ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Τμήμα Ψηφιακών Συστημάτων

Πτυχιακή εργασία

Ρηγινιώτης Δημήτριος ,Ε09147

« Μελέτη και Ανάπτυξη Συστήματος επικοινωνιών φωνής με βάση το πρότυπο NGN»

Επιβλέπων καθηγητής: Μηλιώνης Απόστολος, Επίκουρος Καθηγητής

Πειραιάς , Δεκέμβριος 2016

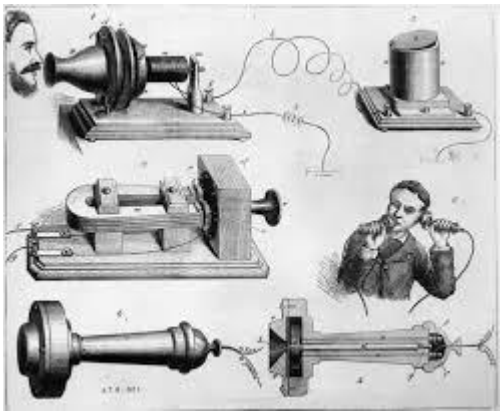
Περιεχόμενα

1. Εισαγωγή

1.1 Ιστορική αναδρομή

Στις σύγχρονες κοινωνίες είναι πλέον γεγονός ότι οι τηλεπικοινωνίες έχουν γίνει αναπόσπαστο κομμάτι της καθημερινότητας του κάθε ανθρώπου, είτε με την μορφή τηλεφώνου ,είτε κινητού είτε του δικτύου του internet.Με τον γενικό όρο **τηλεπικοινωνίες**, χαρακτηρίζεται η κάθε μορφή ενσύρματης ή ασύρματης ,ακουστικής ή και οπτική επικοινωνίας που πραγματοποιείται ανεξαρτήτως απόστασης ανάμεσα σε δυο ή περισσότερα άτομα. Στην επικοινωνία αυτήν καθορίζονται δύο βασικοί ρόλοι. Ο ρόλος του πομπού, που είναι εκείνος που αποστέλλει το μήνυμα και ξεκινάει την επικοινωνία και εκείνος του δέκτη ο οποίος δέχεται την πληροφορία Η ιστορία των τηλεπικοινωνιών ξεκινούν από την αρχαιότητα ακόμα όπου το 1200 πχ ο Όμηρος αναφέρεται στο έργο του «Ιλιάδα» σε σήματα καπνού και σήματα φωτιάς κατά την διάρκεια του Τρωικού πολέμου. Αργότερα κατά την διάρκεια των Ολυμπιακών αγώνων χρησιμοποιήθηκαν για πρώτη φορά ταχυδρομικά περιστέρια τα οποία χρησιμοποιήθηκαν εκτενώς και για την επικοινωνία των στρατιωτών κατά την διάρκεια πολέμων .Αργότερα το 50 πχ οι Ρωμαίοι θα χρησιμοποιήσουν καθρέφτες για την ανταλλαγή μηνυμάτων μέσο του φωτός ενώ θα ιδρύσουν και τα πρώτα ταχυδρομεία. Αρκετά χρόνια αργότερα, με την βιομηχανική επανάσταση η ανάγκη για ένα γρήγορο και αξιόπιστο μέσο επικοινωνίας είχε γίνει πλέον επιτακτική. Έτσι δεν άργησε να εμφανιστεί ο σπουδαιότερος πρόδρομος του τηλεφώνου, ο τηλέγραφος. Η ιδέα του τηλεγράφου αν και προέρχεται, από τα αρχαία χρόνια υλοποιήθηκε το 1774 από τον Ελβετό George Luis που κατασκεύασε μια πρώιμη μορφή τηλεγράφου, αργότερα εμφανίστηκαν οι τηλέγραφοι του Semmering (1810), του Ampere και των Cooke και Wheaton. Ο Αμερικανός, όμως, Samuel Morse (1791-1872) το 1837 παρουσίασε τον τηλέγραφό του που είχε την δυνατότητα να μεταδίδει μηνύματα σε πολύ μακρινές αποστάσεις γρήγορα και χωρίς μεγάλο κόστος. Το πρώτο μήνυμα από αυτόν τον τηλέγραφο στάλθηκε το 1844 από την Ουάσιγκτον στην Βαλτιμόρη. Το 1847 έρχεται η επανάσταση στον τομέα των τηλεπικοινωνιών με την εφεύρεση του τηλεφώνου από τον Graham Bell. Ο Bell γεννήθηκε στις 3 Μαρτίου του 1847, στο Εδιμβούργο της Σκωτίας και σπούδασε στα πανεπιστήμια του Εδιμβούργου και του Λονδίνου. Μετακινήθηκε στον Καναδά το 1870 και στις Ηνωμένες Πολιτείες το 1871 όπου άρχισε να διδάσκει σε κωφάλαλους το σύστημα ορατής (οπτικής) ομιλίας. Το σύστημα αυτό που αναπτύχθηκε από το πατέρα του, Alexander Melville Bell, δείχνει πώς τα χείλη, η γλώσσα, και ο λαιμός χρησιμοποιούνται στην άρθρωση του λόγου. Το 1872 το Bell ίδρυσε ένα σχολείο για τους κωφαλάλους στη Βοστόνη της Μασαχουσέτης. Το σχολείο έγινε στη συνέχεια μέρος του πανεπιστημίου της Βοστόνης, όπου ο Bell διορίστηκε καθηγητής της φωνητικής φυσιολογίας. Έγινε Αμερικανός πολίτης το 1882.
Από την ηλικία των 18 ο Bell εργαζόταν πάνω στην ιδέα της διαβίβασης της ομιλίας. Το 1874, ενώ δούλευε σε ένα είδος τηλεγράφου, ανέπτυξε τις βασικές ιδέες του για

το τηλέφωνο. Το καλοκαίρι του 1875 ο Bell κάνοντας πειράματα μαζί με το βοηθό του Thomas Watson παρατήρησε ότι ένα έλασμα από ατσάλι όταν δονείτο από διάφορους ήχους επηρέαζε το ρεύμα ενός ηλεκτρομαγνήτη. Παρατήρησε ακόμη ότι σε κάποιο άλλο σημείο ένας άλλος ηλεκτρομαγνήτης επηρεαζόταν από αυτές τις αλλαγές του ρεύματος και με τη σειρά του έκανε ένα άλλο έλασμα να πάλλεται. Τα πειράματά του αυτά αποδείχθηκαν τελικά επιτυχή στις 10 Μαρτίου του 1876, όταν διαβιβάστηκε η πρώτη πλήρης πρόταση μέσω του τηλεφώνου: "Watson, έλα εδώ, σε θέλω ". Οι επόμενες επιδείξεις, ιδιαίτερα μια το 1876 στη Φιλαδέλφεια της Πενσυλβάνια, εισήγαγαν το τηλέφωνο στον κόσμο και οδήγησαν στην οργάνωση της τηλεφωνικής επιχείρησης του Bell το 1877. Άλλοι εφευρέτες προσπάθησαν επίσης να δημιουργήσουν μία συσκευή επικοινωνίας. Ειδικά ο Antonio Meutsi, ο οποίος εφηύρε μια ακουστική συσκευή στις αρχές του 1870 και ο Elisha Gray που υπέβαλε μία αίτηση ευρεσιτεχνίας με τη κατασκευή του τηλεφώνου λίγες ώρες μετά τον Bell στο Σικάγο.



Το 1854 ο Γάλλος εφευρέτης Charles Bourseul σκέφτηκε ότι οι δονήσεις που προκαλούνται από την ομιλία σε έναν εύκαμπτο δίσκο ή ένα διάφραγμα θα μπορούσαν να χρησιμοποιηθούν για να συνδέσουν και να αποσυνδέσουν ένα ηλεκτρικό κύκλωμα, παράγοντας με αυτόν τον τρόπο παρόμοιες δονήσεις σε ένα διάφραγμα που βρίσκεται σε μια άλλη θέση, όπου ο αρχικός ήχος θα αναπαραγόταν. Μερικά έτη αργότερα, ο Γερμανός φυσικός Johann Philip Reis εφηύρε ένα όργανο που διαβίβαζε τους μουσικούς τόνους αλλά που δε μπορούσε να αναπαραγάγει την ομιλία. Μια μορφή ακουστικής συσκευής επικοινωνίας αναπτύχθηκε γύρω στα 1870 από τον ιταλοαμερικάνο Antonio Meucci. Το 1876, ανακαλύπτοντας ότι μόνο ένα σταθερό ηλεκτρικό ρεύμα θα μπορούσε να χρησιμοποιηθεί για να διαβιβάσει την ομιλία, ο αμερικανικός εφευρέτης Αλέξανδρος Graham Bell παρήγαγε το πρώτο τηλέφωνο ικανό και την ανθρώπινη ομιλία με την ποιότητα και τη χροιά της.
Τη βασική μονάδα της εφεύρεσης του Bell αποτελούσαν μια συσκευή αποστολής σημάτων (πομπός), μια συσκευή λήψης σημάτων (δέκτης) και ένα ενιαίο καλώδιο που τις συνέδεε. Ο πομπός και ο δέκτης ήταν ίδιοι˙ κάθε ένας περιείχε ένα εύκαμπτο μεταλλικό διάφραγμα και ένα πεταλοειδή μαγνήτη με μια σπείρα καλωδίων. Τα ηχητικά κύματα χτυπούσαν το διάφραγμα αναγκάζοντάς το να δονηθεί στο πεδίο του μαγνήτη. Αυτή η δόνηση παρήγαγε ηλεκτρικό ρεύμα στη σπείρα των καλωδίων. Το ρεύμα ταξίδευε μέσω των καλωδίων σε έναν άλλο δέκτη, ο οποίος μετέφερε τις αλλαγές αυτές στη δύναμη του μαγνητικού πεδίου με αποτέλεσμα να προκαλεί τη δόνηση του διαφράγματος και έχοντας ως σκοπό την αναπαραγωγή του αρχικού ήχου

Το 1880 η Γαλλία απονέμει στο Bell το βραβείο Volta, το οποίο συνοδευόταν από χρηματικό έπαθλο 50.000 φράγκων, για την εφεύρεσή του. Με αυτά τα χρήματα ίδρυσε το εργαστήριο Volta στην Ουάσιγκτον, όπου τον ίδιο χρόνο οι συνεργάτες

του και εκείνος εφηύραν το φωτόφωνο (photophone), το οποίο διαβίβαζε την ομιλία μέσω ακτινών φωτός.
Η είσοδος της μικροηλεκτρονικής άλλαξε το πρόσωπο των τηλεπικοινωνιών και μετά τη δεκαετία του 1950, όπου έχουμε την παράλληλη ανάπτυξη των υπολογιστών και της ηλεκτρονικής επεξεργασίας των πληροφοριών, άρχισαν να καταλαμβάνουν σημαντική θέση στην καθημερινή ζωή του ανθρώπου μέχρι το σημείο να επηρεάζουν άμεσα τον πολιτισμό σε παγκόσμιο επίπεδο.
Η ψηφιακή μετάδοση των σημάτων και η ψηφιακή μεταγωγή και επεξεργασία είναι κάποιες από τις νέες τεχνολογίες που εισήλθαν στις τηλεπικοινωνίες και με αυτό τον τρόπο οι αναλογικές τεχνολογίες στις οποίες στηρίχθηκαν η κλασσική τηλεφωνία, το ραδιόφωνο και η τηλεόραση αντικαθίστανται σταδιακά από την ψηφιακή τεχνολογία που αναπτύχθηκε κυρίως από τις επικοινωνίες data.
Παράλληλα ήρθε η βελτίωση των μέσων μετάδοσης, της υποδομής (οπτικές ίνες, δορυφορικές ζεύξεις, κλπ.), των τεχνικών μετάδοσης (multiplexing, compression, κωδικοποιήσεις, διαμορφώσεις κλπ.)

1.2 Στόχος Εργασίας

Βασικός στόχος της παρούσας εργασίας είναι αφενός η μελέτη λειτουργίας ενός δικτύου επικοινωνιών επόμενης γενιάς και αφετέρου με βάση αυτήν την μελέτη η πραγματοποίηση ενός συστήματος τηλεπικοινωνιών εφαρμοσμένο σε ένα τοπικό δίκτυο για την μεταφορά φωνής σε πραγματικό χρόνο εκμεταλλευόμενοι τα πρωτόκολλα Next Generation Network αλλά και SIP ,RTP .

1.3 Δομή Εργασίας

Η εργασία περιλαμβάνει έξι κεφάλαια. Στο πρώτο κεφάλαιο έγινε μια εισαγωγή στο θέμα της εργασίας και μια ιστορική αναδρομή στην εξέλιξη των τηλεπικοινωνιών. Ακολουθεί το δεύτερο κεφάλαιο όπου γίνεται μελέτη και ανασκόπηση των βασικών πρωτοκόλλων τα οποία αποτέλεσαν τη βάση για την υλοποίηση της εργασίας .Στο τρίτο κεφάλαιο έχουμε την παρουσίαση της υλοποίησης του συστήματος ανταλλαγής φωνής όπου γίνονται αναφορές στην μεθοδολογία που ακολουθήθηκε , στις βασικές οντότητες που λαμβάνουν χώρο και επιτελούν τις λειτουργίες του συστήματος ,τις συσχετίσεις που υπάρχουν μεταξύ τους καθώς και θα δοθεί ένα παράδειγμα σεναρίου εκτέλεσης. Στο τέταρτο κεφάλαιο προχωράμε στην περιγραφή των βασικών μεθόδων και κλάσεων του προγράμματος καθώς και σε ένα παράδειγμα υλοποίησης λειτουργία με τις αντίστοιχες εικόνες του. Στην συνέχεια, πάμε στο πέμπτο κεφάλαιο. Εδώ θα γίνει μια συνολική ανασκόπηση της παρούσας εργασίας και θα παρουσιαστούν κρίσιμα συμπεράσματα τόσο για την επιτυχία της όσο και για την δυνατότητα εξέλιξής της.Τέλος, στο έκτο κεφάλαιο έχουμε τη βιβλιογραφία της εργασίας.

2. Ανασκόπηση Βιβλιογραφίας

Σε αυτό το κεφάλαιο κρίθηκε απαραίτητο να γίνει μια περιγραφή των βασικότερων εννοιών που αφορούν την παρούσα εργασία. Πιο συγκεκριμένα θα αναφερθούμε στους ορισμούς και τα χαρακτηριστικά ενός Next Generation Network (NGN) και των πρωτοκόλλων MEGACO ,RTP και SIP

2.1 Τι είναι ένα Next Generation Network (NGN);

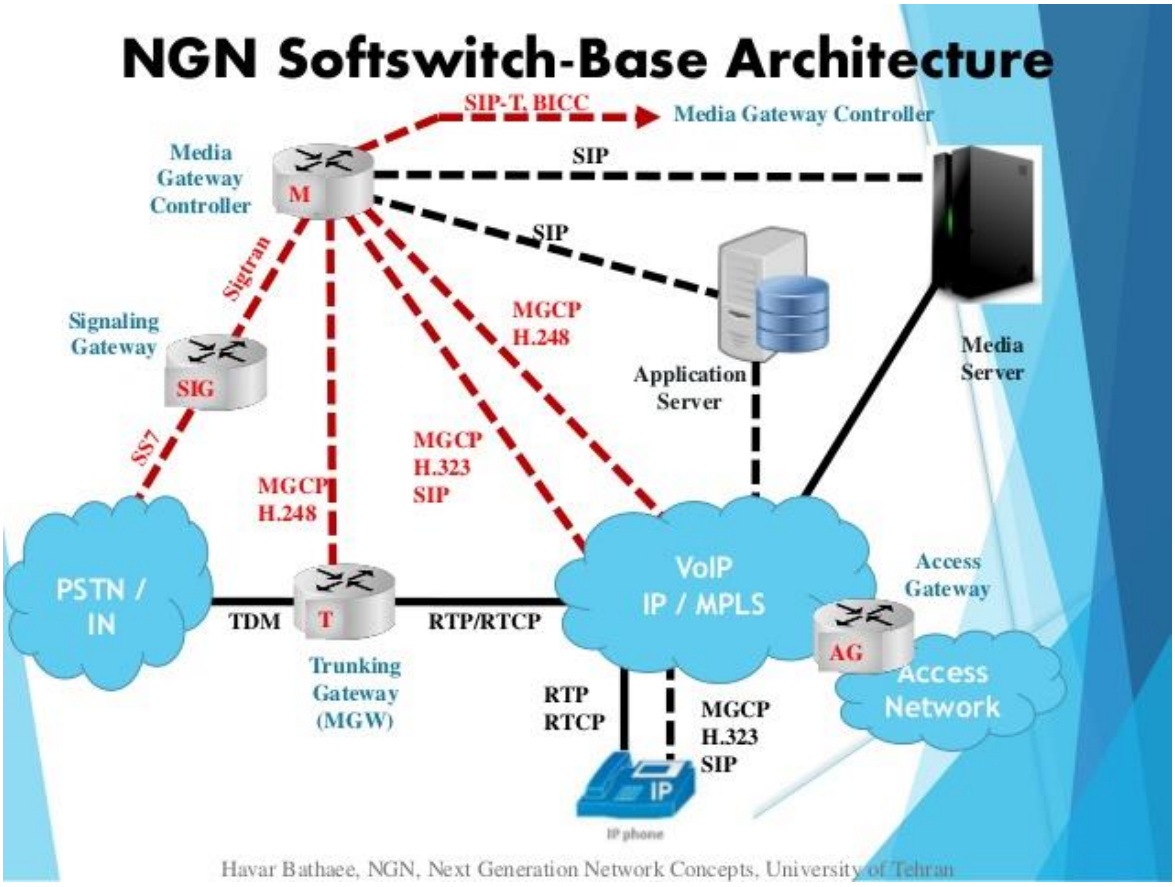
Ένα δίκτυο επόμενης γενιάς (NGN) είναι η απαρχή των βασικών αρχιτεκτονικών αλλαγών στις τηλεπικοινωνίες πυρήνα και πρόσβασης στα δίκτυα.

Η υλοποίηση ενός NGN ξεκίνησε κυρίως από την ανάγκη της μείωσης του κόστους και τη διαφοροποίηση του προϊόντος από τα δίκτυα των παροχών υπηρεσιών τηλεπικοινωνιών .Οι πελάτες πλέον δεν αρκούνται σε μια απλή τηλεφωνική συνδιάσκεψη που παρείχαν μέχρι τώρα οι PSTN υπηρεσίες και στρέφονται προς τις δια δραστικές και ποικίλες υπηρεσίες που τους προσφέρει το διαδίκτυο.

Η γενική ιδέα πίσω από το NGN είναι η δημιουργία ενός δικτύου που μεταφέρει όλες τις πληροφορίες και τις υπηρεσίες (π.χ. φωνή, δεδομένα βίντεο) χρησιμοποιώντας την μέθοδο του εγκλεισμού αυτών σε πακέτα IP, παρόμοια με αυτά που χρησιμοποιούνται στο Διαδίκτυο.
NGN δεν θα πρέπει να ταυτίζεται με την ιδέα του διαδικτύου του μέλλοντος, η οποία επικεντρώνεται περισσότερο στην εξέλιξη του Διαδικτύου από άποψη ποικιλίας και αλληλεπιδράσεων των υπηρεσιών που προσφέρουν.

Ένας αρκετά καλός ορισμός δίνεται από το ITU:
"A Next Generation Network (NGN) is a packet-based network able to provide services including Telecommunication Services and able to make use of multiple broadband, QoS-enabled transport technologies and in which service-related functions are independent from underlying transport-related technologies. It offers

unrestricted access by users to different service providers. It supports generalized mobility which will allow consistent and ubiquitous provision of services to users. ”



Αυτό που μας μένει από τον συγκεκριμένο ορισμό είναι η υποστήριξη του λεγόμενου ***generalized mobility*** που πρακτικά σημαίνει ότι ένα NGN μπορεί να προσφέρει απεριόριστη χωρίς χρονικούς και χωρικούς περιορισμούς πρόσβαση από τους χρήστες του για την παροχή των υπηρεσιών που διαθέτει

Οι βασικές δυνατότητες που υποστηρίζει ένα NGN είναι:

- Packet-based μεταφορά δεδομένων
- διαχωρισμό των λειτουργιών ελέγχου μεταξύ κλήσης/συνεδρίας και εφαρμογής/ υπηρεσίας

- δυνατότητα αποσύνδεσης των υπηρεσιών από το δίκτυο, καθώς και παροχής ανοιχτών διεπαφών
- υποστήριξη για μια ευρεία γκάμα υπηρεσιών, εφαρμογών και μηχανισμών όπως των real time streaming,non real time υπηρεσιών και multimedia)
- διασύνδεση με παλαιότερα δίκτυα μέσω ανοικτών διεπαφών
- Παροχή υπηρεσιών οπουδήποτε και οποτεδήποτε
- απεριόριστη πρόσβαση από τους χρήστες για την παροχή υπηρεσιών
- συγκλίνουσες υπηρεσίες μεταξύ σταθερού/κινητού
- συμβατό με όλες τις απαιτήσεις, όπως για παράδειγμα σχετικά με τις επικοινωνίες έκτακτης ανάγκης ,ασφάλειας και ιδιωτικότητας

2.2 Γιατί Next Generation Network ;

Κατά την τελευταία δεκαετία η έκρηξη της χρήσης του διαδικτύου για την επικοινωνία και την μεταφορά δεδομένων είναι πρωτόγνωρη. Η μετάβαση από την απλή φωνητική επικοινωνία σε εκείνη με το πλούσιο περιεχόμενο αλληλεπιδράσεων (βίντεο και εικόνας) μέσω του Διαδικτύου αυξάνεται ολοένα και περισσότερο με αποτέλεσμα η απλές υπηρεσίες φωνής που παρείχαν οι PSTN να χάνουν έδαφος.
Οι πελάτες προτιμούν το πλούσιο περιεχόμενο και δια δραστικές δυνατότητες επικοινωνίας που προσφέρει το Διαδίκτυο
Οι πάροχοι υπηρεσιών γνωρίζουν αυτές τις αλλαγές της αγοράς και
Έτσι αναγκάστηκαν να στραφούν και εκείνοι σε συνεχή ανάπτυξη των ευρυζωνικών υπηρεσιών τους στο πλαίσιο του υπάρχοντος
υποδομή του δικτύου.
Αυτό βέβαια είναι ιδιαίτερα ζημιογόνο για τους παρόχους όχι μόνο ως προς την προσαρμογή στο ήδη υπάρχον δίκτυο αλλά και ως προς την συντήρηση του με
Από την άλλη πλευρά ωστόσο, από τη σκοπιά των χρηστών , οι υπηρεσίες έχουν γίνει ιδιαίτερα ποιοτικότερες και ασφαλέστερες συνδέοντας τους χρήστες μεταξύ τους χωρίς κανένα γεωγραφικό ή χρονικό περιορισμό.
Επίσης η υπάρχουσα αρχιτεκτονική δικτύου(PSTN) έχει κάποια σημαντικά μειονεκτήματα.
Ένα από τα βασικότερα είναι ότι είναι υπερβολικά περίπλοκη και αντιοικονομική η διαχείριση μιας πιθανής έκρηξης της κίνησης
σε μεγάλη κλίμακα και ως εκ τούτου δεν μπορεί να ανταποκριθεί στην διαρκώς αυξανόμενη απαιτήσεις της αγοράς. Για αυτό τον λόγο κρίθηκε απαραίτητο και επινοήθηκε το νέο πρότυπο NGN προκειμένου να γίνει αποτελεσματική διαχείριση μια πιθανής αύξησης της κυκλοφορίας δεδομένων με ένα οικονομικά αποδοτικό τρόπο σκοπός.

Πιο αναλυτικά τα οφέλη της χρήσης ενός τέτοιου συστήματος σε σχέση με το ήδη υπάρχον είναι τα εξής:
Πρώτον, η ανάπτυξη NGN **μπορεί να μειώσει τις επενδύσεις σε ενημερώσεις εύρους ζώνης του δικτύου**
χρησιμοποιώντας την αυτόματη ανταλλαγή εύρους ζώνης και την τεχνολογία κατανομής σε πυρήνα των
δικτύων για να αξιοποιηθεί πλήρως το δυναμικό του ήδη υπάρχοντος εύρους ζώνης του δικτύου.
Επίσης**, μπορεί να μειώσει βασικά έξοδα** 'όπως της εύρεσης χώρου, της κατανάλωση ρεύματος και το κόστος συντήρησης, χρησιμοποιώντας λιγότερο Εξοπλισμός Δικτύων τόσο σε ποσότητα όσο και σε ποικιλία
σε μια κοινή υποδομή δικτύου.

Το κόστος αποκατάστασης βλαβών του δικτύου μπορεί να μειωθεί, καθώς με τη χρήση της ταχείας επαναφοράς ένα NGN μπορεί να επιστρέψει σε κάποια παλαιότερη έκδοση στην οποία λειτουργούσε σωστά

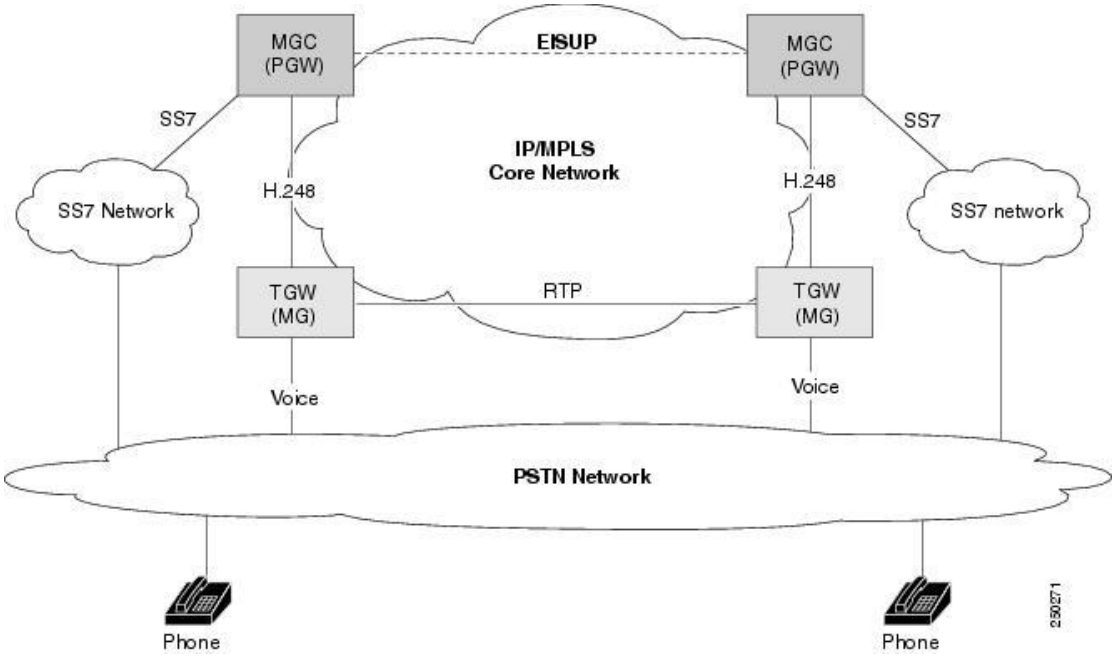
Περισσότερα έσοδα θα παράγονται από την ελαχιστοποίηση του χρόνου παροχής υπηρεσιών χρησιμοποιώντας λιγότερες μεταβιβάσεις και περισσότερες κερδοφόρες υπηρεσίες προστιθέμενης αξίας.
Σε σύγκριση με το υπάρχον PSTN / ISDN υπηρεσίες CPE-based , ένα NGN μπορεί να παρέχει πιο αποτελεσματική και κλιμακούμενη παροχή υπηρεσιών,
χαμηλότερο κόστος και λύσεις που βασίζονται σε ένα δίκτυο που επιτρέπει την κοινή χρήση των πόρων για να ανταποκρίνονται στις αυξανόμενες απαιτήσεις σχετικά με την ποιότητα και τις δυνατότητες των υπηρεσιών.

Η επιτυχής εφαρμογή ενός NGN μπορεί **να διατηρήσει τους υπάρχοντες πελάτες, καθώς και για να προσελκύσει νέου**ς, χρησιμοποιώντας τις καινοτόμες υπηρεσίες που προσφέρονται
. Για παράδειγμα, είναι ιδιαίτερα παρελκυστικό για κάποιον πελάτη η δυνατότητα που του δίνει ένα δίκτυο NGN για εκτέλεση συναλλαγών μέσω του διαδικτύου. Ως ένα από τα πιο σημαντικά χαρακτηριστικά, η σύγκλιση των κινητών και σταθερών δικτύων σε ένα NGN περιβάλλον παρέχει μεγάλη ευελιξία και ευκολία για τους πελάτες, σε αντίθεση με τις παλαιότερες προσεγγίσεις.

Τα οφέλη λοιπόν όπως βλέπουμε είναι πολύπλευρα, τόσο για τους παρόχους οι οποίοι θα πρέπει να ακολουθήσουν τις τάσεις της εποχής και τα θέλω των πελατών ώστε να αυξήσουν τον τζίρο τους όσο και για τους πελάτες οι οποίοι μέσω των δικτύων επόμενης γενιάς μπορούν με οικονομικά, γρήγορα, εύκολα και με ασφάλεια να επωφεληθούν από τις ποικίλες υπηρεσίες και δυνατότητες που τους προσφέρουν καθώς και να τις προσαρμόσουν στις ανάγκες τους.

2.3 Πρωτόκολλο MEGACO/H.248

Σε ένα παραδοσιακό τηλεφωνικό δίκτυο, η υποδομή αποτελείται από διακόπτες που διασυνδέονται μεταξύ τους για να δημιουργήσουν το δίκτυο κορμού και η οποία θα συνδεθεί με τον εξοπλισμό των πελατών (τηλεφωνικά κέντρα, τηλέφωνα). Ενώ το εσωτερικό δίκτυο σήμερα βασίζεται στη ψηφιακή επικοινωνία, η σύνδεση με τους πελάτες μπορεί να είναι είτε αναλογική (PSTN) ή ψηφιακή (ISDN).
Μια παρόμοια κατασκευή θεωρείται σήμερα από μια σειρά από τηλεφωνικές εταιρίες για τα δίκτυα κορμού IP-based που μπορεί διαδοχικά να αντικαταστήσει τμήματα της συνολικής τους υποδομής μεταγωγής του δικτύου. Αντί των διακοπτών, χρησιμοποιούνται οι δρομολογητές IP για να δημιουργήσουν ένα δίκτυο κορμού - η οποία υποστηρίζει δρομολόγηση IP, ενδεχομένως MPLS, και, πιθανότατα, κάποια ρητή μορφή QoS υποστήριξης για τη μεταφορά πακέτων φωνής και δεδομένων από οποιοδήποτε σημείο του δικτύου σε οποιοδήποτε άλλο .



Σε αντίθεση με τη προηγούμενη μορφή με διακόπτες, αυτό δεν απαιτεί ρητή διαμόρφωση των επιμέρους δρομολογητών ανά σύνδεση ομιλίας παρά μόνο πρέπει να ρυθμιστούν τα σημεία εισόδου και εξόδου με τις διευθύνσεις του άλλου, ώστε να γνωρίζουν πού θα σταλούν τα πακέτα φωνής / δεδομένων . Οι δύο τύποι πυλών που χρησιμοποιούνται στα άκρα του δικτύου IP για να συνδεθεί με το συμβατικό τηλεφωνικό δίκτυο είναι: οι πύλες σηματοδότηση για να μετατρέψει την SS7 σηματοδότηση σε πακέτα IP [SIGTRAN]) και πύλες μέσων που εκτελούν σηματοδότηση φωνής.
Η κεντρική οντότητα του πρωτοκόλλου αυτού είναι ο ελεγκτής πυλών μέσων (Media Gateway Controller) .Εκείνοι είναι που ερμηνεύουν την σηματοδότηση και αποφασίζουν πώς θα δρομολογηθούν οι κλήσεις, παρέχουν συμπληρωματικές υπηρεσίες, κλπ. Έχοντας αποφασίσει σχετικά με το πώς μια κλήση θα εγκατασταθεί, θα ενημερώσουν τις πύλες μέσων στις άκρες (πύλες εισόδου και εξόδου) πώς και πού να μεταδώσει τα πακέτα φωνής. Οι Media Gateway Controllers είναι επίσης υπεύθυνοι να αναμορφώσουν τις πύλες σε περίπτωση τυχόν αλλαγών στην πρόσκληση, την επίκληση των συμπληρωματικών υπηρεσιών, κ.λπ. Οι πύλες των μέσων μπορεί να είναι ικανές να ανιχνεύουν την επίκληση χαρακτηριστικών ελέγχου στο κανάλι (π.χ. μέσω τόνων DTMF) και να την κοινοποιήσουν στους Media Gateway Controllers που στη συνέχεια θα κάνουν τις κατάλληλες ενέργειες.

Με άλλα λόγια οι Media Gateway Controllers είναι εκείνοι που αποφασίζουν για τις ενέργειες τις οποίες θα πράξουν τα Media Gateways,θα εγκαταστήσουν την κλήση ,θα φροντίσουν για την σωστή επικοινωνία μεταξύ τους και θα τα ενημερώσουν για οποιεσδήποτε αλλαγές υπάρξουν.

Μια ιδιαίτερη επέκταση του πρωτοκόλλου που συζητούνται σήμερα στην IETF είναι ο ορισμός ενός πρωτοκόλλου για την επικοινωνία με ένα τηλέφωνο IP στις εγκαταστάσεις του πελάτη που ταιριάζει απόλυτα με την αρχιτεκτονική Media Gateway Ελέγχου. Μια τέτοια τηλεφωνική συσκευή θα είναι μια μάλλον απλή οντότητα ουσιαστικά ικανό να εκπέμπει και να λαμβάνει τα γεγονότα και να

αντιδρούν σε αυτές, ενώ οι υπηρεσίες κλήσης που παρέχονται άμεσα από την υποδομή του δικτύου.

Είδαμε μερικά βασικά χαρακτηριστικά του πρωτοκόλλου MEGACO το οποίο δεν θα μας απασχολήσει στην παρούσα εργασία ωστόσο κρίθηκε σκόπιμο να αναφερθούν καθώς είναι ένα πρωτόκολλο που θα μπορούσε να έχει άμεση εφαρμογή στο σύστημα μας σε περίπτωση επέκτασης του. Παρακάτω θα αναφερθούμε στο πρωτόκολλο SIP,ίσως το βασικότερο του συστήματος μας.

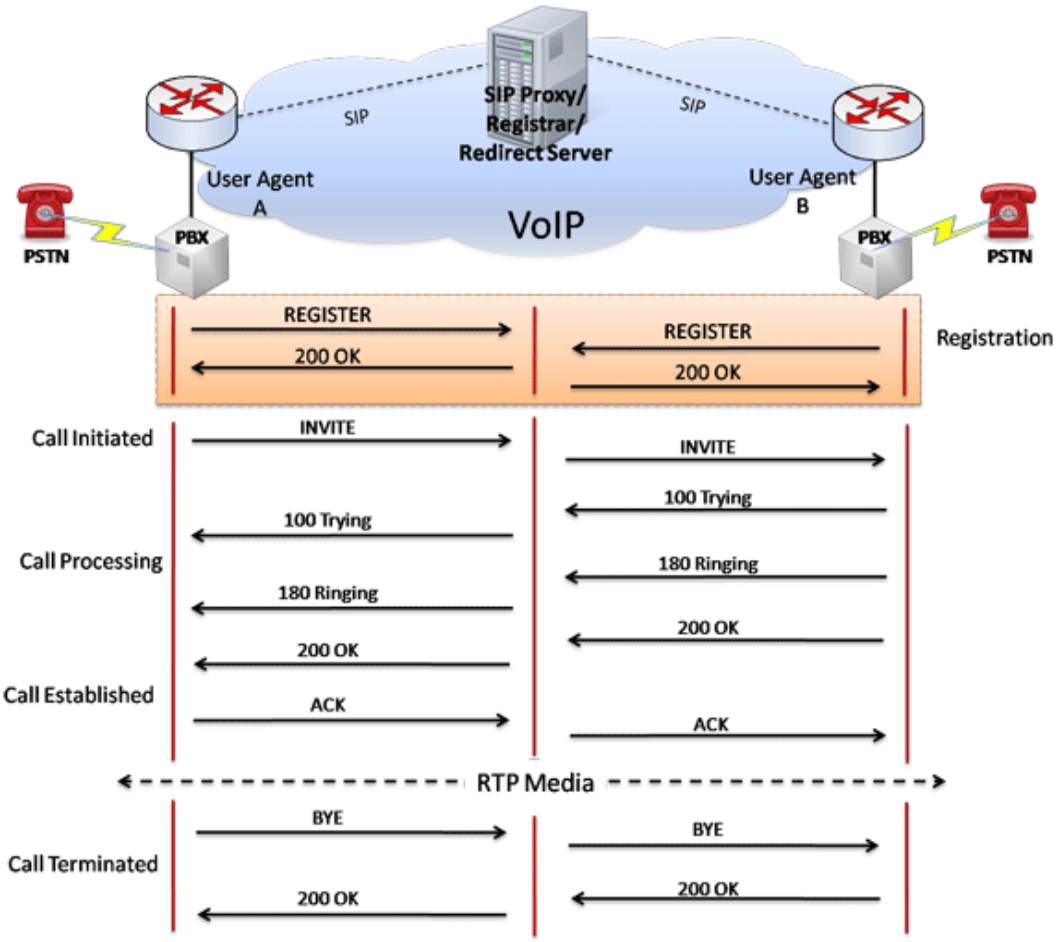
2.4 Πρωτόκολλο SIP

Το SIP είναι ένα πρωτόκολλο σηματοδοσίας που έχει σχεδιαστεί για να δημιουργεί, να τροποποιεί και να τερματίσει μια συνεδρία πολυμέσων μέσω του πρωτοκόλλου Internet. Είναι μια εφαρμογή με πρωτόκολλο στρώματος που ενσωματώνει πολλά στοιχεία του HTTP (Hypertext Transfer Protocol) και SMTP (Simple Mail Transfer Protocol).
Μια συνεδρία δεν είναι τίποτε άλλο παρά μια απλή κλήση μεταξύ δύο τερματικά σημεία. Το τελικό αποτέλεσμα μπορεί να είναι ένα έξυπνο τηλέφωνο, το φορητό υπολογιστή ή άλλη συσκευή που μπορεί να λαμβάνετε και να αποστέλλετε περιεχόμενο πολυμέσων από το Internet.

Εμπεριέχει μια αρχιτεκτονική client-server και κάνει χρήση του URL και URI από το HTTP και ενός κωδικοποιητή κειμένου από το SMTP.
Το SIP λαμβάνει τη βοήθεια του πρωτοκόλλου SDP το οποίο περιγράφει μια συνεδρία και του RTP (Πρωτόκολλο Μεταφοράς πραγματικού χρόνου) που χρησιμοποιείται για την μεταφορά φωνής και βίντεο μέσω δικτύου IP.
Συνήθως, το SIP πρωτόκολλο χρησιμοποιείται για υπηρεσίες τηλεφωνίας και πολυμέσων διανομή μεταξύ δύο ή περισσότερα τερματικών. Για παράδειγμα, ένα άτομο μπορεί να κινήσει μια τηλεφωνική κλήση προς κάποιον άλλο χρησιμοποιώντας SIP, ή κάποιος μπορεί να δημιουργήσει μια κλήση συνδιάσκεψης με πολλούς συμμετέχοντες.

Το SIP πρωτόκολλο σχεδιάστηκε να είναι πολύ απλό, με περιορισμένο σετ εντολών. Είναι επίσης βασισμένο σε κείμενο οπότε μπορεί ο οποιοσδήποτε να καταλάβει κάποιο μήνυμα που μεταφέρεται από το ένα τερματικό στο άλλο.

Για την σωστή δημιουργία του δικτύου το SIP χρησιμοποιεί ένα σύνολο οντοτήτων. Το SIP, κάθε στοιχείο δικτύου αναγνωρίζεται από ένα SIP URI (Uniform Resource Identifier), η οποία είναι σαν μια διεύθυνση.



Τα στοιχεία ενός SIP είναι τα παρακάτω:

User Agent
Είναι το τελικό σημείο και ένα από τα πιο σημαντικά στοιχεία του δικτύου του SIP δικτύου. Ένα τερματικό μπορεί να εκκινήσει, να τροποποιήσει ή να τερματίσει μια συνεδρία. Θα μπορούσε να είναι ένα softphone, κινητό ή φορητό. Τα τερματικά είναι δύο ειδών:

User Agent Client (UAC) - Το πρόσωπο που στέλνει μια αίτηση και λαμβάνει απάντηση.

User Agent διακομιστή (UAS) - Η οντότητα που λαμβάνει ένα αίτημα και στέλνει μια απάντηση.

Το SIP βασίζεται στην αρχιτεκτονική client-server όπου ο καλών αριθμός τηλεφώνου του δρα ως client ο οποίος εκκινεί μια κλήση και ο καλούμενος αριθμός τηλεφώνου του δρα ως διακομιστής που απαντά στην κλήση.

Proxy Server

είναι η οντότητα που λαμβάνει μια αίτηση από το χρήστη και την προωθεί σε άλλον χρήστη. Ο Διακομιστής μεσολάβησης βρίσκεται ανάμεσα σε δύο πράκτορες χρήστη. Μπορούν να υπάρχουν έως 70 διακομιστές μεσολάβησης μεταξύ ενός πομπού και ενός δέκτη.

Υπάρχουν δύο τύποι διακομιστών μεσολάβησης :

Stateless Proxy Server - απλώς προωθεί το μήνυμα που ελήφθη. Αυτό το είδος του διακομιστή δεν αποθηκεύει καμία από τις πληροφορίες της κλήσης
Statefull Proxy Server - Αυτός ο τύπος διακομιστή μεσολάβησης παρακολουθεί κάθε αίτημα και ανταπόκριση και μπορεί να χρησιμοποιηθεί στο μέλλον, εάν απαιτείται. Επίσης μπορεί να αναμεταδώσει το αίτημα του καλούντος, εάν δεν υπάρξει απάντηση από την άλλη πλευρά έγκαιρα.

Registrar Server

ο registrar Server δέχεται αιτήσεις εγγραφής από τα τερματικά και βοηθά τους χρήστες να πιστοποιήσουν την λειτουργία τους εντός του δικτύου. Επίσης αποθηκεύει το URI και τη θέση των χρηστών σε μια βάση δεδομένων για να βοηθήσει άλλους διακομιστές SIP εντός του ίδιου τομέα.

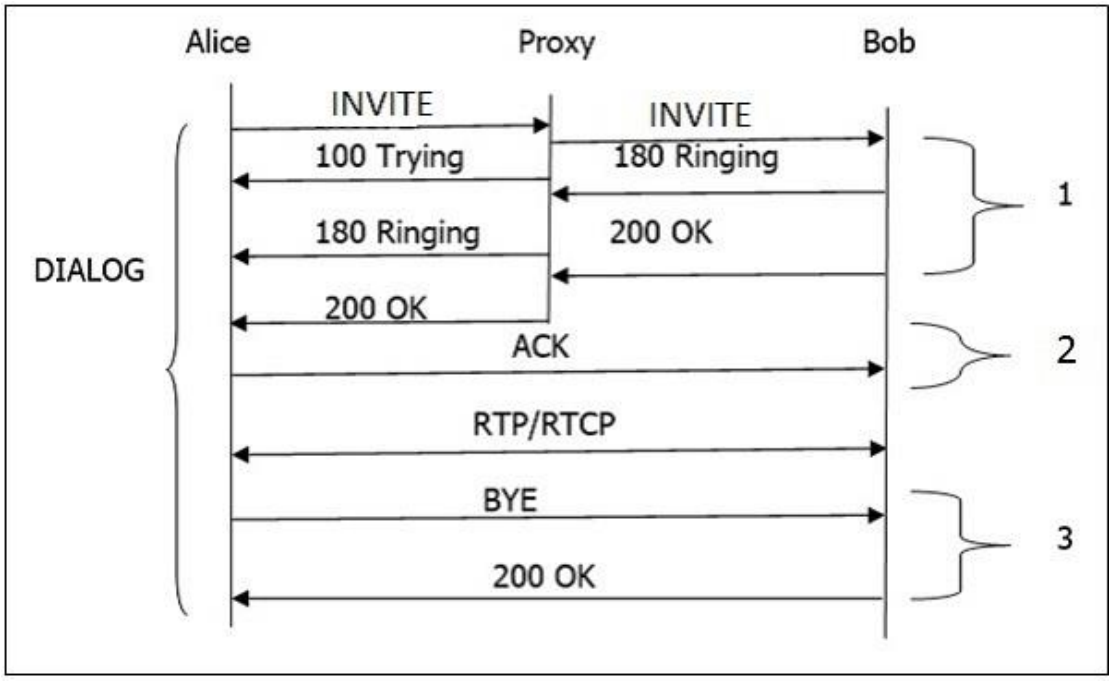
Redirect Server

Ο server επαναπροώθησης λαμβάνει αιτήσεις και αναζητά τον αποδέκτη του αιτήματος στην βάση δεδομένων που δημιουργήθηκε από τον registrar.

Ο server επαναπροώθησης χρησιμοποιεί τη βάση δεδομένων για τη λήψη πληροφοριών θέσης και ανταποκρίνεται με μηνυμα 3xx (απάντηση Ανακατεύθυνσης) για το χρήστη.

Location Server

Ο location server παρέχει πληροφορίες σχετικά με στοιχεία του καλούντος καθώς και πιθανές στους server επαναπροώθησης και μεσολάβησης.
Μόνο ένας Proxy Server ή έναν διακομιστής ανακατεύθυνσης μπορεί να επικοινωνήσει με τον location server



Στην παραπάνω εικόνα βλέπουμε μια βασική χρήση του πρωτοκόλλου SIP για μια κλήση.
Ας το δούμε βήμα -βήμα:

1. Η Alice στέλνει invitation για κλήση σε έναν proxy server
2. proxy server στέλνει μήνυμα trying στην Alice ώστε να αποφύγει την επαναποστολή του μνήματος ενώ ταυτόχρονα έχει βρει τον Bob που πρέπει να δεχθεί την κλήση και προωθεί το INVITE.
3. Στην συνέχεια ο Bob παράγει ήχο Ringing που φτάνει στην Alice
4. Όταν ο Bob σηκώσει το τηλέφωνο στέλνεται μια απάντηση 200 OK
5. Την στιγμή που θα παίρνει και εκείνος ένα 200 OK η Alice στέλνει αίτημα ACK και ξεκινάει η ανταλλαγή πακέτων μέσω RTP.
6. Μετά τη συζήτηση, κάθε συμμετέχων η Alice στέλνει ένα αίτημα BYE για να τερματίσει η κλήση
7. Το BYE φτάνει απευθείας στον Bob.
8. Τέλος, ο Bob στέλνει ένα 200 OK απάντηση για την επιβεβαίωση του BYE και τερματίζεται η συνεδρία.

Στο παράδειγμα παραπάνω είδαμε ότι ανάμεσα στα δυο τερματικά (Bob και Alice) είχαμε κάποιες αιτήσεις (requests) και κάποιες απαντήσεις (responses)
Πιο συγκεκριμένα είδαμε τα requests INVITE,ACK και BYE και τα responses 200 OK ,100 TRYING και 180 RINGING.Στο πρωτόκολλο SIP μπορούν να χρησιμοποιηθούν ποικίλες μέθοδοι αιτήσεων και απαντήσεων.
Οι βασικότερες μέθοδοι αιτήσεων είναι οι εξής:

INVITE
Χρησιμοποιείται για να ξεκινήσει μια συνεδρία με έναν χρήστη. Με άλλα λόγια, μια μέθοδος INVITE χρησιμοποιείται για να δημιουργήσει μια συνεδρία ανάμεσα σε δυο χρήστες ενώ μπορεί να περιέχει και τις πληροφορίες του καλούντος στο εσωτερικό της
Μια συνεδρία θεωρείται εγκατεστημένη εάν μια INVITE έχει λάβει απάντηση επιτυχίας (2xx) ή αν ένα ACK έχει αποσταλεί.

BYE
Είναι η μέθοδος που χρησιμοποιείται για να τερματίσει μια καθιερωμένη συνεδρία και μπορεί να σταλεί μόνο από έναν από τους δυο χρήστες και όχι από τον server μεσολάβησης.

REGISTER
Το αίτημα REGISTER εκτελεί την καταχώριση ενός χρήστη. Αυτή η αίτηση αποστέλλεται από έναν χρήστη σε ένα διακομιστή καταχώρισης.
Το αίτημα REGISTER μπορεί να διαβιβάζεται ή να προσεγγίζεται μέχρι να φτάσει σε έναν έγκυρο διακομιστή καταχώρισης του συγκεκριμένου τομέα.
Ένας χρήστης μπορεί να στείλει ένα αίτημα REGISTER για λογαριασμό άλλου χρήστη. Αυτό είναι γνωστό ως καταχώριση τρίτων.

CANCEL
Η μέθοδος CANCEL χρησιμοποιείται για να τερματίσει μια συνεδρία που δεν είναι εγκατεστημένη. Οι χρήστες μπορούν να χρησιμοποιήσουν αυτό το αίτημα για να ακυρώσουν μια προσπάθεια κλήσης που είχαν ξεκινήσει νωρίτερα.
Μπορεί να αποσταλούν είτε με έναν χρήστη ή μέσω ενός διακομιστή μεσολάβησης.

ACK
Η μέθοδος ACK χρησιμοποιείται για να αναγνωρίσει τις τελικές απαντήσεις σε μια μέθοδο INVITE. Μια ACK πηγαίνει πάντα προς την κατεύθυνση της INVITE.
ACK δεν μπορεί να χρησιμοποιηθεί για να τροποποιήσει την περιγραφή των μέσων που έχει ήδη αποσταλεί στην αρχική INVITE.
Για απαντήσεις με κωδικό 2xx, η ACK λειτουργεί ως end to end, αλλά και για όλες τις άλλες τελικές απαντήσεις, λειτουργεί ως hop by hop όταν εμπλέκονται stateful proxies.

Αντίστοιχα οι τύποι απαντήσεων ανάλογα με τον κωδικό τους είναι:

1xx: Προσωρινές / Ενημερωτικές απαντήσεις
Οι ενημερωτικές απαντήσεις χρησιμοποιούνται για να δείξουν την πρόοδο κλήση.
Για παράδειγμα 100 TRYING,180 RINGING,182 QUEUED

2xx: Απαντήσεις επιτυχίας
Αυτή η κατηγορία των απαντήσεων προορίζεται για την ένδειξη ότι η αίτηση έχει γίνει δεκτή. Για παράδειγμα 200 OK

3xx: Απαντήσεις ανακατεύθυνσης
Αποστέλλονται από servers ανακατεύθυνσης ως απάντηση στο INVITE.
Π.χ. 300 Multiple Choices,301 Moved Permanently

4xx: Απαντήσεις αποτυχίας χρήστη
Δείχνουν ότι το αίτημα δεν μπορεί να ικανοποιηθεί λόγω κάποιου σφάλματος στην πλευρά του αποδέκτη.
Επί παραδείγματι 400 Bad Request,401 Unauthorized,404 Not found

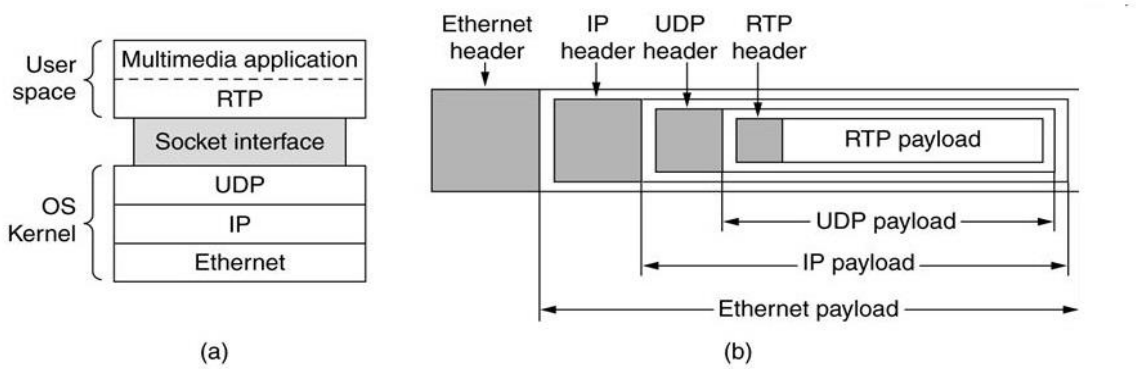
5xx: απόκριση αποτυχίας διακομιστή
Αυτή η απάντηση χρησιμοποιείται για να υποδείξει ότι η αίτηση δεν μπορεί να επεξεργαστεί εξαιτίας ενός σφάλματος στον διακομιστή.
Για παράδειγμα 500 Server Internal Error,501 Not Implemented

6xx: απάντηση αποτυχίας
Αυτή η απάντηση υποδεικνύει ότι ο διακομιστής γνωρίζει ότι η αίτηση θα αποτύχει οπουδήποτε και αν προσπαθήσει να πάει. Έτσι δεν θα πρέπει να αποστέλλεται σε άλλες τοποθεσίες.

Για παράδειγμα 600 Busy Everywhere,606 Not Acceptable

2.5 Πρωτόκολλο RTP

Το Real time transfer protocol (RTP) είναι ένα τυπικό πρωτόκολλο Internet που καθορίζει ένα τρόπο ώστε τα προγράμματα να διαχειρίζονται την μετάδοση δεδομένων πολυμέσων σε πραγματικό χρόνο πάνω είτε σε unicast ή σε multicast υπηρεσίες δικτύου. Το RTP σχεδιάστηκε από την Ομάδα Εργασίας Μεταφορών Ήχου και Εικόνας του IETF ώστε να στηρίξει βιντεοδιασκέψεις με συμμετέχοντες σε γεωγραφικά διάσπαρτες περιοχές. Το RTP χρησιμοποιείται συνήθως σε εφαρμογές τηλεφωνίας μέσω Διαδικτύου. δεν εγγυάται από μόνο του την παράδοση των δεδομένων πολυμέσων σε πραγματικό χρόνο (δεδομένου ότι αυτό εξαρτάται από τα χαρακτηριστικά του δικτύου) ωστόσο, παρέχει τα μέσα για τη διαχείριση των δεδομένων, ώστε να επιτευχθεί το καλύτερο δυνατό αποτέλεσμα.



(a) The position of RTP in the protocol stack. (b) Packet nesting.

Το RTP συνδυάζει τη μεταφορά δεδομένων με το πρωτόκολλο ελέγχου (RTCP), το οποίο καθιστά δυνατή την παρακολούθηση της υποβολής στοιχείων για μεγάλα multicast δίκτυα. Η παρακολούθηση επιτρέπει στο δέκτη να ανιχνεύσει αν υπάρχει οποιαδήποτε απώλεια πακέτων και για να αντισταθμίσει οποιαδήποτε

καθυστέρηση. Οι πληροφορίες που βρίσκονται στην κεφαλίδα του RTP λέει πώς να ανακατασκευάσει ο δέκτης τα δεδομένα και περιγράφει πώς τα κωδικοποιημένα bit πακετάρονται. Κατά κανόνα, το RTP τρέχει πάνω από το User Datagram Protocol (UDP), αν και μπορούν να χρησιμοποιηθούν και άλλα πρωτόκολλα μεταφοράς.

Τα βασικά συστατικά ενός RTP είναι:

- έναν αριθμό ακολουθίας(*sequence number)*, που χρησιμοποιείται για την ανίχνευση χαμένων πακέτων
- αναγνώριση ωφέλιμου φορτίου(*payload identification)*, το οποίο περιγράφει τη συγκεκριμένη κωδικοποίηση των μέσων, έτσι ώστε να μπορεί να αλλάξει αν πρέπει να προσαρμοστεί σε κάποια μεταβολή στο bandwidth
- frameindication, που σηματοδοτεί την αρχή και το τέλος κάθε πλαισίου
- αναγνώριση πηγής(*source identification*), η οποία προσδιορίζει τον εντολέα του πλαισίου
- και *intramedia synchronization*ι, που χρησιμοποιεί timestamps για να ανιχνεύσει ενδεχόμενη καθυστέρηση στο buffer και να φροντίσει για την αντιστάθμιση της.

Σε αυτό το κεφάλαιο έγινε μια συνοπτική παρουσίαση των πρωτοκόλλων τα οποία κρίθηκε απαραίτητο να μελετηθούν για την υλοποίηση την συγκεκριμένης εργασίας και τα οποία συνδυάζονται αρμονικά μεταξύ τους προκειμένου να πάρουμε το βέλτιστο αποτέλεσμα. Στα επόμενα κεφάλαια θα εμβαθύνουμε στην υλοποίηση και στον τρόπο που χρησιμοποιήθηκαν τα παραπάνω πρωτόκολλά.

3. Παρουσίαση Συστήματος

3.1 Εισαγωγή

Όπως είπαμε και πιο πάνω τα τελευταία χρόνια οι χρήστες αποζητούν κάτι παραπάνω από μια απλή επικοινωνία μέσω του τηλεφώνου τόσο από άποψη οικονομίας όσο και από άποψη δυνατοτήτων. Με αυτό ως γνώμονα η συγκεκριμένη εργασία μελέτα να συγκεκριμένα δίκτυα επόμενης γενιάς που πετυχαίνουν αυτόν τον σκοπό και στην συνέχεια παρατίθεται η υλοποίηση ενός πειραματικού συστήματος τέτοιου με το οποίο καταφέραμε να πετύχουμε την μεταφορά φωνής σε real time χρησιμοποιώντας ως πυλώνες τα πρωτόκολλα που περιγράψαμε στο προηγούμενο κεφάλαιο. Η υλοποίηση του πραγματοποιήθηκε σε οικιακό δίκτυο με την χρήση δύο τερματικών .

3.2 Επιλεγμένος τρόπος υλοποίησης

Προκειμένου να προχωρήσουμε στην υλοποίηση ενός τέτοιου συστήματος πρέπει να σκεφτούμε κάποια βασικά πράγματα:

1) Ποια γλώσσα προγραμματισμού θα χρησιμοποιήσουμε;
2) Ποιο εργαλείο θα χρησιμοποιηθεί για την συγγραφή του κώδικα;
3) Πως θα προσαρμοστούν τα πρωτόκολλα που έχουν επιλεγεί στις ανάγκες του συστήματος ;
4) Ποια θα είναι η μορφή του GUI ώστε να είναι φιλικό προς τον χρήστη;

Πρώτα από όλα η γλώσσα προγραμματισμού που επιλέχθηκε είναι η Java λόγω τόσο των βιβλιοθηκών που διαθέτει και μας βοήθησαν ιδιαίτερα στην υλοποίηση όσο και στο ότι πρόκειται γενικά για μια γλώσσα ιδιαίτερα απλή και ευέλικτη που προσφέρει μεγαλύτερη ασφάλεια στο δίκτυο που δημιουργεί.

Η συγγραφή του κώδικα και η εκτέλεση του πραγματοποιήθηκε στο Netbeans προκειμένου να υπάρχει η δυνατότητα εύκολης και γρήγορης αναγνώρισης και επιδιόρθωσης λαθών στον κώδικα όπως επίσης και για την άμεση

εκτέλεση του που θα μας επιτρέψει να προσδιορίσουμε γρηγορότερα τις επιδιορθώσεις στις οποίες θα πρέπει να προβούμε.

Έτσι αφού βρήκαμε στο διαδίκτυο το εργαλείο στην σελίδα http://netbeans.org/features/index.html κατεβάσαμε το πρόγραμμα και το εγκαταστήσαμε στον υπολογιστή προκειμένου να ξεκινήσει η συγγραφή του κώδικα.

Η υλοποίηση την οποία επιδιώκουμε να πετύχουμε περιλαμβάνει την επικοινωνία μεταξύ δυο φορητών υπολογιστών με απώτερο σκοπό την αμφίδρομη μεταφορά φωνής σε πραγματικό χρόνο.

Πηγές για την επιτυχή υλοποίηση του προγράμματος αποτέλεσαν σελίδες με tutorial στο διαδίκτυο αλλά και εκπαιδευτικά βίντεο της διαδικτυακής υπηρεσία του youtube με αντίστοιχα πειράματα.

Τα βασικά πρωτόκολλα τα οποία επιλέχθηκαν να χρησιμοποιηθούν για την υλοποίηση μας είναι το SIP και το RTP.

Οι βασικές οντότητες οι οποίες θα χρησιμοποιηθούν θα είναι δυο Media Gateways τα οποία πρακτικά θα είναι τα δύο laptop και δυο Control Servers ένα για κάθε χρήστη. Πιο κάτω θα αναφερθούμε πιο συγκεκριμένα στις δυο αυτές οντότητες.

Προκειμένου να γίνει ποιο σωστή διαχείριση των πηγών και το αποτέλεσμα να είναι το καλύτερο δυνατό προτιμήθηκε η υλοποίηση να διαχωριστεί σε κάποια κομμάτια προκειμένου να είμαστε σίγουροι ότι εκτελείται σωστά το καθένα από αυτά ξεχωριστά και έπειτα να γίνει η σύνδεση τους.

Βήμα 1:

Το πρώτο πράγμα που κάναμε είναι η δημιουργία ενός project για την επικοινωνία μέσω socket ανάμεσα στα Media Gateways και τους control servers τους μέσω ενός socket ακρόασης που ανοίγει ο δεύτερος.

Βήμα 2:

Στην συνέχεια με την βοήθεια του συγκεκριμένου εξαιρετικού άρθρου στον διαδικτυακό τόπο
http://alex.bikfalvi.com/teaching/upf/2013/architecture_and_signaling/lab/sip/
προχωρήσαμε στην σύνδεση των δυο server με την βοήθεια του πρωτοκόλλου SIP το οποίο πραγματοποιείται με ένα δικό του περιβάλλον εκτέλεσης με τον καλών να κάνει αναζήτηση των ενεργών χρηστών του δικτύου, επιλέγει στην συνέχεια τον χρήστη που θέλουμε να καλέσουμε και στην άλλη πλευρά ο κληθέντας δέχεται αρχικά μήνυμα σύνδεσης από τον καλών και στην συνέχεια δέχεται αίτημα κλήσης στο οποίο έχει την επιλογή είτε να δεχτεί και να προχωρήσει σε συνδιάσκεψη είτε να την απορρίψει.

Βήμα 3:

Σε αυτό το στάδιο αυτό που θέλαμε να πετύχουμε ήταν η ανταλλαγή φωνής σε πραγματικό χρόνο. Αρχικά έγινε προσπάθεια να γίνει μια απλή καταγραφή φωνής από το μικρόφωνο, να αποθηκευτεί με την μορφή .wav στον υπολογιστή και στην συνέχεια να αποσταλεί στον αποδέκτη εκεί όπου θα αναπαραχθεί από τα ηχεία.Το βήμα αυτό ήταν κομβικό για την υλοποίηση καθώς θα αποτελούσε την βάση και για την αμφίδρομη μεταφορά φωνής από τον έναν χρήστη στον άλλον.

Βήμα 4:

Αφού επετεύχθη αυτό έγινε προσπάθεια να γίνει το ίδιο χωρίς τοπική αποθήκευση της φωνής αλλά άμεσης αποστολής της. Έτσι καταφέραμε να πετύχουμε αρχικά την άμεση καταγραφή και μεταφορά σε πραγματικό χρόνο της φωνής και την αναπαραγωγής της από τα ηχεία

Βήμα 5:

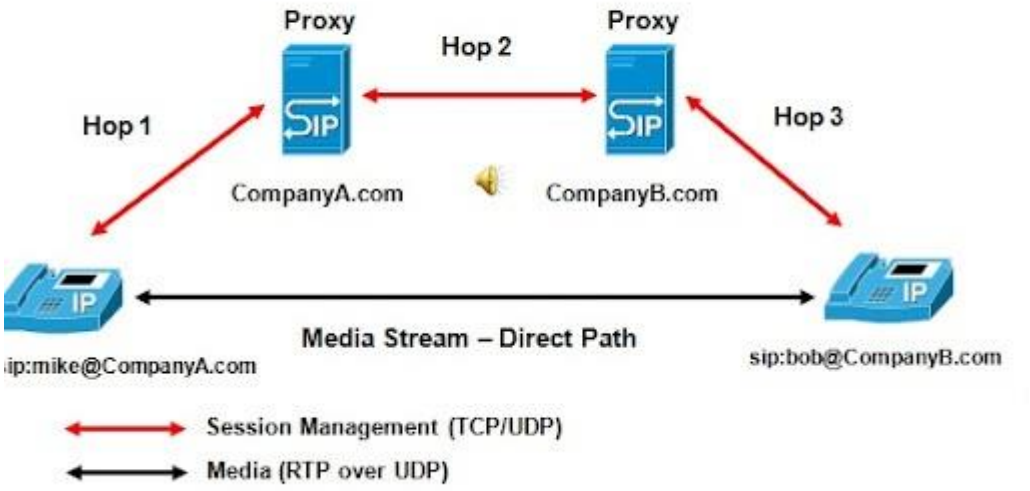
Το επόμενο βήμα το οποίο θα έπρεπε να γίνει ήταν η δυνατότητα αμφίδρομης τόσο καταγραφής και αποστολής φωνής όσο και ανάγνωσης και αναπαραγωγής της. Αυτό το καταφέραμε δημιουργώντας στις δυο πλευρές των χρηστών κλάσεις που επιτελούν την αντίστροφή λειτουργία δηλαδή στον κληθέντα χρήστη να μπορεί να

αποστείλει φωνή και στον καλούντα να μπορεί να αναγνώσει και να την αναπαράγει.

Βήμα 6:

Τελικά συνδέοντας όλα τα παραπάνω κομμάτια και δημιουργώντας το περιβάλλον που θα δούμε πιο κάτω καταφέραμε να ολοκληρώσουμε την υλοποίηση του συστήματος κατά το οποίο ένας χρήστης συνδέεται ,καλεί οποιονδήποτε άλλο χρήστη βρίσκεται μέσα στο δίκτυο και μπορεί τόσο να στείλει όσο και να διαβάσει πακέτα ήχου.

3.3 Βασικές οντότητες



Η υλοποίηση την οποία επιδιώκουμε να πετύχουμε περιλαμβάνει την επικοινωνία μεταξύ δυο φορητών υπολογιστών με απώτερο σκοπό την αμφίδρομη μεταφορά φωνής σε πραγματικό χρόνο.

Τα Media Gateways:

Ουσιαστικά θα είναι οι δυο βασικές οντότητες του συστήματος οι οποίες θα εκτελούν και την μεταφορά φωνής μεταξύ τους. Θα μπορούν να ξεκινήσουν μια κλήση να την διακόψουν και να την τερματίσουν. Η λειτουργία τους εξαρτάται άμεσα από τους Control Servers και οποιαδήποτε ενέργει εκτελείται πηγάζει από έλεγχο και εντολή από εκείνους.

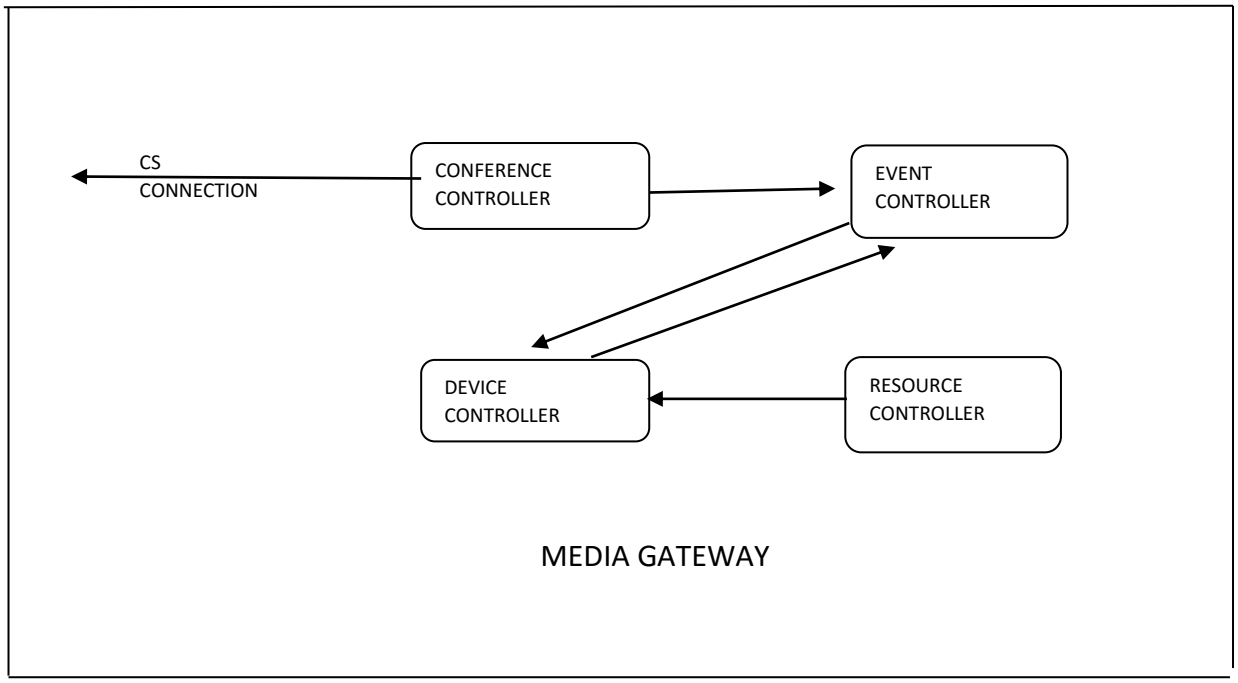
Για την υποστήριξη της λειτουργία των MG δημιουργήσαμε ένα σύνολο διαχειριστών ο καθένας από τους οποίους θα επιτελεί μια βασική λειτουργία προκειμένου να λειτουργήσει σωστά το σύστημα .Οι διαχειριστές αυτοί είναι:

Ο διαχειριστής συνεδριών (conference controller)

Ο διαχειριστής πόρων (resource controller)

Ο διαχειριστής συσκευών (device controller)

Ο διαχειριστής γεγονότων(event controller)



Ο διαχειριστής συνεδριών:

Η οντότητα αυτή είναι υπεύθυνη για την επικοινωνία του Media Gateway με τον Control Server και είναι υπεύθυνος τόσο για την έναρξη όσο και για τον τερματισμό μια συνεδρίας.

Ο διαχειριστής γεγονότων :

Ο συγκεκριμένος διαχειριστής αφού γίνει η σύνδεση του MG με τον CS είναι εκείνος που θα ξεκινήσει την διαδικασία για την έναρξη της μεταφοράς ήχου από τον έναν χρήστη στον άλλον και είναι εκείνος που ενεργοποιεί τόσο τον κωδικοποιητή φωνής όσο και τον αποκωδικοποιητή για την ακρόαση.

Ο διαχειριστής συσκευών:

Είναι η οντότητα που είναι υπεύθυνη για την διασύνδεση των δυο χρηστών και που θα εκκίνηση την διαδικασία της μεταφοράς φωνής σε πραγματικό χρόνο. Αρχικά ανοίγει ένα port μέσω του οποίου ο πομπός θα μιλάει και ο δέκτης θα ακούει την φωνή ενώ στο άλλο άκρο ενώ υπάρχει και η αντίστροφη λειτουργία ούτως ώστε να είναι αμφίδρομη η επικοινωνία. Επίσης είναι εκείνος ο οποίος θα καλέσει τον κωδικοποιητή και τον αποκωδικοποιητή ώστε να γίνει κωδικοποίηση και αποκωδικοποίηση του ήχου ώστε να γίνει η μεταφορά του

Ο διαχειριστής πόρων:

Είναι η οντότητα που εμπεριέχει τις συσκευές κωδικοποίησης και αποκωδικοποίησης .Λαμβάνει χώρα και στον πομπό και στον δέκτη και ανοίγει έπειτα από εντολή του διαχειριστή συσκευών

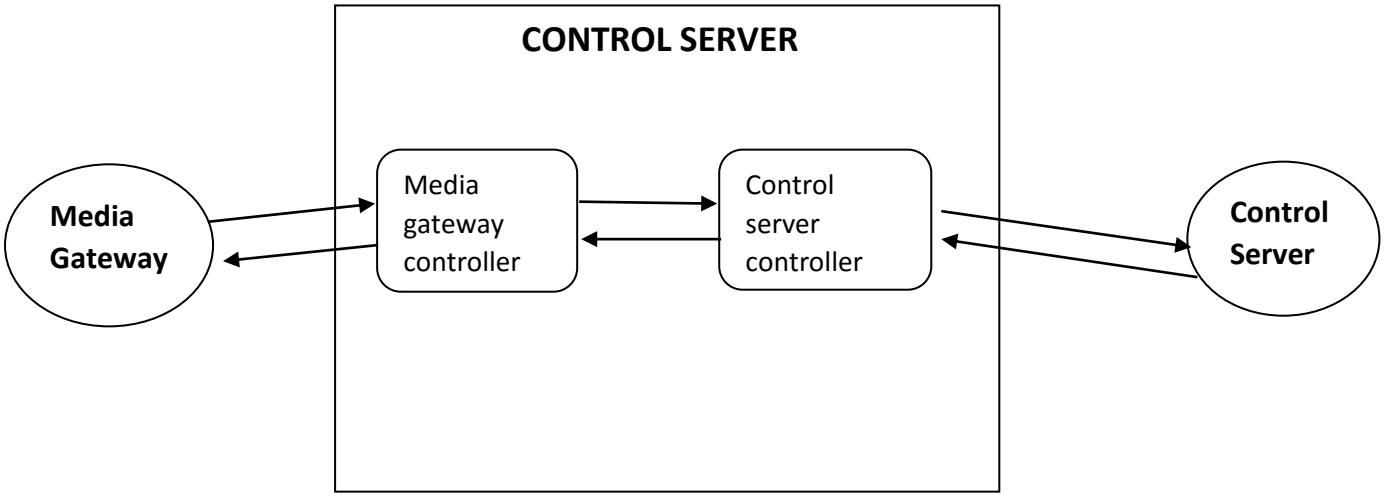
Η όλη διαδικασία λοιπόν ξεκινάει από τον διαχειριστή συνεδριών η οποί αρχικά συνδέεται με τον control Server, στην συνέχεια αιτείται την έναρξη κλήσης και μετά από μια σειρά εντολών που θα δούμε παρακάτω δέχεται εντολή από τον Server να θέσει σε λειτουργία τον διαχειριστή γεγονότων μέσω του οποίου ξεκινάει η διαδικασία της μεταφοράς φωνής. Ποιο κάτω θα δούμε αναλυτικό παράδειγμα λειτουργίας στο οποίο θα αποσαφηνιστεί η όλη διαδικασία λειτουργίας.

Οι Control Servers

Η λειτουργία των Media Gateways όπως είδαμε και παραπάνω εξαρτάται άμεσα από τις εντολές που θα δεχτούν από τους Control Servers.Οι control servers είναι εκείνοι που θα πάρουν το αίτημα για κλήση από τον διαχειριστή συνεδριών του Media Gateway όπως είδαμε πιο πάνω, θα αναζητήσουν τον χρήστη τον οποίον προσπαθεί να καλέσει το MG και θα επικοινωνήσει μέσω του πρωτοκόλλου SIP με τον server του αποδέκτη της κλήσης ο οποίος θα ενημερώσει τον δικό του χρήστη θα λάβει μια απάντηση και με την αντίστροφη διαδικασία θα φτάσει στον άλλο MG ώστε να γίνουν οι αντίστοιχες ενέργειες.

Για την σωστή και απρόσκοπτη λειτουργία της λειτουργίας αυτής κάθε control server διαθέτει δυο βασικούς controlers:

- Ο Media Gateway controller
- Ο Control Server controller



Ο Media Gateway controller

Είναι υπεύθυνος για την επικοινωνία του Media Gateway με τον server.Ο διαχειριστής συνεδριών του media gateway του στέλνει τα στοιχεία του χρήστη με

τον οποίον θέλει να συνδεθεί και έπειτα από αναζήτηση θα επικοινωνήσει με τον Control Server Controller προκειμένου να συνδεθεί να τον αντίστοιχο server και να ξεκινήσει η κλήση.

Ο Control Server controller

Ο Control server controller είναι εκείνος που εκτελεί το πρωτόκολλο SIP για την επικοινωνία των δυο server και κατ' επέκταση και των δυο τερματικών. Ο ρόλος του είναι ιδιαίτερα σημαντικός γιατί μέσω της γεφύρωσης των δυο server γίνεται η έμμεση επικοινωνία των δυο Media gateways προκειμένου να γίνει έναρξη συνομιλίας ή κατάργηση της κλήσης.

Οι οντότητες αυτές είναι οι ίδιες και για τα δυο τερματικά. Για λόγους ευκολίας και ο control server αλλά και τα media gateway βρίσκονται στον ίδιο υπολογιστή. Ωστόσο έχει γίνει πειραματική δοκιμή για την ορθή λειτουργία του συστήματος και σε διαφορετικές συσκευές με θετικά αποτελέσματα. Παρακάτω θα γίνει μια αναλυτική παρουσίαση του τρόπου λειτουργίας του συστήματος

3.4 Παράδειγμα λειτουργίας του συστήματος

Ποιο πάνω είδαμε αποσπασματικά τα βήματα που ακολουθήθηκαν για την υλοποίηση της παρούσας εργασίας καθώς και της βασικές οντότητες που χρησιμοποιήθηκαν ώστε να δομηθούν σωστά οι λειτουργίες του συστήματος.

Στο συγκεκριμένο σημείο δεν θα ασχοληθούμε τόσο με το γραφικό περιβάλλον που αναπτύχθηκε για να υποστηριχθεί η λειτουργία του δικτύου μας αλλά στις αλυσιδωτές ενέργειες που εκτελούνται προκειμένου να πετύχουμε το επιθυμητό αποτέλεσμα.

Το συγκεκριμένο πρόγραμμα είναι στημένο και στους δυο υπολογιστές εκ των οποίων η μια θα είναι ο καλών που θα ξεκινήσει την κλήση και η δεύτερη θα είναι ο καλούμενος.

Αρχικά ο καλών μόλις μπει στο σύστημα ο διαχειριστής συνεδριών του συνδέεται με τον media gateway controller του control server.

Μόλις συνδεθεί ανοίγει ο control server controller ο οποίος θα ψάξει μέσω τοπικής αναζήτησης στο δίκτυο για ενεργούς χρήστες και στην συνέχεια ο χρήστης θα επιλέξει ποιον θέλει να καλέσει.

Εφόσον ο καλούμενος είναι και αυτός αντίστοιχα συνδεδεμένος με τον server του μέσω του διαχειριστή συνεδριών αντίστοιχα ενεργοποιηθεί το γραφικό περιβάλλον του control server controller στο οποίο θα εμφανιστεί η προσπάθεια σύνδεσης από την άλλη πλευρά και θα του απαντήσει με ένα ok response ότι είναι συνδεδεμένος.

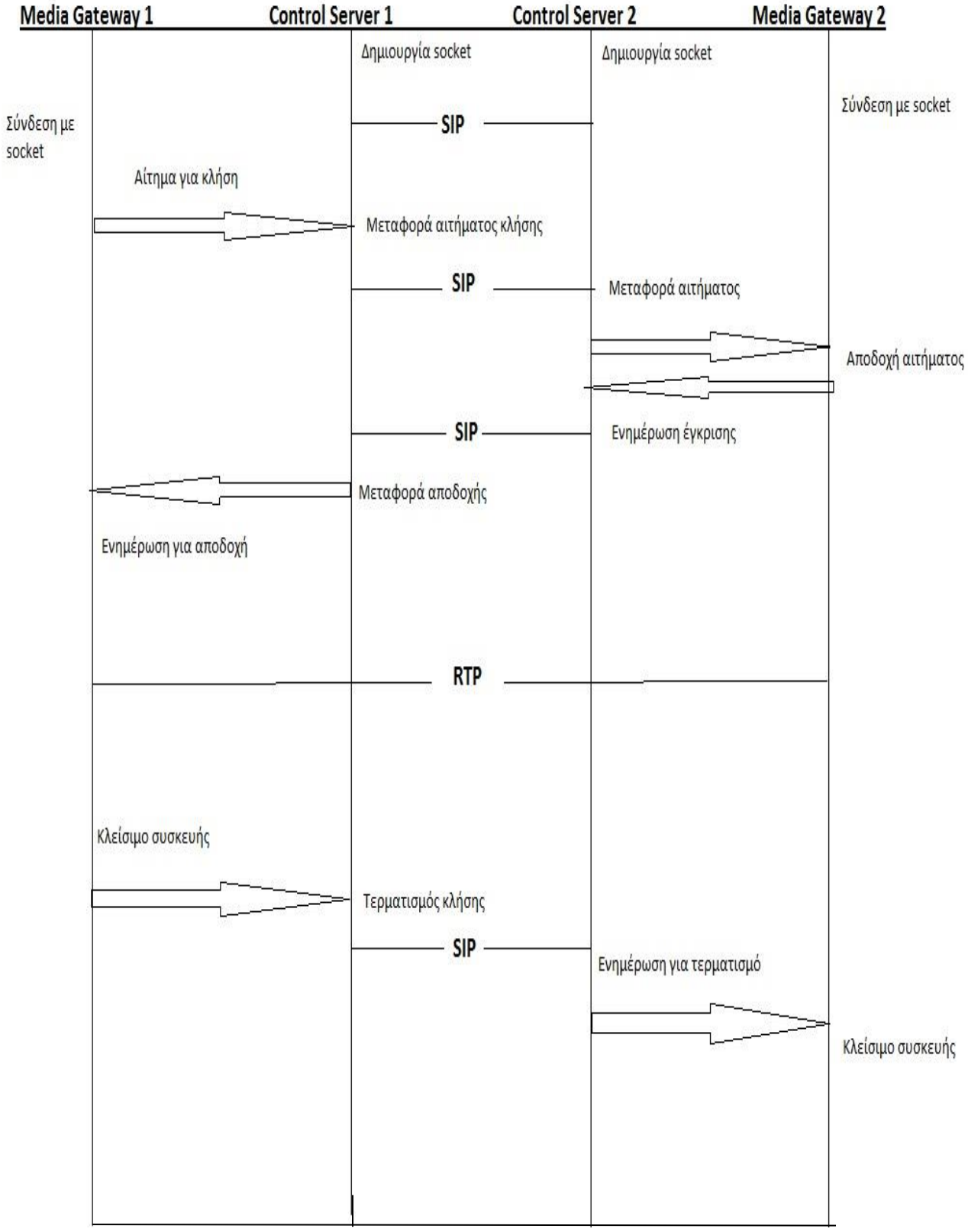
Στην συνέχεια ο control server controller του καλούντος θα στείλει request invite. Εκείνη την στιγμή θα ξεκινήσει ο διαχειριστής γεγονότων του χρήστη ,ο οποίος με την σειρά του θα ανοίξει τον διαχειριστή συσκευών και στην συνέχεια τον κωδικοποιητή και τον αποκωδικοποιητή ώστε να είναι σε ετοιμότητα σε περίπτωση αποδοχής της κλήσης.

Στο περιβάλλον του control server controller του κληθέντος θα εμφανιστεί το request invite που θα ενεργοποιήσει παράθυρο διαχείρισης αιτήματος για αποδοχή ή απόρριψη της κλήσης.

Σε περίπτωση αποδοχής θα εκκινήσει αντίστοιχα τον διαχειριστής γεγονότων ,ο διαχειριστής συσκευών και τον διαχειριστή πόρων και θα ξεκινήσει η διαδικασία καταγραφής ,κωδικοποίησης και αποκωδικοποίησης της φωνής.

Σε περίπτωση που δεν αποδεχθεί την κλήση θα αποστέλλεται response BYE το οποίο στον control server controller του καλούντος το οποίο θα κλείσει τους διαχειριστές συσκευών ,γεγονότων και πόρων και θα τερματίσει έτσι την κλήση.

Αντίστοιχα σε περίπτωση που έχει ολοκληρωθεί η κλήση ο καλών θα στείλει BYE στον κληθέντα και εκείνος θα κλείσει τους διαχειριστές που είναι ανοιγμένοι για την μεταφορά φωνής


Media Gateway 1
Control Server 1
Control Server 2
Media Gateway 2
Δημιουργία socket
Δημιουργία socket
Σύνδεση με socket
Σύνδεση με socket
SIP
Αίτημα για κλήση
Μεταφορά αιτήματος κλήσης
SIP
Μεταφορά αιτήματος
Αποδοχή αιτήματος
SIP
Ενημέρωση έγκρισης
Μεταφορά αποδοχής
Ενημέρωση για αποδοχή
RTP
Κλείσιμο συσκευής
Τερματισμός κλήσης
SIP
Ενημέρωση για τερματισμό
Κλείσιμο συσκευής

4. Υλοποίηση

4.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα επικεντρωθούμε στο πρακτικό-εκτελεστικό κομμάτι της υλοποίησης. Αυτό σημαίνει ότι θα δούμε αναλυτικά τις κλάσεις που χρησιμοποιήθηκαν με τις λειτουργίες που επιτελούν μέσω των μεθόδων που διαθέτουν ,πώς συνδέονται τόσο λογικά όσο και γραφικά. Επίσης θα γίνει μια παρουσίαση της εκτέλεσης του προγράμματος με εικόνες που τραβήχτηκαν κατά την λειτουργία του ούτως ώστε να καταλάβουμε ακριβώς την λειτουργία του. Όπως προ είπαμε η γλώσσα προγραμματισμού που χρησιμοποιήσαμε είναι η Java και το εργαλείο συγγραφής κώδικα είναι το NetBeans. Αρχικά έγινε ένα βασικός έλεγχος τους κώδικα στον υπολογιστή που πραγματοποιήθηκε η συγγραφή και στην συνέχεια πραγματοποιήθηκε έλεγχος με ταυτόχρονη εκτέλεση του κώδικα σε δυο laptop προκειμένου να ελέγξουμε και την αποτελεσματικότητα του.

4.2 Περιγραφή

Στο προηγούμενο κεφάλαιο χωρίσαμε την βασική υλοποίηση σε κάποια βασικά στάδια προκειμένου να γίνει πιο εύκολη και ακριβής η λειτουργία του συστήματος. Επιγραμματικά τα στάδια αυτά ήταν:

1. Η σύνδεση Media Gateway με Control Server
2. Η σύνδεση Control Server με Control Server
3. Η ανταλλαγή φωνής σε πραγματικό χρόνο μεταξύ των Media Gateways

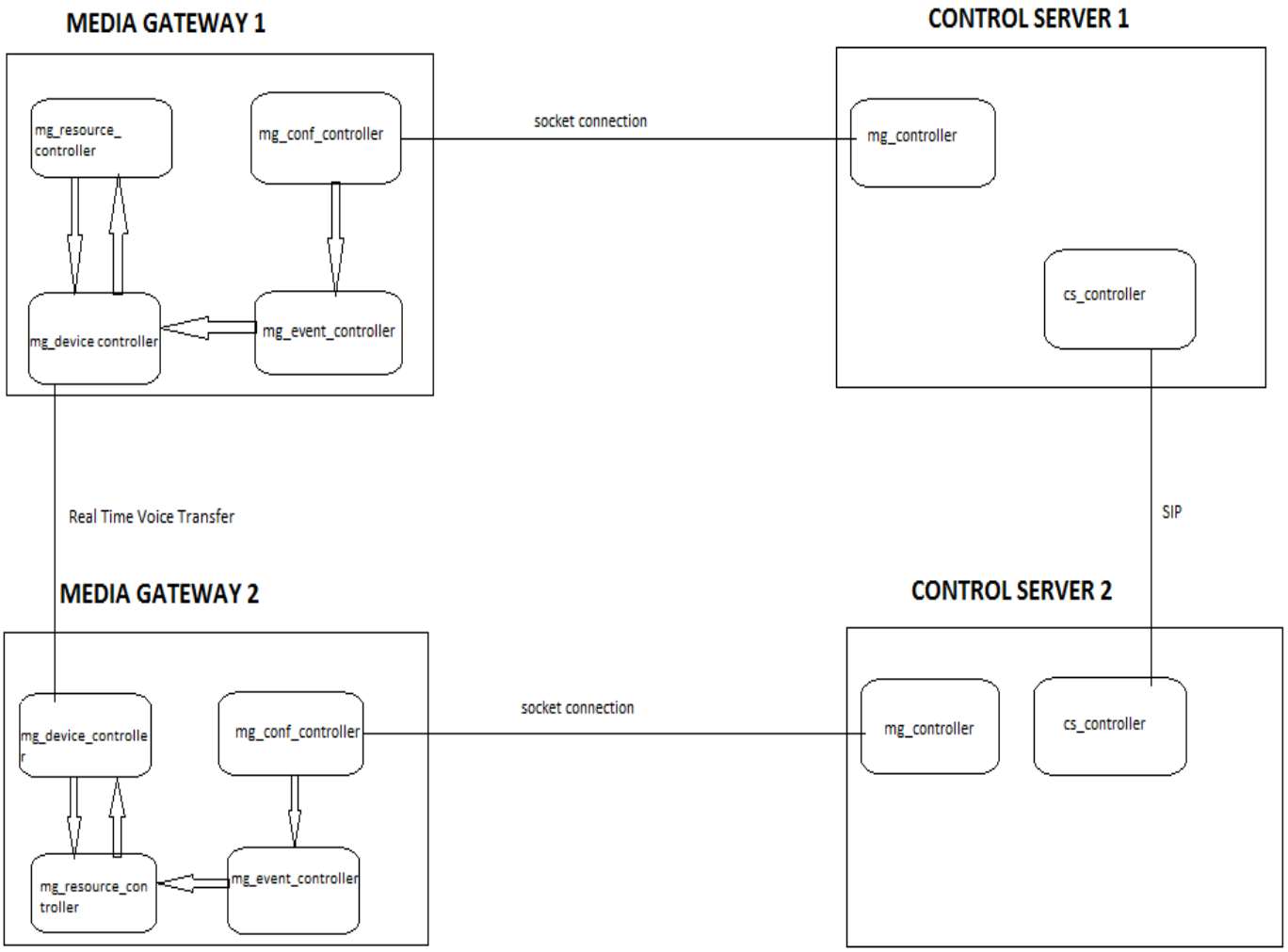
Το σύστημα μας αποτελείται από δύο χρήστες στον καθέναν από τους οποίους δημιουργήσαμε τις οντότητες:

- Ngn.java: που αποτελεί την βασική κλάση (main class) για την έναρξη της εκτέλεσης του προγράμματος
- Mg_controller: η κλάση για την δημιουργία του socket και την σύνδεση με το Media Gateway
- CS_controller_server: που επιτελεί την λειτουργία του SIP από την πλευρά του αποδέκτη της κλήσης
- CS_controller_client: που επιτελεί την λειτουργία του SIP από την πλευρά του χρήστη που εκκινεί την κλήση
- Mg_conf_controller: για την σύνδεση του Media Gateway στον Control Server
- Mg_event_controller: για την εκκίνησης την κωδικοποίησης /αποκωδικοποίησης φωνής
- Mg_resource_controller_recorder: για την εκκίνηση της καταγραφής της φωνής

- Mg_resource_controller_player: για την εκκίνηση της αναπαραγωγής φωνής
- Mg_device_controller_server: που περιλαμβάνει τον περιβάλλον έναρξης και ολοκλήρωσης της κλήσης καθώς και την εκκίνηση του mg_resource_controller_recorder και mg_resource controller_player

Η σύνδεση Media Gateway με Control Server

Για την σύνδεση των Media Gateway με τους αντίστοιχούς Control Servers τους ακολουθήσαμε μια απλή σύνδεση με socket.Συγκεκριμένα στην κλάση mg_controller ανοίξαμε ένα socket ακρόασης στο port 81 και στην συνέχεια με την εκτέλεση του mg_conf_controller πραγματοποιήθηκε η σύνδεση



Η σύνδεση Control Server με Control Server

Στην συνέχεια εκτελείται η κλάση cs_controller του control server που του δίνει την δυνατότητα να επιλέξει τον χρήστη με τον οποίον θέλει να συνδεθεί. Αρχικά καθορίζουμε τα δεδομένα που θα αποσταλούν στον αποδέκτη όπως η διεύθυνση το port ακρόασης κλπ. Στην συνέχεια δημιουργείται το σημείο ακρόασης SIP και δεσμεύονται οι πληροφορίες .Τέλος καταχωρούμε την κλάση ως SIP Linstener.Στην συνέχεια στο γραφικό περιβάλλον επιλέγουμε το κουμπί search ώστε να καλέσει την αντίστοιχη μέθοδο που θα μας εμφανίσει μέσω της διενέργειας μιας αναζήτησης στο δίκτυο όλους τους συνδεδεμένους χρήστες. Τώρα αφού επιλέξουμε την διεύθυνση μας δίνεται μια σειρά από αιτήματα που μπορούν να γίνουν ώστε να πραγματοποιηθεί η σύνδεση. Πιο συγκεκριμένα έχουμε την επιλογή Register Stateless η μέθοδος της οποίας αρχικά παίρνει την διεύθυνση του καλούμενου ,δημιουργεί της κεφαλίδες με τις πληροφορίες που θα εμφανιστούν, δημιουργείται το μήνυμα με την χρήση της μεθόδου message factory και γίνεται η αποστολή του αιτήματος. Αντίστοιχα έχουμε την επιλογή Register Statefully η μέθοδος της οποίας είναι αντίστοιχη ωστόσο δημιουργείται μια νέα ClientTransaction έτσι ώστε να υπάρχει και αποθήκευση πληροφοριών της σύνδεση και το αίτημα στέλνεται μέσω αυτού του Transaction.Με αντίστοιχό τρόπο αποστέλλονται και αιτήματα για έναρξη κλήσης μέσω του αντίστοιχου κουμπιού και μεθόδου Invite αλλά και για ολοκλήρωση μέσω του Bye.Στην πρώτη περίπτωση εκτελείται το mg_device_controller και στην δεύτερη σταματάει να εκτελείται. Στην άλλη πλευρά του SIP αρχικά καθορίζονται οι πληροφορίες που θα αποστέλλονται στις απαντήσεις του και δημιουργούνται οι ανάλογες κεφαλίδες ,δημιουργείται σημείο επικοινωνίας SIP και τέλος σε κάθε αίτημα που δέχεται ενημερώνεται μέσω της μεθόδου table update ο πίνακας που περιέχει το γραφικό περιβάλλον ανάλογα με τα headers που έχει δημιουργήσει ο πομπός του αιτήματος. Σε περίπτωση που το αίτημα είναι Register Stateless ή Statefully η απόκριση αποτελείται από τα Headers που δημιουργήθηκαν και το μήνυμα με κωδικό 200 OK. Αν το αίτημα είναι Invite αρχικά συντάσσεται απόκριση με της κεφαλίδες και κωδικό 180 Ringing θέτοντας έτσι την επικοινωνία σε κατάσταση αναμονής για αποδοχή. Ταυτόχρονα εκτελείται η κλάση calling.java η οποία δίνει την δυνατότητα είτε για αποδοχή είτε για απόρριψη του αιτήματος. Αν γίνει αποδοχή αρχικά αποστέλλεται απάντηση 200 OK από τον cs_controller_server.java και εκτελείται η κλάση mg_device_controller.java.Σε περίπτωση είτε αιτήματος BYE είτε απόρριψης της κλήσης αποστέλλεται απάντηση με τις κεφαλίδες και με κωδικό αυτή την φορά 603 .

Η ανταλλαγή φωνής σε πραγματικό χρόνο μεταξύ των Media Gateways

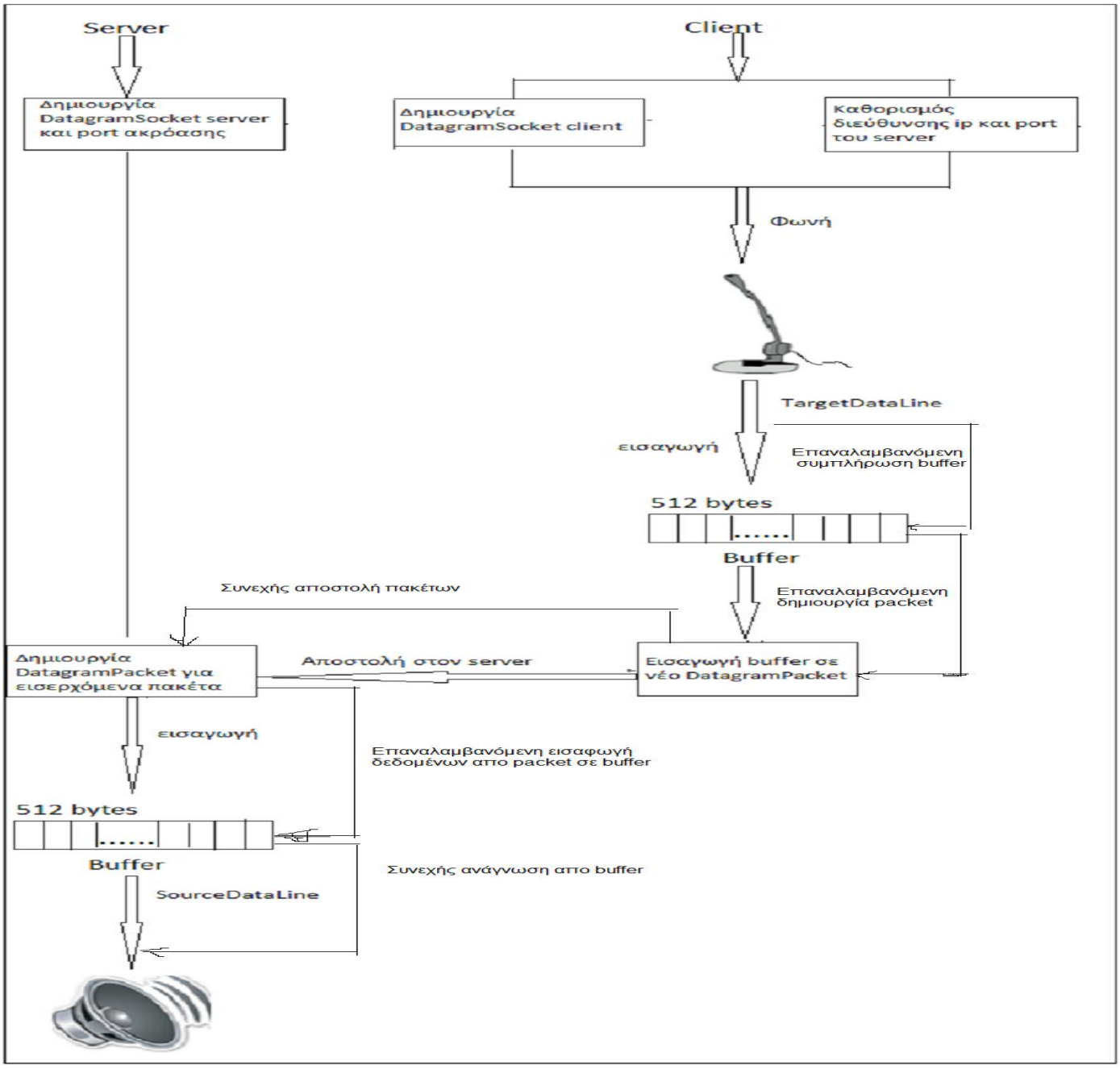
Αρχικά η πρώτη κλάση που εκτελείται είναι η mg_event_controller η οποία δημιουργεί δυο λογικές μεταβλητές ώστε να ρυθμίζεται η έναρξη και λήξη της καταγραφής και αναπαραγωγής της φωνής. Έτσι καταρχήν ανοίγει port για την λήψη πακέτων ήχου και συνδέεται με το αντίστοιχο port του άλλου χρήστη για την αποστολή των πακέτων ήχου. Αφού καθοριστούν τόσο τα port όσο και η διεύθυνση του δεύτερου συνδιαλεγόμενου της κλήσης εκτελούνται οι δυο βασικές μέθοδοι της κλάσης ταυτόχρονα και στους δυο ώστε να έχουμε αμφίδρομη επικοινωνία μεταξύ των χρηστών. Σε αυτές τις δυο κλάσεις επιλέξαμε να δημιουργήσουμε 3 buffer: ένα για την εισαγωγή των πακέτων φωνής και άλλα δυο που θα μεταφέρουν

πληροφορίες σχετικά με το σύνολο των bytes που έχουν αποσταλεί και τον χρόνο αποστολής προκειμένου να υπολογιστούν στην αντίστοιχη μέθοδο η καθυστέρηση(delay) ο χρόνος δηλαδή που μεσολάβησε από την αποστολή μέχρι την λήψη του πακέτου, το bandwith έτσι ώστε να προσδιοριστεί η ταχύτητα του δικτύου και να εισαχθεί αν χρειαστεί κάποια καθυστέρηση και το jitter δηλαδή η διαφορά της καθυστέρησης του παρόντος πακέτου με το αμέσως προηγούμενο του. Παρακάτω θα προσδιορίσουμε τον τρόπο λειτουργίας των δύο αυτών βασικών μεθόδων της rec_audio και της play_audio

Η μέθοδος rec_audio χρησιμοποιείται για την καταγραφή φωνής θέτοντας την τιμή της μεταβλητής calling ως true προκειμένου καλώντας αργότερα την μέθοδο rec του mg_resource_contr2_recorder να ξεκινήσει η κωδικοποίηση της φωνής. Του δίνει δηλαδή τις πληροφορίες σχετικά με την διεύθυνση αποστολής ,το port ακρόασης καθώς και το μέσο καταγραφής της φωνής τα οποία απαιτούνται από τον constructor Datagram Packet. Η συγκεκριμένη μέθοδος της κλάσης mg_resource_controller_recorder δέχεται την είσοδο των πακέτων φωνής μέσω του μικροφώνου εισάγοντας τα στο buffer και την αποστέλλει στον κληθέντα χρήστη μέσω του DaragramPacket που δημιουργείται. Πιο αναλυτικά αρχικά δημιουργείται ένα buffer από 512 bytes ,και καταχωρούνται η διεύθυνση ip και το port στο οποίο θα αποσταλεί η φωνή. Όσο η συνθήκη calling του mg_event controller είναι αληθής πακέτα ήχου εισάγονται στο buffer μέχρι να γεμίσει και στην συνέχεια προστίθενται στο DatagramPacket το οποίο μέσω του DatagramSocket θα αποσταλεί στον άλλο χρήστη ώστε τα bytes να εισαχθούν στο αντίστοιχο buffer που έχει δημιουργήσει Να σημειωθεί εδώ ότι το πλήθος των δεδομένων που θα εισαχθούν στο buffer θα πρέπει να είναι μικρότερο ή ίσο των θέσεων που διαθέτει ώστε να αποφευχθεί η υπερχείλιση. Προκειμένου να υπάρξει μια εικόνα για την καθυστέρηση στην εισαγωγή τον πακέτων καταγράφεται ο χρόνος εισαγωγής στο buffer ο οποίος μετατρέπεται σε bytes και αποστέλλεται μέσω του socket προκειμένου να υπολογιστεί η καθυστέρηση και το jitter.Τέλος αντίστοιχη λειτουργία πραγματοποιείται και για το σύνολο των bytes που θα αποσταλούν μέσω των πακέτων προκειμένου να καθοριστεί το ποσοστό αποτυχίας

Η μέθοδος play_audio χρησιμοποιείται για την αναπαραγωγή φωνής θέτοντας την τιμή της μεταβλητής playing ως true προκειμένου καλώντας αργότερα την μέθοδο rec του mg_resource_contr2_player να ξεκινήσει η αποκωδικοποίηση της φωνής. Και σε αυτήν την περίπτωση προσδιορίζονται η διεύθυνση από την οποία δέχεται τη φωνή όπως επίσης και το μέσο αναπαραγωγής (δηλαδή τα ηχεία) τα οποία θα χρησιμοποιηθούν από την μέθοδο mg_resource_controller_player. Η μέθοδος run του mg_resource_controller_player δέχεται τα πακέτα φωνής (DatagramPacket) από τον χρήστη με τον οποίο πραγματοποιείται η κλήση μέσω του Datagram Socket ,τα αποκωδικοποιεί και τα αναπαράγει. Πιο συγκεκριμένα καθορίζονται ένα socket ακρόασης και ένα buffer μεγέθους 512 bytes. Όσο η συνθήκη playing του mg_event controller είναι αληθής γίνεται αποδοχή των πακέτων φωνής μέσω του Datagram Socket και θα εισαχθούν στο buffer που δημιουργήσαμε με συνεχή ροή. Για την

μεταφορά στην μονάδα αναπαραγωγής θα χρησιμοποιήσουμε την μέθοδο SourseDataLine .Η συγκεκριμένη μέθοδος ουσιαστικά δρα ως πηγή για τα ηχεία καθώς δέχεται τα bytes ήχου που διαχειρίζεται το buffering τους και τα αποστέλλει σε αυτά για την αναπαραγωγή.



Σχηματική αναπαράσταση λειτουργίας

Σε αυτή την κλάση επίσης γίνεται ο υπολογισμός τόσο των χαμένων πακέτων όσο και της καθυστέρησης αποστολής ενός πακέτου μέσω των πληροφοριών που στέλνει ο πομπός και συγκρίνοντας τες με τις τιμές των bytes και του χρόνου που έφτασαν τελικά .Επίσης για να διασφαλιστεί η ομαλή λειτουργία και σε δίκτυα

ευρείας κλίμακας (WAN) όπου η ταχύτητα είναι πιο χαμηλή και η διακύμανση καθυστέρησης μεγαλύτερη γίνεται έλεγχος του εύρους του jitter και αν είναι μεγαλύτερο από το όριο που έχουμε θέσει αυξάνεται το μέγεθός του buffer ώστε να συνεχίσει την ομαλή αναπαραγωγή πακέτων χωρίς διακοπές .Σε επόμενο κεφάλαιο θα γίνει εκτενής έλεγχος των αποτελεσμάτων με διάφορες τιμές τόσο του delay που θα θέσουμε όσο και του jitter που θα αποτελεί την μέγιστη αποδεκτή τιμή πριν μεγαλώσει το buffer όπως προείπαμε.

Να σημειωθεί ότι οι ρόλοι server και client που παρουσιάζονται στο σχήμα αναφέρονται ο μεν server στον δέκτη της κλήσης και ο δε client στον πομπό που εκκινεί την κλήση. Για να πετύχουμε την αμφίδρομη ανταλλαγή φωνής, δηλαδή συνομιλία αρκεί αυτοί οι δυο ρόλοι να αναστραφούν δηλαδή στον server να υπάρχει και ο μηχανισμός του client και το αντίστροφο και κατά την έναρξη της κλήσης να τρέχουν παράλληλα όλες οι αντίστοιχες μέθοδοι.

4.3 Βασικές κλάσεις και μέθοδοι

Ας δούμε τώρα αναλυτικότερα τι συμβαίνει σε κάθε μια από τις βασικές οντότητες που περιγράψαμε.

Ngn1.java

Με την έναρξη της συγκεκριμένης κλάσης ξεκινάει και η εκτέλεση του συστήματος εμφανίζοντας το γραφικό περιβάλλον του mg_controller.java

```
mg_controler2 fr = new mg_controler2();
    fr. setVisible (true);
    mg_conf_contr2.main(args);
```

Mg_controller.java

Η κλάση αυτή αποτελεί το αρχικό panel που θα εμφανιστεί στον χρήστη κατά την εκτέλεση του προγράμματος. Θα του δώσει την επιλογή να εισέλθει στο σύστημα είτε ως καλλών είτε ως κληθής και επιπλέον θα ανοίξει το port για την σύνδεση με το media gateway

```
final int portNumber = 81;
//Δημιουργεί ένα socket ακρόασης στο port 81
ServerSocket = new ServerSocket(portNumber);


public mg_controler2() throws IOException {
System.out.println("waiting for connection... " + portNumber);

initComponents ( );

// εμφανίζει αυτό το μήνυμα μέχρι κάποιος χρήστης να   συνδεθεί στο port
```

Mg_conf_controller1.java

Με την έναρξη της εκτέλεσης του mg_controller γίνεται και η σύνδεση του χρήστη στον συγκεκριμένο server που θα του επιτρέψει να προχωρήσει σε κάποια κλήση

```
final String host = InetAddress.getLocalHost().getHostAddress();;
```

//η ip του server που στην προκειμένη περίπτωση που βρίσκονται στον ίδιο υπολογιστή είναι η ίδια με την δικιά του και το port που θα συνδεθεί

```
final int portNumber = 81;

System.out.println("Creating socket to '" + host + "' on port " + portNumber);
//μήνυμα σύνδεσης


                                                                    }
```

Cs_controller1_server.java

Σε περίπτωση που ο χρήστης επιλέξει να εισέλθει ως callee εκτελείται η συγκεκριμένη κλάση η οποία είναι έτοιμη ανά πάσα στιγμή να δεχθεί αίτημα κλήσης

Μέθοδος onOpen :

αυτή η μέθοδος καλείται όταν αρχίσει να εκτελείται η συγκεκριμένη κλάση

```
  private void onOpen(java.awt.event.WindowEvent evt) {

    try {
      this.ip = InetAddress.getLocalHost().getHostAddress();

      this.sipFactory = SipFactory.getInstance();
      this.sipFactory.setPathName("gov.nist");
      this.properties = new Properties();
      this.properties.setProperty("javax.sip.STACK_NAME", "stack");
      this.sipStack = this.sipFactory.createSipStack(this.properties);
      this.messageFactory = this.sipFactory.createMessageFactory();
      this.headerFactory = this.sipFactory.createHeaderFactory();
      this.addressFactory = this.sipFactory.createAddressFactory();
      this.listeningPoint = this.sipStack.createListeningPoint(this.ip, this.port,
this.protocol);
      this.sipProvider = this.sipStack.createSipProvider(this.listeningPoint);
      this.sipProvider.addSipListener(this);

      this.contactAddress = this.addressFactory.createAddress("sip:" + this.ip + ":" +
this.port);

      this.contactHeader                                                    =
this.headerFactory.createContactHeader(contactAddress);

      this.jTextArea.append("Local address: " + this.ip + ":" + this.port + "\n");
    }
    catch(Exception e) {
      JOptionPane.showMessageDialog(this,e.getMessage(),"Error",
JOptionPane.ERROR_MESSAGE);
    }
```

Μέθοδος processRequest:

Όταν κάποιος Server συνδεθεί εκτελείται η συγκεκριμένη μέθοδος

```
  public void processRequest(RequestEvent requestEvent) {
    // Δέχεται το αίτημα
    Request = requestEvent.getRequest();

    this.jTextArea.append("\nRECV " + request.getMethod() + " " +
request.getRequestURI().toString());

    try {
      ServerTransaction transaction = requestEvent.getServerTransaction();
      if(null == transaction) {
        transaction = this.sipProvider.getNewServerTransaction(request);
      }
```

```
        // Συμπληρώνεται ο πίνακας με της πληροφορίες του συνδεδεμένου χρήστη
        this.updateTable(requestEvent, request, transaction);

        // Ανάλογα με το αίτημα που δέχτηκε δίνει και τη κατάλληλη απάντηση
        Response;
        if(request.getMethod().equals("REGISTER")) {

       //Αν το αίτημα είναι Register αποστέλλεται απάντηση με κωδικό 200 OK

           response = this.messageFactory.createResponse(200, request);
           ((ToHeader)response.getHeader("To")).setTag(String.valueOf(this.tag));
           response.addHeader(this.contactHeader);
           transaction.sendResponse(response);
           this.jTextArea.append(" / SENT " + response.getStatusCode() + " " +
response.getReasonPhrase());
        }
        else if(request.getMethod().equals("INVITE")) {

           // αν το αίτημα είναι INVITE ανοίγει παράθυρο αποδοχή  ή μη της κλήσης
και στέλνει μήνυμα αναμονής για απάντηση.

calling cd= new calling();
          cd.setVisible(true);
           response = this.messageFactory.createResponse(180, request);
           ((ToHeader)response.getHeader("To")).setTag(String.valueOf(this.tag));
           response.addHeader(this.contactHeader);
           transaction.sendResponse(response);
           this.jTextArea.append(" / SENT " + response.getStatusCode() + " " +
response.getReasonPhrase());
        }
        else if(request.getMethod().equals("BYE")) {
           // Αν είναι BYE δίνεται αντίστοιχει απάντηση και τερματίζεται το
πρόγραμμα.
           response = this.messageFactory.createResponse(603, request);
           ((ToHeader)response.getHeader("To")).setTag(String.valueOf(this.tag));
           response.addHeader(this.contactHeader);
           transaction.sendResponse(response);
           this.jTextArea.append(" / SENT " + response.getStatusCode() + " " +
response.getReasonPhrase());
          new  mg_device_controler2_1().setVisible(false);
        }
     }
     catch(SipException e) {
        this.jTextArea.append("\nERROR (SIP): " + e.getMessage());
     }
     catch(Exception e) {
        this.jTextArea.append("\nERROR: " + e.getMessage());
     }
  }
```

Μέθοδος updateTable:

Μέθοδος για την συμπλήρωση του πίνακα με βάση τα στοιχεία που έστειλε ο καλλών server

```
private void updateTable(RequestEvent Request, ServerTransaction transaction) {

    DefaultTableModel tableModel = (DefaultTableModel) this.jTable.getModel();
    // δέχεται τα headers
    FromHeader from = (FromHeader)request.getHeader("From");

    ToHeader to = (ToHeader)request.getHeader("To");
    CallIdHeader callId = (CallIdHeader)request.getHeader("Call-Id");
    CSeqHeader cSeq = (CSeqHeader)request.getHeader("CSeq");
    // παίρνει το αίτημα SIP
    Dialog = transaction.getDialog();
    // προσθήκη νέας γραμμής στον πίνακα
    tableModel.addRow(new Object[] {
      (new Date()).toString(),
      request.getRequestURI() != null ? request.getRequestURI().toString() :
"(unknown)",
      from != null ? from.getAddress() : "(unknown)",
      to != null ? to.getAddress() : "(unknown)",
      callId != null ? callId.getCallId() : "(unknown)",
      cSeq != null ? cSeq.getSeqNumber() + " " + cSeq.getMethod() : "(unknown)",
      dialog != null ? dialog.getDialogId() : "",
      transaction.getBranchId(),
      "Request",
      request.getMethod() });
  }
```

Cs_controller1_client.java

Στη περίπτωση που ο χρήστης επιλέξει να πραγματοποίηση κάποια κλήση και εισέλθει ως caller στο σύστημα εκτελείται η συγκεκριμένη κλάση. Στην οποία αρχικά θα αναζητήσει του ενεργούς χρήστες που μπορεί να καλέσει και στην συνέχεια θα κάνει μια σειρά ενεργειών προκειμένου να ξεκινήσει η κλήση

Μέθοδος onOpen:

Η συγκεκριμένη μέθοδος εκτελείται κατά το άνοιγμα της κλάσης

```
private void onOpen(java.awt.event.WindowEvent evt)
{
    buttonRegisterStateless.setEnabled(true);
    buttonRegisterStatefull.setEnabled(true);
```

```
      buttonInvite.setEnabled(true);
      buttonBye.setEnabled(true);

       try {
    //Δέχεται την τοπική IP.
    this.ip = InetAddress.getLocalHost().getHostAddress();

    this.sipFactory = SipFactory.getInstance();
    this.sipFactory.setPathName("gov.nist");
    // Δημιουργούνται τα δεδομένα που θα αποσταλούν
    this.properties = new Properties();
    this.properties.setProperty("javax.sip.STACK_NAME", "stack");

    this.sipStack = this.sipFactory.createSipStack(this.properties);
    this.messageFactory = this.sipFactory.createMessageFactory();
    this.headerFactory = this.sipFactory.createHeaderFactory();
    this.addressFactory = this.sipFactory.createAddressFactory();
    //Δημιουργείται το σημείο ακρόασης SIP καθώς και δεσμεύονται η IP ,το port και
το πρωτόκολλο
    this.listeningPoint=this.sipStack.createListeningPoint(this.ip,this.port,
this.protocol);
    this.sipProvider = this.sipStack.createSipProvider(this.listeningPoint);
    // Καταχωρούμε την κλάση ως SIP listener
    this.sipProvider.addSipListener(this);
    // Δημιουργείται η διεύθυνση επικοινωνίας
    this.contactAddress = this.addressFactory.createAddress("sip:" + this.ip + ":" +
this.port);
    // Δημιουργεί την κεφαλίδα για όλα τα SIP μηνύματα
    this.contactHeader = this.headerFactory.createContactHeader(contactAddress);
catch(Exception e) {
    // αν βρεθεί κάποιο λάθος κλείνει το σύστημα

    JOptionPane.showMessageDialog(this,e.getMessage(),"Error",
JOptionPane.ERROR_MESSAGE);
    System.exit(-1);
}
  ……..
```

Μέθοδος onRegisterStateless:

Αυτή η μέθοδος καλείται όταν πατηθεί το κουμπί για αίτημα REGISTER

```
    private void onRegisterStateless(java.awt.event.ActionEvent evt)
{
      try {
     Address  addressTo=this.addressFactory.createAddress(this.jTextField1.getText());
    javax.sip.address.URI requestURI = addressTo.getURI();
    ArrayList viaHeaders = new ArrayList();
    ViaHeader = this.headerFactory.createViaHeader(this.ip, this.port, "udp", null);
    viaHeaders.add(viaHeader);
```

```
    MaxForwardsHeader
  maxForwardsHeader=this.headerFactory.createMaxForwardsHeader(70);
    CallIdHeader = this.sipProvider.getNewCallId();
    CSeqHeader = this.headerFactory.createCSeqHeader(1L,"REGISTER");
  FromHeaderfromHeader=this.headerFactory.createFromHeader(this.contactAddress
  , String.valueOf(this.tag));
    ToHeader toHeader = this.headerFactory.createToHeader(addressTo, null);

    Request request = this.messageFactory.createRequest(
      requestURI,
      "REGISTER",
      callIdHeader,
      cSeqHeader,
      fromHeader,
      toHeader,
      viaHeaders,
      maxForwardsHeader);
    // Προσθήκη της κεφαλίδας επαφής
    request.addHeader(contactHeader);
    // Αποστολή αιτήματος μέσω SIP
    this.sipProvider.sendRequest(request);

    // Εμφάνιση μηνύματος στο παράθυρο γεγονότων
    this.textArea.append(
      "Request sent:\n" + request.toString() + "\n\n");
  }
  catch(Exception e) {
    // μήνυμα σε περίπτωση σφάλματος
    this.textArea.append("Request sent failed: " + e.getMessage() + "\n");
```

Μέθοδος onRegisterStatefull:

Η μέθοδος αυτή καλείται όταν ο χρήστης πατήσει το κουμπί "Reg (SL)" .

```
    private void onRegisterStatefull(java.awt.event.ActionEvent evt)
      try {
    // Παίρνει την διεύθυνση του καλούμενου
    Address addressTo = this.addressFactory.createAddress(this.jTextField1.getText());
    // Δημιουργείται το αίτημα για το μήνυμα SIP
    javax.sip.address.URI requestURI = addressTo.getURI();
    // Δημιουργία των headers
   ArrayList viaHeaders = new ArrayList();

ViaHeader viaHeader = this.headerFactory.createViaHeader(this.ip, this.port, "udp", null);

    viaHeaders.add(viaHeader);

    MaxForwardsHeadermaxForwardsHeader=
  this.headerFactory.createMaxForwardsHeader(70);

    CallIdHeader callIdHeader = this.sipProvider.getNewCallId();
    CSeqHeader cSeqHeader = this.headerFactory.createCSeqHeader(1L,"REGISTER");
```

```
FromHeaderfromHeader=this.headerFactory.createFromHeader(this.contactAddress
, String.valueOf(this.tag));
   ToHeader toHeader = this.headerFactory.createToHeader(addressTo, null);

   // Δημιουργία  request.
   Request request = this.messageFactory.createRequest(
      requestURI,
      "REGISTER",
      callIdHeader,
      cSeqHeader,
      fromHeader,
      toHeader,
      viaHeaders,
      maxForwardsHeader);
   // Προσθήκη του contact header
   request.addHeader(contactHeader);

   // αποστολή αιτήματος
   this.sipProvider.sendRequest(request);
    ClientTransactiontransaction =this.sipProvider.getNewClientTransaction(request);
    transaction.sendRequest();
   this.textArea.append(
      "Request sent:\n" + request.toString() + "\n\n");
}
catch(Exception e) {
   // μήνυμα σε περίπτωση αδυναμίας αποστολής
   this.textArea.append("Request sent failed: " + e.getMessage() + "\n");
}
```

Μέθοδος onInvite:

Η συγκεκριμένη μέθοδος καλείται όταν πατήσουμε το κουμπί invite για να πραγματοποιηθεί μια κλήση

```
   private void onInvite(java.awt.event.ActionEvent evt)
     try {
   // λαμβάνει την διεύθυνση αποστολής
   Address addressTo = this.addressFactory.createAddress(this.jTextField1.getText());
   // δημιουργέιται το αίτημα που θα σταλεί μέσω του SIP
   javax.sip.address.URI requestURI = addressTo.getURI();
   // Δημιουργία των κεφαλίδων
   ArrayList viaHeaders = new ArrayList();
   ViaHeader viaHeader = this.headerFactory.createViaHeader(this.ip, this.port,
"udp", null);
   viaHeaders.add(viaHeader);
   MaxForwardsHeadermaxForwardsHeader=
this.headerFactory.createMaxForwardsHeader(70);
   CallIdHeader callIdHeader = this.sipProvider.getNewCallId();
   CSeqHeader cSeqHeader = this.headerFactory.createCSeqHeader(1L,"INVITE");
```

```
FromHeaderfromHeader=this.headerFactory.createFromHeader(this.contactAddress
, String.valueOf(this.tag));
   ToHeader toHeader = this.headerFactory.createToHeader(addressTo, null);
   // Δημιουργία μηνύματος
   Request request = this.messageFactory.createRequest(
      requestURI,
      "INVITE",
      callIdHeader,
      cSeqHeader,
      fromHeader,
      toHeader,
      viaHeaders,
      maxForwardsHeader);
   request.addHeader(contactHeader);


    ClientTransactiontransaction                                                =
this.sipProvider.getNewClientTransaction(request);
    transaction.sendRequest();
   this.textArea.append(
      "Request sent:\n" + request.toString() + "\n\n");
}

catch(Exception e) {
   // σε περίπτωση σφάλματος εμφανίζεται το παρακάτω μήνυμα
   this.textArea.append("Request sent failed: " + e.getMessage() + "\n");
```

Μέθοδος onBye:

Η συγκεκριμένη μέθοδος καλείται όταν ολοκληρωθεί η κλήση και πατήσουμε το κουμπί Bye

```
   private void onBye(java.awt.event.ActionEvent evt)
      try {
   // δέχεται την διεύθυνση προορισμού
   Address addressTo = this.addressFactory.createAddress(this.jTextField1.getText());
   // δημιουργείται το URI Request του μηνύματος
   javax.sip.address.URI requestURI = addressTo.getURI();
   // δημιουργία των headers
  ArrayList viaHeaders = new ArrayList();

 ViaHeader viaHeader = this.headerFactory.createViaHeader(this.ip, this.port, "udp", null);

viaHeaders.add(viaHeader);

   MaxForwardsHeadermaxForwardsHeader=
this.headerFactory.createMaxForwardsHeader(70);
   CallIdHeader callIdHeader = this.sipProvider.getNewCallId();
   CSeqHeader cSeqHeader = this.headerFactory.createCSeqHeader(1L,"BYE");

FromHeaderfromHeader=this.headerFactory.createFromHeader(this.contactAddress
, String.valueOf(this.tag));
```

```
    ToHeader toHeader = this.headerFactory.createToHeader(addressTo, null);

    // Δημιουργία αιτήματος
    Request request = this.messageFactory.createRequest(
      requestURI,
      "BYE",
      callIdHeader,
      cSeqHeader,
      fromHeader,
      toHeader,
      viaHeaders,
      maxForwardsHeader);
    request.addHeader(contactHeader);


    // δημιουργία νεάς επαφης μέσω SIP.
     ClientTransactiontransaction= this.sipProvider.getNewClientTransaction(request);
// αποστολή μηνύματος
     transaction.sendRequest();
    // Εμφάνιση του μηνύματος στο παράθυρο του γραφικού περιβάλλοντος
    this.textArea.append(
      "Request sent:\n" + request.toString() + "\n\n");
}
catch(Exception e) {
    // σε περίπτωση σφάλματος
    this.textArea.append("Request sent failed: " + e.getMessage() + "\n");
}
```

Μέθοδος search:

Καλείται όταν πατήσουμε το κουμπί search προκειμένου να εμφανιστούν οι συνδεδεμένοι χρήστες για την έναρξη μιας κλήσης

```
     InetAddress localhost = null;
     try {
       localhost = InetAddress.getLocalHost();
     } catch (UnknownHostException ex) {
       Logger.getLogger(cs_controler2_c.class.getName()).log(Level.SEVERE, null, ex);
     }
 // IPv4 usage
byte[] ip = localhost.getAddress();// επαναλαμβανόμενος έλεγχος του τελευταίου σκέλους της ip εκτελώντας ping
    while (true)  {
          for (int i = 1; i <= 10; i++)
        {

            ip[3] = (byte)i;
            InetAddress address = null;

              try {
```

```
                address = InetAddress.getByAddress(ip);

            } catch (UnknownHostException ex) {

Logger.getLogger(cs_controler2_c.class.getName()).log(Level.SEVERE,null, ex);
            }


            try {
              if (address.isReachable(1000))//σε περίτπωση που απαντήσει
εμφανίζεται η ip και η φράση Pinging…pinging
              {

                 System.out.println(address + " - Pinging... Pinging");
                 DefaultTableModelmodel = (DefaultTableModel) jTable1.getModel();
                  model.addRow(new Object[]{address});
```

Mg_resource_controller1_recorder.java

Η συγκεκριμένη κλάση καλείται για την κωδικοποίηση της φωνής έπειτα από την καταγραφή μέσω του μικροφώνου και γίνεται αποστολή των πακέτων μέσω socket.

Μέθοδος run:

Είναι η βασική μέθοδος της κλάσης η οποία δέχεται την είσοδο των πακέτων φωνής μέσω του μικροφώνου και στην αποστέλλει στην κληθέντα χρήστη. Επίσης παράλληλα αποστέλλονται τόσο τα bytes που έχουν σταλεί όσο και ο χρόνος αποστολής

```
private static final int MAX_TIMEOUT = 1;

  public TargetDataLine audio_in = null;

  public DatagramSocket dout; //dhmiourgia datagramm socket send

  byte byte_buff[] = new byte[512];//dhmiourgia buffer gia paketa pros apostoli

  public InetAddress server_ip;//ip tou apodekti server apo to mg_device controller

  public int server_port;//port tou apodekti server

  public void run(){

    int i = 0;

    while(mg_event_contr1.recording){

      Date now = new Date();
```

```
long msSend = now.getTime()
try {
    dout.setSoTimeout(MAX_TIMEOUT);
    audio_in.read(byte_buff, 0, byte_buff.length);//anakta ta dedomena apo tin suskeui eisodou
    long SendTime = System.currentTimeMillis();
    DatagramPacketdata = new DatagramPacket(byte_buff, byte_buff.length,server_ip, server_port);//dhmiourgei to datagram packet pros apostoli
    System.out.println("send #"+i++)
    dout.send(data);//apostelei mesw tou datagramm socket to datagramm packet
    now = new Date();
   long msReceived = now.getTime() //χρόνος εισαγωγή  στο πακέτο
   long delayTime = msReceived - msSen;// εκτύπωση πακέτου και καθυστέρησης
 System.out.println(  "Received from " + data.getAddress().getHostAddress() +  ": "  + " Delay: " + delayTime );
 } catch (IOException ex) {
    System.out.println("Timeout for packet " + i);
 } catch (Exception ex) {
    Logger.getLogger(recorder_thread.class.getName()).log(Level.SEVERE, null, ex);
 }
audio_in.close();
audio_in.drain();
System.out.println("Thread stop");
```

Mg_resource_controller1_player.java

Η συγκεκριμένη κλάση καλείται για την ανάγνωση και αποκωδικοποίηση των δεδομένων φωνής προκειμένου να αναπαραχθεί μέσω των ηχείων

Μέθοδος run:

Η μέθοδος αυτή δέχεται τα πακέτα φωνής από τον χρήστη με τον οποίο πραγματοποιείται η κλήση ,τα αποκωδικοποιεί και τα αναπαράγει. Όσο εκτελείται αυτή η μέθοδος παράλληλα δέχεται από την από τον άλλο χρήστη τόσο τα bytes που έχει αποστείλει ώστε να τα συγκρίνει με όσα έχει λάβει και να καθοριστεί το packet_loss όσο και ο χρόνος αποστολής ώστε να υπολογιστεί η καθυστέρηση.

```
int time_receive;//xronos lipsis
```

```
int time =0;

int i = 0;//paketa pou pirame

int j = 0;//paketa pou xathikan

int l =0; // gia to jitter pinakas

int bytes=0;// pragmatika bytes tou buffer

int total_delay=0;

int delay=0;//xronos pou mesolavei apo apostoli mexri lipsi

int time_save=0;

int jitter_int= 0;

DatagramPacket incoming_time=new DatagramPacket(buff, buff.length);//afiksi paketwn hxou

DatagramPacket incoming = new DatagramPacket(jitter_buffer, jitter_buffer.length);//afiksi xronou

DatagramPacket incoming_bytes = new DatagramPacket(bytes send, bytes_send.length);//afiksi xronou

int total_bytes=0;

int bytes_lost=0;

int max_jitter = 0;

 int[] array = new int[800000];

////////////////////////////////////////////////////////////////////////////////////////////

while (mg_event_contr1.calling) {


   LocalDateTime now = LocalDateTime.now();//xronos lipsis se 16diko gia upologismo delay

  // System.out.println("Time received: "+dtf.format(now) );//xronos lipsis me pragmatiki morfi

   ////////////////////////////////////////////////////////////////////

   try {

     din.receive(incoming_time);//mesw tou socket lamvanoume to datagram packet apo ton client

     buff = incoming_time.getData();//eisagwgi sto buffer twn paketwn apo to datagram packet

     for (byte b:jitter_buffer) {
```

```
            if (b != 0)
            {
               bytes=(bytes+1);
            }
          }
          total_bytes=Math.abs( total_bytes+bytes);
           System.out.println("total bytes received until now "+total_bytes);
        ////////////////////////////////////////////////////
           din.receive(incoming);//mesw tou socket lamvanoume to datagram packet apo ton client
           jitter_buffer = incoming.getData();//eisagwgi sto buffer twn paketwn apo to datagram packet
           ByteBuffer serverResponseBuffer = ByteBuffer .wrap(buff);//lamvanei ta bytes tou buffer
           int clientsend = (int)serverResponseBuffer.getLong();//metatropi bytes se long
      ////////////////////////////////////////////////////////////////////////
           din.receive(incoming_bytes);//mesw tou socket lamvanoume to datagram packet apo ton client
           bytes send = incoming_bytes.getData();//eisagwgi sto buffer twn paketwn apo to datagram packet
           ByteBuffer ResponseBuffer = ByteBuffer .wrap(bytes send);//lamvanei ta bytes tou buffer
           int client_bytes = Math.abs((int)ResponseBuffer.getLong());//metatropi bytes se long

      ////////////////////////////////////////////////////////////////////////
           audio_out.write(jitter_buffer, 0, jitter_buffer.length);//eisagei ta dedomena stin suskeui eksodo
       ////////////////////////////////////////////////////////////////////////////////
             time_receive= Math.abs((int) System.currentTimeMillis());
             time = Math.abs((time_receive - clientsend));
             JITTER =time-time_save;
             if (JITTER <0 ){
              JITTER= - JITTER;
```

```
                }
                l=l+1;
                array[l] = JITTER ;
                if (array[l]> array[l-1]){
                 max_jitter = array[l];
                 System.out.println("max jitter"+ max_jitter);
                }
                jitter_int=jitter_int+JITTER;
               // System.out.println("jitter"+ JITTER);
                time_save= Math.abs(time);
               // System.out.println("time"+ time);
                 total_delay=Math.abs(total_delay+time);
                System.out.println("#" +k + " Delay from previous packet :  " + time+"   bytes send from client: "+client_bytes+ "jitter "+ JITTER);
                k= k+1;

 ////////////////////////////////////////////////////////////////////////////////////////////
                bytes_lost = Math.abs(client_bytes - total_bytes);
                System.out.println("bytes lost   " + bytes_lost);
                double packet_loss = Math.abs(((100*bytes_lost)/client_bytes)/100);

               // System.out.println("   Current packet loss."+ packet_loss);
                  if (packet_loss > LOSS_RATE) {// an to xasimo twn bytes sto paketo twn 512 bytes einai megalutero tou 30 % upologizoume ta paketa pou xathikan
                   System.out.println("   Reply not sent."+ packet_loss);
                   j++;
                  }
          if (JITTER >= AVERAGE_MAX_JITTER) {
                  jitter_buffer = new byte[2048];
                try {
                try {
                  Thread.sleep((int) (packet_loss * 2 * AVERAGE_DELAY));
```

```
            } catch (InterruptedException ex) {
              Logger.getLogger(mg_resource_contr1.class.getName()).log(Level.SEVERE, null,
ex);
            }
          }
          else{

          try {
            Thread.sleep((int) (packet_loss * 2 * AVERAGE_DELAY));
            } catch (InterruptedException ex) {
              Logger.getLogger(mg_resource_contr1.class.getName()).log(Level.SEVERE, null,
ex);
            }
    ////////////////////////////////////////////////////////////////////////////////
```

Mg_device_controller1.java

Η συγκεκριμένη κλάση αποτελεί το βασικό γραφικό περιβάλλον για την επικοινωνία των δυο επαφών και δίνει την δυνατότητα τόσο καταγραφής όσο και αναπαραγωγής φωνής. Περιλαμβάνει δυο βασικά κουμπιά το Start με το οποίο οι λογικές μεταβλητές recording και playing του mg_event_controller ενεργοποιούνται και μπορεί να γίνει καταγραφή και αναπαραγωγή φωνής και το Stop με το οποίο θέτονται ως false και σταματάει τόσο η αναπαραγωγή όσο και η καταγραφή φωνής. Πατώντας και από τις δυο πλευρές το κουμπί Start επιτυγχάνεται η συγχρονισμένη καταγραφή και αναπαραγωγή φωνής. Σε αυτή την κλάση έχουμε δυο βασικές μεθόδους:

Μέθοδος rec_audio:

Η μέθοδος αυτή χρησιμοποιείται για την καταγραφή φωνής θέτοντας την τιμή της μεταβλητής calling ως true προκειμένου καλώντας αργότερα την μέθοδο rec του mg_resource_contr2_recorder να ξεκινήσει η κωδικοποίηση της φωνής. Του δίνει δηλαδή τις πληροφορίες σχετικά με την διεύθυνση αποστολής ,το port ακρόασης καθώς και το μέσο καταγραφής της φωνής τα οποία απαιτούνται από τον constructor Datagram Packet.

```
        public void rec_audio(){
            try {
                AudioFormat format = getaudioformat();//περιλαμβάνει τις πληροφοριες
      σχετικά με την διεύθυνση και το port
                DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);
                if(!AudioSystem.isLineSupported(info)){
```

```
            System.out.println("not suport");
            System.exit(0);
        }
        audio_in = (TargetDataLine) AudioSystem.getLine(info);
        audio_in.open(format);
        audio_in.start();
        mg_resource_contr2 r = new mg_resource_contr2();
        InetAddress inet = InetAddress.getByName(address);
        r.audio_in = audio_in;
        r.dout = new DatagramSocket();
        r.server_ip = inet;
        r.server_port = port;
        mg_event_contr2.calling = true;
        r.start();

    } catch (LineUnavailableException | UnknownHostException | SocketException
ex) {

Logger.getLogger(mg_device_controler2_1.class.getName()).log(Level.SEVERE,   null,
ex);
    }
  }

////////////////////////////////////////////////////////////////////////////////
//
```

Μέθοδος play_audio:

Αντίστοιχα με πιο πάνω η μέθοδος αυτή χρησιμοποιείται για την αναπαραγωγή φωνής θέτοντας την τιμή της μεταβλητής playing ως true προκειμένου καλώντας αργότερα την μέθοδο rec του mg_resource_contr2_player να ξεκινήσει η αποκωδικοποίηση της φωνής. Και σε αυτήν την περίπτωση προσδιορίζονται η διεύθυνση και το port από το οποίο δέχεται τη φωνή όπως επίσης και το μέσω αναπαραγωγής (δηλαδή τα ηχεία) τα οποία θα χρησιμοποιηθούν από την μέθοδο mg_resource_controller_player

```
    public void play_audio() throws UnknownHostException{
    try {
        AudioFormat format = getaudioformat2();
        DataLine.Info info_out = new DataLine.Info(SourceDataLine.class, format);
        if(!AudioSystem.isLineSupported(info_out)){
            System.out.println("not suport");
            System.exit(0);
        }
        audio_out = (SourceDataLine)AudioSystem.getLine(info_out);
        audio_out.open(format);
        audio_out.start();
        player_thread p = new player_thread();
```

```
            InetAddress inet = InetAddress.getByName(address);
            p.din = new DatagramSocket(port_server);
            p.audio_out = audio_out;

            mg_event_contr2.playing = true;
            p.start();

        } catch (LineUnavailableException | SocketException ex) {

    Logger.getLogger(mg_device_controler2_1.class.getName()).log(Level.SEVERE,  null,
    ex)
```

Mg_event_controller1.java

Η συγκεκριμένη μέθοδος είναι εκείνη μέσω της οποίας αρχικοποιούνται οι λογικές μεταβλητές αναπαραγωγής και καταγραφής φωνής που θα χρησιμοποιηθούν από το mg_device_controller

Μέθοδος main

Η συγκεκριμένη μέθοδος εκκινεί το mg_device_controler για την έναρξη της ανταλλαγής φωνής

```
public static void main(String[] args) {
    mg_device_controler2_2 fr = new mg_device_controler2_2();
    fr.setVisible(true);
}
```

Παρακάτω θα δούμε με χαρακτηριστικές οθόνες την αλληλοσυσχέτιση των παραπάνω μεθόδων για την εκτέλεση του προγράμματος

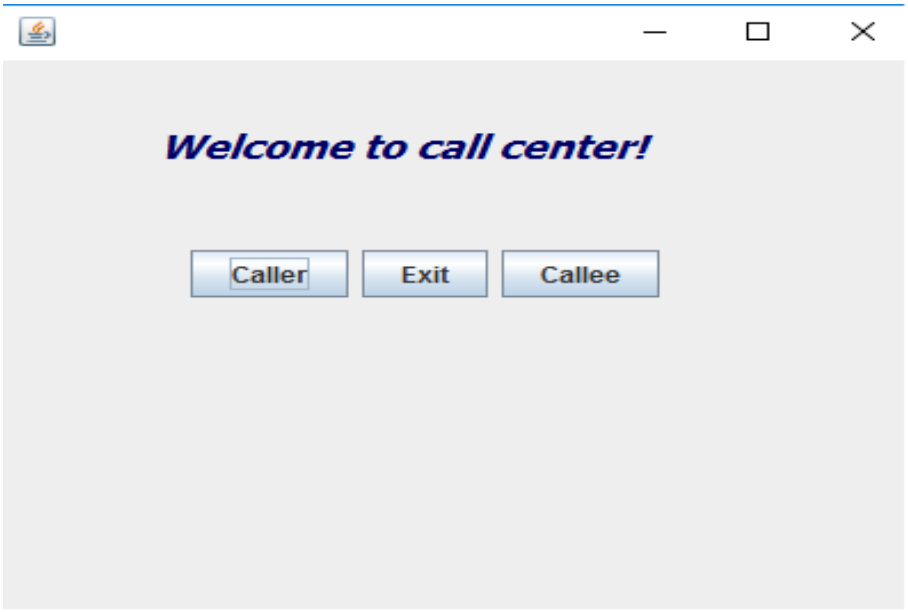
4.4 Σενάριο υλοποίησης

Σε αυτό το σημείο αφού έγινε αναλυτική περιγραφή του συστήματος και των μεθόδων που εκτελούνται θα δούμε ένα χαρακτηριστικό παράδειγμα εκτέλεσης του βήμα με τις αντίστοιχες οθόνες για να γίνει πλήρως κατανοητή λειτουργία του.

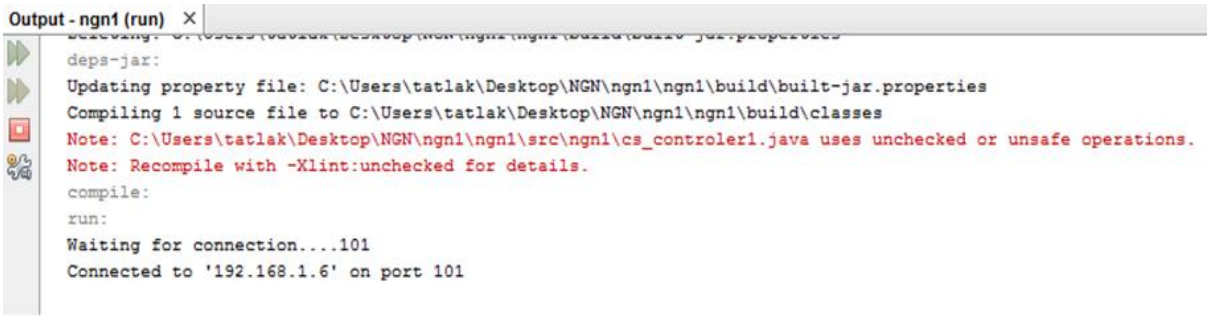
Για να εκτελέσει κάποιος τοπικά το πρόγραμμα στον υπολογιστή του αρκεί απλά να έχει εγκατεστημένο στο Java Runtime Environment ,και να διαθέτει μια συσκευή αναπαραγωγής ήχου και ένα μικρόφωνο. Στην συνέχεια με ένα απλό διπλό click στο εικονίδιο της εφαρμογής στην επιφάνεια εργασίας μπορεί να εκτελέσει το πρόγραμμα. Εμείς παρακάτω προκειμένου να ελέγξουμε αναλυτικότερα την κλιμάκωση της εκτέλεσης του προγράμματος θα προχωρήσουμε στην εκτέλεση και ανάλυση του μέσω του NetBeans στο οποίο έγινε και η συγγραφή του κώδικα.
Αυτό επίσης θα μας βοηθήσει κυρίως ώστε να εξακριβώσουμε και την σωστή και μεταφορά αλλά και ανάγνωση πακέτων φωνής πράγμα το οποίο δεν μπορεί να φανεί στην απλή τοπική εκτέλεση του προγράμματος
Ακολουθεί περιγραφή ενός σεναρίου χρήσης με τις αντίστοιχες οθόνες

Έτσι λοιπόν όταν κάποιος χρήστης αρχίσει να εκτελεί το πρόγραμμα αρχικά θα εμφανιστεί μπροστά του η παρακάτω οθόνη που του δίνει τις παρακάτω επιλογές:

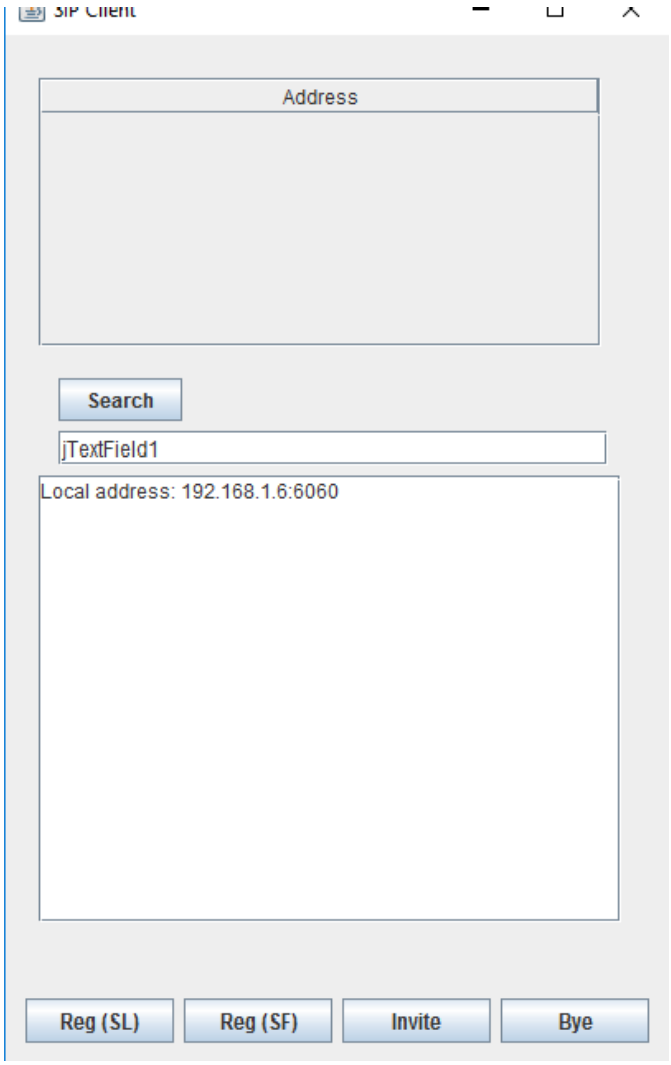
- Να εισέλθει ως caller,δηλαδή να εισέλθει προκειμένου να καλέσει κάποιον άλλον που βρίσκεται στο δίκτυο
- Να εισέλθει ως callee,ώστε να μπορεί να δεχτεί κάποια κλήση
- Ή με την επιλογή exit να κλείσει το πρόγραμμα



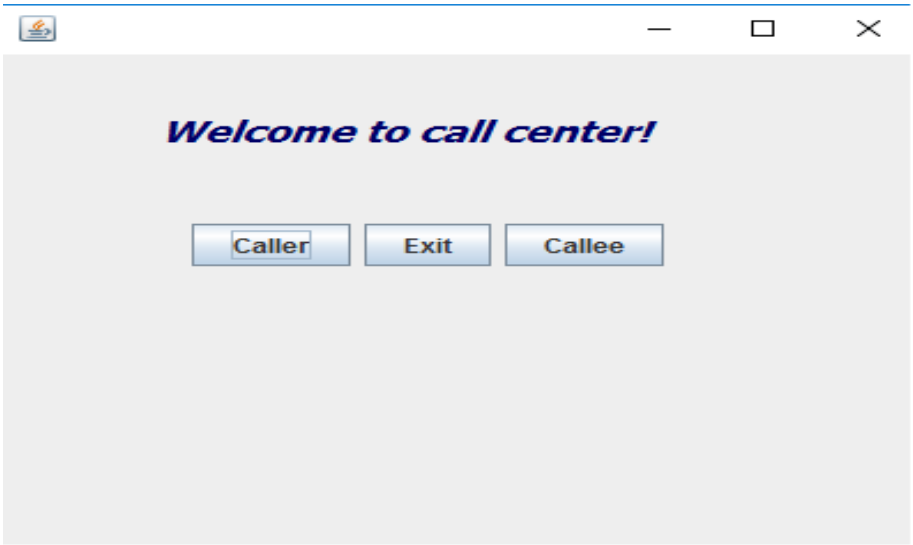
Με την είσοδο στο σύστημα ταυτόχρονα εκτελούνται τόσο το mg_controller.java όσο και το mg_conf_controller για την σύνδεση των Media Gateway και Control Server μέσω του socket ακρόασης που έχει ανοίξει, σύνδεση η οποία φαίνεται στον πίνακα εξόδου στο πρόγραμμα Netbeans όπως χαρακτηριστικά φαίνεται στην πιο κάτω εικόνα



Σε περίπτωση που η επιλογή του χρήστη είναι η είσοδος ως Caller εμφανίζεται το παρακάτω παράθυρο το οποίο συμπεριλαμβάνει ένα panel εμφάνισης των διαθέσιμων διευθύνσεων για κλήση, την επιλογή search προκειμένου να γίνει αναζήτηση για διαθέσιμους χρήστες στο δίκτυο, και τις επιλογές για αιτήματα Register(Stateless και Statefully) όπως είδαμε και πιο πάνω,Invite για έναρξη της κλήσης και Bye για αποχώρηση από την συνομιλία.

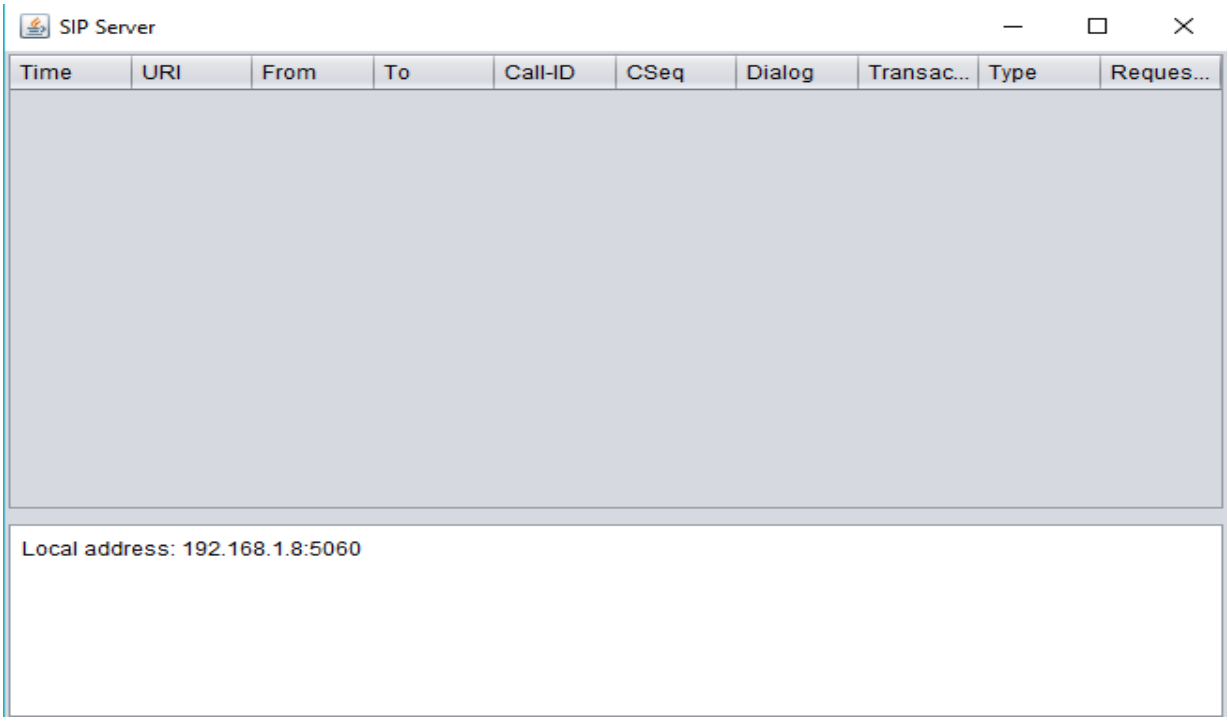


Προκειμένου να δούμε το παράδειγμα εκτέλεσης συνολικά εκτελέσαμε και σε ένα διαφορετικό υπολογιστή το πρόγραμμα έτσι αρχικά εμφανίζεται και πάλι η κλασική αρχική οθόνη

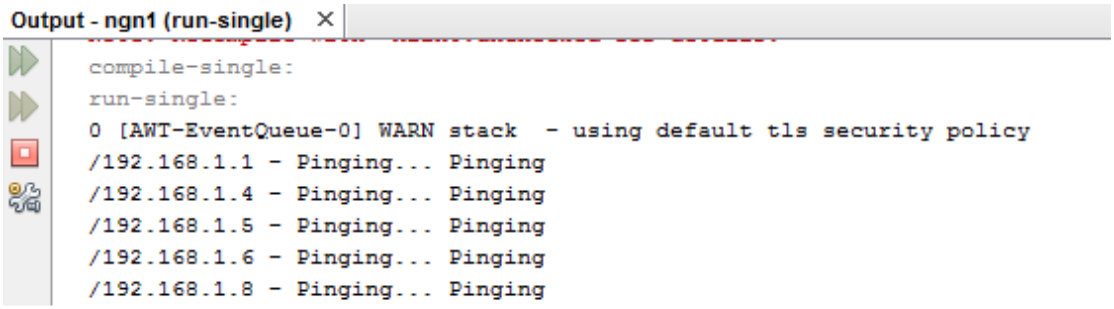


Αντίστοιχα και σε αυτή την περίπτωση έχουμε την σύνδεση Media Gateway με Control Server όπως φαίνεται παρακάτω

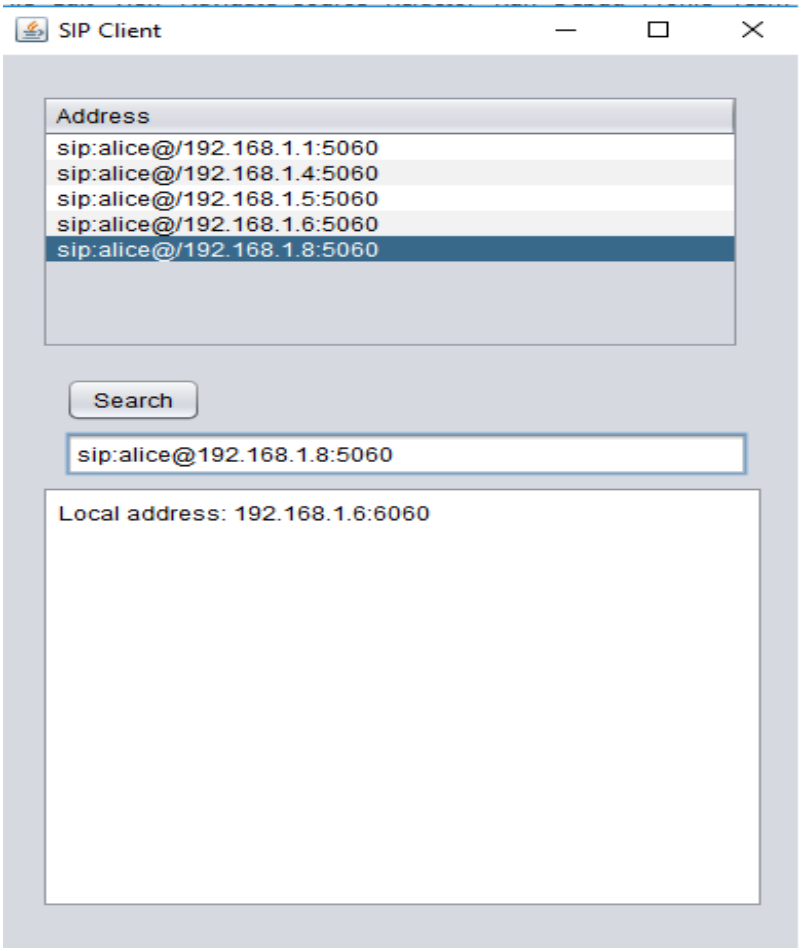
Σε αυτή την περίπτωση όμως επιλέξαμε να εισέλθουμε ως callee και έχουμε μπροστά μας το συγκεκριμένο περιβάλλον το οποίο μόλις δεχθεί κάποιο αίτημα όπως θα δούμε συμπληρώνεται ο πίνακας που διαθέτει



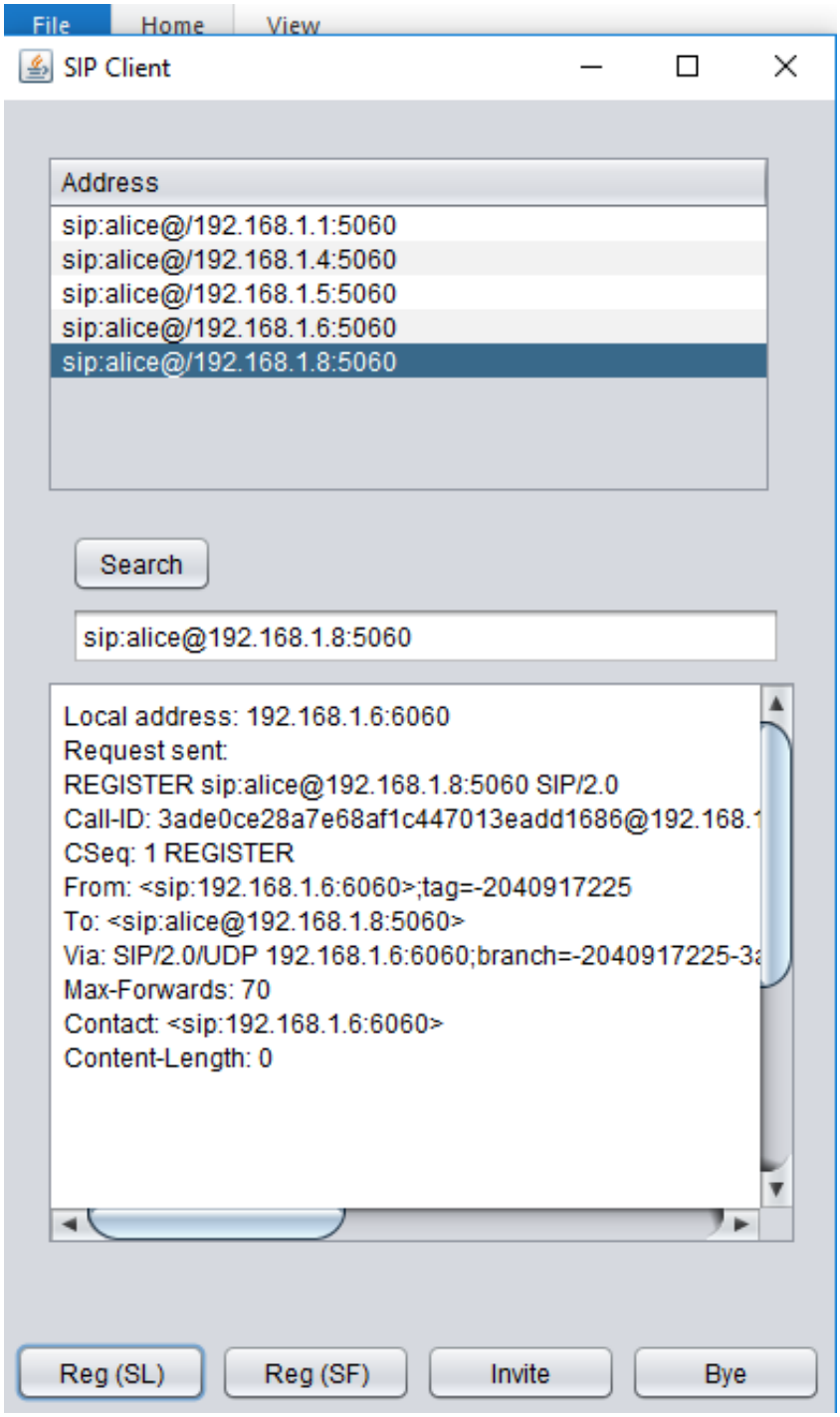
Επιστρέφοντας τώρα στον υπολογιστή ο οποίος θέλει να πραγματοποιήσει κάποια κλήση. Το πρώτο βήμα που θα κάνει είναι να πραγματοποιήσει μια αναζήτηση πατώντας το κουμπί Search προκειμένου να ελέγξει ποιοι υπολογιστές στο δίκτυο είναι διαθέσιμοι για κλήση. Εδώ φαίνεται μια ενδεικτική εκτέλεση όπως φαίνεται στο NetBeans με τις ενεργές διευθύνσεις που έχουν βρεθεί



Τώρα μετά το πέρας την αναζήτησης ο πίνακας διευθύνσεων συμπληρώνεται και προκύπτει η παρακάτω εικόνα

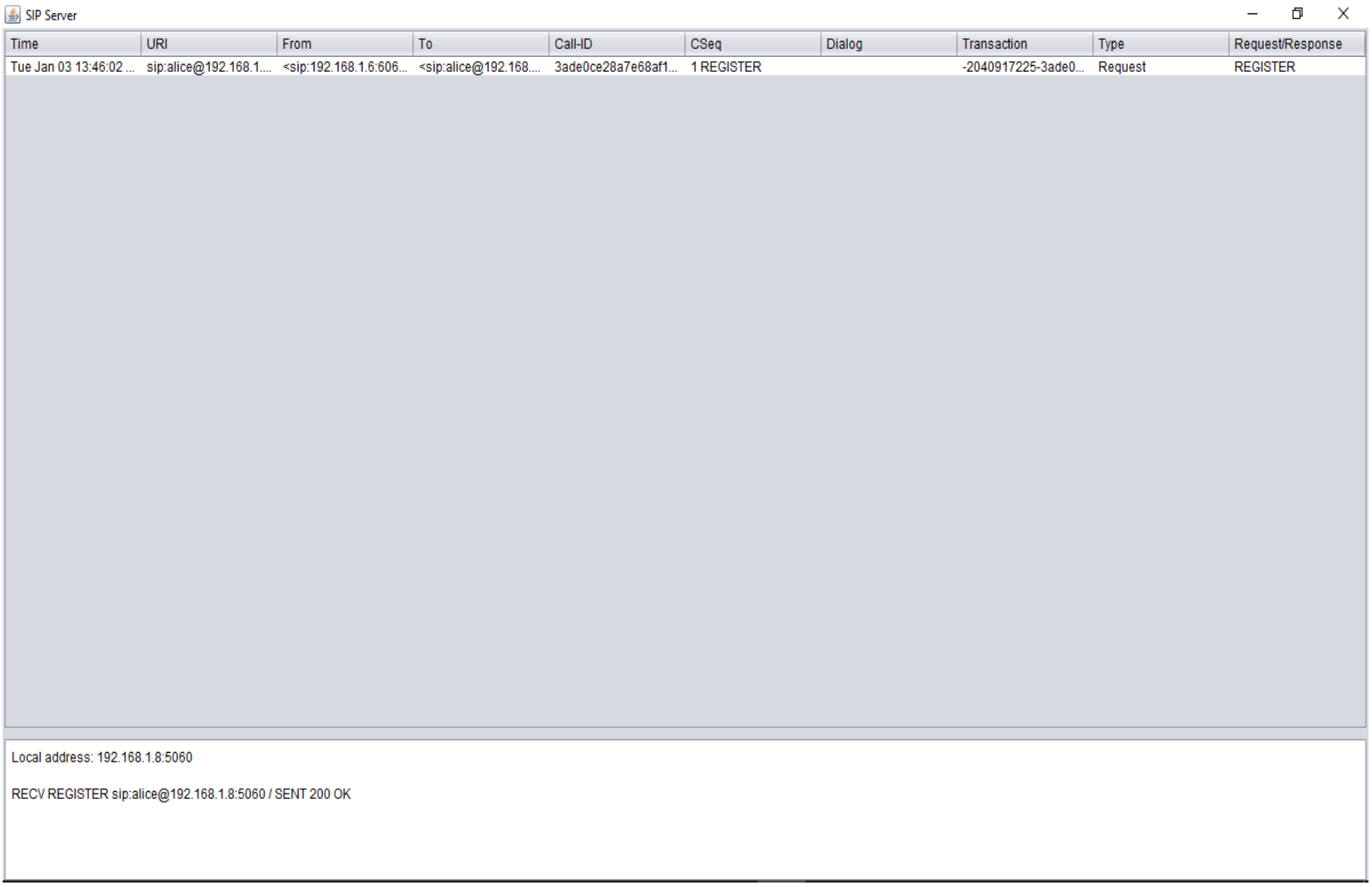


Όπως φαίνεται επιλέξαμε την διεύθυνση του υπολογιστή στον οποίον έχουμε εισέλθει ως callee

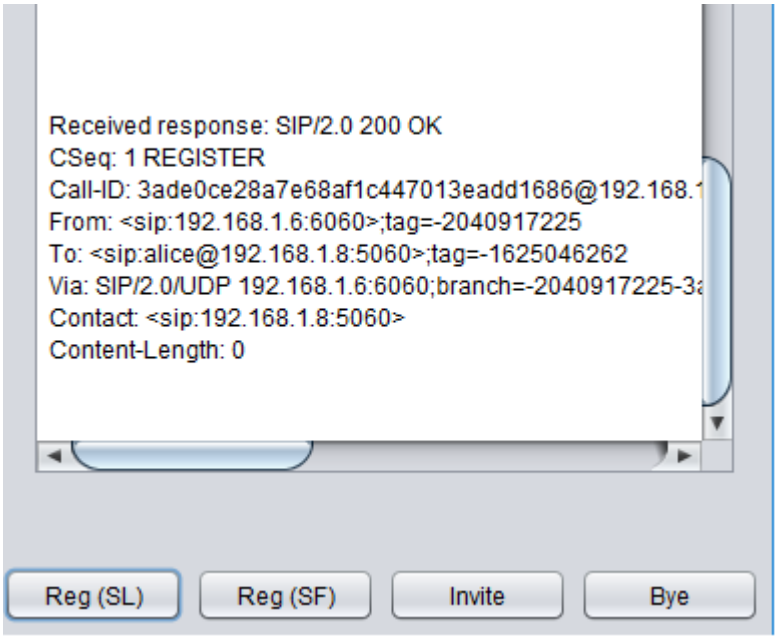


Σε αυτή την φάση θα αποστείλουμε στον δεύτερο υπολογιστή ένα αίτημα Register πατώντας το κουμπί Reg(SL).Η σύνταξη του αιτήματος φαίνεται στο κατώτερο πλαίσιο του παραθύρου της παραπάνω εικόνας

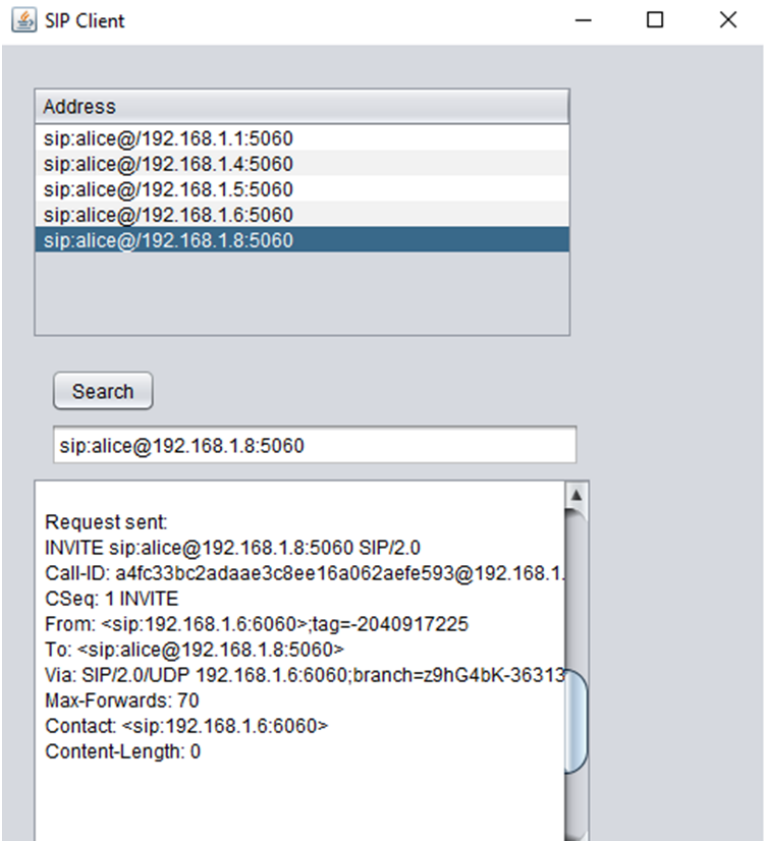
Με την αποστολή του αιτήματος συμπληρώνεται ο πίνακας με τα στοιχεία του δηλαδή, τον αποστολέα ,τον αποδέκτη, την μέρα και ώρα αποστολής καθώς και το είδος του αιτήματος ενώ αποστέλλει και μια απάντηση αποδοχής όπως φαίνεται στο κάτω μέρος της εικόνας



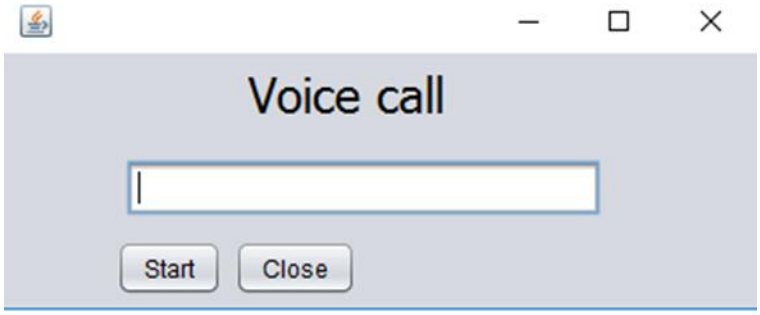
Η συγκεκριμένη απάντηση εμφανίζεται και στον Caller και έχει την παρακάτω μορφή



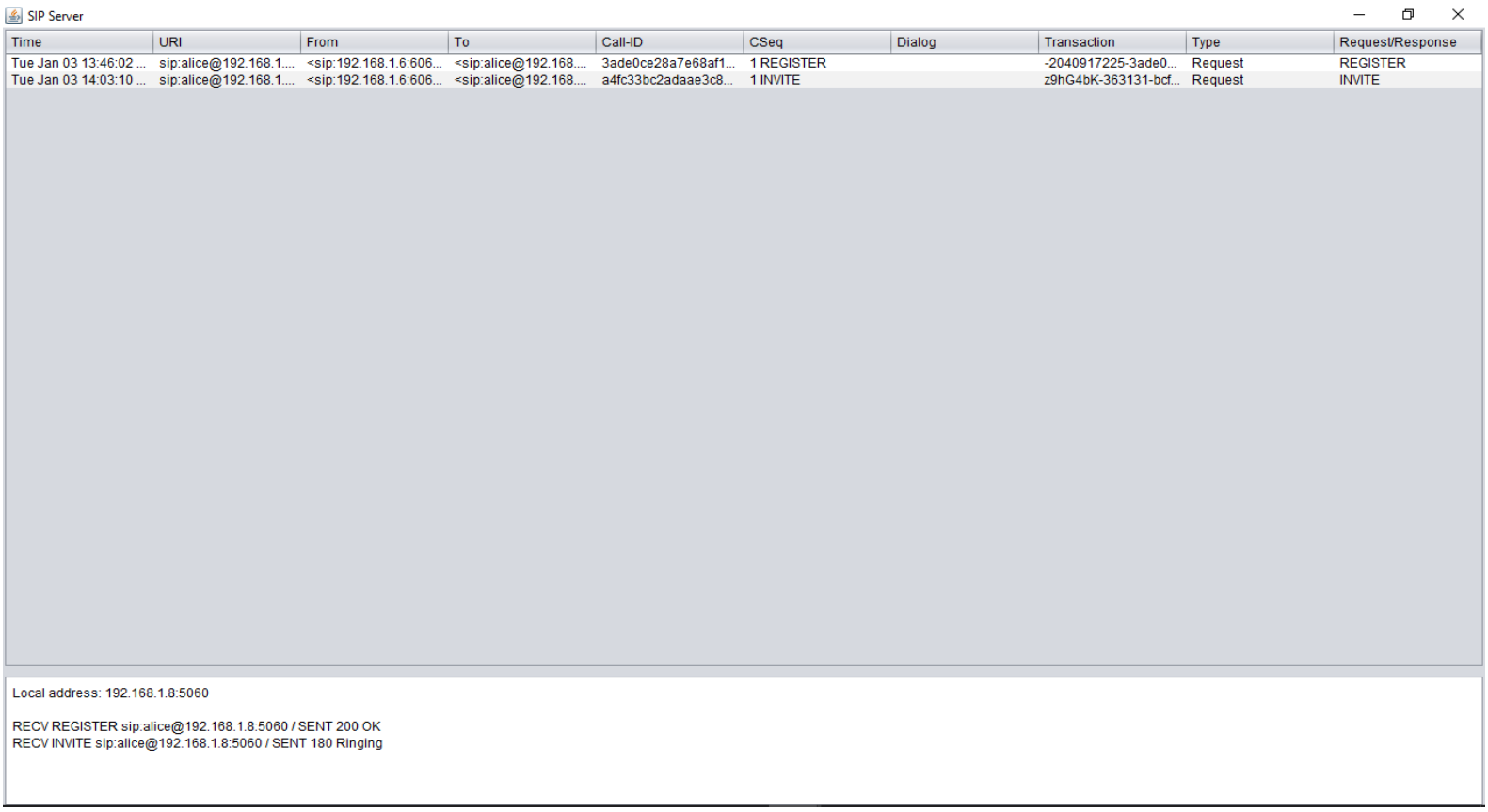
Στην συνέχεια θα πιέσει το κουμπί INVITE προκειμένου να αποσταλεί αίτημα κλήσης



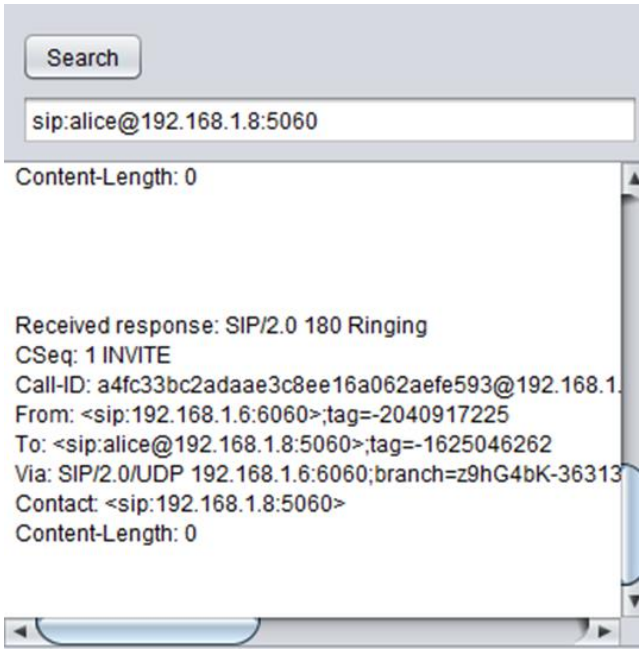
Με το πάτημα του κουμπιού ενεργοποιείται το παράθυρο εκτέλεσης της κλήσης



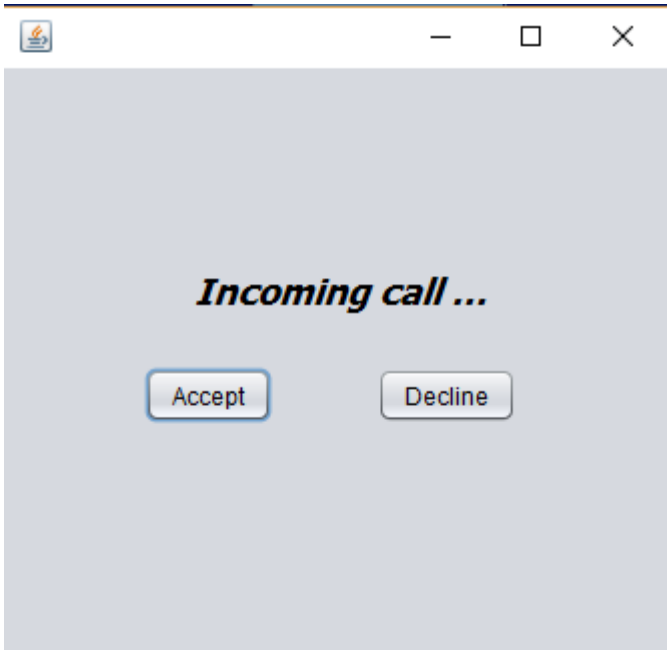
Αντίστοιχα ο πίνακας στο παράθυρο του Callee συμπληρώνεται και πάλι μόνο που αυτή τη φορά το μήνυμα που αποστέλλεται ως απάντηση είναι μήνυμα αναμονής 180 RINGING



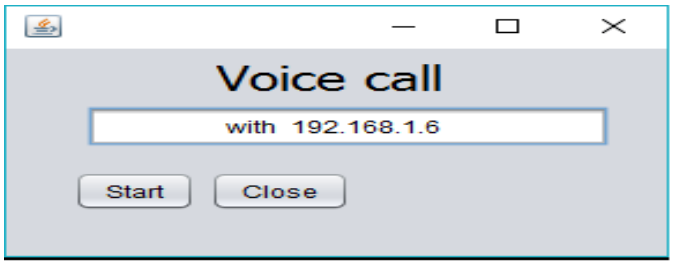
Το μήνυμα στον καλλών έχει την παρακάτω μορφή



Επίσης εμφανίζεται στον αποδέκτη της κλήσης οθόνη διαχείρισης του αιτήματος



Στην περίπτωση που επιλεγεί το Accept ανοίγει ένα παράθυρο κλήσης



Σε αυτή την φάση πιέζοντας το κουμπί Start τόσο ο πομπός όσο και ο αποδέκτης ξεκινάει η ανταλλαγή πακέτων και η αναπαραγωγή τους σε πραγματικό χρόνο όπως φαίνεται χαρακτηριστικά στα δυο output του προγράμματος στο NetBeans

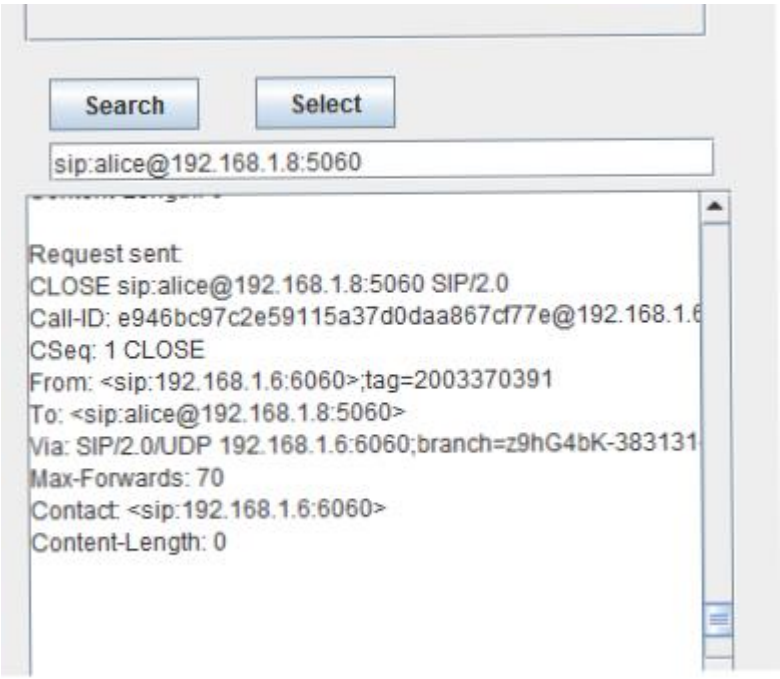
Output - ngn1 (run-single)

```
#708
Received from 192.168.1.66
   Reply sent.
#709
Received from 192.168.1.66
   Reply sent.
#710
Received from 192.168.1.66
   Reply sent.
#711
Received from 192.168.1.66
   Reply sent.
#712
Received from 192.168.1.66
   Reply sent.
send #701
Received from 192.168.1.66:  Delay: 126
send #702
Received from 192.168.1.66:  Delay: 0
send #703
Received from 192.168.1.66:  Delay: 1
send #704
Received from 192.168.1.66:  Delay: 0
send #705
Received from 192.168.1.66:  Delay: 1
send #706
Received from 192.168.1.66:  Delay: 0
send #707
Received from 192.168.1.66:  Delay: 1
send #708
Received from 192.168.1.66:  Delay: 0
```

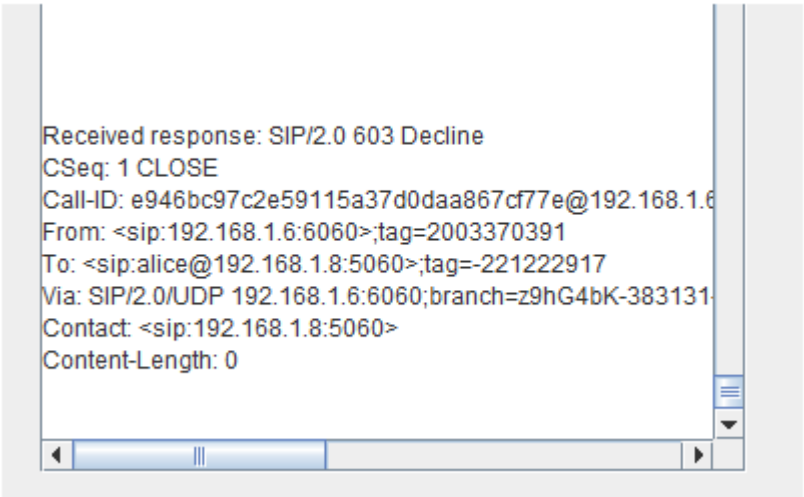
Output - ngn_2 (run-single)

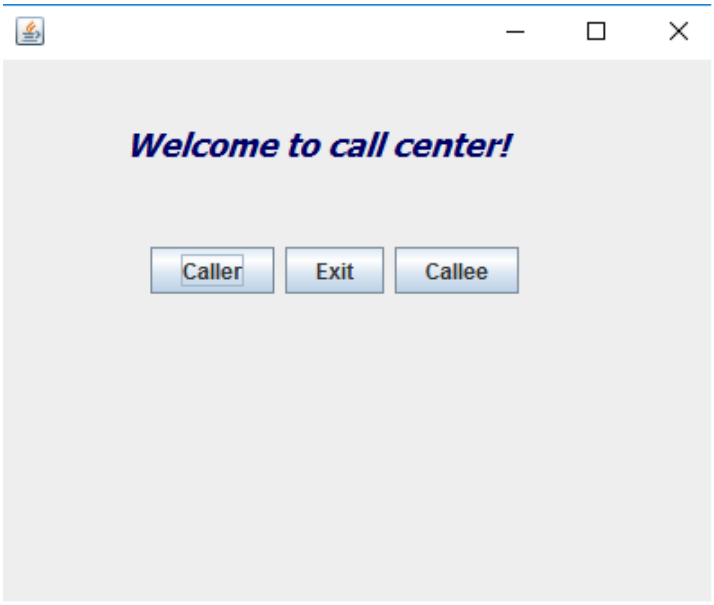
```
Received from 192.168.1.65:  Delay: 1
send #752
Received from 192.168.1.65:  Delay: 1
send #753
Received from 192.168.1.65:  Delay: 0
send #754
Received from 192.168.1.65:  Delay: 1
send #755
Received from 192.168.1.65:  Delay: 0
send #756
Received from 192.168.1.65:  Delay: 1
#642
Received from 192.168.1.65
   Reply sent.
#643
Received from 192.168.1.65
   Reply sent.
#644
Received from 192.168.1.65
   Reply sent.
#645
Received from 192.168.1.65
   Reply sent.
#646
Received from 192.168.1.65
   Reply sent.
```

Με την ολοκλήρωση της κλήσης ο πομπός πιέζοντας το πλήκτρο BYE θα τερματίσει την κλήση ενώ θα κλείσουν όλα τα παράθυρα εκτέλεσης της συνομιλίας Το Request που θα αποσταλεί είναι το εξής:

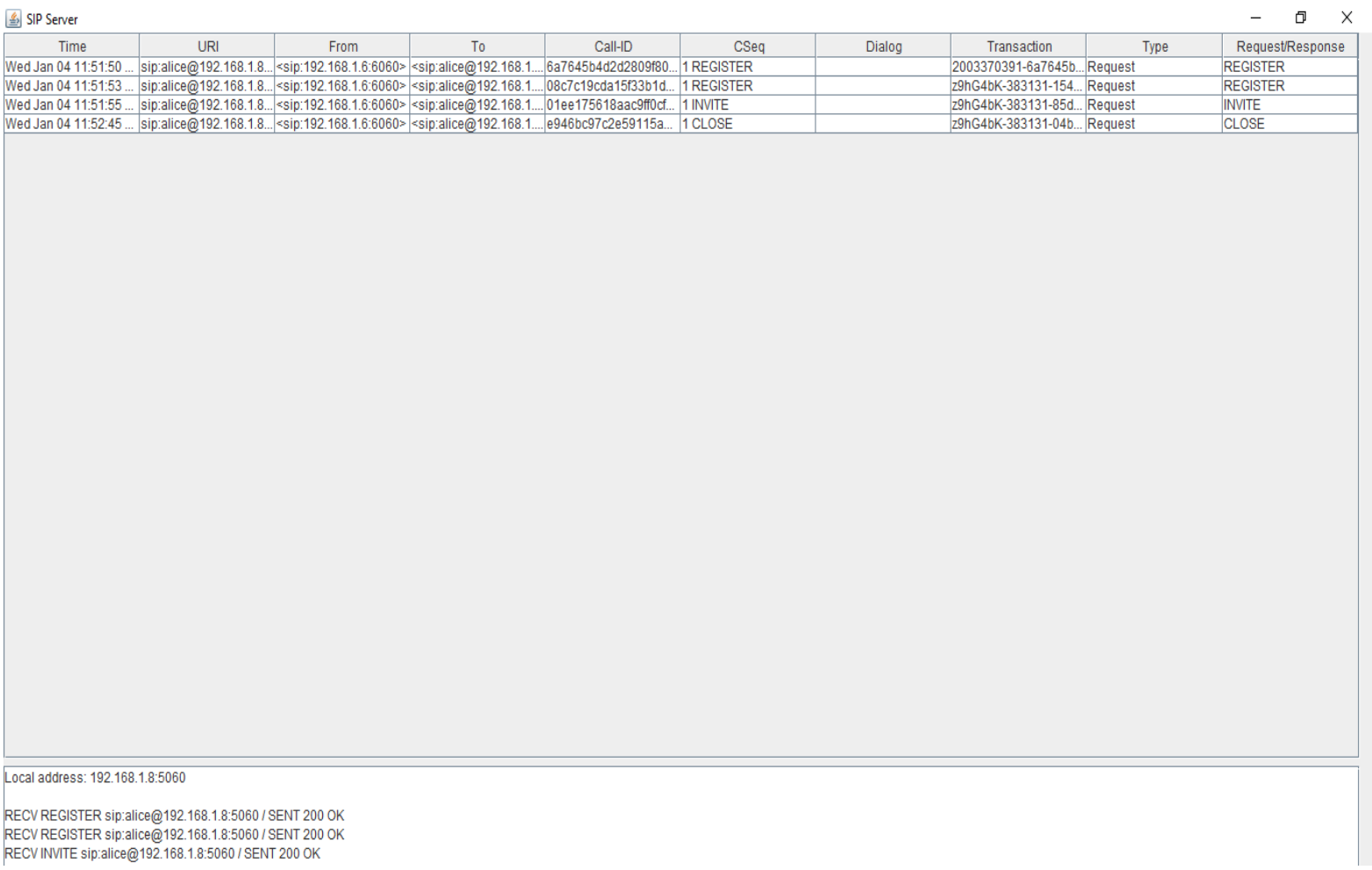


Θα λάβει επίσης Response για έξοδο από τον κληθέντα και θα επανέλθει στο αρχικό μενού επιλογής ρόλου χρήσης.

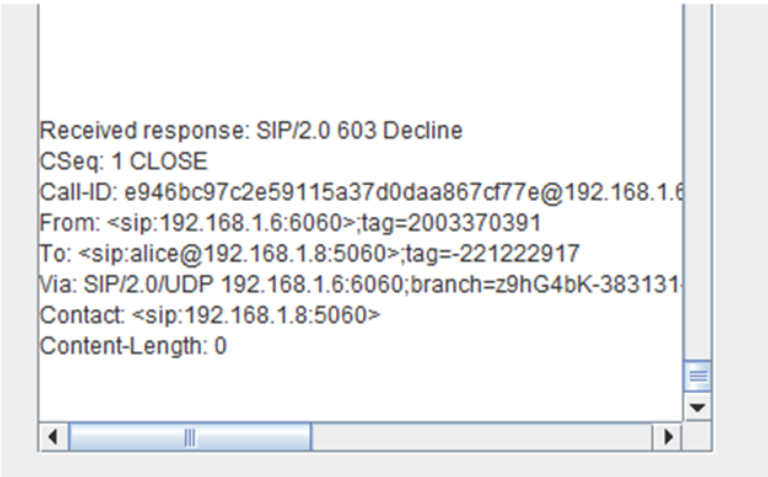




Όταν η άλλη πλευρά επιλέξει BYE τότε αρχικά θα συμπληρωθεί ο πίνακας με τα δεδομένα που θα λάβει και αφετέρου θα σταματήσει την μεταφορά φωνής ώστε να ολοκληρωθεί η κλήση. Και στον συγκεκριμένο υπολογιστή επανέρχεται το αρχικό μενού επιλογής



Στην περίπτωση πάλι που οι χρήστης δεν επιλέξει Accept αλλά Decline θα αποσταλεί στον πομπό Response σφάλματος με το οποίο και εκείνος με την μεριά του θα κλείσει το παράθυρο που άνοιξε για την καταγραφή της φωνής .



5. Συμπέρασμα

5.1 Αποτελέσματα

Για την δοκιμή και εξέταση της αποτελεσματικότητας της συγκεκριμένης εργασίας χρησιμοποιήθηκαν δυο φορητοί υπολογιστές ένας της εταιρίας Dell και ένα της εταιρίας Acer με επεξεργαστές intel core i3 τρίτης γενιάς με επεξεργαστική ισχύ στα 2,4 και 3 GHz αντίστοιχα και μνήμη RAM στα 6GB και λειτουργικό Windows 10.Ωστόσο θα μπορούσαμε να θέσουμε ως ελάχιστες απαιτήσεις συστήματος για την ορθή λειτουργία του προγράμματος οποιονδήποτε υπολογιστή με επεξεργαστή Pentium 4 ή νεότερο με ισχύ από 2,6 GHz και πάνω, μνήμη RAM τουλάχιστον 1GB και υπολειπόμενη χωρητικότητα δίσκου πάνω από 2 GB.

Οι δύο υπολογιστές είναι συνδεδεμένοι σε τοπικό δίκτυο ασύρματα. Προκειμένου να έχουμε μια πλήρη εικόνα της λειτουργικότητας του συστήματος υπολογίσαμε ένα σύνολο τιμών : τα πακέτα που αποστάλθηκαν ,τα πακέτα που διαβάστηκαν ,ο χρόνος αποστολής ,ο χρόνος λήψης ,η καθυστέρηση ,το jitter και η ταχύτητα αποστολής.

Το jitter αποτελεί ουσιαστικά την απεικόνιση της διαφοράς της καθυστέρησης του πακέτου από το προηγούμενο που λάβαμε και μετριέται σε milliseconds. Αυτό περιγράφεται με τον τύπο

Jitter= (time receive(i)-time send(i))- (time receive(i-1)-time send(i-1))

Με τη ταχύτητα αποστολής πακέτων περιγράφεται το σύνολο των bytes που μεταφέρονται ανά ms σε ένα δίκτυο. Ο τύπος προσδιορισμού της συγκεκριμένης τιμής είναι

Network_speed= total_bytes/total_time

Στους παρακάτω πίνακες παρουσιάζονται ενδεικτικά αποτελέσματα από πειραματική χρήση του προγράμματος με διαφορετικό delay και μέγεθος buffer.Τα βασικά στοιχεία του πίνακα είναι το **average delay** που είναι μια αρχική καθυστέρηση που θέσαμε στο σύστημα ώστε να προσομοιώσουμε δίκτυα με διαφορετικό εύρος και jitter,το **jitter buffer size** που προσδιορίζει το διορθωμένο μέγεθος του buffer σε περίπτωση που το jitter είναι μεγαλύτερο από το προβλεπόμενο **average maximum jitter** ,και το αρχικό **μέγεθος του buffer**.Από αυτά παίρνουμε αποτελέσματα σχετικά με το **ποσοστό των χαμένων bytes** ,**η ταχύτητα αποστολής πακέτων** του συστήματος ,το σύνολο των bytes που αποστάλθηκαν και που παρελήφθησαν και το **μέγιστο jitter** που είχαμε κατά την διάρκεια της συνομιλίας . Οι τιμές αυτές άλλαξαν στον κώδικα στις κλάσεις mg_resource_controller_recorder και mg_resource_controller_player και καθορίστηκαν σε αναλογία η μια με την άλλη ώστε το αποτέλεσμα να είναι όσο γίνεται πιο σωστό.

Buffer size	**512**	**512**	**512**	**1024**
Average Delay (ms)	**0**	**10**	**20**	**30**
Jitter Buffer size (bytes)	**1024**	**1024**	**1024**	**2048**
Average maximum jitter(ms)	**10**	**20**	**30**	**40**
Bytes Received (bytes)	859531830	1144086229	1963724821	584485614
Bytes Send (bytes)	877496815	1153517559	2000437515	592041785
Total time(ms)	89390	378782	519281	63546
Total byte loss (%)	2,05	0,82	1,83	1,28
Ταχύτητα αποστολής(bytes/ms)	11017,9	3078,8	4080,5	7894
Maximum Jitter (ms)	4	6	9	21
Total packet loss	2	3	5	5

Πίνακας αποτελεσμάτων για το Media Gateway 1

Buffer size	**512**	**512**	**512**	**1024**
Average Delay (ms)	**0**	**10**	**20**	**30**
Jitter Buffer size (bytes)	**1024**	**1024**	**1024**	**2048**
Average maximum jitter(ms)	**10**	**20**	**30**	**40**
Bytes Received (bytes)	844758953	1141042092	1949246382	547386976
Bytes Send (bytes)	861398561	1150673668	1977076846	554284461
Total time(ms)	79643	374657	490233	74999
Total byte loss (%)	1,94	0,84	1,41	1,25
Ταχύτητα αποστολής(bytes/ms)	9636,4	3037,8	3807,3	8722,57
Maximum Jitter (ms)	6	7	11	37
Total packet loss	1	2	6	8

Πίνακας αποτελεσμάτων για το Media Gateway 2

Με βάση τους παραπάνω πίνακες μπορούμε να προβούμε σε ασφαλή συμπεράσματα σχετικά με την συμπεριφορά του συστήματος σε δίκτυα διαφορετικού εύρους. Έτσι μεταβάλλοντας αρχικά το buffer τόσο ως προς την αρχική τιμή του όσο και σε περίπτωση μεγάλου jitter και στην συνέχεια το delay που προστίθεται για την προσομοίωση μεγαλύτερου δικτύου όπως και του maximum jitter που ανάλογα με αυτήν την τιμή θα κρίνεται αν χρειάζεται αύξηση του buffer ή όχι. Όσο μεγαλύτερο είναι το εύρος του δικτύου τόσο μεγαλύτερο θα είναι και το jitter οπότε για την σωστή λειτουργία του δικτύου θα πρέπει να θέτουμε και την ανάλογη καθυστέρηση. Αυτό που παρατηρούμε είναι ότι το bytes loss είναι σε πάρα πολύ χαμηλά επίπεδα και αυτό φαίνεται και κατά την διάρκεια της αναπαραγωγής καθώς ο ήχος αναπαράγεται καθαρά και με αρκετά καλή ακρίβεια. Αυτό που παρατηρούμε είναι ότι όσο μικρότερη είναι η ταχύτητα αποστολής τόσο λιγότερα πακέτα έχουν σταλεί οπότε τόσο μικρότερο είναι και το packet loss.Επίσης παρατηρούμε ότι σε όσο μεγαλύτερα επίπεδα έχουμε θέσει το maximum jitter η μέγιστη διακύμανση των καθυστερήσεων αυξάνεται ενώ αυξάνονται και τα χαμένα πακέτα, δηλαδή τα πακέτα που το σύνολο των bytes που έχουν παραδοθεί δεν υπερβαίνει το 70% του αρχικού πακέτου. Λόγω επίσης της καθυστέρησης που έχουμε θέσει καθυστερεί τόσο η αποστολή όσο και η αναπαραγωγή των πακέτων ήχου πράγμα που έχει τα αντίθετα αποτελέσματα για τη ταχύτητα αποστολής η οποία μειώνεται. Συμπερασματικά ανάλογα με το δίκτυο μπορούμε να θέσουμε και ένα χαμηλό jitter προκειμένου να έχουμε μικρότερο πλήθος χαμένων πακέτων.

5.2 Συνοπτική αποτίμηση αποτελέσματος

Στόχος μας στην παρούσα εργασία ήταν να μελετήσουμε τον τρόπο λειτουργίας ενός δικτύου τηλεπικοινωνιών επόμενης γενιάς (NGN) δίνοντας ιδιαίτερη έμφαση στην δομή του. Έτσι προβήκαμε στην υλοποίηση ενός τέτοιου συστήματος μεταφοράς φωνής σε πραγματικό χρόνο με σκοπό να πετύχουμε το καλύτερο δυνατό αποτέλεσμα. Η κλιμάκωση της εργασίας έγινε προοδευτικά αρχικά μελετώντας τόσο κάποια ιστορικά στοιχεία σχετικά με την εξέλιξη του κλάδου όσο αρκετές επιστημονικές δημοσιεύσεις προκειμένου να κατανοήσουμε τόσο το πρωτόκολλο NGN όσο και τα υπόλοιπα πρωτόκολλα που θα μας ήταν χρήσιμα για την υλοποίηση της εργασίας.

Στην συνέχεια ξετυλίχθηκε η σκέψη μας σχετικά με την μεθοδολογία που θα ακολουθούσαμε για την πραγματοποίηση του προγράμματος. Ξεδιαλύναμε σχετικά πρακτικά θέματα όπως την γλώσσα προγραμματισμού που χρησιμοποιήθηκε και το εργαλείο συγγραφής του κώδικα ενός προσαρμόσαμε και τα πρωτόκολλα που

μελετήσαμε στα δεδομένα της εργασίας. Έτσι προχωρήσαμε στην συγγραφή του κώδικα αρχικά δημιουργώντας τον «σκελετό» με τις βασικές κλάσεις και μεθόδους που χρησιμοποιήθηκαν και αργότερα με τον υπόλοιπο κώδικα. Τελικά καταλήξαμε στο αποτέλεσμα που παρουσιάστηκε στο προηγούμενο κεφάλαιο και που χρησιμοποιήθηκε για μια πειραματική μελέτη αποτελεσμάτων.

Η παρούσα εργασία κατάφερε σε ένα καλό επίπεδο να αξιοποιήσει αρκετά στοιχεία από τα πρωτόκολλα SIP και RTP όπως επίσης και του γενικότερου πρωτοκόλλου NGN.Στο τελικό στάδιο καταφέραμε τόσο την επιτυχή επικοινωνία των Server μέσω του SIP όσο και την καλύτερη δυνατή ποιότητα ήχου που μπορούσαμε να πετύχουμε με τις παροντικές μας γνώσεις.

Ωστόσο στο μέλλον έχει πολλά περιθώρια περεταίρω μελέτης και κλιμάκωσης της χρηστικότητας του ώστε να παρέχει μεγαλύτερο εύρος υπηρεσιών σε περισσότερους χρήστες . Υπάρχουν περιθώρια εμβάθυνσης και χρήσης τόσο στο πρωτόκολλο MEGACO που δεν χρησιμοποιήθηκε, το οποίο θα μας επιτρέψει την επέκταση της χρήσης του συγκεκριμένου συστήματος σε περισσότερους χρήστες όπως και των πρωτοκόλλων RTP και SIP ώστε να δοθεί στον χρήστη η δυνατότητα επικοινωνίας και με περισσότερους από έναν συν διαλεγόμενους καθώς και μεταφορά εκτός από φωνής σε πραγματικό χρόνο και εικόνας.

6.Βιβλιογραφία

1) http://www.rrmediagroup.com/Features/FeaturesDetails/FID/70
2) Crimi, J. C. (2005). Next Generation Network (NGN) Services. Retrieved December 2, 2008, from http://www.mobilein.com/NGN_Svcs_WP.pdf
3) ITU-T. (2004). ITU-T Recommendation Y.2001: General overview of NGN. Retrieved December 2, 2008, from http://www.itu.int/itudoc/itut/aap/sg13aap/history/y2001/y2001.html
4) Developing a SIP Application in Java ,http://alex.bikfalvi.com/teaching/upf/2013/architecture_and_signaling/lab/sip/
5) Σ. Αποστολάκος, Α. Μηλιώνης: "Δορυφορικές ραδιοεπικοινωνίες IP στον Έλεγχο Εναέριας Κυκλοφορίας: Σχεδίαση, Ανάπτυξη και Αξιολόγηση Τηλεπικοινωνιακών Συστημάτων", υπό έκδοση
6) https://dev.gentoo.org/~solar/Cookbook_D1/ch02s02.html
7) https://el.wikipedia.org/wiki/SIP
8) https://www.tutorialspoint.com/session_initiation_protocol/index.htm
9) http://www.javaworld.com/article/2071781/java-web-development/sip-programming-for-the-java-developer.html
10) https://en.wikipedia.org/wiki/List_of_SIP_request_methods
11) https://en.wikipedia.org/wiki/List_of_SIP_response_codes
12) https://en.wikipedia.org/wiki/H.248
13) http://searchnetworking.techtarget.com/definition/Real-Time-Transport-Protocol
14) http://www.csee.umbc.edu/~pmundur/courses/CMSC691C/lab5-kurose-ross.html
15) http://docs.oracle.com/javase/tutorial/networking
16) http://docs.oracle.com/javase/7/docs/api/javax/sound
17) http://slideplayer.com/slide/4900023/
18) http://www.slideshare.net/lineking/udp-33422210
19) https://github.com/awadalaa/Socket-Programming-Java/blob/master/UDP-Pinger