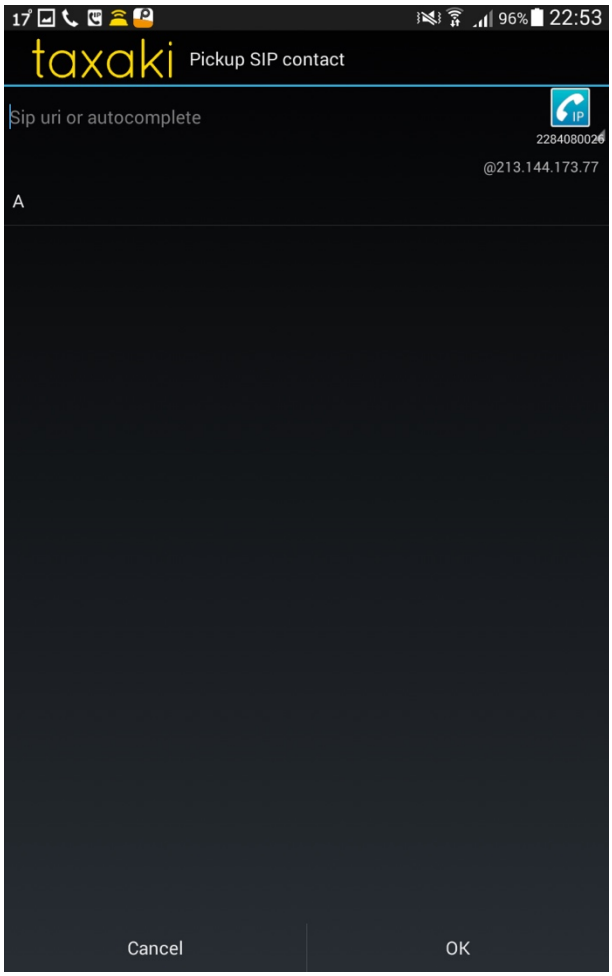
Εικόνα 7.1: Οθόνη παρατήρησης θέσης οδηγών πάνω στον χάρτη

Όταν ο χρήστης δεχτεί μία κλήση από κάποιον πελάτη κι αφού την αποδεχτεί εμφανίζεται η οθόνη της εφαρμογής για την διαχείριση της ενεργής κλήσης. Όπως φαίνεται και στην εικόνα 7.2 ο χρήστης βλέπει κάποιες πληροφορίες για την κλήση, όπως είναι τα στοιχεία του συνομιλητή του, η διάρκεια της κλήσης και άλλα. Επίσης υπάρχει και πλήθος πλήκτρων για την διαχείριση της κλήσης με βασικότερα από αυτά, το πλήκτρο τερματισμού τής κλήσης αλλά και το πλήκτρο για την προώθηση της κλήσης σε άλλον οδηγό.



Εικόνα 7.2: Οθόνη διαχείρισης εισερχόμενης κλήσης

Στην περίπτωση όπου ο χρήστης επιλέξει να προωθήσει την κλήση, στην οθόνη τής συσκευής του εμφανίζεται η οθόνη με την λίστα των ζωνών στις οποίες υπάρχουν διαθέσιμοι οδηγοί για να εξυπηρετήσουν τον πελάτη. Η οθόνη αυτή φαίνεται στην εικόνα 7.3 στην οποία η μόνη ζώνη με ελεύθερο οδηγό για να εξυπηρετήσει τον πελάτη είναι η ζώνη Α. Σε περίπτωση που ο οδηγός κρίνει ότι ο οδηγός που βρίσκεται στη ζώνη Α είναι σε θέση να εξυπηρετήσει ταχύτερα τον πελάτη, το μόνο που έχει να κάνει είναι να επιλέξει την ζώνη Α και να πατήσει το πλήκτρο OK. Το σύστημα θα προωθήσει την κλήση τού πελάτη στον κατάλληλο οδηγό για την ταχύτερη δυνατή εξυπηρέτησή του.



Εικόνα 7.3: Οθόνη επιλογής ζώνης για προώθηση κλήσης

ΒΙΒΛΙΟΓΡΑΦΙΑ - ΑΝΑΦΟΡΕΣ

[1] Η επανάσταση τηλεφωνικών υπηρεσιών VoIP- http://www.answeringphoneservices.com/voip/voiptelephoneservices.php

[2] H.323 : Packet-based multimedia communications systems - http://www.itu.int/rec/T-REC-H.323/e

[3] RFC 2543 - SIP: Session Initiation Protocol - https://tools.ietf.org/html/rfc2543

[4] RFC 3261 - SIP: Session Initiation Protocol - https://tools.ietf.org/html/rfc3261

[5] RFC 2822 - Internet Message Format - https://tools.ietf.org/html/rfc2822

[6] RFC 3621 - Power Ethernet MIB – https://tools.ietf.org/html/rfc3621

[7] Internet Communications Using SIP: Delivering VoIP and Multimedia Services with Session Initiation Protocol (Networking Council), Henry Sinnreich, Alan B. Johnston ISBN: 978-0-471-41399-8

[8] RFC 3261 - SIP: Session Initiation Protocol - http://www.faqs.org/rfcs/rfc3261.html

[9] Programming SIP / About SIP / Programming SIP - http://www.sipcenter.com/sip.nsf/html/Programming+SIP

[10] Session Initiation Protocol - Wikipedia, the free encyclopedia - http://en.wikipedia.org/wiki/Session_Initiation_Protocol

[11] Android - https://www.android.com/

[12] Mjsip Library Documentation - http://www.mjsip.org/doc/index.html

[13] Pjsip Library Documentation - http://trac.pjsip.org/repos/wiki

[14] Doubango Library / Framework - https://doubango.org/

[15] Java History - http://www.oracle.com/technetwork/java/javase/overview/javahistory-index-198355.html