ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Ανάλυση & Υλοποίηση Επικοινωνίας μέσω του Πρωτοκόλλου SIP»

(«Analysis and Implementation of SIP Communications Protocol»)

Φοιτητής: Ιωάννης Ε. Παπουτσής

Εισηγητής: Γεώργιος Λιοδάκης,
Καθηγητής Εφαρμογών

ΤΕΙ ΚΡΗΤΗΣ / ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΗΣ
ΧΑΝΙΑ ΚΡΗΤΗΣ
ΣΕΠΤΕΜΒΡΙΟΣ 2004

Περιεχόμενα

ΠΡΟΛΟΓΟΣ

Αντικείμενο της εργασίας αποτέλεσε η ανάλυση του πρωτοκόλλου SIP και η υλοποίηση των δομικών στοιχείων αυτού (SIP clients, Proxy servers) στο δικτυακό περιβάλλον του COMNETTA Lab του ΤΕΙ Κρήτης/Τμήμα Ηλεκτρονικής. Απώτερο στόχο αποτέλεσε η εξέταση της ευελιξίας των δυνατοτήτων του SIP πρωτοκόλλου στο πλαίσιο της άμεσα μελλοντικής ευρείας χρήσης του στα επικοινωνιακά δίκτυα.

Το ερέθισμα για την ενασχόλησή μου με το πρωτόκολλο SIP αποτέλεσε, αφενός το ενδιαφέρον μου στα IP δίκτυα επικοινωνίας, αφετέρου η εργασιακή εμπειρία που απέκτησα ως Developer στο Software Center της SIEMENS Ελλάδας.

Ευχαριστώ ιδιαίτερα την οικογένεια μου για την ηθική αλλά κυρίως υλική συμπαράσταση που μού παρείχαν κατά την διάρκεια της φοίτησής μου στο ΤΕΙ ΚΡΗΤΗΣ /Τμήμα Ηλεκτρονικής.

Αυτή η πτυχιακή εργασία δεν θα είχε ολοκληρωθεί χωρίς την συμπαράσταση κάποιων καλών φίλων. Ευχαριστώ την Μαρία Καλημέρη για την υπομονή και την υποστήριξη που μού παρείχε όταν αυτό ήταν αναγκαίο. Αξίζει, επίσης, να αναφέρω την εξαιρετική συνάδελφο Μιχαίλα Δημητροπούλου για την πολύτιμη βοήθεια που μου προσέφερε, όσο εργαζόμουν στην εταιρία SIEMENS. Ακόμη θα ήθελα να ευχαριστήσω τον συνάδελφο και καλό φίλο Νικόλαο Χασάναγα, για την ηθική υποστήριξη και συνεργασία που προσέφερε καθ'όλη την διάρκεια των σπουδών μου.

Τέλος, θα ήθελα να ευχαριστήσω την επιτροπή εξέτασης (Ιωάννη Βαρδιάμπαση, Επίκουρο Καθηγητή και Τωμαδάκη Ιωάννη, Εργαστηριακό Συνεργάτη) και ιδιαίτερα τον κ. Γιώργο Λιοδάκη, Καθηγητή Εφαρμογών για την πολύτιμη καθοδήγησή του στο δύσκολο έργο της οργάνωσης, διόρθωσης, και συγγραφής της πτυχιακής εργασίας.

Abstract

Session Initiation Protocol (SIP) is an application-layer signaling protocol that can establish, modify, and terminate interactive multimedia sessions over IP networks. SIP is expected to revolutionize tomorrow's telephony market and promises significant improvements for business communications by its integration with other business communication applications.

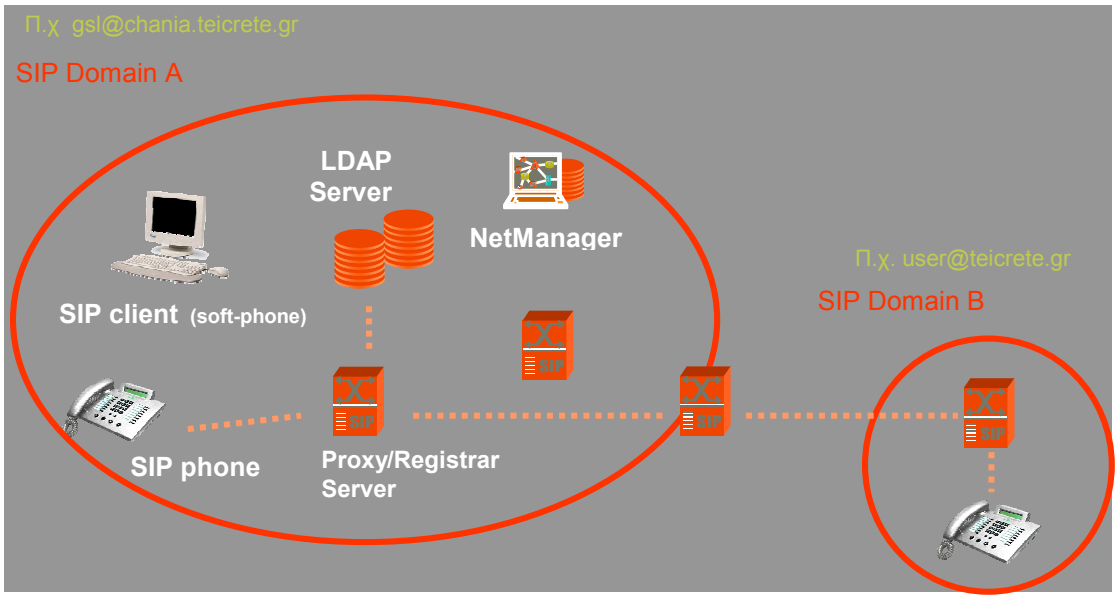
The purpose of the thesis is to examine in detail the protocol's architecture as well as SIP elements (Clients, SIP proxy servers, registrar servers, etc.) for development purposes. In particular, SIP hardware and software available for implementers and interoperability testing with other network devices is presented.

Furthermore, various communication scenarions exploiting the network infrastructure of the Communication Networks and Telematic. Applications Lab (COMNETTA Lab) of the Technological Education Institute of Crete / Department of Electronics were examined and implemented. The aforementioned set-ups in a real communications environment clearly demonstrated the SIP's protocol advantages within a packet voice network and its capabilities to define new services for users.

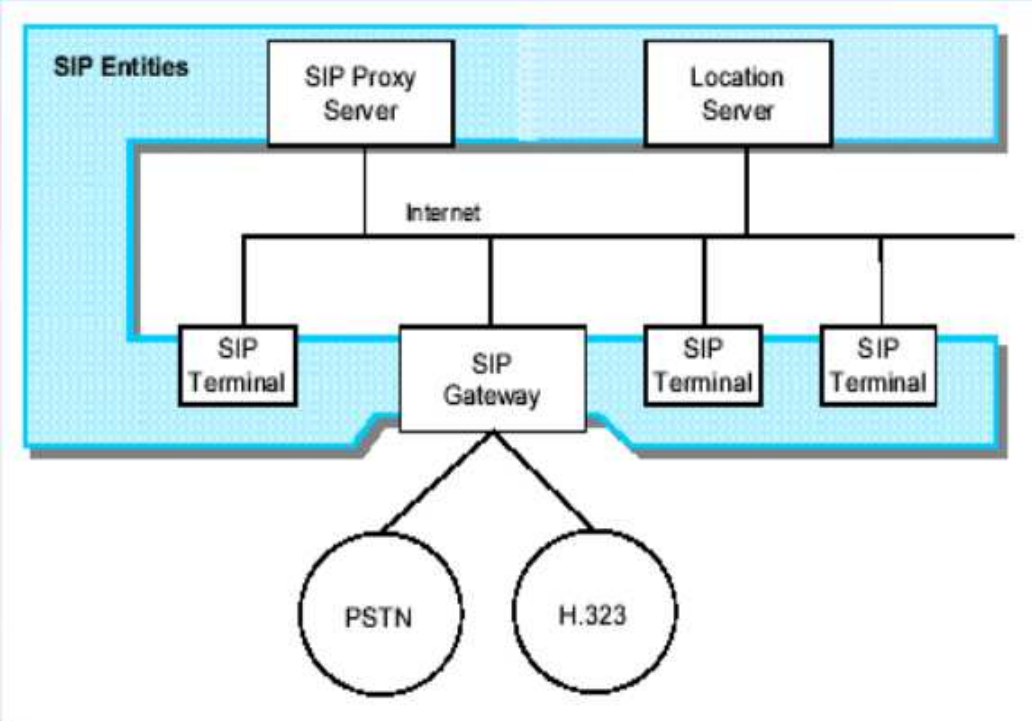
ΚΕΦΑΛΑΙΟ: 1 ΠΕΡΙ ΤΟΥ SESSION INITIATION PROTOCOL (SIP) ΠΡΩΤΟΚΟΛΛΟΥ

1.1 Εισαγωγή

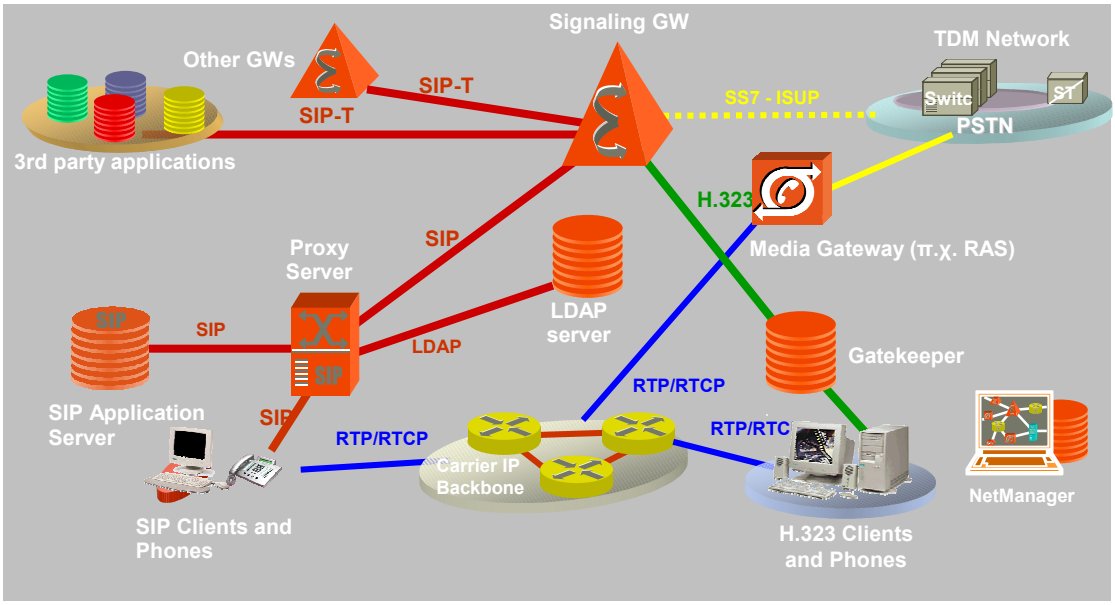
Στο *Internet* βρίσκονται πολλές εφαρμογές που απαιτούν την δημιουργία και διαχείριση μιας συνόδου (*session)*. Λέγοντας "*session*" θεωρούμε την ανταλλαγή δεδομένων μεταξύ μια ομάδας συμμετεχόντων. Η υλοποίηση αυτών των εφαρμογών δεν μπορεί να θεωρηθεί εύκολη υπόθεση εξαιτίας της ιδιομορφίας που επικρατεί μεταξύ των συμμετεχόντων. Για παράδειγμα, είναι πιθανό οι χρήστες να μετακινούνται μεταξύ *end points*, να διευθυνσιοδοτούνται με πολλαπλά ονόματα, ή να επικοινωνούν με την ταυτόχρονη χρήση διαφόρων τύπων δεδομένων (*media*). Αρκετά πρωτόκολλα έχουν προταθεί και υλοποιηθεί στο παρελθόν (όπως H.323), για την διεξαγωγή *real time multimedia sessions* μεταφέροντας δεδομένα όπως φωνή, εικόνα *video*, ή μηνύματα κειμένου. Το SIP μπορεί να συνεργαστεί άψογα με αυτά τα πρωτόκολλα αφού όπως ειπώθηκε δεν μεταφέρει τα δεδομένα του *session*, αλλά μονάχα της πληροφορίες που αφορούν το ίδιο το *session*. Σε ένα *session,* το SIP επιδιώκει να γνωρίσει τις συσκευές των χρηστών που επιδιώκουν να επικοινωνήσουν – *Internet endpoints* (ή *user agents* όπως ονομάζονται) και να συμφωνήσουν σε ένα *session* που θα συμμετάσχουν. Για τον εντοπισμό των διαφόρων πιθανών συμμετεχόντων ενός *session,* το SIP καθιστά ικανή την δημιουργία και χρήση δικτυακής υποδομής αποτελούμενη από *network hosts* (καλούνται *proxy servers*) στους οποίους οι *user agents* στέλνουν *registrations, invitations* για *sessions*, και άλλα *requests* (βλέπε σχήμα 1.1). Το SIP, λοιπόν, αποτελεί ένα ευέλικτο εργαλείο γενικού σκοπού για την δημιουργία, τροποποίηση και τερματισμό *sessions* το οποίο λειτουργεί ανεξάρτητα από το πρωτόκολλο μεταφοράς (*transport protocol*) που τρέχει από κάτω, με τελικό σκοπό την παροχή επικοινωνίας πολυμέσων (βλέπε σχήμα 1.2) εντός ενός ενοποιημένου περιβάλλοντος επικοινωνίας (βλέπε σχήμα 1.3).



Σχήμα 1.1



Σχήμα 1.2



Σχήμα 1.3

1.2 Γενικά περί του SIP πρωτοκόλλου

Το SIP είναι ένα *application layer protocol* (βλέπε σχήμα 1.4) που έχει την δυνατότητα να δημιουργεί, τροποποιεί ή να τερματίζει *multimedia sessions* (*conferences*), όπως οι κλήσεις Ιντερνετικής τηλεφωνίας. Επίσης το SIP έχει την

δυνατότητα να διαχειριστεί *multicast sessions* ή καταστάσεις όπου ο τύπος των δεδομένων που διακινούνται από το *session* μπορεί να αλλάξει. Το SIP παρέχει διαφανώς, υπηρεσίες όπως *name mapping* και *redirection* οι οποίες δίνουν στους χρήστες ελευθερία κινήσεων – οι χρήστες απλά διατηρούν έναν εξωτερικό ορατό *identifier* ανεξαρτήτου φυσικής τοποθεσίας τους στο δίκτυο. Όταν επιτευχθεί επικοινωνία SIP μεταξύ κάποιων συμμετεχόντων ορίζονται κάποια χαρακτηριστικά που αφορούν τα κάτωθι:

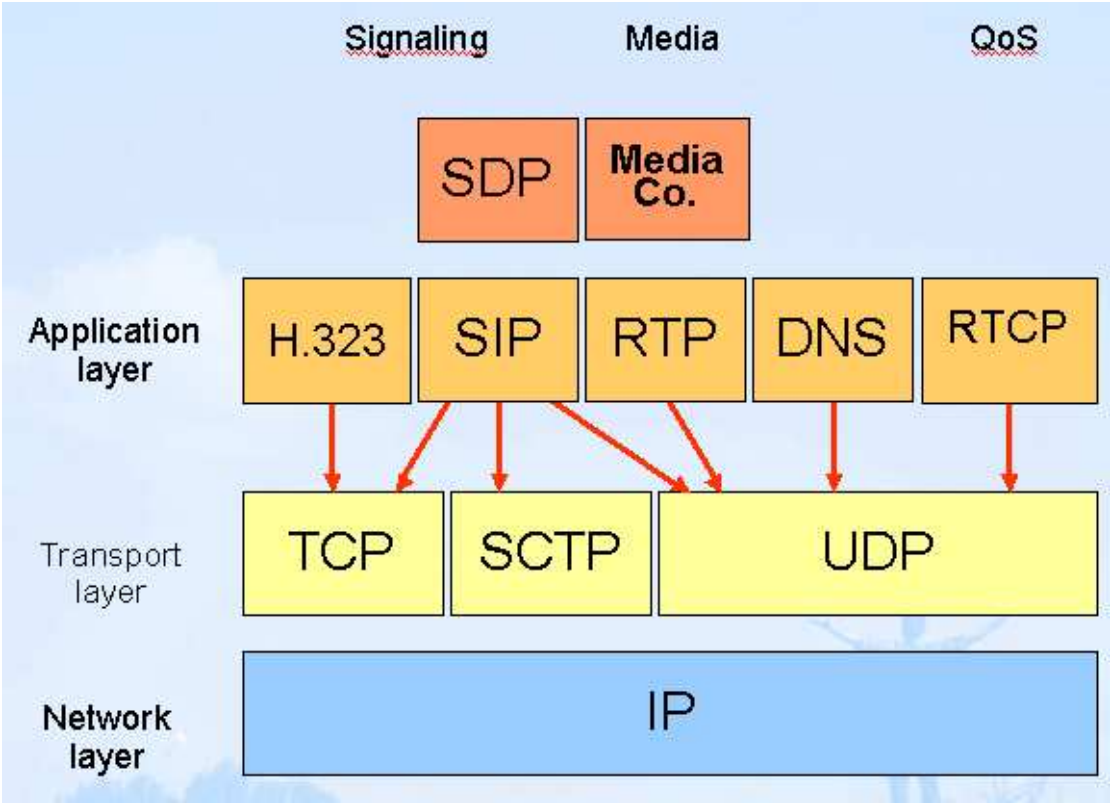
User location: καθορίζεται το *end system* που θα χρησιμοποιηθεί στο *session*

User availability: καθορίζεται η πρόθεση του καλούμενου να πάρει μέρος σε ένα SIP *session*

User capabilities: καθορισμός των *media* αλλά και των παραμέτρων τους (*media parameters*)

Session setup: διαπραγμάτευση και καταχώρηση των χρησιμοποιούμενων παραμέτρων και από τις δύο πλευρές (καλών – καλούμενος)

Session management: μεταφορά, τερματισμός, και τροποποίηση παραμέτρων του *session*.



Σχήμα 1.4

Το SIP δεν αποτελεί ένα ολοκληρωμένο σύστημα επικοινωνίας. Το SIP είναι περισσότερο ένα στοιχείο που μπορεί να χρησιμοποιηθεί μαζί με άλλα πρωτόκολλα της **IETF** για να χτίσει κανείς μια ολοκληρωμένη αρχιτεκτονική επικοινωνίας

multimedia. Τυπικά αυτές οι αρχιτεκτονικές θα μπορούσαν να περιέχουν πρωτόκολλα όπως το *Real-time Transport Protocol RTP* [1] για την μεταφορά δεδομένων *real-time*, και παροχή QoS (*Quality of Service*), το *Real-time Streaming Protocol* RTSP [2] για τον έλεγχο της παράδοσης των *streaming media*, το *Media Gateway Control Protocol* MEGACO [3] για τον έλεγχο των **gateways** στα *Public Switched Telephone Networks* (PSTNs), και το *Session Description Protocol* (SDP) [4] για την περιγραφή των *multimedia sessions*. Έτσι λοιπόν θα πρέπει να περιμένει κανείς πως το SIP μπορεί να χρησιμοποιηθεί σε συνδυασμό με άλλα πρωτόκολλα για να μπορεί να παρέχει ολοκληρωμένες υπηρεσίες σε χρήστες. Η κύρια λειτουργία του όμως δεν εξαρτάται από κανένα από τα παραπάνω πρωτόκολλα.

Ειδικότερα, σε σχέση με το παλαιότερο H.323 πρωτόκολλο μπορούν να αναφερθούν τα κάτωθι: Το H.323 προέρχεται από τον «κόσμο» των τηλεπικοινωνιών (αποτελεί προτυποποίηση της ITU), απαρτίζεται από ένα σύνολο διασυνδεδεμένων προτύπων (H.225, H.224, κ.λ.π.), κάνει χρήση μόνο του TCP στρώματος) δεν παρέχει ευέλικτη αριθμοδοτηση μέσω της σύστασης E.164 ή μέσω email διευθύνσεων, δεν υποστηρίζει κινητικότητα των χρηστών, δεν διαθέτει διαδικασίες εύκολης αποκατάστασης των κλήσεων, απαιτεί την ύπαρξη gatekeeper, συνεργάζεται εύκολα με PSTN δίκτυα και αποτελεί μία ώριμη τεχνολογία. Αντίθετα το SIP πρωτόκολλο προέρχεται από τον κόσμο της τηλεματικής (αποτελεί προτυποποίηση της IETF μέσω μιας σειρά RFC εγγράφων) υποστηρίζει εύκολα δεδομένα τύπου multimedia, υποστηρίζει την κοινότητα των χρηστών, εμπεριέχει κοινά στοιχεία των προτοκόλλων HTTP και SMTP, κάνει χρήση των TCP/SCTP/UDP στρωμάτων και ολοκληρώνεται εύκολα σε IP δίκτυα, παρέχει ευέλικτη αριθμοδότηση μέσω της σύστασης E.164 ή μέσω URIs και βρίσκεται σε φάση συνεχούς εξέλιξης με συνεχή προώθηση νέων προιόντων ως αποτέλεσμα έρευνας.

Το SIP δεν παρέχει από μόνο του υπηρεσίες. Ωστόσο παρέχει μεταβλητές, οι οποίες μπορούν να χρησιμοποιηθούν για την υλοποίηση υπηρεσιών. Για παράδειγμα, το SIP μπορεί να εντοπίσει ένα χρήστη και να τού παραδώσει διαφανώς ένα αντικείμενο στην τρέχουσα τοποθεσία. Αν μια μεταβλητή είχε χρησιμοποιηθεί για να μεταφέρει ένα *session description* γραμμένο σε SDP τα *end points* θα συμφωνούσαν στις παραμέτρους του *session*. Αν η ίδια μεταβλητή είχε χρησιμοποιηθεί για να μεταφέρει μια φωτογραφία του καλούντα (*caller A*) όπως το *session description* τότε θα είχαμε μια υπηρεσία *caller–id* στο **SIP-phone** (ζιπίφωνο) μας. Το παραπάνω παράδειγμα αποδεικνύει την ευκολία με την οποία μπορεί κανείς να ορίσει νέες υπηρεσίες βασιζόμενος στο SIP *protocol*.

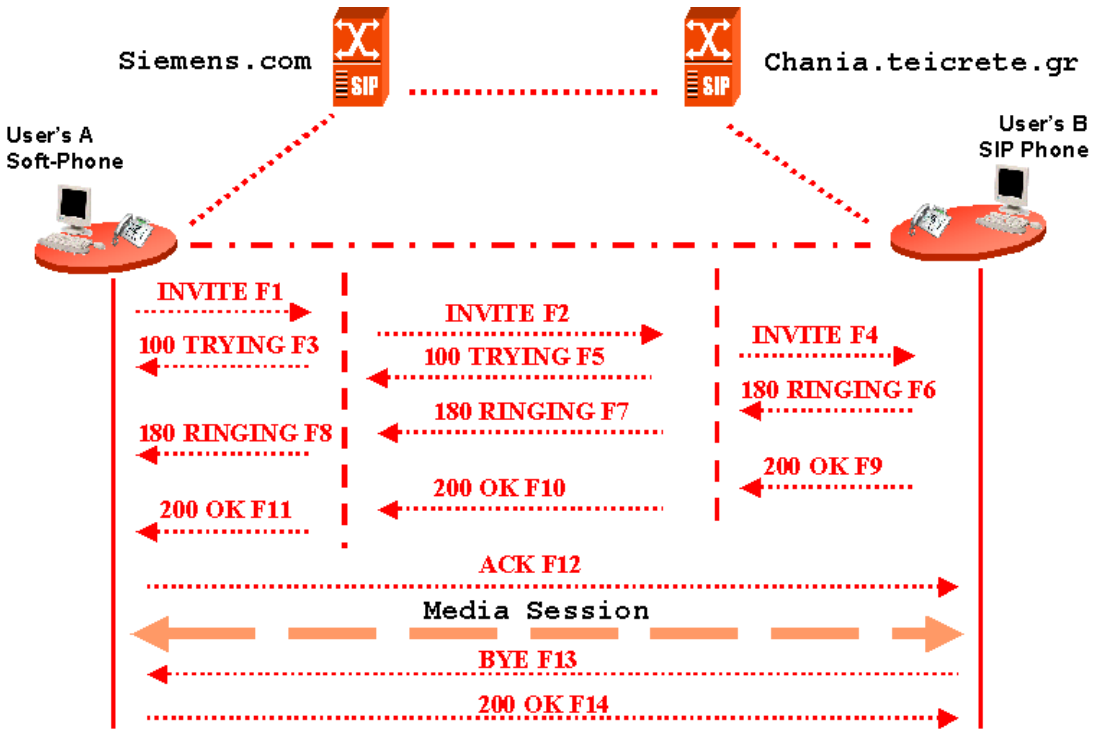
Το SIP δεν παρέχει υπηρεσίες ελέγχου τηλε-συνδιάσκεψης όπως *floor control* και *voting* όπως και δεν καθορίζει την διαχείριση μιας κλήσης τηλε-συνδιάσκεψης. Αυτό, όμως, που μπορεί να γίνει με το SIP είναι να οριστεί ένα *session* που θα χρησιμοποιεί κάποια άλλα *conference control protocols* (όπως συμβαίνει στο δίκτυο *Mbone* [14]). Επιπλέον, δεδομένου ότι τα SIP μηνύματα και τα δεδομένα χρήστη κατά την διάρκεια των *sessions* που δημιουργεί μπορεί να δρομολογούνται μέσα από εντελώς διαφορετικά μεταξύ τους δίκτυα που υποστηρίζουν είτε IPv4 είτε IPv6, το SIP δεν θα μπορούσε να παρέχει υπηρεσίες κράτησης πόρων δικτύου (*network resources reservation*).

Η εφαρμογή του SIP πρωτοκόλλου καθιστά τα θέματα ασφαλείας σημαντικά για εξέταση. Θα πρέπει να αναφερθεί πως το SIP παρέχει ένα set από υπηρεσίες

ασφαλείας όπως *denial-of-service prevention*, *authentication (user – user & user – proxy)* *integrity protection*, *encryption* και *privacy services*. Οι παραπάνω υπηρεσίες αποτελούν όμως ένα ιδιαίτερο θέμα και δεν θα αναλυθούν περαιτέρω στην παρούσα πτυχιακή εργασία.

1.3 Ανάλυση λειτουργίας SIP πρωτοκόλλου

Σε αυτή την ενότητα αναλύεται η υλοποίηση διαφόρων διαδικασιών μέσω του SIP πρωτοκόλλου Αρχικά έχουμε την δυνατότητα να δούμε μερικές από τις βασικές λειτουργίες του SIP: εντοπισμός θέσης ενός *end point*, σηματοδοσία της κλήσης, διαπραγμάτευση των παραμέτρων του *session*, τερματισμός *session*.



Σχήμα 1.5

Το σχήμα 1.5 δείχνει ένα τυπικό παράδειγμα ανταλλαγής μηνυμάτων SIP μεταξύ των δύο χρηστών A και B (κάθε μήνυμα έχει μια ετικέτα με το γράμμα F και ένα νούμερο ως αύξοντα αριθμό για ευκολότερη αναφορά). Στο συγκεκριμένο παράδειγμα ο χρήστης A χρησιμοποιεί μια SIP εφαρμογή στο PC του (ονομάζεται αλλιώς και SIP **soft-phone**) ενώ ο χρήστης B χρησιμοποιεί ένα **SIP-phone**. Η IP επικοινωνία επιτυγχάνεται είτε μέσω *Internet*, είτε μέσω ενός δικτύου VPN (όπως πράττουν οι περισσότεροι πάροχοι IP τηλεφωνίας). Στο σχήμα 1.5 φαίνονται οι δύο *proxy servers* που βρίσκονται «πίσω» από κάθε χρήστη και στόχος τους είναι η ευκολότερη διεξαγωγή του *session*. Αυτή η τυπική συμφωνία που γίνεται μεταξύ των *servers-users* συχνά ονομάζεται και SIP *trapezoid* λόγω του γεωμετρικού σχήματος που σχηματίζεται από τις διακεκομμένες γραμμές.

Για την δημιουργία μιας κλήσης ο χρήστης Α χρησιμοποιεί την SIP ταυτότητα του χρήστη Β που είναι ένας τύπος *Uniform Resource Identifier* (URI) [15]. Σε γενικές γραμμές θα μπορούσε να σημειωθεί ότι η URI έχει πολλές ομοιότητες με τις γνωστές σε όλους *email addresses* αφού περιέχει ένα *username* και ένα *domain*. Στην δική μας περίπτωση η URI του χρήστη Β είναι:

```
sip:userB@teicrete.gr
```

Όπου `teicrete.gr` είναι το *domain* του παρόχου υπηρεσιών όπου ανήκει ο χρήστης Β. Αντίστοιχα ο χρήστης Α έχει την ακόλουθη URI:

```
sip:userA@siemens.com
```

Για την διεξαγωγή της κλήσης ο χρήστης Α μπορεί να πάτησε σε κάποιο *hyperlink*, σε κάποια καταχώρηση ενός βιβλίου διευθύνσεων ή μπορεί να δακτυλογράφησε την URI. Επιπρόσθετα το SIP παρέχει ασφαλείς URIs που καλούνται SIPS URIs. Μια SIPS URI θα μπορούσε να είναι:

```
sips:userA@siemens.com
```

Μια κλήση σε μια SIPS URI εγγυάται την ασφαλή και με χρήση κρυπτογράφησης (TLS [21]) μεταφορά των μηνυμάτων SIP από τον καλούντα έως το *domain* του καλούμενου. Το μέρος της κλήσης από το *domain* του καλούμενου μέχρι τον ίδιο γίνεται πάλι με μεθόδους ασφάλειας μόνο που οι τελευταίες συμμορφώνονται με την εκάστοτε πολιτική ασφαλείας που εφαρμόζεται στο *domain* του Β και όχι του Α.

Το SIP είναι βασισμένο στον ίδιο μηχανισμό αποστολής-λήψης *responses/requests* του γνωστού σε όλους HTTP πρωτοκόλλου. Κάθε συναλλαγή (*transaction*) αποτελείται από ένα *request* που χρησιμοποιεί μια συγκεκριμένη λειτουργία και τουλάχιστον ένα *response*. Στο συγκεκριμένο παράδειγμα η συναλλαγή ξεκινάει με ένα INVITE *request* που στέλνει το **soft-phone** του χρήστη Α με προορισμό την SIP URI του χρήστη Β. Το INVITE είναι ένα παράδειγμα μιας SIP λειτουργίας που καθορίζει την ενέργεια που ο χρήστης Α (*requestor*) ζητάει από τον *server* του χρήστη Β. Το INVITE περιέχει έναν αριθμό από κάποια *headers*. Στα πεδία των *headers* (*fields*) τοποθετούνται επιπλέον πληροφορίες του μηνύματος. Έτσι λοιπόν εκεί υπάρχει ένας μοναδικός αριθμός που χαρακτηρίζει την κλήση, η διεύθυνση προορισμού του μηνύματος, η διεύθυνση του καλούντα (χρήστης Α), πληροφορίες σχετικά με το είδος του *session* που ο χρήστης Α θέλει να αποκαταστήσει με τον Β. Το μήνυμα INVITE έχει την ακόλουθη γενική μορφή:

```
INVITE sip:userB@teicrete.gr SIP/2.0
Via: SIP/2.0/UDP pc33.siemens.com;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: User B <sip:userB@teicrete.gr>
From: User A <sip:userA@siemens.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.siemens.com
CSeq: 314159 INVITE
Contact: <sip:userA@pc33.siemens.com>
Content-Type: application/sdp
Content-Length: 142
```

Η πρώτη γραμμή του μηνύματος περιέχει το όνομα της μεθόδου `(INVITE)`. Οι γραμμές που ακολουθούν αποτελούν τα πεδία των *headers*. Στο παραπάνω παράδειγμα χρησιμοποιούνται τα λιγότερα δυνατά *headers*. Ακολουθεί μια σύντομη επεξήγηση των *headers*.

Via: περιέχει την διεύθυνση (*pc33.siemens.com*) στην οποία ο χρήστης Α περιμένει τα *responses*. Επιπλέον περιέχει μια μοναδική τυχαία παράμετρο που χαρακτηρίζει την συγκεκριμένη συναλλαγή.

Max-Forwards: χρησιμεύει ώστε να μειώνει την κίνηση από πακέτα που έχουν χάσει τον προορισμό τους. Αποτελείται από έναν ακέραιο που μειώνεται για κάθε μεταπήδηση (*hop*) μέσα στο δίκτυο. Όταν ο αριθμός μηδενιστεί το πακέτο σταματάει να μεταδίδεται.

To: περιέχει το όνομα του καλούμενου (που μπορεί να εμφανίζεται ένα σύνολο αλφαριθμητικών χαρακτήρων) και την SIP ή SIPS URI στην οποία πρέπει να φτάσει το *request*.

From: περιέχει επίσης το όνομα που έχει επιλέξει ο χρήστης Α να εμφανίζεται και την SIP ή SIPS URI που δείχνει τον *originator* του *request*. Σε αυτό το *header* υπάρχει ένα σύνολο αλφαριθμητικών χαρακτήρων που έχει επιλεχτεί τυχαία από το **soft-phone** και χρησιμοποιείται για λόγους ταυτοποίησης.

Call-ID: περιέχει ένα αλφαριθμητικό σύνολο χαρακτήρων που αναφέρεται στην κλήση και είναι μοναδικό παγκοσμίως. Δημιουργείται από τον συνδυασμό ενός τυχαίου *string* και του *host name* ή την IP διεύθυνση που έχει το **soft-phone**. Ο συνδυασμός των πεδίων *To tag*, *From tag*, και *Call-ID* καθορίζει μια σχέση *peer-to-peer* μεταξύ των χρήστη Α και χρήστη Β και ονομάζεται *dialog*.

CSeq: περιέχει έναν ακέραιο αριθμό και το όνομα μιας SIP λειτουργίας. Ο ακέραιος αυξάνεται για κάθε καινούργιο *request* μέσα στο ίδιο *dialog*.

Contact: περιέχει μια SIP ή SIPS URI η οποία δηλώνει την άμεση διαδρομή για να επικοινωνήσει κάποιος με τον χρήστη Α. Συνήθως αποτελείται από κάποιο *username* και ένα FQDN (= *Fully Qualified Domain Name)* [16]. Αν και τα *domain names* είναι προτιμότερα, μπορεί κάποιοι χρήστες να μην έχουν *registered domain names*, οπότε σε αυτή την περίπτωση χρησιμοποιούνται οι IP διευθύνσεις. Έτσι λοιπόν γίνεται σαφές ότι ενώ το πεδίο *VIA* δίνει πληροφορίες για το πού θα σταλούν τα *responses*, το πεδίο *Contact* δίνει πληροφορίες για την διεύθυνση που θα ζητηθούν μελλοντικά *requests*.

Content-Type: περιέχει μια περιγραφή του περιεχόμενου του μηνύματος. (ωστόσο το περιεχόμενο του μηνύματος δεν φαίνεται στο παραπάνω παράδειγμα)

Content-Length: περιέχει τον αριθμό των *bytes* του μηνύματος

Οι λεπτομέρειες του *session* όπως ο τύπος των *media* που θα χρησιμοποιηθούν, ο *codec*, ή το *sampling rate* για την χρησιμοποιούμενη

δειγματοληψία και κωδικοποίηση δεν προσδιορίζονται από το SIP. Το κύριο σώμα ενός SIP μηνύματος περιέχει μια περιγραφή του session κωδικοποιημένη με κάποιο άλλο πρωτόκολλο. Ένα τέτοιο πρωτόκολλο θα μπορούσε να είναι το SDP [4]. Το κατά SDP κωδικοποιημένο, κύριο σώμα του μηνύματος (δεν φαίνεται στο παράδειγμα) μεταφέρεται από το SIP με ανάλογο τρόπο όπως ένα έγγραφο του *Microsoft Word* μεταφέρεται σαν *attachment* από ένα *e-mail*.

Γυρνώντας στο παράδειγμα του σχήματος, το **soft-phone** του χρήστη Α καθώς δεν γνωρίζει την διεύθυνση του χρήστη Β ή του αντίστοιχου *server* στο *domain* `teicrete.gr` στέλνει το INVITE στον SIP server που εξυπηρετεί το *domain* του χρήστη Α (`siemens.com`). Η διεύθυνση του SIP server στο `siemens.com` θα μπορούσε να είχε γίνει γνωστή στο **soft-phone** χειροκίνητα (*manually)* κατά το *setup* της εφαρμογής ή μέσω ενός DHCP *server*.

Ο `siemens.com` *server* είναι ένας SIP *server* τύπου *proxy server*. Ο *proxy server* δέχεται τα *requests* και τα προωθεί για χάρη του *requestor* (χρήστης Α). Στο συγκεκριμένο παράδειγμα ο *proxy server* δέχεται το *request* και στέλνει πίσω στο soft-phone του χρήστη Α το *response 100 (Trying)*. Το *response 100 (Trying)* δείχνει ότι ο *proxy server* έλαβε το *request* και έχει αναλάβει να το προωθήσει στον προορισμό του για χάρη του χρήστη Α. Τα *responses* στο SIP χαρακτηρίζονται από ένα τριψήφιο κωδικό ακολουθούμενο από μια περιγραφική έκφραση. Τα *responses* περιέχουν επίσης τα ίδια *headers* όπως τα *requests* (**To, From, Call-ID, CSeq**) έτσι ώστε το soft-phone του χρήστη Α να είναι σε θέση να αντιστοιχήσει το κάθε *response* που φτάνει πίσω στο αντίστοιχο *request* που έγινε. Ο *proxy server* του `siemens.com` εντοπίζει την διεύθυνση του *proxy server* για το `teicrete.gr`, πιθανότατα εκτελώντας ένα ειδικό DNS (*Domain Name Service*) *lookup*. Το DNS *lookup* θα δώσει την IP διεύθυνση του SIP *proxy* για το `teicrete.gr`, έτσι ώστε ο *proxy* να το προωθήσει το INVITE *request*. Πριν γίνει η προώθηση του request ο SIP *proxy server* `tou siemens.com` προσθέτει μια επιπλέον τιμή στο *Via header* του μηνύματος, η οποία περιέχει την διεύθυνσή του. Ο *proxy server* του `teicrete.gr` λαμβάνει το INVITE *request* και απαντάει με ένα *respond 100 (Trying)* πίσω στον *proxy server* του `siemens.com` για να δείξει ότι έλαβε και επεξεργάζεται το *request*. Ο *proxy server* του `teicrete.gr` συμβουλεύεται μια βάση δεδομένων που συνήθως ονομάζεται *location service* και περιέχει τις τρέχουσες IP διευθύνσεις των χρηστών που εξυπηρετεί. Μόλις εντοπίσει την διεύθυνση του χρήστη Β προωθεί το *request* στο SIP-phone προσθέτοντας ακόμη ένα *VIA header* το οποίο θα περιέχει την διεύθυνση του εαυτού του.

Μόλις το SIP-phone του χρήστη Β δεχτεί το INVITE από τον χρήστη Α προειδοποιεί τον χρήστη Β για την εισερχόμενη κλήση, όπως ακριβώς συμβαίνει και στην συμβατική σταθερή τηλεφωνία. Ο χρήστης Β μπορεί να επιλέξει αν θα απαντήσει την κλήση ή όχι. Μόλις το SIP-phone του χρήστη Β δεχτεί την κλήση στέλνει ένα *response 180 (ringing)* πίσω στον χρήστη Α το οποίο δρομολογείται μέσω των δύο *proxies* για να τού δηλώσει ότι είναι σε κατάσταση *ringing*. Ο κάθε *proxy* χρησιμοποιεί το πεδίο *Via* για να καθορίσει πού θα στείλει το *response* και αφαιρεί την δική του διεύθυνση από την κορυφή. Έτσι, ενώ οι υπηρεσίες όπως DNS *lookup* και *location service* είναι αναγκαίες για την δρομολόγηση ενός INVITE *request*, το *response 180 (ringing)* μπορεί να δρομολογηθεί πίσω στον καλούντα

χωρίς DNS *lookups*. Επίσης φαίνεται ότι *κάθε proxy server* που βλέπει τα INVITEs θα βλέπει και κάθε *response* που προορίζεται για αυτά.

Όταν το soft-phone του χρήστη Α λάβει το *response 180 (ringing)*, μεταβιβάζει με κάποιο τρόπο αυτήν την πληροφορία στον χρήστη Α ίσως με κάποιο ηχητικό τόνο ή οπτική ένδειξη στην οθόνη.

Σε αυτό το σενάριο, έστω ότι ο χρήστης Β αποφασίζει να απαντήσει στην κλήση. Όταν σηκώσει το ακουστικό το SIP-*phone* στέλνει ένα *response 200 (OK)* για να δείξει ότι η κλήση απαντήθηκε. Το *response 200 (OK)* περιέχει στο κυρίως μέρος αυτού την περιγραφή του *session* (σε SDP μορφή) που ο χρήστης Β είναι πρόθυμος να συμμετάσχει. Σαν αποτέλεσμα έχουμε μια αμφίδρομη ανταλλαγή μηνυμάτων SDP. Ο χρήστης Α στέλνει στον χρήστη Β και ο χρήστης Β απαντάει στον χρήστη Α. Αυτή η αμφίδρομη ανταλλαγή μηνυμάτων παρέχει τις βασικές δυνατότητες διαπραγμάτευσης και είναι βασισμένες στο απλούστατο μοντέλο του *offer/answer* του SDP *exchange*. Αν ο χρήστης Β δεν ήθελε να απαντήσει στην κλήση ή ήταν απασχολημένος σε κάποια άλλη κλήση (*busy*), τότε θα είχε σταλεί ένα *error response* αντί για το *200 (OK)*. Το *response 200 (OK)* θα μπορούσε να έχει την παρακάτω μορφή (μήνυμα F9 στο σχήμα 1.5):

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP server10.teicrete.gr
   ;branch=z9hG4bKnashds8;received=192.0.2.3
Via: SIP/2.0/UDP bigbox3.site3.siemens.com
   ;branch=z9hG4bK77ef4c2312983.1;received=192.0.2.2
Via: SIP/2.0/UDP pc33.siemens.com
   ;branch=z9hG4bK776asdhds ;received=192.0.2.1
To: UserB <sip:userb@teicrete.gr>;tag=a6c85cf
From: UserA <sip:usera@siemens.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.siemenes.com
CSeq: 314159 INVITE
Contact: <sip:userb@192.0.2.4>
Content-Type: application/sdp
Content-Length: 131
```

Η πρώτη γραμμή του *response* περιέχει τον κωδικό του *response* και την ερμηνεία αυτού. Οι επόμενες γραμμές αποτελούν τους *headers*. Τα *Via, To, From, Call-ID, and CSeq* συμπληρώνονται αντιγράφοντας τις πληροφορίες που παρέχει το INVITE *request*. Παρατηρεί κανείς ότι υπάρχουν 3 πεδία *Via*, γιατί όπως προαναφέρθηκε κάθε σταθμός στην πορεία του μηνύματος προσθέτει την δική του πληροφορία. Το SIP-phone του χρήστη Β πρόσθεσε μια *tag* παράμετρο στο πεδίο *To:*. Αυτή η παράμετρος θα ενσωματωθεί και από τα δύο *end points* στο συγκεκριμένο *dialog* και θα χρησιμοποιείται σε όλα τα επόμενα *requests / responses* της κλήσης. Όπως στην περίπτωση του χρήστη Α, έτσι και στον χρήστη Β το πεδίο *Contact:* περιέχει μια URI στην οποία ο χρήστης Β μπορεί να κληθεί άμεσα στο SIP-*phone* του. Τα *Content-Type* και *Content-Length* αναφέρονται στο κυρίως μέρος του μηνύματος που περιέχει πληροφορίες για τα SDP *media* που χρησιμοποιεί ο χρήστης Β.

Σε αυτή την περίπτωση το *200 (OK)* δρομολογείται πίσω μέσω των δύο *proxies* και το λαβαίνει το *soft-phone* του χρήστη Α το οποίο τότε σταματάει τον *ring-back* τόνο και υποδεικνύει ότι η κλήση απαντήθηκε. Τελικά το *soft-phone* του χρήστη Α στέλνει ένα *acknowledge message* (μήνυμα επιβεβαίωσης) ACK στο SIP-

phone του χρήστη Β για να επιβεβαιώσει την λήψη του τελευταίου *response 200 (OK)*. Σε αυτό το παράδειγμα το ACK στάλθηκε άμεσα από το *soft-phone* του χρήστη Α στο SIP-*phone* του χρήστη Β χωρίς την μεσολάβηση των δύο *proxy servers*. Αυτό συνέβη επειδή τα *end-points* έμαθαν την διεύθυνση του απέναντι τερματικού (*partner end-point*) από το *Contact header*. Τα *lookups* που διεξάγονται από τους δύο *proxies* πλέον δεν χρειάζονται και στο επόμενο βήμα οι *proxies* σταματούν να παρεμβάλλονται στην ροή της κλήσης. Έτσι, λοιπόν, ολοκληρώνεται το *handshake* τριών καταστάσεων (INVITE/200/ACK) που χρησιμοποιείται για να επιτευχθεί ένα SIP *session*.

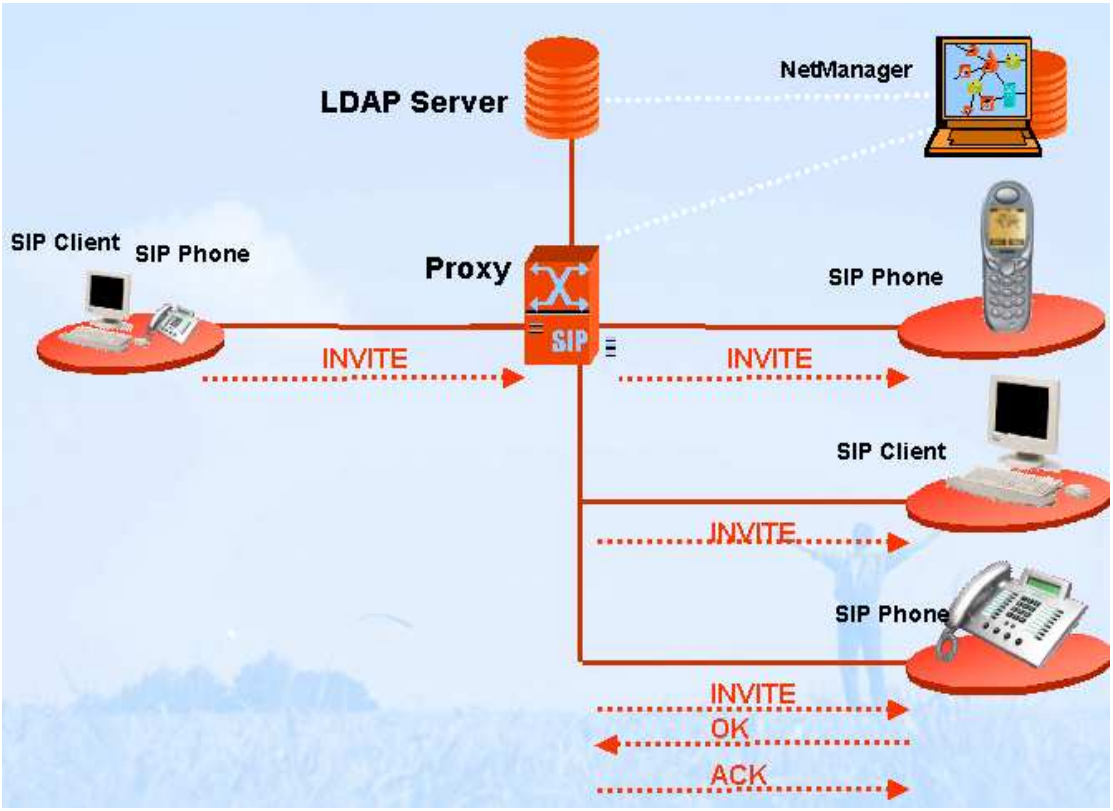
Το *media session* ανάμεσα στους δύο χρήστες έχει, πλέον, αρχίσει και οι χρήστες στέλνουν πακέτα *media* χρησιμοποιώντας το *format* στο οποίο συμφώνησαν από την ανταλλαγή των μηνυμάτων SDP. Γενικά τα *end-to-end* πακέτα χρησιμοποιούν διαφορετικό *path* από τα μηνύματα σηματοδοσίας SIP.

Κατά την διάρκεια του *session* οποιοσδήποτε από τους δύο χρήστες μπορεί να αλλάξει τα χαρακτηριστικά του *media session*. Αυτό πετυχαίνεται με την αποστολή ενός re-INVITE το οποίο θα περιγράφει το νέο *media*. Αυτό το re-INVITE αναφέρεται στο ίδιο *dialog*, έτσι ώστε στην απέναντι πλευρά το SIP-*phone* να αντιληφθεί ότι πρέπει να τροποποιήσει ένα υπάρχον *session* αντί να δημιουργήσει ένα καινούργιο. Η άλλη πλευρά στέλνει ένα μήνυμα *response 200 (OK)* για να δεχτεί την αλλαγή. Ο αιτών της αλλαγής απαντάει στο *response* με ένα μήνυμα ACK. Όμως, αν η άλλη πλευρά από αυτήν που ζητήθηκε το re-INVITE δεν δεχτεί την αλλαγή, τότε απαντάει με ένα *error response* (π.χ. *488 Not Acceptable Here*) το οποίο επίσης λαμβάνει ένα ACK. Ωστόσο η μη αλλάγη των χαρακτηριστικών του media session δεν σημαίνει και αποτυχία της κλήσης, καθώς το *session* συνεχίζεται χρησιμοποιώντας τις προηγούμενες παραμέτρους.

Στο τέλος της κλήσης ο χρήστης Β αποσυνδέεται πρώτος και προκαλεί ένα μήνυμα *BYE*. Αυτό το μήνυμα δρομολογείται άμεσα στο *soft-phone* του χρήστη Α παρακάμπτοντας τους *proxies*. Ο χρήστης Α επιβεβαιώνει την λήψη του μηνύματος με ένα *response 200 (OK)* το οποίο τερματίζει το *session*. Θα πρέπει να σημειωθεί ότι δεν στέλνεται ACK σε αυτήν την περίπτωση, καθώς μηνύματα ACK στέλνονται μονάχα σαν απάντηση στα *responses* τα οποία δημιουργούνται σαν απάντηση στα INVITE *requests*.

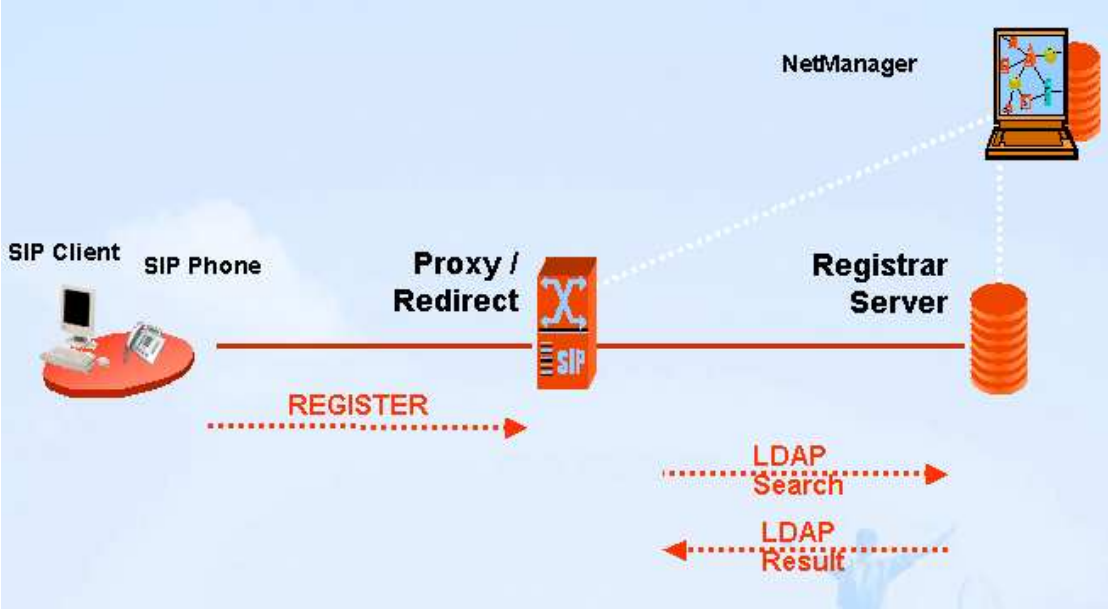
Σε μερικές περιπτώσεις μπορεί να είναι χρήσιμο για τους *proxies* να βλέπουν όλα τα μηνύματα σηματοδοσίας μεταξύ των *end-points* κατά την διάρκεια του *session*. Για παράδειγμα αν ο `teicrete.gr` ήθελε να παραμείνει στο *path* του SIP-signaling θα πρόσθετε στους *headers* του μηνύματος ένα πεδίο γνωστό σαν *Record-route* το οποίο θα περιείχε μία URI η οποία θα οδηγούσε στο *hostname* ή την IP διεύθυνση του *proxy server*. Αυτή η πληροφορία θα φαινόταν και από τις δύο SIP-enabled συσκευές (*soft-phone* & SIP-phone) αφού το πεδίο *Record-route* θα περνούσε απέναντι στο *soft-phone* του χρήστη Α με το *response 200 (OK)* και θα αποθηκευόταν για όλη την διάρκεια του *dialog*. Τότε ο *server* `teicrete.gr` θα λάμβανε και θα προωθούσε (*proxy*) όλα τα μηνύματα (ACK, BYE, 200 (OK)). Κάθε *proxy* μπορεί ανεξάρτητα να αποφασίσει αν θα λαμβάνει τα διαδοχικά μηνύματα, τα οποία μπορεί να τελικά να δρομολογούνται μέσα από κάθε *proxy server* που ζητάει μια τέτοια ενέργεια. Αυτή η δυνατότητα χρησιμοποιείται συχνά σε περιπτώσεις όπου οι *proxies* παρέχουν *mid-call* χαρακτηριστικά (π.χ. call transfer, charging κ.α.).

Επιπρόσθετα με τις υπηρεσίες DNS *lookup* και *location service*, οι *proxy servers* μπορούν να πάρουν ευέλικτες αποφάσεις δρομολόγησης (*routing decisions*) για να αποφασίσουν πού θα στείλουν τα *requests*. Θα μπορούσε για παράδειγμα ο *server* του χρήστη Β να δρομολογήσει το INVITE στον *voice mail* του χρήστη Β αν το SIP-*phone* είχε απαντήσει με κάποιο *response 486 (Busy Here)*. Ένας *proxy server* θα μπορούσε επίσης να στείλει ένα INVITE σε περισσότερες από μια διευθύνσεις ταυτόχρονα. Αυτός ο τύπος της παράλληλης εύρεσης είναι γνωστός σαν **forking** (βλέπε σχήμα 1.6).



Σχήμα 1.6

Μια άλλη κοινή διαδικασία στο SIP είναι αυτή της καταχώρησης (***registration***). Το *registration* είναι ακόμα ένας τρόπος για να πληροφορηθεί ο *proxy server* του `teicrete.gr` την τρέχουσα διεύθυνση του χρήστη Β. Σε κάθε *initialization* αλλά και σε περιοδικά χρονικά διαστήματα το SIP-*phone* του χρήστη Β στέλνει REGISTER μηνύματα σε κάποιο *server* στο *domain* `teicrete.gr` που ονομάζεται SIP *Registrar* (βλέπε σχήμα 1.7). Το μήνυμα REGISTER εμπεριέχει την σχέση μεταξύ της SIP ή SIPS URI του χρήστη Β με την IP διεύθυνση του τερματικού στο οποίο βρίσκεται (*logged in*). Ο *Registrar server* καταγράφει αυτήν την αντιστοίχηση σε μία βάση δεδομένων που καλείται και *location service*. Η υπηρεσία εύρεσης χρήστη (*location service*) μπορεί να χρησιμοποιηθεί από τον *proxy server* του *domain* `teicrete.gr`. Συχνά ο *registrar server* για ένα *domain* είναι τοποθετημένος στο ίδιο σημείο όπου λειτουργεί και ο *proxy server,* καθώς η διαφορά μεταξύ των SIP *servers* είναι λογική και όχι απαραίτητα φυσική.



Σχήμα 1.7

Στο SIP δεν υπάρχει περιορισμός στο *registration* των συσκευών για κάποιον χρήστη. Θα μπορούσε π.χ. ο χρήστης να κάνει *registration* από δύο σημεία (στο σπίτι και στο γραφείο). Οι πληροφορίες αυτές κρατούνται μαζί στην *location service* και επιτρέπουν στον *proxy server* να διεξάγει διάφορες έρευνες για να εντοπίσει τον χρήστη (π.χ. κατά το *setup* μιας κλήσης). Όμοια είναι δυνατόν περισσότεροι από ένας χρήστης να είναι *registered* στην ίδια συσκευή ταυτόχρονα.

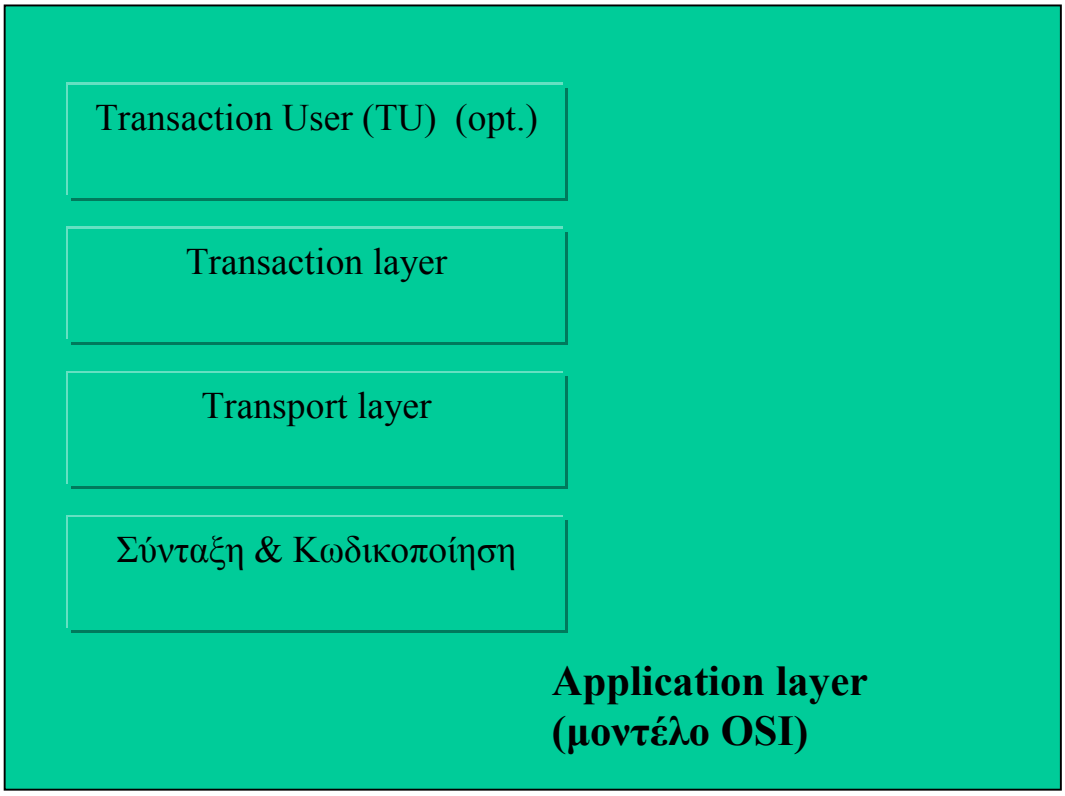
Γενικά η *location service* περιέχει πληροφορίες που επιτρέπουν σε κάποιον *proxy* να δώσει σαν *input* μία URI και να πάρει σαν *output* μία σειρά από μηδενικά ή επιπλέον URIs οι οποίες υποδεικνύουν στον *proxy* πού να προωθήσει το *request*. Τα *registrations* είναι ένας τρόπος για να δημιουργηθούν αυτές οι πληροφορίες άλλα όχι ο μοναδικός τρόπος. Ειδικές λειτουργίες εικονικής σύνδεσης (*mapping*) μπορούν να σχηματιστούν από τον διαχειριστή του δικτύου.

Τέλος θα πρέπει να σημειωθεί ότι στο πρωτόκολλο SIP η λειτουργία του *registration* είναι απαραίτητη για τη δρομολόγηση των εισερχόμενων SIP *requests* και δεν παίζει κανένα ρόλο στην εξουσιοδότηση των εξερχόμενων *requests*. Τα θέματα *authorization & authentication* αντιμετωπίζονται από το SIP ανά *request* με βάση έναν μηχανισμό *challenge / response*.

1.4 Αρχιτεκτονική Δομή του SIP πρωτοκόλλου

Το SIP είναι δομημένο σαν ένα πρωτόκολλο αποτελούμενο από επίπεδα (*layers)* (βλέπε σχήμα 1.8). Η παραπάνω έκφραση σημαίνει ότι η συμπεριφορά του πρωτοκόλλου μπορεί να περιγραφεί από ένα σύνολο ανεξάρτητων μεταξύ τους βαθμίδων επεξεργασίας. Η συμπεριφορά του πρωτοκόλλου περιγράφεται με *layers* για λόγους παρουσίασης, επιτρέποντας την περιγραφή λειτουργιών που είναι κοινές για κάποια στοιχεία σε κάποιο τομέα. Δεν θα πρέπει να παρερμηνευτεί η παραπάνω πρόταση αφού σε καμία περίπτωση δεν ορίζει την υλοποίηση. Με την έκφραση ότι

κάποιο στοιχείο περιέχει ένα *layer* εννοούμε ότι συμμορφώνεται σε κάποιο σύνολο από κανόνες που ορίζονται από το *layer*. Επιπλέον τα στοιχεία που ορίζονται από το SIP είναι λογικά στοιχεία, όχι φυσικά.



Σχήμα 1.8

Το κατώτερο *layer* του SIP αφορά τη σύνταξη και κωδικοποίησή του. Ειδικότερα η κωδικοποίηση καθορίζεται από μια γραμματική *Backus – Naur* (BNF) [17].

Το δεύτερο *layer* είναι το *transport layer*. Καθορίζει πώς ο *client* στέλνει τα *requests* και λαμβάνει τα *responses*. Επίσης καθορίζει πως ο *server* λαμβάνει τα *requests* και στέλνει τα *responses*. Όλα τα SIP στοιχεία (*Proxy, Registrar, Redirect Servers, Clients*) διαθέτουν το *transport layer*.

Το τρίτο *layer* είναι το *transaction layer*. Τα *transactions* είναι θεμελιώδη συστατικά του SIP. Ένα *transaction* (συναλλαγή) αποτελείται από ένα *request* σταλμένο από έναν *client transaction* (χρησιμοποιώντας το *transport layer*) σε έναν *server transaction*, μαζί με όλα τα *responses* στο συγκεκριμένο *request* που στάλθηκαν από τον *server transaction* πίσω στον *client*. Το *transaction layer* είναι υπεύθυνο για την επανεκπομπή των πακέτων του *application layer* που δεν έφτασαν στον προορισμό τους, για να αντιστοιχίζει τα *responses* στα *requests*, και να ελέγχει τα *timeouts* του *application layer*. Για την υλοποίηση του SIP πρωτοκόλλου το απαιτούμενο λογισμικό που τρέχει σε ένα δομικό στοιχείο της όλης υλοποίησης και αλληλεπιδρά με την χρήση αναφέρεται ως *User Agent*. Το λογισμικό *User Agent* αποτελείται από δύο τμήματα: τον *User Agent Client* (UAC) και τον *User Agent*

Server (UAS). Το τμήμα UAC είναι επιφορτισμένο με τη έναρξη των κλήσεων (αποστολή *requests*), το δε τμήμα UAS για την απάντηση των κλήσεων (αποστολή *responses*). Γενικά, μια τυπική συσκευή ή εφαρμογή που κάνει χρήση του SIP πρωτοκόλλου για διεξαγωγή κλήσεων περιλαμβάνει και τα δύο ανωτέρω τμήματα (σε αντιδιαστολή με έναν *web browser* που λειτουργεί μόνο ως *client*)

Κάθε *task* που πετυχαίνει ο *User Agent Client* (UAC) αποτελεί μια σειρά από *transactions*. Οι *user agents* που λειτουργούν σαν *stateful proxies* περιέχουν *transaction layer*. Οι *stateless proxies* δεν περιέχουν *transaction layer* (βλέπε ενότητα 1.5.1). Το *transaction layer* αποτελείται από ένα *client component* και ένα *server component* καθένα από τα οποία αναπαριστά μια πεπερασμένη κατάσταση μηχανής η οποία είναι σχεδιασμένη να εκτελεί μια συγκεκριμένη λειτουργία.

Το επόμενο ανώτερο *layer* ονομάζεται *transaction user* (TU). Κάθε μία από τις SIP οντότητες εκτός της *stateless proxy* είναι ένας *transaction user*. Όταν το TU θέλει να στείλει ένα *request*, δημιουργεί μία *client transaction instance* και στέλνει το *request* δίνοντας επιπλέον πληροφορίες για την IP διεύθυνση του παραλήπτη, το *port* και τον χρήστη που θα παραλάβει το *request*. Όπως το TU μπορεί να δημιουργήσει ένα *request*, είναι δυνατόν και να το ακυρώσει. Όταν ο *client* ακυρώνει το *request*, ζητάει από τον *server* να σταματήσει την περαιτέρω επεξεργασία του *request*, να γυρίσει στην προηγούμενη κατάστασή του και να δημιουργήσει ένα συγκεκριμένο *error response* για το *transaction*. Η παραπάνω διαδικασία συμβαίνει με ένα CANCEL *request* το οποίο απαιτεί το δικό του *transaction*.

Τα SIP στοιχεία όπως οι *user agent clients* & *servers*, *stateless* & *stateful proxies* και *registrars* περιέχουν ένα πυρήνα (*core*) που ξεχωρίζει το ένα από το άλλο. Οι πυρήνες (εκτός από τους *stateless proxies*) είναι *transactions users* (*TU*). Ενώ η συμπεριφορά ενός UAS ή UAC εξαρτάται από την λειτουργία που ακολουθείται. Ωστόσο υπάρχουν κάποιοι κοινοί κανόνες για όλες τις λειτουργίες που διεκπεραιώνονται. Για τους UACs οι κανόνες αυτοί ελέγχουν την δημιουργία των *requests* ενώ για τους UASs ελέγχουν την επεξεργασία του *request* και την δημιουργία του *response*. Όπως έχει ειπωθεί, στο SIP τα *registrations* παίζουν σημαντικό ρόλο. Ο UAS που μεταχειρίζεται τα μηνύματα REGISTER ονομάζεται *registrar*.

Διάφορα άλλα *requests* μπορούν να σταλούν μέσω ενός *dialog*. Το *dialog* είναι μία *peer-to-peer* σχέση μεταξύ δύο *user agents* για κάποιο χρονικό διάστημα. Το *dialog* διευκολύνει την ακολουθία των μηνυμάτων και την σωστή δρομολόγηση των *requests* μεταξύ των *user agents*. Η λειτουργία INVITE αποτελεί τον μοναδικό τρόπο που περιγράφεται στο *Technical Specification Document* του SIP για να δημιουργηθεί ένα *dialog*. Όταν ένας UAC στέλνει ένα request σε ένα υπάρχον *dialog* τότε αυτό ακολουθεί τους κοινούς κανόνες ενός UAC, επιπρόσθετα ακολουθεί και τους κανόνες για τα *mid-dialog requests*.

1.4.1 Περι Stateful και Stateless UAS

Ο *stateless* UAS είναι ένας UAS που δεν διατηρεί την *transaction state*. Απαντάει στα *request* κανονικά, αλλά απορρίπτει οποιαδήποτε κατάσταση (*state*) που κανονικά θα ήταν διατηρητέα από τον UAS μετά την αποστολή του *response*. Όταν

ένας *stateless* UAS λαμβάνει μία επανεκπομπή ενός *request*, τότε δημιουργεί ξανά το *response* και το ξαναστέλνει σαν να απαντούσε στο πρώτο *instance* του *request*. Ένας UAS δεν μπορεί να είναι *stateless* εκτός και αν η επεξεργασία ενός *request* οδηγεί πάντα στο ίδιο *response* (τα *requests* θα πρέπει να είναι πανομοιότυπα). Αυτό αποκλείει την ύπαρξη κάποιων *stateless Registrars*. Οι *stateless* UASs δεν χρησιμοποιούν το *transaction layer*, λαμβάνουν τα *requests* απευθείας από το *transport layer* και στέλνουν *responses* αντίστοιχα στο *transport layer*.

Ο ρόλος ενός *stateless* UAS είναι χρήσιμος κυρίως για να διευθύνει τα *unauthenticated requests* για τα οποία εκπέμπεται ένα *challenge response*. Αν τα *unauthenticated requests* επεξεργάζονταν *statefully*, τότε διάφορα *malicious unauthenticated requests* θα μπορούσαν να δημιουργήσουν μεγάλα ποσά από *transaction states* τα οποία θα μπορούσα να έχουν ως αποτέλεσμα την καθυστέρηση ή την διακοπή της επεξεργασίας (*Call Processing*) στον UAS, καταλήγοντας σε μια επίθεση τύπου *Denial Of Service (DoS) [12]*.

Οι παρακάτω προτάσεις χαρακτηρίζουν τον *stateless* UAS:

- Ο *stateless* UAS δεν μπορεί να στέλνει *provisional responses* (π.χ. *100 Trying*)
- Ο *stateless* UAS δεν μπορεί να επανεκπέμπει *responses*
- Ο *stateless* UAS πρέπει να αγνοεί τα ACK *requests*
- Ο *stateless* UAS πρέπει να αγνοεί τα CANCEL *requests*
- Τα "*To*" *header tags* των *responses* θα πρέπει να δημιουργούνται με έναν *stateless* τρόπο, δηλαδή θα δημιουργείται ίδιο *tag* για κάθε ίδιο *request*.

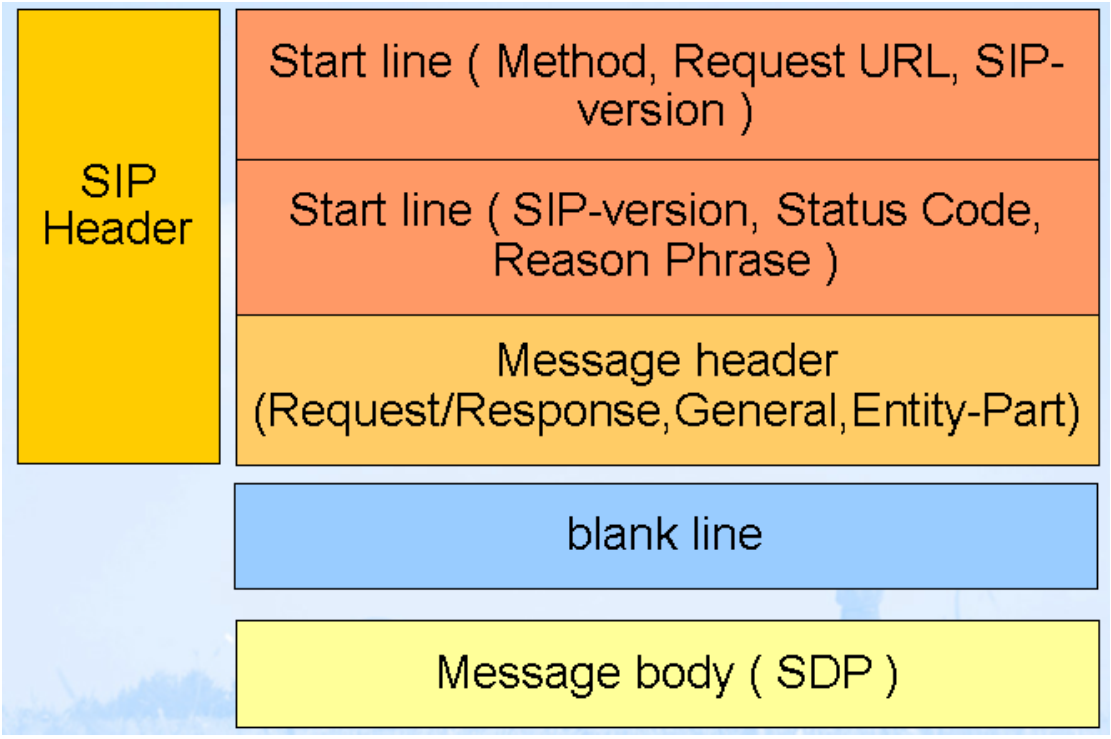
Από όλες τι άλλες απόψεις ο *stateless* UAS συμπεριφέρεται όμοια με τον *stateful* UAS. Ένας UAS μπορεί να λειτουργεί με οποιονδήποτε τρόπο (*stateless* ή *stateful*) για κάθε νέο *request*.

ΚΕΦΑΛΑΙΟ: 2 ΔΟΜΗ ΤΩΝ SIP ΜΗΝΥΜΑΤΩΝ

2.1 Μηνύματα SIP

Το SIP είναι ένα πρωτόκολλο βασισμένο σε χαρακτήρες κειμένου και ειδικότερα το *charset* (σετ χαρακτήρων) UTF-8. Ένα μήνυμα SIP μπορεί να είναι είτε ένα *request* από τον *client* στον *server* είτε ένα *response* από τον *server* στον *client*.

Τα μηνύματα και των δύο τύπων (*requests*/*responses*) ακολουθούν το βασικό *format* που περιγράφεται στο έγγραφο της IETF που περιγράφει το *Internet Message Format* [5]. Η σύνταξη παρουσιάζει κάποιες διαφορές στο σετ χαρακτήρων και στις συντακτικές λεπτομέρειες αφού το SIP επιτρέπει *header fields* που θα ήταν *invalid* σύμφωνα με τους κανόνες του παραπάνω εγγράφου. Και οι δύο τύποι μηνυμάτων περιέχουν μια αρχική γραμμή (*start line*), έναν ή περισσότερους *headers* μία άδεια γραμμή που δείχνει το τέλος των *headers* και προαιρετικά το κυρίως μέρος του μηνύματος.



Σχήμα 2.1

Η *start-line*, ο κάθε *message header* και η άδεια γραμμή θα πρέπει να τερματίζονται με έναν χαρακτήρα επιστροφής γραμμής (CR). Θα πρέπει να σημειωθεί ότι η άδεια γραμμή πρέπει να υπάρχει ακόμα και αν δεν υπάρχει κυρίως μέρος στο μήνυμα.

Επιπλέον, αν και το SIP δεν αποτελεί επέκταση του HTTP πρωτοκόλλου, τα περισσότερα από τα μηνύματα του SIP καθώς και η σύνταξη των *headers* παρουσιάζουν αρκετές ομοιότητες με το HTTP/1.1 [6].

Requests

Τα SIP *requests* διαφέρουν από τα υπόλοιπα *requests* αφού περιέχουν μία *request line* για *start line*. Η *request line* περιέχει το όνομα της λειτουργίας (*method*) , την *request* URI και την *version* του πρωτοκόλλου. Τις παραπάνω πληροφορίες χωρίζουν χαρακτήρες κενού (SP).

```
Request-Line   =   Method SP Request-URI SP SIP-Version
CRLF
```

Λειτουργία: Υπάρχουν έξι πιθανές λειτουργίες: **REGISTER, INVITE, ACK, CANCEL, BYE, OPTIONS**. Πιθανές επεκτάσεις του SIP μπορεί να παρέχουν περισσότερες λειτουργίες. Στην λειτουργία OPTIONS μπορεί να γίνει αναζήτηση σε έναν *server* για τις δυνατότητες που προσφέρει.

Request-URI: Η *request URI* μπορεί να είναι μια SIP ή SIPS URI ή μια κοινή URI όπως περιγράφεται στο *Uniform Resource Identifiers – URI* [7]. Υποδεικνύει τον χρήστη ή την υπηρεσία στην οποία απευθύνεται. Η *request URI* δεν πρέπει να περιέχει κενά και δεν επιτρέπεται να τοποθετείται η διεύθυνση σε "<>".

Τα SIP *elements* μπορεί να υποστηρίζουν *request URIs* με διαφορετικό πλάνο από το συνηθισμένο SIP ή SIPS. Θα μπορούσε να χρησιμοποιηθεί το "tel" το οποίο περιγράφεται στο *URLs for Telephone Calls* [8]. Τα SIP *elements* μπορούν να χρησιμοποιούν *non*-SIP URIs αρκεί να χρησιμοποιούν ένα μηχανισμό που να μεταφράζει τα *non*-SIP URIs σε SIP URIs ή SIPS URIs.

SIP-Version: Τα μηνύματα *requests/responses* περιέχουν την έκδοση (*version*) του SIP που χρησιμοποιούν και ακολουθούν διάφορους κανόνες συμμόρφωσης που έχουν να κάνουν με *version control*. Έτσι για να είναι συμβατές με συγκεκριμένες προδιαγραφές, οι εφαρμογές που στέλνουν SIP μηνύματα πρέπει να περιέχουν την έκδοση SIP του "SIP/2.0". Το SIP *version string* είναι *case insensitive* αλλά οι υλοποιήσεις θα πρέπει να γίνονται χρησιμοποιώντας κεφαλαία.

Αν και στο HTTP/1.1 δεν συμβαίνει το ίδιο, το SIP μεταχειρίζεται τον αριθμό *version* σαν *string*. Στην πράξη αυτό δεν δημιουργεί καμία διαφορά.

Responses

Τα SIP *responses* ξεχωρίζουν από τα *requests* έχοντας μία *status line* σαν *start line*. Η *status line* αποτελείται από το *protocol version* ακολουθούμενο από ένα αριθμητικό κωδικό κατάστασης (*status code*) και από την αντίστοιχη έκφρασή του. Όλα τα παραπάνω χωρίζονται από χαρακτήρες κενού (SP).

```
Status-Line   =   SIP-Version SP Status-Code SP Reason-
Phrase CRLF
```

Ο *status code* είναι ένας τριψήφιος ακέραιος αριθμός που δείχνει το αποτέλεσμα μιας προσπάθειας για κατανόηση και ικανοποίηση ενός *request*. Η *reason phrase* σκοπεύει να δώσει μια περιγραφή του *status code*. Ο *status code* χρησιμοποιείται από το *software* της SIP-*enabled* συσκευής ενώ η *reason phrase* έχει τοποθετηθεί για να την διαβάζει ο άνθρωπος όπου αυτό κρίνεται αναγκαίο.

Ενώ το *specification SIP: Session Initiation Protocol* [9] προτείνει συγκεκριμένη ορολογία για την *reason phrase*, οι διάφορες υλοποιήσεις μπορεί να επιλέξουν διαφορετική ορολογία για την περιγραφή των *status codes*.

Το πρώτο ψηφίο του *status* προσδιορίζει το είδος του *response* (*class*). Τα επόμενα δύο ψηφία δεν κατηγοριοποιούνται. Για αυτό το λόγο οποιοδήποτε *response* μεταξύ 100 ~ 199 αναφέρεται σαν "*response 1xx*", οποιοδήποτε *response* μεταξύ 200~299 αναφέρεται ως "*response 2xx*" κ.ο.κ. Το SIP/2.0 επιτρέπει 6 πιθανές τιμές σαν πρώτο ψηφίο και ειδικότερα:

1xx : Προσωρινή κατάσταση – το *request* έχει ληφθεί, συνεχίζεται η επεξεργασία του.

2xx : Επιτυχής κατάσταση – η ενέργεια ελήφθη επιτυχημένα, κατανοήθηκε και έγινε δεκτή

3xx : *Redirection* – πρέπει να εκτελεσθεί και επιπλέον ενέργεια για να ολοκληρωθεί το *request*

4xx : *Client Error* – το *request* περιέχει συντακτικό πρόβλημα ή δεν μπορεί να εκπληρωθεί στον συγκεκριμένο *server*

5xx : *Server Error* – ο *server* απέτυχε να εκπληρώσει ένα έγκυρο *request*

6xx : *Global Failure* – το *request* δεν μπορεί να εκπληρωθεί σε κανένα *server*.

Πεδία Header (Header Fields)

Τα *header fields* στο SIP παρουσιάζουν αρκετές ομοιότητες με τα *headers* στο HTTP συντακτικά αλλά και εννοιολογικά. Συγκεκριμένα τα SIP *header fields* ακολουθούν την αναφορά [H4.2] για την σύνταξη και τους κανόνες για τα *extending header fields* σε πολλαπλές γραμμές.

Η αναφορά [H4.2] καθορίζει επίσης ότι πολλαπλά *header fields* με το ίδιο *field name* τα οποία έχουν τιμή μία λίστα με στοιχεία χωρισμένα από κόμμα, μπορούν να συνδυαστούν σε ένα *header field*. Αυτό μπορεί να εφαρμοστεί και στο SIP αλλά ο συγκεκριμένος κανόνας διαφέρει λόγω της διαφορετικής γραμματικής. Συγκεκριμένα κάθε SIP *header* του οποίου η γραμματική είναι του τύπου

```
header = "header-name" HCOLON header-value *(COMMA header-value)
```

επιτρέπει τον συνδυασμό των *header fields* του ίδιου ονόματος σε κάποια *comma-separated* λίστα. Το *header field* "*Contact*" επιτρέπει λίστες *comma-separated* εκτός αν η τιμή είναι "*".

Μορφοποίηση των Header Fields

Τα *Header Fields* ακολουθούν το ίδιο γενικό *header format* όπως αυτό περιγράφεται στο *section 2.2* του *Internet Message Format* [5]. Κάθε *header field* αποτελείται από ένα *field name* (το όνομα του πεδίου) ακολουθούμενο από ένα (" : ") και την τιμή του πεδίου.

```
field-name: field-value
```

Η επίσημη γραμματική που χρησιμοποιείται για *message headers* επιτρέπει έναν αριθμό από κενά δεξιά και αριστερά από το (":"), ωστόσο καλό είναι κάτι τέτοιο να αποφεύγεται και να χρησιμοποιείται ένα κενό (SP) μονάχα μεταξύ του διαχωριστικού (":") και της τιμής.

```
Subject:              lunch
Subject      :        lunch
Subject              :lunch
Subject: lunch
```

Έτσι, λοιπόν, όλα τα παραπάνω είναι έγκυρα και ισοδύναμα, αλλά το τελευταίο είναι προτιμότερο.

Τα *header fields* μπορούν να επεκταθούν σε πολλαπλές γραμμές ξεπερνώντας κάθε επιπλέον γραμμή με τουλάχιστον ένα SP ή οριζόντιο διάστημα (*Horizontal Tab HT*). Το *line break* και το κενό στην αρχή της επόμενης γραμμής αντιστοιχούν απλούς κενούς χαρακτήρες (SP). Έτσι λοιπόν τα παρακάτω είναι ισοδύναμα.

```
Subject: I know you're there, pick up the phone and talk to me!
Subject: I know you're there,
         pick up the phone
         and talk to me!
```

Η σχετική σειρά των *header fields* με διαφορετικά *field names* δεν είναι ιδιαίτερα σημαντική. Ωστόσο, προτείνεται τα *header fields* που χρειάζονται για την επεξεργασία που εκτελεί ο *proxy* (*Via, Route, Record-Route, Proxy-Require, Max-Forwards*, και *Proxy-Authorization*) να φαίνονται πάνω από τα υπόλοιπα *header fields* για να γίνεται γρηγορότερα η προσπέλαση τους. Η σχετική σειρά των γραμμών *header fields* με το ίδιο όνομα είναι σημαντική. Σε ένα μήνυμα επιτρέπεται να υπάρχουν πολλαπλά *header fields* με το ίδιο *field name*, αρκεί η τιμή του *header* να ορίζεται σαν *comma separated list*. Ο συνδυασμός των πολλαπλών *header fields* σε ένα ζευγάρι (*field name : field value*) θα πρέπει να είναι δυνατός χωρίς αλλαγή στη σημασία του μηνύματος και προσθέτοντας τα επιπλέον *field values* στο πρώτο χωρίζοντας τα με *commas*. Εξαιρέσεις σε αυτόν το κανόνα αποτελούν τα *WWW-Authenticate, Authorization, Proxy-Authenticate*, και *Proxy-Authorization header fields*. Πολλαπλά *header fields* με αυτά τα ονόματα μπορούν να υπάρξουν σε κάποιο μήνυμα, όμως όταν η γραμματική που ακολουθούν δεν είναι η γενική που

παρουσιάστηκε στην προηγούμενη παράγραφο δεν μπορούν να συνδυαστούν σε ένα *header field*.

Οι υλοποιήσεις θα πρέπει να είναι σε θέση να επεξεργάζονται μηνύματα που περιέχουν πολλαπλά *header fields* με το ίδιο όνομα, σε οποιονδήποτε συνδυασμό των : μια τιμή ανά γραμμή και *comma-separated* τιμές

Τα παρακάτω *groups* των *header fields* είναι έγκυρα και ισοδύναμα.

```
Route: <sip:alice@atlanta.com>
Subject: Lunch
Route: <sip:bob@biloxi.com>
Route: <sip:carol@chicago.com>

Route: <sip:alice@atlanta.com>, <sip:bob@biloxi.com>
Route: <sip:carol@chicago.com>
Subject: Lunch

Subject: Lunch
Route: <sip:alice@atlanta.com>, <sip:bob@biloxi.com>,
       <sip:carol@chicago.com>
```

Τα παρακάτω groups των *header fields* είναι έγκυρα αλλά όχι ισοδύναμα.

```
Route: <sip:alice@atlanta.com>
Route: <sip:bob@biloxi.com>
Route: <sip:carol@chicago.com>

Route: <sip:bob@biloxi.com>
Route: <sip:alice@atlanta.com>
Route: <sip:carol@chicago.com>

Route: <sip:alice@atlanta.com>,<sip:carol@chicago.com>,
       <sip:bob@biloxi.com>
```

Το *format* μιας *header field* τιμής, ορίζεται από το *header name*. Στις περισσότερες τον περιπτώσεων θα είναι μια σειρά από χαρακτήρες UTF-8, ή ένας συνδυασμός χαρακτήρων με κενά ή αλφαριθμητικά μέσα σε εισαγωγικά (“ ”).

Στην σύγκριση των *header fields* θα πρέπει να γνωρίζει κανείς ότι τα *field names* είναι πάντοτε *case insensitive*. Τα *field values*, *parameter names*, και *parameter values* είναι *case-insensitive* εκτός και αν έχει δηλωθεί κάτι διαφορετικό σε κάποιο συγκεκριμένο *header field.* Οι τιμές που εκφράζονται μέσα σε εισαγωγικά είναι *case sensitive* εκτός και αν έχει δηλωθεί το αντίθετο.

Παράδειγμα 1 (Ισοδυναμία μηνυμάτων):

```
Contact: <sip:alice@atlanta.com>;expires=3600
```

&

```
CONTACT: <sip:alice@atlanta.com>;ExPiReS=3600
```

Παράδειγμα 2 (Ισοδυναμία μηνυμάτων):

```
Content-Disposition: session;handling=optional
```

&

```
content-disposition: Session;HANDLING=OPTIONAL
```

Αντίθετα, τα επόμενα δύο *header fields* δεν είναι ισοδύναμα.

```
Warning: 370 devnull "Choose a bigger pipe"
Warning: 370 devnull "CHOOSE A BIGGER PIPE"
```

Ταξινόμηση των Header Fields

Κάποια *header fields* γίνονται κατανοητά μόνο από συγκεκριμένα *requests* και *responses*. Αυτά ονομάζονται *request header fields* και *response header fields*. Αν κάποιο *header field* μοιάζει να μην ταιριάζει στο μήνυμα (π.χ. ένα *request header field* σε ένα *response*) τότε θα πρέπει να αγνοηθεί.

Συντομευμένη μορφή (Compact Form)

Το SIP παρέχει έναν μηχανισμό για να αναπαριστά τα ονόματα των κοινών *header fields* σε μια συντομευμένη μορφή. Αυτή η δυνατότητα μπορεί να φανεί χρήσιμη σε μηνύματα που διαφορετικά θα ήταν αρκετά μεγάλα σε μήκος για να μεταφερθούν με το *transport protocol* (π.χ. UDP). Έτσι δίνεται η δυνατότητα να μειώσουμε το μήκος των μηνυμάτων το οποίο επιβάλλεται από το *maximum transmission unit* MTU που υπάρχει σε κάθε *transport protocol*. Ο *compact form* μπορεί να αντικαταστήσει τον *longer form* χωρίς να υπάρξει αλλαγή στην πληροφορία του *header field*. Μέσα σε ένα μήνυμα είναι δυνατόν να συναντήσουμε και τους δύο τύπους. Ως επακόλουθο θα πρέπει και οι υλοποιήσεις να ακολουθούν την παραπάνω οδηγία.

Σώμα μηνύματος (Message Body)

Τα *requests* (ακόμα και τα καινούργια που αποτελούν *extension* στο RFC 3261) μπορεί να περιέχουν στο μήνυμα τους κυρίως σώμα εκτός αν έχει σημειωθεί το αντίθετο. Η υλοποίηση του κυρίως σώματος εξαρτάται από την λειτουργία του *request*.

Στα *response* μηνύματα η λειτουργία του *request* καθώς και ο *response status code* καθορίζουν τον τύπο και την ερμηνεία οποιουδήποτε *message body*. Όλα τα *response* μπορεί να περιέχουν *message body*.

Τύπος του Message Body

Ο τύπος δεδομένων (*media type*) του κυρίως μηνύματος πρέπει να δηλωθεί στο *Content-type header field*. Αν το κυρίως σώμα φέρει οποιουδήποτε είδους

κωδικοποίηση όπως συμπίεση, τότε αυτό θα πρέπει εξίσου να δηλωθεί στο *Content-Encoding header field*, διαφορετικά το *Content-Encoding* θα πρέπει να παραληφθεί. Επιπρόσθετα θα πρέπει να αναφέρεται το *character set* του *message body* σαν μέρος της τιμής του *Content-Type header field* αν αυτό είναι εφαρμόσιμο.

Το "*multipart*" τύπου MIME που ορίζεται στο έγγραφο της IETF σχετικά με *Multipurpose Internet Mail Extensions* (MIME) [18] μπορεί να χρησιμοποιηθεί στο κυρίως σώμα του μηνύματος. Οι υλοποιήσεις που στέλνουν *requests* τα οποία περιέχουν *multipart message bodies* πρέπει να στέλνουν ένα *session description* σαν *non-multipart message* αν η απομακρυσμένη υλοποίηση το απαιτεί μέσω ενός *Accept header field* που δεν περιέχει *multipart*.

Τα SIP μηνύματα μπορεί να περιέχουν *message bodies* με δυαδικά δεδομένα. Όταν δεν παρέχεται καμία παράμετρος σχετικά με το *charset* από τον αποστολέα (*sender*) τα *media subtypes* τύπου *text* παίρνουν εξ'ορισμού σαν *charset* την τιμή "UTF-8"

Μήκος του Message Body

Το μήκος του *message body* σε *bytes* παρέχεται από το *Content-Length header field*.

Η μεταφορά κατά τμήματα (chunks) που χρησιμοποιείται στο HTTP/1.1 [19] δεν μπορεί να χρησιμοποιηθεί από το SIP. (Σημείωση: Η "*chunked*" κωδικοποίηση τροποποιεί το σώμα του μηνύματος σαν μια σειρά από *blocks* το καθένα με τον δικό του δείκτη μήκους.)

Το SIP μήνυμα μέσα στο FRAME

Αντίθετα με το HTTP, οι SIP υλοποιήσεις μπορούν να χρησιμοποιήσουν UDP ή άλλα μη αξιόπιστα *datagram protocols*.

Οι υλοποιήσεις που επεξεργάζονται SIP μηνύματα πάνω από *stream-oriented transports* πρέπει να αγνοούν χαρακτήρες CRLF πριν την *start line* [H4.1]

Η τιμή του *content-length header* χρησιμοποιείται για να εντοπιστεί το τέλος ενός SIP μηνύματος σε μια ροή δεδομένων (*data stream*). Ο *content-length header* θα είναι πάντα παρών όταν τα SIP μηνύματα στέλνονται πάνω από *stream oriented transports*.

2.2 Γενική Συμπεριφορά ενός User Agent

Ο *user agent* αναπαριστά ένα *end system*. Περιέχει ένα *user agent client* (UAC) ο οποίος δημιουργεί τα *requests* και ένα *user agent server* (UAS) ο οποίος αντιδρά στα *requests*. Ο UAC είναι ικανός να δημιουργεί κάποιο *request* βασισμένο σε κάποια εξωτερική ενέργεια (το *click* ενός χρήστη σε κάποιο κουμπί ή το σήμα σε μια PSTN γραμμή). Ο UAS είναι ικανός να δέχεται κάποιο *request* και να δημιουργεί ένα *response* βασισμένο στην είσοδο από τον χρήστη, σε εξωτερική διέγερση, στο αποτέλεσμα ενός προγράμματος, η κάποιον άλλο μηχανισμό.

Όταν ένας UAC στέλνει ένα *request*, το *request* περνάει από κάποιους *proxy servers* οι οποίοι το προωθούν προς τον UAS. Όταν ο UAS δημιουργεί το *response*, τότε το *response* προωθείται πίσω στον UAC.

Οι διαδικασίες των UAC και UAS εξαρτώνται από δύο βασικούς παράγοντες:

- αν το *request* ή το *response* βρίσκεται μέσα ή έξω από ένα *dialog*
- την λειτουργία του *request*

Τα *dialogs* αναπαριστούν μια *peer-to-peer* σχέση μεταξύ των *user agents* και δημιουργούνται με ειδικές SIP λειτουργίες (ή *methods* όπως εμαφανίζονται στην Αγγλική βιβλιογραφία) όπως για παράδειγμα το INVITE.

Για την ασφάλεια των χρηστών υπάρχουν διαδικασίες ασφάλειας για τα *requests* και *responses* που βρίσκονται τοποθετημένα έξω από ένα *dialog*. Συγκεκριμένα υπάρχουν ειδικοί μηχανισμοί *authenticate* για τους UASs και UACs. Ένα περιορισμένο *set* από *privacy features* πετυχαίνετε με χρήση κρυπτογράφησης των *message bodies* χρησιμοποιώντας S/MIME [20].

2.2.1 Συμπεριφορά του User Agent Client

Δημιουργία του request

Ένα SIP *request* για να μπορεί να θεωρείται έγκυρο θα πρέπει να διατυπωθεί από τον UAC χρησιμοποιώντας τα επόμενα *header fields* : **To, From, CSeq, Call-ID, Max-Forwards,** και **Via**. Όλα από τα παραπάνω *header fields* είναι υποχρεωτικά σε όλα τα SIP *requests*. Αυτά τα 6 *header fields* αποτελούν τις βάσεις ενός SIP μηνύματος καθώς παρέχουν πολύ κρίσιμες πληροφορίες για την δρομολόγηση των μηνυμάτων (όπως τη διεύθυνση των μηνυμάτων, τη δρομολόγηση των *responses*, την μέγιστη διάδοση ενός μηνύματος, την σειρά των μηνυμάτων, και το *unique identification* ενός *transaction*). Θα πρέπει να σημειωθεί και η ύπαρξη της *request line* η οποία παρέχει εξίσου πολύ σημαντικές πληροφορίες όπως την λειτουργία, την *request* URI και την SIP *version*.

Request-URI

```
INVITE sip:userB@teicrete.gr SIP/2.0
Via: SIP/2.0/UDP pc33.siemens.com;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: User B <sip:userB@teicrete.gr>
From: User A <sip:userA@siemens.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.siemens.com
CSeq: 314159 INVITE
Contact: <sip:userA@pc33.siemens.com>
Content-Type: application/sdp
Content-Length: 142
```

Η αρχική *request URI* ενός μηνύματος θα πρέπει να τίθεται στην τιμή της URI στο *To: field*. Μια αξιοσημείωτη εξαίρεση αποτελεί η λειτουργία REGISTER.

Ωστόσο μπορεί να είναι ανεπιθύμητη η συμπλήρωση των δύο αυτών *fields* με την ίδια τιμή για λόγους *privacy*.

Σε κάποιες ειδικές περιστάσεις, η παρουσία μίας προϋπάρχουσας διαδρομής (*route*) μπορεί να επηρεάσει την *request URI* του μηνύματος. Η προϋπάρχουσα διαδρομή είναι μια σειρά από URIs οι οποίες συνθέτουν μια σειρά από *servers* στους οποίους ο UAC θα στείλει εξερχόμενα *requests* τα οποία δεν βρίσκονται σε κάποιο *dialog*. Συνήθως οι ρυθμίσεις αυτές είναι σεταρισμένες στον UA από τον χρήστη ή τον *service provider* μέσω ενός *non*-SIP μηχανισμού. Όταν ο *service provider* επιθυμεί να σετάρει τον UA με έναν *outbound proxy*, προτείνεται αυτό να συμβαίνει παρέχοντας μια *pre-existing* διαδρομή μέσω ενός απλού URI αυτό του *outbound proxy*.

Όταν έχει σεταριστεί μια *pre-existing* διαδρομή θα πρέπει να ακολουθούνται οι διαδικασίες για τον υπολογισμό της *Request-URI* και του *Route header field* χρησιμοποιώντας το επιθυμητό *request URI* σαν το *remote target* URI.

To

```
INVITE sip:userB@teicrete.gr SIP/2.0
Via: SIP/2.0/UDP pc33.siemens.com;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: User B <sip:userB@teicrete.gr>
From: User A <sip:userA@siemens.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.siemens.com
CSeq: 314159 INVITE
Contact: <sip:userA@pc33.siemens.com>
Content-Type: application/sdp
Content-Length: 142
```

Το *header field "To"* καθορίζει τον επιθυμητό «**λογικό**» παραλήπτη του *request* ή κάποιο *resource* το οποίο είναι ο στόχος του *request*. Μπορεί να περιέχει τον απόλυτο παραλήπτη του *request* αλλά αυτό δεν είναι απόλυτο. Το *header field "To"* μπορεί να περιέχει μια SIP ή SIPS URI, επίσης μπορεί να κάνει χρήση των υπόλοιπων URIs *schemes* (π.χ. tel URI RFC 2806 *URLs for Telephone Calls*) όπου αυτό είναι επιτρεπτό. Οι υλοποιήσεις που στηρίζουν το TLS [21] πρέπει να στηρίζουν και το SIPS URI *scheme*. Στο *"To" header field* επιτρέπεται να δοθεί ένα *display name*.

Ένας UAC μπορεί να διδαχθεί πώς να υπολογίζει το *"To" header field* με διάφορους τρόπους για συγκεκριμένα *requests*. Συνήθως ο χρήστης ορίζει το *"To" header field* μέσω ενός *human interface*, δίνοντας την URI *manually* ή κάνοντας *click* σε κάποια καταχώρηση ενός *address book*. Συνήθως δεν είναι αναγκαίο από τον χρήστη να δώσει ολόκληρη την URI, αλλά ένα *string* από γράμματα και αριθμούς (για παράδειγμα «*bob*»). Αφήνεται πλέον στον UA πώς θα εκμεταλλευτεί το συγκεκριμένο *input*. Χρησιμοποιώντας το "*string to form*", το μέρος του χρήστη μιας SIP URI υπονοεί ότι ο UA επιθυμεί το όνομα να γίνεται *resolved* στο *domain* δεξιά από το *at-sign in* SIP URI (π.χ. *sip:bob@example.com*). Χρησιμοποιώντας το "*string to form*" το *user part* της SIPS URI, ο UA υπονοεί ότι θέλει να επικοινωνεί με ασφάλεια και ότι το όνομα θέλει να γίνεται *resolved* στο *at-sign in domain*.

Ένα *request* έξω από από κάποιο *dialog* δεν μπορεί να περιέχει το *"To" tag*. Το *"To" tag* υποδεικνύει ότι το *dialog* υπάρχει. Εφόσον δεν υπάρχει *dialog*, δεν μπορεί να υπάρξει και *tag "To"*.

Το παρακάτω παράδειγμα αποτελεί ένα έγκυρο *"To" header field*.

```
To: Carol <sip:carol@chicago.com>
```

From

```
INVITE sip:userB@teicrete.gr SIP/2.0
Via: SIP/2.0/UDP pc33.siemens.com;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: User B <sip:userB@teicrete.gr>
From: User A <sip:userA@siemens.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.siemens.com
CSeq: 314159 INVITE
Contact: <sip:userA@pc33.siemens.com>
Content-Type: application/sdp
Content-Length: 142
```

Το *From header field* υποδηλώνει την λογική ταυτότητα του *initiator* ενός *request*, πιθανότατα το *address-of-record* ενός χρήστη. Όπως και το "*To" header field*, περιέχει μία URI και προαιρετικά το εμφανιζόμενο όνομα. Χρησιμοποιείται από τα SIP *elements* για να καθοριστεί ποιοι κανόνες επεξεργασίας θα εφαρμοστούν σε κάποιο *request* (για παράδειγμα αυτόματη απόρριψη κλήσης). Γίνεται κατανοητό πως είναι πολύ σημαντικό το *From* URI να μην περιέχει IP διεύθυνση ή το FQDN του *host* στο οποίο ο UA τρέχει εφόσον αυτά δεν είναι λογικά ονόματα.

Το *From header field* επιτρέπει να τοποθετηθεί ένα εμφανιζόμενο όνομα. Ένας UAC θα πρέπει να χρησιμοποιεί για εμφανιζόμενο όνομα "*Anonymous*" αν η ταυτότητα του *client* πρέπει να μείνει κρυφή.

Συνήθως η τιμή που βρίσκεται στο *From header field* των *requests* τα οποία δημιουργούνται από κάποιον UA, είναι προ-τοποθετημένη από τον χρήστη ή τους *administrators* του *local domain*. Αν κάποιος UA χρησιμοποιείται από πολλούς χρήστες, μπορεί να διαθέτει *switchable profiles* τα οποία περιέχουν ένα URI για κάθε ταυτότητα χρήστη. Οι παραλήπτες των *requests* μπορούν να διαπιστώσουν την ταυτότητα του *originator* ενός *request* ούτως ώστε να εξακριβώσουν πραγματικά αν είναι αυτός που δηλώνει το *From header field*.

Το *From* θα πρέπει να περιέχει μια νέα *tag* παράμετρο διαλεγμένη από τον UAC. Αυτή η παράμετρος χρησιμοποιείται για να αναγνωρίζεται το *dialog*.

Παραδείγματα:

```
From: "Bob" <sips:bob@biloxi.com> ;tag=a48s
From: sip:+12125551212@phone2net.com;tag=887s
From: Anonymous <sip:c8oqz84zk7z@privacy.org>;tag=hyh8
```

Call-ID

```
INVITE sip:userB@teicrete.gr SIP/2.0
```

```
Via: SIP/2.0/UDP pc33.siemens.com;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: User B <sip:userB@teicrete.gr>
From: User A <sip:userA@siemens.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.siemens.com
CSeq: 314159 INVITE
Contact: <sip:userA@pc33.siemens.com>
Content-Type: application/sdp
Content-Length: 142
```

Το *Call-ID header field* λειτουργεί σαν μοναδικό πιστοποιητικό ώστε να ομαδοποιούνται μαζί μια σειρά μηνυμάτων. Πρέπει να είναι το ίδιο για όλα τα *requests* και *responses* που στέλνονται από οποιοδήποτε UA μέσα σε κάποιο *dialog*. Θα πρέπει να είναι το ίδιο και για κάθε *registration* από κάποιον UA.

Σε ένα καινούργιο *request* που δημιουργείται από κάποιον UAC έξω από κάποιο *dialog*, το Call-ID *header field* θα πρέπει να επιλεγεί από τον UAC σαν ένα γενικό αναγνωριστικό στοιχείο (*globally unique identifier*) στον χώρο και στον χρόνο εκτός αν η παραπάνω αρχή υπερκαλύπτεται από συγκεκριμένη συμπεριφορά κάποιας λειτουργίας. Όλοι οι SIP UAs πρέπει να διαθέτουν ένα τρόπο για να εγγυηθούν ότι τα *Call-ID header fields* που δημιουργούν δεν θα δημιουργηθούν και από κάποιον άλλο UA. Θα πρέπει να σημειωθεί ότι, όταν τα requests ξαναστέλνονται μετά από συγκεκριμένα *failure responses* που ζητούν την βελτίωση κάποιας παραμέτρου, τότε αυτά τα *retried requests* δεν θεωρούνται καινούργια requests. Έτσι λοιπόν δεν υπάρχει ανάγκη για νέο *Call-ID header field*.

Η χρήση κρυπτογραφημένων τυχαίων αναγνωριστικών παρέχει κάποια προστασία από *session hijackings* και μειώνει την πιθανότητα ακούσιων *Call-ID collisions*.

Για την επιλογή του *Call-ID header field* ενός *request*, ο ανθρώπινος παράγοντας δεν παίρνει μέρος είτε με την μορφή χρήστη είτε με την μορφή *network administrator*.

Παράδειγμα:

```
Call-ID: f81d4fae-7dec-11d0-a765-00a0c91e6bf6@foo.bar.com
```

Cseq

```
INVITE sip:userB@teicrete.gr SIP/2.0
Via: SIP/2.0/UDP pc33.siemens.com;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: User B <sip:userB@teicrete.gr>
From: User A <sip:userA@siemens.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.siemens.com
CSeq: 314159 INVITE
Contact: <sip:userA@pc33.siemens.com>
Content-Type: application/sdp
Content-Length: 142
```

Το *CSeq header field* υπάρχει για να παρέχει ένα τρόπο αναγνώρησης και διαχείρισης των *transactions*. Αποτελείται από έναν *sequence number* (αριθμός

σειράς) και μια λειτουργία. Η λειτουργία πρέπει να ταιριάζει με αυτήν που αναφέρθηκε στο *request*. Για τα *non-registered requests* έξω από κάποιο *dialog* ο *sequence number* είναι αυθαίρετος. Ο *sequence number* θα πρέπει να εκφράζεται σαν ένας 32 *bit integer* μη προσημασμένος και θα πρέπει να είναι μικρότερος από 2^31. Όσο ακολουθεί τους παραπάνω κανόνες ο *client* μπορεί να ακολουθήσει οποιοδήποτε μηχανισμό επιθυμεί για να επιλέγει τιμές για το *CSeq header field*.

Παράδειγμα

```
CSeq: 4711 INVITE
```

Max-Forwards

```
INVITE sip:userB@teicrete.gr SIP/2.0
Via: SIP/2.0/UDP pc33.siemens.com;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: User B <sip:userB@teicrete.gr>
From: User A <sip:userA@siemens.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.siemens.com
CSeq: 314159 INVITE
Contact: <sip:userA@pc33.siemens.com>
Content-Type: application/sdp
Content-Length: 142
```

Το *Max-Forwards header field* εξυπηρετεί στον περιορισμό των μεταπηδήσεων (*hops*) που μπορεί να υποστεί ένα *request* στον δρόμο προς τον προορισμό του. Αποτελείται από κάποιον *integer* ο οποίος μειώνεται κατά ένα για κάθε *hop* που διασχίζει το *request*. Αν η *Max-Forwards* τιμή φτάσει στο 0 πριν το *request* φτάσει στον προορισμό του τότε το *request* θα απορριφθεί με *error response 486 (Too Many Hops)*

Ο UAC πρέπει να εισάγει ένα *Max-Forwards header field* σε κάθε *request* που δημιουργεί, με μέγιστη τιμή το 70. Ο αριθμός αυτός επιλέχτηκε ώστε να είναι αρκετά μεγάλος για να μπορεί να εγγυηθεί ότι κάποιο *request* δεν θα χαθεί σε ένα SIP *network* όπου δεν υπάρχουν *loops*, αλλά όχι τόσο μεγάλος ώστε να καταναλώνονται *proxy resources* όταν συμβαίνει κάποιο *loop*. Μικρότερες τιμές μπορούν να χρησιμοποιηθούν αλλά με προσοχή και όταν είναι γνωστή από τον UA, η τοπολογία του δικτύου.

Via

```
INVITE sip:userB@teicrete.gr SIP/2.0
Via: SIP/2.0/UDP pc33.siemens.com;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: User B <sip:userB@teicrete.gr>
From: User A <sip:userA@siemens.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.siemens.com
CSeq: 314159 INVITE
Contact: <sip:userA@pc33.siemens.com>
Content-Type: application/sdp
Content-Length: 142
```

Το *Via header field* υποδηλώνει το *transport* πρωτόκολλο που χρησιμοποιήθηκε για το *transaction* και αναγνωρίζει την τοποθεσία όπου πρέπει να σταλεί το *response*. Θα πρέπει να σημειωθεί ότι μια συγκεκριμένη τοποθεσία στο *via header field* προστίθεται μόνο μετά την επιλογή της επόμενης διαδρομής που θα χρησιμοποιηθεί για να μεταφερθεί το μήνυμα στο επόμενο *hop*.

Όταν ο UAC δημιουργεί ένα *request* θα πρέπει να εισάγει ένα *Via header field* μέσα στο *request*. Το όνομα του πρωτοκόλλου καθώς και η έκδοση αυτού θα πρέπει να είναι στο *header field* "SIP" και "2.0" αντίστοιχα. Το *Via header field* θα πρέπει να περιέχει μια *branch parameter* (παράμετρος διακλάδωσης). Αυτή η παράμετρος χρησιμοποιείται για να αναγνωρίζεται η *transaction* που δημιουργήθηκε από το *request*. Η παράμετρος αυτή χρησιμοποιείται από τον *client* και τον *server*.

Η τιμή της *branch parameter* πρέπει να είναι μοναδική στο χρόνο και στο χώρο για όλα τα *requests* που στέλνονται από τον UA. Οι εξαιρέσεις σε αυτόν τον κανόνα είναι τα CANCEL και ACK για τα *non-2xx responses*. Όπως έχει αναφερθεί προηγουμένως ένα CANCEL *request* θα έχει την ίδια τιμή στην *branch parameter* με το *request* που θέλει να ακυρώσει. Επίσης ένα ACK για ένα *non-2xx response* θα έχει επίσης το ίδιο *branch ID* όπως το INVITE του οποίου το *response* σκοπεύει να αναγνωρίσει (*acknowledge*).

Η μοναδική ιδιότητα της παραμέτρου *branch ID*, να διευκολύνει με την χρήση της ως *transaction ID* δεν αποτελούσε μέρος του εγγράφου της IETF με τίτλο *SIP: Session Initiation Protocol* [11].

Η *branch ID* που έχει εισαχθεί από ένα στοιχείο συμβατό με την συγκεκριμένη προδιαγραφή πρέπει πάντα να αρχίζει με τους χαρακτήρες "z9hG4bK". Αυτοί οι 7 χαρακτήρες χρησιμοποιούνται σαν μαγικό *cookie*. Το 7 μπορεί να διαβεβαιώσει ότι μια παλαιότερη υλοποίηση από την έκδοση του *SIP: Session Initiation Protocol* [11] δεν θα διάλεγε την ίδια τιμή, έτσι αυτό εξυπηρετεί ώστε οι *servers* που λαμβάνουν το *request* μπορούν να αποφασίσουν ότι το *branch ID* δημιουργήθηκε με βάση την προδιαγραφή *RFC 3261 (SIP: Session Initiation Protocol)* [9] ή οποία είναι παγκοσμίως μοναδική.

Contact

```
INVITE sip:userB@teicrete.gr SIP/2.0
Via: SIP/2.0/UDP pc33.siemens.com;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: User B <sip:userB@teicrete.gr>
From: User A <sip:userA@siemens.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.siemens.com
CSeq: 314159 INVITE
Contact: <sip:userA@pc33.siemens.com>
Content-Type: application/sdp
Content-Length: 142
```

Το *Contact header field* παρέχει μια SIP ή SIPS URI η οποία μπορεί να χρησιμοποιηθεί για να έρθει αυτή σε επαφή με τον UA ώστε να διεκπεραιωθούν συνεπακόλουθα *requests*. Το *Contact header field* πρέπει να είναι παρών και να περιέχει ακριβώς μία SIP ή SIPS URI σε οποιοδήποτε *request* που μπορεί να καταλήξει στην δημιουργία ενός *dialog*. Από τις μεθόδους που περιγράφονται στη

προδιαγραφή *RFC 3261 (SIP: Session Initiation Protocol)* [9], η μοναδική λειτουργία που το απαιτεί είναι η INVITE *request*. Για αυτά τα *requests* ο σκοπός του *Contact* είναι γενικός, δηλαδή η τιμή του *Contact header field* περιέχει την URI στην οποία ο UA θα ήθελε να λαμβάνει τα *requests*. Αυτή η URI θα πρέπει να είναι έγκυρη ακόμα και αν χρησιμοποιείται σε διαδοχικά *requests* έξω από κάποιο *dialog*.

Αν η *request URI* ή το *Router header field* περιέχουν μία SIPS URI τότε και το *Contact header field* θα πρέπει να περιέχει αντίστοιχα μία SIPS URI.

Supported & Require

Αν ο UAC είναι συμβατός με προεκτάσεις (*extensions*) του SIP που μπορούν να εφαρμοστούν από τον *server* στο *response* τότε θα πρέπει να εισαχθεί στο *request* ένα *Supported header field* που να περιγράφει της επιλογές για τις προεκτάσεις.

Το *tag* επιλογών (*option*) θα πρέπει να αναφέρεται στις προεκτάσεις που καθορίζονται στα *standards RFC*. Αυτή η ενέργεια έγινε για να αποφευχθούν παρεξηγήσεις μεταξύ **servers-clients**. Έτσι όλα τα *elements* ακολουθούν κάποιους κοινούς κανόνες και δεν ακολουθούν διαφορετικές οδηγίες που θα μπορούσε να είχε ορίσει ο κάθε κατασκευαστής χωριστά. Οι προεκτάσεις που ορίζονται σε πειραματικά και πληροφοριακά RFCs δεν χρησιμοποιούνται στο *Supported header field* ενός *request*.

Αν ο UAC επιμένει ότι ένας UAS καταλαβαίνει ένα *extension* το οποίο ο UAC επιθυμεί να εφαρμόσει στο *request* ούτως ώστε να επεξεργαστεί κατάλληλα το *request*, θα πρέπει να εισάγει το *Require header field* το οποίο περιγράφει τα *option tags* για το *extension*. Αν ο UAC επιθυμεί να εφαρμόσει ένα *extension* στο *request* και επιμένει πως κάθε *proxy* από τους οποίους διέρχεται το καταλαβαίνουν, τότε θα πρέπει να εισάγει ένα *Proxy-Require header field* το οποίο θα περιγράφει τα *option tags* για το συγκεκριμένο *extension*.

Όπως και με το *Supported header field* τα *option tags* στα *Require and Proxy-Require header fields* θα πρέπει να αναφέρουν *extensions* που υπάρχουν στα *standards* των RFCs.

Επιπρόσθετα στοιχεία μηνυμάτων

Όταν δημιουργηθεί ένα νέο *request*, θα έχουν δημιουργηθεί επίσης και τα *header fields* που περιγράφηκαν παραπάνω. Οποιαδήποτε προαιρετικά *header fields* προσθέτονται τότε.

Τα SIP *requests* μπορούν να περιέχουν ένα *message body* κωδικοποιημένο κατά MIME [18]. Ανεξάρτητα από τον τύπο του σώματος ενός *request*, υπάρχουν συγκεκριμένα *header fields* που πρέπει να τροποποιηθούν ώστε να χαρακτηρίζουν το σώμα.

Αποστολή του Request

Πριν την αποστολή ενός *request* υπολογίζεται ο προορισμός του. Ο προορισμός πρέπει να καθοριστεί εφαρμόζοντας DNS διαδικασίες, εκτός και αν ισχύουν άλλοι κανόνες λόγω εφαρμογής κάποιας τοπικής πολιτικής ασφαλείας (*Local policy)*. Αν το πρώτο στοιχείο στο *route set* καθορίζει κάποια καθορισμένη διαδρομή οι διαδικασίες θα πρέπει να εφαρμοστούν στην *request URI* του *request*. Διαφορετικά οι διαδικασίες εφαρμόζονται στην πρώτη τιμή του *Route header field* του *request* (αν υπάρχει) ή στην *request URI* αν δεν υπάρχει *Route header field*. Αυτές οι διαδικασίες προδίδουν μια καθορισμένη σειρά από διευθύνσεις, *ports* και *transports* που θα πρέπει να χρησιμοποιηθούν. Ανεξάρτητα από ποια URI χρησιμοποιείται σαν *input* σε αυτές τις διαδικασίες, αν η *request URI* υποδεικνύει μια SIPS *resource* τότε ο UAC πρέπει να ακολουθήσει τις αντίστοιχες διαδικασίες που θα ακολουθούσε για *input* μια SIPS URI.

Η *local policy* (τοπική πολιτική – σύνολο από κανόνες) μπορεί να καθορίζει ένα σύνολο από εναλλακτικούς προορισμούς διαθέσιμους για προσπάθεια αποστολής. Αν η *request URI* περιέχει μία SIPS URI, οποιοσδήποτε εναλλακτικός προορισμός πρέπει να συνοδεύεται από TLS [21]. Πέρα από αυτό, δεν υπάρχουν περιορισμοί στους εναλλακτικούς προορισμούς αν το *request* δεν περιέχει *Route header field*. Αυτή η λειτουργία παρέχει ένα τρόπο για να ορίσει κανείς έναν *outbound proxy* σαν ένα εναλλακτικό *pre-existing route set*. Ωστόσο αυτή η προσέγγιση για τον ορισμό ενός *outbound proxy* δεν προτείνεται, καθώς είναι προτιμότερο να χρησιμοποιείται μια *pre-existing route set* με ένα απλό URI. Αν το *request* περιέχει κάποιο *Route header field* τότε θα πρέπει να σταλεί στις τοποθεσίες που προκύπτουν από την αρχική τιμή, αλλά μπορεί να σταλεί και σε οποιονδήποτε *server* όπου ο UA θεωρεί πως θα τηρηθεί η πολιτική δρομολόγησης που περιγράφεται στο RFC 3261 [9]. Συγκεκριμένα ο UAC που έχει ρυθμιστεί με έναν *outbound proxy* θα πρέπει να προσπαθήσει να στείλει το *request* στην τοποθεσία που περιγράφεται από την πρώτη τιμή του *Route header field* αντί να το στείλει στον *outbound proxy* όπως θα έκανε με τα υπόλοιπα μηνύματα.

Έτσι εξασφαλίζεται ότι οι *outbound proxies* που δεν προσθέτουν τιμές στο *Record-Route header field* δεν θα συμπεριλαμβάνονται στο *path* των επόμενων μηνυμάτων. Επιτρέπεται στα *end-points* που δεν μπορούν να εντοπίσουν την πρώτη *Route* URI να μεταβιβάσουν αυτό το *task* στον *outbound proxy*.

Ο UAC θα πρέπει να ακολουθεί τις διαδικασίες που έχουν οριστεί για τα *stateful elements* προσπαθώντας κάθε διεύθυνση μέχρι να έρθει σε επαφή με κάποιον *server*. Κάθε προσπάθεια σημαίνει και διαφορετικό *transaction* και όπως γίνεται κατανοητό το κάθε ένα έχει διαφορετικό *via header* με μια καινούργια *branch parameter*. Επιπλέον η τιμή *transport* στο *via header field* καθορίζεται ανάλογα με το *transport* που επιλέχτηκε για τον *server*.

Επεξεργασία των Responses

Τα *responses* επεξεργάζονται πρώτα από το *transport layer* και μετά περνούν για επεξεργασία από το *transaction layer*. Το *transaction layer* εκτελεί την επεξεργασία που πρέπει να κάνει και στέλνει το *response* πάνω στον *Transaction User*. Ανάλογα με την λειτουργία γίνεται και διαφορετικά η επεξεργασία από το *Transaction User*. Ωστόσο υπάρχουν κάποιοι γενικοί κανόνες που εφαρμόζονται, ανεξάρτητοι από την λειτουργία και περιγράφονται παρακάτω.

Σφάλματα στο Transaction Layer

Σε μερικές περιπτώσεις τα *responses* που επιστρέφονται από το *transaction layer* δεν είναι SIP *messages* αλλά ένα *transaction layer error*. Όταν ένα *time out error* λαμβάνεται από το *transaction layer* τότε πρέπει να ερμηνευθεί σαν ένα *status code 408* (*Request Timeout*). Αν αναφερθεί ένα *fatal transport error* από το *transaction layer* (συνήθως εξαιτίας των *fatal* ICMP *errors* στο UDP ή σε *connection failures* του TCP) η κατάσταση πρέπει να ερμηνευθεί σαν ένα *status code 503* (*Service Unavailable*).

Μη αναγνωρίσιμα Responses

Ο UAC πρέπει να μεταχειρίζεται οποιαδήποτε *responses* που δεν αναγνωρίζει σαν ισοδύναμα με *code x00 response* του ίδιου *class*, και θα πρέπει να είναι ικανός να επεξεργαστεί τα *codes x00* για όλα τα *classes*. Για παράδειγμα, αν ένας UAC λάβει ένα μη αναγνωρίσιμο *code 431*, μπορεί ασφαλέστατα να θεωρήσει ότι κάτι πήγε στραβά με το *request* που δημιούργησε και θα μεταχειριστεί το *response* σαν να έλαβε ένα *response code 400 (Bad Request)*. Ο UAC πρέπει να μεταχειρίζεται οποιαδήποτε προσωρινό *response* διαφορετικό από *100* σαν *status code 183* (*Session Progress*). Συνεπώς ένας UAC πρέπει να είναι οπωσδήποτε ικανός να επεξεργαστεί *responses* με *status code 100* και *183*.

Αν σε κάποιο *response* υπάρχουν περισσότερες από μία τιμές στο *via header field* τότε ο UAC πρέπει να απορρίψει το μήνυμα. Η παρουσία επιπλέον τιμών στο *via header field* που δημιουργείται από τον *originator* υποδηλώνει ότι το μήνυμα δρομολογήθηκε λανθασμένα ή πιθανότατα είναι αλλοιωμένο (*corrupted*).

Επεξεργασία των 3xx Responses

Όταν λαμβάνεται ένα *redirection response* (για παράδειγμα ένα *response* με *status code 301*) οι *clients* θα πρέπει να χρησιμοποιούν την (τις) URI(s) στο *Contact header field* για να σχηματοποιήσουν ένα ή περισσότερα νέα *requests* βασισμένα στο *redirected request*. Αυτή η διαδικασία παρουσιάζει ομοιότητες με αυτή του *proxy recursing* σε ένα *response* με *class 3xx*. Ο *client* αρχίζει με έναν αρχικό στόχο ο οποίος περιέχει ακριβώς μία URI. Η URI αντλείται από την *request URI* του *original request*. Αν ο *client* επιθυμεί να σχηματοποιήσει νέα *requests* βασισμένα σε *responses* με *class 3xx* τοποθετεί τις URIs μέσα στο *target set*. Όπως και στο *proxy recursion* ο *client* που επεξεργάζεται *responses class 3xx* δεν μπορεί να τοποθετήσει περισσότερες από μια φορά τις δοσμένες URI στο *target set*. Αν το *original request* είχε μια SIPS URI στο *request URI* ο *client* μπορεί να επιλέξει την επιστροφή (*recurse*) σε μια *non*-SIPS URI αλλά θα πρέπει να ειδοποιήσει τον χρήστη για το *redirection* σε μια μη-ασφαλή URI.

Οποιοδήποτε νέο *request* μπορεί να λάβει *response* της *class 3xx* περιέχοντας την *original* URI σαν *contact*. Δύο τοποθεσίες μπορούν να σεταριστούν ώστε να κάνουν *redirect* η μία στην άλλη. Τοποθετώντας κάποιο δοσμένο URI στο *target set* μία φορά εμποδίζονται άπειρα *redirection loops*.

Καθώς το *target set* αυξάνεται, ο *client* μπορεί να δημιουργήσει νέα *requests* στα URIs σε οποιαδήποτε σειρά. Υπάρχει ένας κοινός μηχανισμός για να καθορίζει το *set* από την *q* παράμετρο στο *Contact header field*. Τα *requests* στα URIs μπορούν δημιουργηθούν σειριακά ή παράλληλα. Η κάθε προσέγγιση επεξεργάζεται ομάδες ελαττωμένων *q*-τιμών σειριακά ή επεξεργάζεται τα URIs σε κάθε *q*- ομάδα παράλληλα.

Αν η επαφή με μια διεύθυνση της λίστας καταλήξει σε αποτυχία, όπως αυτές ορίζονται αμέσως παρακάτω, το στοιχείο μετακινείται στην επόμενη διεύθυνση της λίστας έως ότου η λίστα εξαντληθεί. Αν η λίστα εξαντληθεί τότε το *request* έχει αποτύχει.

Οι αποτυχίες θα πρέπει να ανιχνεύονται μέσω των *failure response codes* (*codes* μεγαλύτεροι από 399). Για τα *network errors* ο *client transaction* θα αναφέρει οποιαδήποτε *transport layer failures* στον *transaction user*. Θα πρέπει να σημειωθεί πως κάποιοι *response codes* υποδεικνύουν πως το *request* μπορεί να επαναληφθεί. Τα *requests* που δοκιμάζονται εκ νέου δεν πρέπει να θεωρούνται σαν *failures*.

Όταν ληφθεί ένα *failure* για κάποια συγκεκριμένη *contact address*, ο *client* θα πρέπει να δοκιμάσει την επόμενη *contact address*. Αυτό περικλείει την δημιουργία ενός νέου *transaction* για την παράδοση του νέου *request*.

Για την δημιουργία του *request* βασισμένου σε μια *contact address* ενός *response 3xx*, ο UAC πρέπει να αντιγράψει ολόκληρη την URI από το *target set* και να την τοποθετήσει στο *request URI*, εκτός των "*method-param*" και "*header*" URI παραμέτρων. Χρησιμοποιεί τις "*header*" παραμέτρους για να δημιουργήσει τιμές για τα *header fields* του *request* γράφοντας πάνω στις *header field* τιμές του *redirected request*.

Θα πρέπει να σημειωθεί πως σε μερικές περιπτώσεις τα *header fields* που επικοινωνούσαν με την *contact address* μπορεί να προσαρτώνται στα υπάρχοντα *header fields* του *original redirected request*. Σαν γενικός κανόνας, αν κάποιο *header field* μπορεί να δεχτεί μια λίστα από *comma separated* τιμές τότε η νέα τιμή του *header field* μπορεί να προσαρτηθεί σε οποιαδήποτε προυπάρχουσα τιμή του *original redirected message*. Αν το *header field* δεν δέχεται πολλαπλές τιμές, η τιμή στο *original redirected request* μπορεί να γραφεί από πάνω με την τιμή του *header field* που αναγράφεται στο *contact address*. Για παράδειγμα, αν μια *contact address* επιστρέφεται με την επόμενη τιμή

```
sip:user@host?Subject=foo&Call-Info=http://www.foo.com>
```

τότε οποιοδήποτε *subject header field* στο *original redirected request* πανωγράφεται, αλλά το HTTP URL προστίθεται μερικά σε οποιαδήποτε υπάρχουσα τιμή του *Call-Info header field*.

Στον UAC προτείνεται η επαναχρησιμοποίηση των *header fields To,From* και *Call-ID* που χρησιμοποιήθηκαν στο *original redirected request*, αλλά ο UAC μπορεί να επιλέξει την ενημέρωση των πληροφοριών αυτών γα παράδειγμα το *Call-ID header field* όταν πρόκειται για νέα *requests*.

Τέλος μόλις «κατασκευαστεί» το νέο *request* στέλνεται χρησιμοποιώντας ένα νέο *client transaction*, έτσι πρέπει να έχει νέο *branch ID* στο *top via field* όπως περιγράφτηκε προηγουμένως.

Σε όλες τις υπόλοιπες περιπτώσεις τα *requests* που στέλνονται σαν απάντηση σε *redirected responses* πρέπει να χρησιμοποιούν τα *header fields* και *bodies* του *original request*.

Σε μερικές περιπτώσεις, οι τιμές στο *Contact header field* μπορούν να αποθηκευτούν προσωρινά ή μόνιμα στον UAC εξαρτώμενα από το ληφθέν *status code* ή την παρουσία κάποιου *expiration interval*.

Επεξεργασία 4xx Responses

Κάποια συγκεκριμένα *4xx response codes* απαιτούν συγκεκριμένη επεξεργασία από τον UA ανεξάρτητα της μεθόδου που χρησιμοποιήθηκε.

Αν ληφθεί ένα *401 (Unauthorized)* ή *407 (Proxy Authenticate Required) response* τότε ο UA θα πρέπει να ακολουθήσει τις *authorization procedures* (διαδικασίες εξουσιοδότησης) για να δοκιμάσει ξανά το *request* παρέχοντας *credentials* (διαπιστευτήρια)

Αν ληφθεί ένα *413 (Request Entity Too Large) response*, τότε το *request* περιείχε κάποιο *message body* που ήταν μεγαλύτερο από αυτό που μπορούσε να δεχτεί ο UAS. Αν είναι δυνατόν ο UAC πρέπει να δοκιμάσει ξανά το *request* είτε παραλείποντας το σώμα είτε χρησιμοποιώντας μικρότερο σώμα μηνύματος.

Αν ληφθεί ένα *415 (Unsupported Media Type) response*, τότε το *request* περιείχε τύπους *media* οι οποίοι δεν υποστηρίζονται από τον UAS. Ο UAC θα πρέπει να ξαναστείλει το *request*, αυτή τη φορά χρησιμοποιώντας σαν περιεχόμενο κάποιο που να περιέχεται στην λίστα του *Accept header field* του *response*, καθώς και κωδικοποιήσεις αυτές που υπάρχουν στο *Accept-Encoding header field* του ίδιου *response*, και γλώσσες αυτές που φαίνονται στο *Accept-Language header field* του *response*.

Αν ληφθεί ένα *416 (Unsupported URI Scheme) response*, το *request URI* χρησιμοποίησε ένα *URI-scheme* που δεν παρέχεται από τον *server*. Ο *client* θα πρέπει να δοκιμάσει ξανά το *request* αυτή τη φορά χρησιμοποιώντας ένα SIP URI.

Αν ληφθεί ένα *420 (Bad extension) response* τότε το *request* περιείχε ένα *Require* ή *Proxy-Require header field* το οποίο χρησιμοποιούσε ένα *option-tag* για κάποιο *feature* (χαρακτηριστικό) που δεν παρείχε ο *proxy* ή ο UAS. Ο UAC θα πρέπει να ξαναπροσπαθήσει το *request* αυτή τη φορά παραλείποντας οποιαδήποτε *extensions* που περιέχονται στο *Unsupported header field* του *response*.

Σε όλες τις παραπάνω περιπτώσεις, το *request* επαναλαμβάνεται δημιουργώντας ένα καινούργιο *request* με τις κατάλληλες τροποποιήσεις. Αυτό το νέο *request* αποτελείται από ένα νέο *transaction* και θα πρέπει να έχει την ίδια τιμή στα πεδία *Call-ID, To,* και *From* με το προηγούμενο *request*. Το πεδίο *CSeq* θα

πρέπει να περιέχει ένα νέο *sequence number* αυξημένο κατά ένα σε σχέση με το προηγούμενο.

2.2.2 Συμπεριφορά του User Agent Server

Όταν ένα *request* το οποίο δεν ανήκει σε κάποιο *dialog* υφίσταται επεξεργασία από τον UAS, υπάρχει ένα σύνολο από κάποιους κανόνες που ακολουθούνται ανεξάρτητα της λειτουργίας που χρησιμοποιήθηκε.

Θα πρέπει να σημειωθεί πως η επεξεργασία ενός *request* είναι καθαρά ατομική διαδικασία. Αν ένα *request* γίνει δεκτό από τον UAS τότε πρέπει να εκτελεστούν όλες οι αλλαγές κατάστασης που σχετίζονται με το *request*.

Οι UASs θα πρέπει να επεξεργάζονται τα *requests* στην σειρά των βημάτων που ακολουθούν (δηλαδή ξεκινώντας με *authentication*, μετά ερευνώντας την λειτουργία, τα *header fields* και όλα τα υπόλοιπα που αναφέρονται σε αυτή την ενότητα).

Έρευνα λειτουργίας

Μόλις ένα *request* έχει πιστοποιηθεί (*authenticated*), εκτός και αν έχει επιλεγεί η παράλειψη του *authentication*, ο UAS πρέπει να ερευνήσει την λειτουργία του *request*. Αν ο UAS αναγνωρίζει αλλά δεν υποστηρίζει την λειτουργία πρέπει να δημιουργήσει ένα *response 405 (Method Not Allowed)*. Οι διαδικασίες για την δημιουργία του *response* περιγράφονται αργότερα. Ο UAS πρέπει επίσης να προσθέσει ένα *Allow header field* στο *response 405 (Method Not Allowed)*. Το *Allow header field* πρέπει να καταγράφει όλες τις λειτουργίες που παρέχονται από τον UAS ο οποίος δημιούργησε το μήνυμα.

Αν η λειτουργία είναι μία από τις παρεχόμενες από τον UA *Server*, τότε η επεξεργασία συνεχίζεται κανονικά.

Μη Αναγνωρίσιμο Header Field

Αν ο UAS δεν καταλαβαίνει ένα *header field* ενός *request* (πιθανότατα *το header field* δεν χαρακτηρίζεται στο RFC 3261 [9] ή κάποιο υποστηριζόμενο *extension*) τότε ο UAS πρέπει να αγνοήσει το συγκεκριμένο *header field* και θα πρέπει να συνεχίσει την επεξεργασία του μηνύματος. Ένας UAS θα πρέπει να αγνοεί οποιαδήποτε δυσνόητα *header fields* τα οποία δεν είναι αναγκαία για την επεξεργασία του *request*.

Το και Request-URI

Το *header field "To"* προσδιορίζει τον *original* παραλήπτη του *request* που προορίστηκε από τον χρήστη ο οποίος προσδιορίζεται στο *From field*. Ο *original* παραλήπτης πιθανότατα θα είναι ο UAS που επεξεργάζεται το *request*. Αν δεν ισχύει κάτι τέτοιο τότε αυτό μπορεί να έχει προκύψει εξαιτίας του *call forwarding* ή άλλων *proxy operations*. Ο UAS μπορεί να εφαρμόσει οποιαδήποτε *policy* επιθυμεί για να καθορίσει πότε θα δέχεται *requests* ανάλογα με το αν το "*To*" *header field* περιέχει

την ταυτότητα του αναφερόμενου UAS ή όχι. Ωστόσο προτείνεται ο UAS να δέχεται *requests* ακόμα και αν δεν αναγνωρίζει το URI *scheme* (για παράδειγμα ένα tel URI) στο "*To*" *header field* ή το "*To*" *header field* δεν απευθύνεται σε κάποιον γνωστό ή τρέχων χρήστη του UAS. Στην τελευταία περίπτωση θα πρέπει να δημιουργήσει ένα *response* με *status code 403* (*Forbidden*) και να το στείλει στον *transaction server* για εκπομπή.

Το *request URI* προσδιορίζει τον UAS που πρέπει να επεξεργαστεί το *request*. Αν το *request URI* χρησιμοποιεί κάποιο *scheme* που δεν παρέχεται από τον UAS, ο τελευταίος θα πρέπει να απορρίψει το *request* με ένα *response 416 (Unsupported URI Scheme)*. Αν το *request URI* δεν προσδιορίζει μια διεύθυνση, στην οποία ο UAS είναι πρόθυμος να δεχτεί *requests*, θα πρέπει να απορρίψει το *request* με ένα *response 404 (Not Found)*. Τυπικά ένας UA που χρησιμοποιεί την λειτουργία REGISTER για να προσαρτήσει την *address-of-record* σε μια συγκεκριμένη *contact address* θα βλέπει τα *requests* των οποίων η *request URI* ταυτίζεται με την προαναφερθείσα *contact address*.

Merged Requests

Αν το *request* δεν έχει κάποιο *tag* στο "*To*" *header field*, τότε ο πυρήνας του UAS πρέπει να διερευνήσει το *request* ως προς τα απερχόμενα *transactions*. Αν τα *From tag*, *Call-ID*, και *CSeq* ταιριάζουν απόλυτα με κάποια που ανήκουν σε κάποιο εξερχόμενο *transaction* αλλά το *request* δεν ταιριάζει με αυτό, τότε ο UAS θα πρέπει να δημιουργήσει ένα *response 482 (Loop Detected)* και να το περάσει στον *server transaction*.

Αν το ίδιο *request* φτάσει στον *server* περισσότερες από μία φορές ακολουθώντας διαφορετικά *paths* αυτό μπορεί να συμβαίνει εξαιτίας του *forking*. Ο UAS θα επεξεργαστεί το πρώτο *request* και θα ανταποκριθεί με ένα *response 482 (Loop Detected)* στα υπόλοιπα.

Require

Υποθέτοντας ότι ο UAS αποφασίζει πως είναι ο κατάλληλος για την επεξεργασία του *request*, ελέγχει το *Require header field* (αν υπάρχει).

Το *Require header field* χρησιμοποιείται από τον UAC για να δηλώσει σε κάποιον UAS τα SIP *extensions* που ο UAC περιμένει από τον UAS ώστε ο τελευταίος να είναι σε θέση να επεξεργαστεί το *request κατάλληλα*. Αν ο UAS δεν καταλαβαίνει ένα *option tag* που βρίσκεται στο *Require header field*, θα πρέπει να αντιδράσει με ένα *respond 420* (*Bad Extension*). Επιπλέον ο UAS πρέπει να προσθέσει ένα *Unsupported header field* όπου θα καταγράφονται εκεί τα *option tags* που δεν έγιναν κατανοητά στο *request*.

Θα πρέπει να σημειωθεί ότι τα *Require* και *Proxy-Require* δεν μπορούν να χρησιμοποιηθούν σε ένα SIP CANCEL *request* ή σε ένα ACK *request* που στάλθηκε για κάποιο *non-2xx response*. Αυτά τα *header fields* θα πρέπει να αγνοηθούν αν υπάρχουν στα *requests*.

Ένα ACK *request* για κάποιο *non-2xx response* πρέπει να περιέχει μόνο τις τιμές των *Require* και *Proxy-Require* που υπήρχαν και στο *initial request*.

Παράδειγμα

UAC->UAS:
```
INVITE sip:watson@bell-telephone.com SIP/2.0
Require: 100rel
```

UAS->UAC:
```
SIP/2.0 420 Bad Extension
Unsupported: 100rel
```

Με αυτόν τον τρόπο εξασφαλίζεται ότι η αλληλεπίδραση client-server θα προχωρήσει χωρίς καθυστερήσεις όταν όλα τα *options* έχουν κατανοηθεί και από τις δύο πλευρές. Καθυστερήσεις μπορούν να προκύψουν μόνο όταν κάποιο *option* είναι μη κατανοητό. Σε ένα ταιριαστό ζευγάρι *client-server* η αλληλεπίδραση εξελίσσεται αρκετά γρήγορα κερδίζοντας χρόνο από *round trips* που συμβαίνουν σε μηχανισμούς διαπραγμάτευσης.

Επεξεργασία περιεχομένου μηνύματος

Θεωρώντας ότι ο UAS κατανοεί οποιαδήποτε *extensions* απαιτούνται από τον *client*, θα περάσει στην εξέταση του σώματος του μηνύματος και των *header fields* που το περιγράφουν. Αν υπάρχουν οποιαδήποτε *message bodies* των οποίων τα *type*, (φαίνεται από *Content-type*), *language* (φαίνεται από *Content-Language*), ή κωδικοποίηση (φαίνεται από *Content-Encoding*) δεν είναι κατανοητά, και αυτά τα *bodies* δεν είναι προαιρετικά (φαίνεται από το *Content-Disposition header field*), τότε ο UAS πρέπει να απορρίψει το *request* με ένα *response 415* (*Unsupported Media Type*). Το *respond* θα πρέπει να περιέχει ένα *Accept header field* στο οποίο θα καταγράφονται όλοι οι τύποι των *bodies* που είναι κατανοητοί στην περίπτωση που το *request* περιείχε *type body not-supported* από τον UAS. Αν το *request* περιείχε *content-encodings* μη κατανοητό από τον UAS, το *response* θα πρέπει να περιέχει ένα *Accept-Encoding header field* που να καταγράφει όλα τα *encodings* που είναι κατανοητά από τον UAS. Αν το *request* περιείχε κάποιο περιεχόμενο με *languages* μη κατανοητές από τον UAS τότε το *response* θα πρέπει να περιέχει ένα *Accept-Language header field* που να καταγράφει τις *languages* που είναι κατανοητές από τον UAS. Πέρα από αυτούς τους ελέγχους, το *body handling* εξαρτάται από την λειτουργία και τον τύπο αυτού.

Εφαρμογή των Extensions

Ένας UAS που θα επιθυμούσε να εφαρμόσει κάποια *extensions* δημιουργώντας το *response,* δεν έχει αυτό το δικαίωμα εκτός και αν για αυτό το *extension* υπάρχει *support* που φαίνεται στο *Supported header field* του request. Αν το επιθυμητό *extension* δεν παρέχεται (*not supported*), ο *server* θα πρέπει να βασιστεί μονάχα στο *baseline* SIP και σε οποιαδήποτε άλλα *extensions* υποστηρίζει ο *client*. Σε σπάνιες *περιπτώσεις*, όπου ο *server* δεν μπορεί να επεξεργαστεί το *request* χωρίς το *extension*, τότε μπορεί να στείλει το *response 421 (Extension Required)*. Το *421* σημαίνει πως ο *server* αδυνατεί να δημιουργήσει το κατάλληλο *response* χωρίς να χρησιμοποιήσει κάποιο *extension*. Το αναγκαίο *extension* πρέπει να περιέχεται στο *require header field* του *response*. Αυτή η συμπεριφορά όμως δεν προτείνεται από το

RFC 3261 [9] αφού κάτι τέτοιο θα μπορούσε να καταστρέψει την συνεργασία SIP *elements* διαφορετικών *vendors*.

Επεξεργασία του Request

Θεωρώντας ότι όλοι οι έλεγχοι στις προηγούμενες παραγράφους έχουν τελειώσει με επιτυχία, η επεξεργασία που εφαρμόζει ο UAS αποκτάει ιδιαίτερο χαρακτήρα ανάλογα με την λειτουργία που έχει χρησιμοποιηθεί στο request.

Αποστολή προσωρινού Response

Μία σημαντική οδηγία ανεξάρτητη της λειτουργίας, για την δημιουργία *responses* αναφέρει ότι οι UASs δεν πρέπει να στέλνουν κάποιο *provisional response* (προσωρινό) για κάποιο *non*-INVITE *request*. Ουσιαστικά οι UASs θα πρέπει να εκδίδουν ένα *final response* σε οποιοδήποτε *non*-INVITE *request* το συντομότερο δυνατό.

Όταν δημιουργείται ένα *response 100 (Trying)* οποιοδήποτε *Timestamp header field* είναι παρών στο *request* πρέπει να αντιγραφεί στο *response*. Αν υπάρξει κάποιο *delay* στην δημιουργία του *response*, ο UAS θα πρέπει να προσθέσει ένα *delay value* στο *Timestamp value* του *response*. Η τιμή αυτή θα περιέχει την διαφορά χρόνου μεταξύ της χρονικής στιγμής που λήφθηκε το *request* και της χρονικής στιγμής που στάλθηκε το *response*.

Headers & Tags

Το *From field* ενός *response* πρέπει να ισούται με το *From field* του *request*. Το *Call-ID header field* που ανήκει στο *response* θα πρέπει να ισούται με το *Call-ID header field* του *request*. Το *CSeq header field* που βρίσκουμε στο *response* θα πρέπει να ισούται με το *CSeq header field* που υπάρχει στο αντίστοιχο *header field* του *request*. Οι τιμές στο *Via header field* θα πρέπει να ταυτίζονται με τις τιμές στο *Via header field* του *request* και θα πρέπει να διατηρούν και την ίδια σειρά.

Αν το *request* περιείχε κάποιο *"To" tag*, τότε το *"To" header field* στο *response* πρέπει να ισούται με αυτό του *request*. Ωστόσο αν το *"To" header field* στο *request* δεν περιείχε κάποιο *tag*, η URI στο *"To" header field* του *response* θα πρέπει να ισούται με την URI που υπάρχει στο *"To" header field*. Επιπλέον ο UAS πρέπει να προσθέσει ένα *tag* στο *"To" header field* του *response* (με εξαίρεση το *response 100 (Trying)* όπου σε αυτήν την περίπτωση το *tag* είναι προαιρετικό). Αυτό εξυπηρετεί ώστε να προειδοποιεί ο UAS ότι αντιδράει, πιθανότατα καταλήγοντας σε κάποιο *dialog*. Το ίδιο *tag* πρέπει να χρησιμοποιηθεί για όλα τα *responses* σε αυτό το *request*, *final* και *provisional* (εξαιρώντας ξανά τα *responses 100*).

2.3 Redirect Servers

Σε κάποιες αρχιτεκτονικές μπορεί να είναι επιθυμητή η μείωση του *processing load* στους *proxy servers* που είναι υπεύθυνοι για την δρομολόγηση των

requests, και η βελτίωση της δυναμικότητας του *signaling path* χρησιμοποιώντας μεθόδους *redirection*.

Το *redirection* επιτρέπει στους *servers* να στέλνουν την *routing information* για κάποιο *request* πίσω σαν *response* στον *client*, αφαιρώντας τον εαυτό τους από το *loop* άλλων μηνυμάτων για την συγκεκριμένη *transaction*. Έτσι λοιπόν δεν χάνουν τον στόχο τους (εύρεση της τοποθεσίας ενός χρήστη) απλά δεν μεσολαβούν στο *transaction*. Όταν ο *originator* ενός *request* λάβει κάποιο *redirection response* θα στείλει ένα νέο *request* βασισμένο στη URI(s) που έλαβε. Με την εκπομπή των URIs από τον πυρήνα του δικτύου στις άκρες του, το *redirection* επιτρέπει μια αξιοσημείωτη κλιμάκωση του δικτύου (*network trafic shaping*).

Ένας *redirection server* λογικά αποτελείται από ένα *transaction server layer* και έναν *transaction user* ο οποίος έχει πρόσβαση σε μια *location service*. Η *location service* ουσιαστικά είναι μια *database* που περιέχει ως καταχωρίσεις, τις αντιστοιχήσεις (*mappings*) μεταξύ ενός URI και κάποιων *set* ενός ή περισσοτέρων *alternative locations* στα οποία ο στόχος του συγκεκριμένου URI μπορεί να εντοπιστεί.

Ένας *redirect server* δεν διαπραγματεύεται με SIP *requests* από μόνος του. Αν δεχτεί ένα *request* (εκτός από CANCEL) τότε ή θα απορρίψει το *request* ή θα ετοιμάσει μια λίστα με τα *alternative locations* από την *location service* και θα επιστρέψει ένα τελικό *response* με *class 3xx*. Στα CANCEL *requests* θα πρέπει να απαντήσει με κάποιο *2xx response*. Αυτό το *response* ολοκληρώνει και το SIP *transaction*. Αποτελεί ευθύνη του *client* η ανίχνευση των *forwarding loops* μεταξύ των *redirect servers*.

Όταν ο *redirect server* απαντάει με ένα *3xx response* σε κάποιο *request*, τοποθετεί την λίστα των alternative locations (μία η περισσότερες) στο *Contact header field*. Στο *Contact header field* μπορεί επίσης να δοθεί μια "*expires*" παράμετρος η οποία δηλώνει την διάρκεια ζωής των *Contact data*.

Το *Contact header field* περιέχει URIs τα οποία πληροφορούν για νέα *locations* ή *usernames* στα οποία μπορεί να επιχειρηθεί κάποιο *request*, μπορεί απλά να καθορίζει κάποιες επιπλέον *transport parameters*. Ένα *301 (Moved Permanently)* ή *302 (Moved Temporarily) response* μπορεί να δίνει την ίδια *location* ή *username* που χρησιμοποιήθηκε και στο αρχικό *request* απλά μπορεί να δίνει κάποιες επιπλέον παραμέτρους (*transport parameters*) όπως διαφορετικό *server* ή *multicast address* ή αλλαγή του *transport* στο SIP π.χ. από UDP σε TCP και το αντίστροφο.

Ωστόσο οι *redirect servers* δεν πρέπει να επιστρέφουν ένα *request* σε κάποιο URI ίσο με αυτό που υπάρχει στο *request URI*, αντί αυτού ο *server* μπορεί να στέλνει το *request* στον προορισμό του (*destination* URI) με την προϋπόθεση ότι το URI δεν δείχνει προς τον εαυτό του ή μπορεί να το απορρίψει με ένα *404*.

Αν ένας *client* χρησιμοποιεί κάποιον *outbound proxy* και αυτός ο *proxy* ουσιαστικά κάνει *redirection* στα *requests*, προκύπτει η προοπτική για άπειρα *redirection loops*.

Θα πρέπει να σημειωθεί πως η τιμή στο *Contact header field* μπορεί να αναφέρεται σε διαφορετική *resource* από αυτή που κλήθηκε αρχικά. Για παράδειγμα μια SIP κλήση που συνδέθηκε σε ένα PSTN *gateway* μπορεί να επιθυμεί μια ειδική ανακοίνωση όπως «Ο αριθμός που καλέσατε έχει αλλάξει».

Το *Contact header field* ενός *response* μπορεί να περιέχει οποιαδήποτε κατάλληλη URI η οποία δείχνει την διεύθυνση στην οποία μπορεί να εντοπιστεί ο καλούμενος – δεν είναι *limited* στα SIP URIs. Θα μπορούσε να περιέχει URI για τηλέφωνο, FAX, ή *irc* ή *mailto:* (RFC 2368) [13].

Η παράμετρος *expires* του *Contact header field* δείχνει για πόση διάρκεια η URI είναι *valid*. Η τιμή αυτή είναι ένας αριθμός που δείχνει τον χρόνο σε δευτερόλεπτα. Αν αυτή η παράμετρος δεν δίνεται τότε η τιμή του *Expires header field* καθορίζει την διάρκεια ζωής του URI.

Οι *redirect servers* θα πρέπει να αγνοούν *features* που δεν καταλαβαίνουν (όπως *unrecognized header fields*, *unknown options tags* του *Require*, ή ονόματα λειτουργιών) και θα πρέπει να προχωρούν στο *redirection* του *request*.

2.4 Λειτουργία CANCEL

Στο προηγούμενο κεφάλαιο έγινε μια μικρή αναφορά στην συμπεριφορά ενός UA κατά την δημιουργία των *requests* και επεξεργασία των *responses* για κάθε λειτουργία. Σε αυτή την ενότητα θα γίνει λόγος για μια λειτουργία γενικών καθηκόντων, την CANCEL.

Το CANCEL *request*, όπως υπονοεί το όνομα, χρησιμοποιείται για να ακυρώσει ένα *previous request* που στάλθηκε από τον *client*. Συγκεκριμένα ζητάει από τον UAS τον τερματισμό της επεξεργασίας για το *request* και την δημιουργία ενός *error response* για το ίδιο *request*. Το CANCEL δεν έχει καμία επίπτωση σε ένα *request* για το οποίο ο UAS έχει δημιουργήσει κάποιο *final response*. Εξαιτίας αυτού, είναι πιο πρακτικό να γίνεται CANCEL σε *requests* τα οποία απαιτούν από τον *server* αρκετό χρονικό διάστημα για δημιουργήσει το *response*. Για αυτό το λόγο το CANCEL χρησιμοποιείται κυρίως στα INVITE *request* στα οποία μπορεί να χρειαστεί αρκετό χρονικό διάστημα για να δημιουργηθεί το *response*. Σε αυτή την περίπτωση ο UAS που λαμβάνει ένα CANCEL *request* για κάποιο INVITE και δεν έχει στείλει ένα *final response*, θα σταματήσει το *ringing* και θα ανταποκριθεί στο INVITE με ένα *error response*.

Τα CANCEL *requests*, δημιουργούνται και στέλνονται από τους UAs αλλά και από τους *proxies*. Ένας *stateful proxy* ανταποκρίνεται σε κάποιο CANCEL όχι όπως θα ανταποκρινόταν σε ένα οποιοδήποτε *request* (δηλ. απλά κάνοντας *forward*). Για αυτό το λόγο το CANCEL αναφέρεται και σαν "*hop-by-hop*" *request*, αφού χαίρει ειδικής μεταχείρισης σε κάθε *hop* που εκτελεί.

Συμπεριφορά του Client

Ένα CANCEL *request* δεν θα πρέπει να στέλνεται για να ακυρώσει κάποιο *request* διαφορετικό από INVITE.

Εφόσον στα *requests* εκτός των INVITE η ανταπόκριση είναι άμεση, η αποστολή ενός CANCEL σε ένα *non*-INVITE *request* θα δημιουργούσε έναν αγώνα δρόμου με αμφίβολα αποτελέσματα.

Οι παρακάτω διαδικασίες λαμβάνουν μέρος στην κατασκευή ενός CANCEL *request*. Τα *Request-URI*, *Call-ID*, *To*, το αριθμητικό μέρος του *CSeq*, και το *From header fields* σε ένα CANCEL *request* πρέπει να είναι πανομοιότυπα με αυτά που υπάρχουν στο *request* που ακυρώνεται, ακόμα και τα *tags*. Ένα CANCEL που δημιουργήθηκε από κάποιον *client* πρέπει να έχει μονάχα το *Via header field* ίδιο με την τιμή που βρίσκεται στην κορυφή του *Via header field* στο *request* που ακυρώνεται. Χρησιμοποιώντας τις ίδιες τιμές στα παραπάνω *header field* γίνεται ευκολότερη η αντιστοίχηση του CANCEL με το *request* που πρόκειται να ακυρωθεί. Ωστόσο το *method part* του *CSeq header field* πρέπει να έχει την τιμή ενός CANCEL.

Αν το *request* που ακυρώνεται περιέχει ένα *Route header field*, τότε το CANCEL *request* θα πρέπει να περιέχει και αυτό την τιμή του *Route header field*. Αυτό συμβαίνει έτσι ώστε να είναι δυνατόν από τους *stateless proxies* να δρομολογήσουν σωστά το CANCEL *request*.

Το CANCEL *request* δεν μπορεί να περιέχει *require* ή *Proxy-Require header fields*.

Μόλις το CANCEL *request* ετοιμαστεί, ο *client* ελέγχει αν έχει δεχτεί κάποιο *response* (*provisional* ή *final*) για το *request* που ακυρώνεται. (στη βιβλιογραφία αναφέρεται και "*original request*")

Αν δεν έχει ληφθεί κάποιο *provisional response*, τότε δεν μπορεί να σταλεί το CANCEL. Αντιθέτως ο *client* πρέπει να περιμένει για την άφιξη ενός *provisional response* πριν στείλει το CANCEL. Αν για το *original request* δημιουργήθηκε ένα *final response*, το CANCEL δεν μπορεί να σταλεί εφόσον τα CANCEL δεν έχουν καμία επίδραση σε *requests* για τα οποία έχει δημιουργηθεί ένα *final response*. Όταν ο *client* αποφασίζει να στείλει ένα CANCEL, δημιουργεί ένα *client transaction* και στέλνει απέναντι το CANCEL μαζί με το *destination address*, *port*, και *transport*. Τα *destination address*, *port*, και *transport* πρέπει να είναι ίδια με αυτά που χρησιμοποιήθηκαν για το *original request*.

Αν ήταν επιτρεπτή η αποστολή ενός CANCEL πριν την λήψη ενός *response* για το προηγούμενο *request*, ο *server* θα μπορούσε να λάβει το CANCEL πριν το *original request*.

Θα πρέπει να σημειωθεί ότι και τα δύο *transactions* (*original request* και *cancel*) θα ολοκληρωθούν ανεξάρτητα. Ωστόσο ένας UAC που ακυρώνει ένα *request* δεν μπορεί να βασιστεί στη λήψη ενός *response 487 (Request Terminated)* για *original request* εφόσον ένας UAS συμβατός με το RFC 2543 [11] δεν θα δημιουργήσει ένα τέτοιο *response*. Αν δεν υπάρξει κάποιο *final response* για το *original request* σε 64*T1 δευτερόλεπτα (όπου T1 το interval στο οποίο αν δεν ληφθεί ένα *response* σε κάποιο INVITE το τελευταίο ξαναστέλνεται) ο *client* θα πρέπει να θεωρήσει πως η *original transaction* ακυρώθηκε και θα πρέπει να τερματίσει το *transaction* που ασχολείται με το *original request*

Συμπεριφορά του Server

Η λειτουργία CANCEL απαιτεί από τον *Transaction User* στο *server side* την ακύρωση του *pending transaction*. Ο *Transaction User* καθορίζει το *transaction* που θα ακυρωθεί διαβάζοντας το CANCEL *request*, και στην συνέχεια υποθέτοντας πως η *request method* είναι οποιαδήποτε εκτός από CANCEL ή ACK εφαρμόζει τις διαδικασίες αντιστοίχησης. Η *matching transaction* είναι αυτή που πρόκειται να ακυρωθεί.

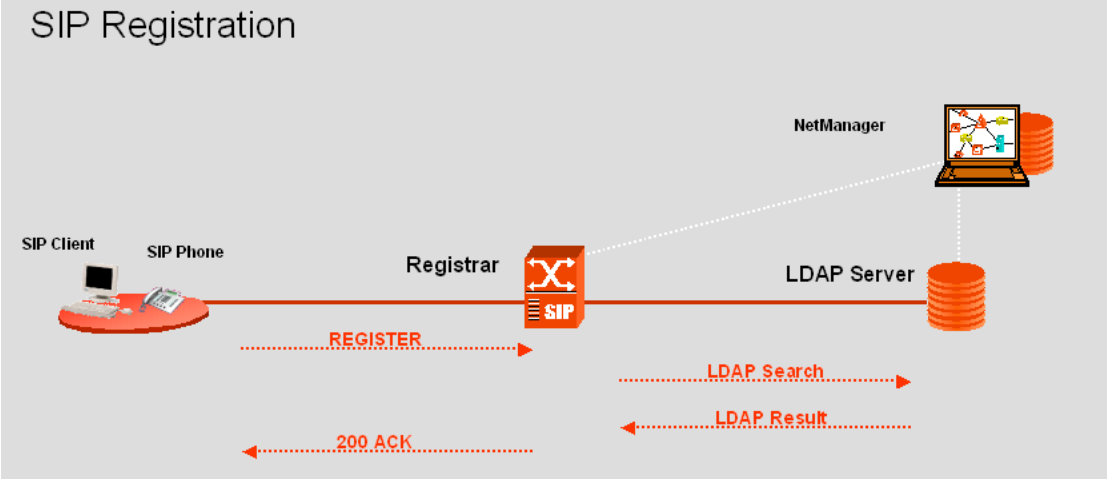
Η επεξεργασία ενός CANCEL *request* στον *server* εξαρτάται από το είδος του *server*. Ένας *stateless proxy* θα προωθήσει το *request*. Ένας *stateful proxy* μπορεί να ανταποκριθεί στο *request*, και να δημιουργήσει κάποια CANCEL *requests* για τον εαυτό του, ενώ ο UAS θα ανταποκριθεί κανονικά.

Ο UAS επεξεργάζεται το CANCEL *request* σύμφωνα με την γενική συμπεριφορά που παρουσιάστηκε σε προηγούμενη παράγραφο. Ωστόσο εφόσον τα CANCEL *requests* διαδίδονται *hop-by-hop* και δεν μπορεί να γίνει η επιβεβαίωση τους, δεν μπορεί να γίνει πρόσκληση από τον *server* για επιπλέον διαπιστευτήρια σε κάποιο *Authorization header field*. Επιπλέον θα πρέπει να σημειωθεί ότι τα CANCEL *requests* δεν περιέχουν κάποιο *Require header field*.

Αν ο UAS δεν εντοπίσει κάποιο *transaction* που να ταιριάζει με το CANCEL σύμφωνα με την διαδικασία παραπάνω, θα πρέπει να ανταποκριθεί στο CANCEL με ένα *response 481* (*Call Leg/Transaction Does Not Exist*). Ωστόσο αν το *transaction* για *το original request* υπάρχει, η συμπεριφορά του UAS στην λήψη του CANCEL *request* εξαρτάται από το αν έχει σταλεί ένα *final response* ή όχι για το *original request*. Αν έχει σταλεί, το CANCEL *request* δεν έχει καμία επίδραση στην επεξεργασία του *original request*, καμία επίδραση στην κατάσταση του *session*, και καμία επίδραση στα *responses* που δημιουργήθηκαν για το *original request*. Αν ο UAS δεν δημιούργησε κάποιο *final response* για το *original request*, η συμπεριφορά του εξαρτάται στην λειτουργία που χρησιμοποιήθηκε κατά το *original request*. Αν αυτό ήταν INVITE, ο UAS πρέπει να ανταποκριθεί άμεσα στο INVITE με ένα *response 487* (*Request Terminated*). Ένα CANCEL *request* δεν έχει καμία επίδραση στην επεξεργασία των *transactions* που χρησιμοποιούν κάποια λειτουργία η οποία ορίζεται σε *specification* διαφορετικό του RFC 3261 [9].

Ανεξάρτητα της μεθόδου που χρησιμοποιείται στο *original request*, όταν το CANCEL αντιστοιχηθεί με κάποιο υπαρκτό *transaction*, ο UAS απαντάει αυτόματα με ένα *response 200 (OK)*. Το συγκεκριμένο *response* κατασκευάζεται ακολουθώντας τις διαδικασίες που περιγράφηκαν παραπάνω, σημειώνοντας πως το "*To*" *tag* του *response* στο CANCEL και το "*To*" *tag* στο *response* του *original request* θα πρέπει να είναι τα ίδια. Το *response* που αναφέρεται στο CANCEL δίνεται στον *server transaction* για την εκπομπή του.

2.5 Λειτουργία Καταχώρησης (Registrations)



Σχήμα 2.2

Το SIP προσφέρει δυνατότητες *discovery*. Αν ένας *user* επιθυμεί να αρχικοποιήσει (*initiate*) ένα *session* με κάποιον άλλο *user*, το SIP πρέπει να ανακαλύψει το *host* στο οποίο ο *user*-στόχος γίνεται διαθέσιμος. Αυτή η διαδικασία *discovery* συχνά εκτελείται από SIP *network elements* όπως οι *proxy servers* ή οι *redirect servers* που είναι υπεύθυνοι για την λήψη των *requests*, τον υπολογισμό του επόμενου σταθμού τους με βάση την γνώση που παίρνουν από μια βάση δεδομένων, και την αποστολή αυτών των *requests*. Για να γίνει κάτι τέτοιο, τα SIP elements λαμβάνουν υπ' όψιν μια υπηρεσία γνωστή και ως *location service* η οποία παρέχει καταχωρίσεις για ένα συγκεκριμένο *domain*. Αυτές οι καταχωρήσεις αντιστοιχούν ένα εισερχόμενο SIP ή SIPS URI π.χ.

```
sip:userB@teicrete.com
```

σε ένα ή περισσότερα URI(s) που είναι κάπως πιο κοντά στον επιθυμητό χρήστη π.χ.

```
userB@chania.teicrete.gr
```

Τελικά ο *proxy* θα λάβει υπ' όψιν την *location service* η οποία αντιστοιχεί το ληφθέν URI σε ένα η περισσότερους UA στους οποίους έχει κάνει *log in* ο *userB*.

Η διαδικασία του *Registration* δημιουργεί καταχωρήσεις σε μια *location service* ενός συγκεκριμένου *domain*, οι οποίες συνδέουν μία *address-of-record* URI με μία *contact address*. Έτσι όταν ο *proxy* ενός *domain* λάβει ένα *request* του οποίου το *request URI* ταιριάξει με κάποιο *address-of-record*, ο *proxy* θα προωθήσει το *request* στην *contact address* που είναι *registered* με το *address-of-record*. Το τελευταίο είναι λογικό, αν σκεφτεί κανείς πως το *registration* της *address-of-record* γίνεται στην *location service* ενός *domain* όπου όλα τα *requests* με στόχο την συγκεκριμένη *address-of-record* δρομολογούνται στο εν λόγω *domain*. Στις περισσότερες των περιπτώσεων αυτό σημαίνει ότι το *domain* στο οποίο γίνεται το *registration* θα πρέπει να ταιριάζει με το *domain* στο URI του *address-of-record*.

Η διαδικασία του *registration* απαιτεί την αποστολή ενός REGISTER *request* σε έναν ειδικό τύπο UAS γνωστό σαν *registrar*. Ένας *registrar* αντιδρά σαν το *front end* της *location service* ενός *domain*, διαβάζοντας και γράφοντας αντιστοιχήσεις

(*mappings*) βασισμένες στα περιεχόμενα των REGISTER *requests*. Αυτή η *location service* τυπικά ελέγχεται από έναν *proxy server* που είναι υπεύθυνος για την δρομολόγηση των *requests* που αφορούν το παραπάνω *domain*.

Το σχήμα 2.2 περιγράφει την διαδικασία του *registration*. Θα πρέπει να σημειωθεί ότι ο *registrar* και ο *proxy server* εκτελούν διαφορετική λογική λειτουργία και στην πράξη μπορούμε να τους συναντήσουμε στην ίδια δικτυακή συσκευή. Για λόγους καλύτερης κατανόησης οι δύο *servers* έχουν διαχωριστεί σε αυτήν την εργασία. Επίσης μπορεί να αναφερθεί πως ένας UA πρέπει να στέλνει *requests* μέσω ενός *proxy server* έτσι ώστε να φτάσει τον *registrar* αν οι τελευταίοι αποτελούν ξεχωριστά *elements*.

Το SIP δεν ορίζει κάποιον συγκεκριμένο μηχανισμό για την υλοποίηση της *location service*. Η μόνη απαίτηση είναι ότι ο *registrar* ενός συγκεκριμένου *domain* πρέπει να έχει την δυνατότητα να διαβάζει και να γράφει δεδομένα στην *location service*. Επιπρόσθετα, ο *proxy* και ο *redirect server* για το συγκεκριμένο *domain* θα πρέπει επίσης να έχουν δικαιώματα εγγραφής στα δεδομένα της *location service*. Ένας *registrar* και ένας *proxy server* για το ίδιο *domain*, μπορούν να είναι τοποθετημένοι στον ίδιο χώρο (όπως και συνηθίζεται).

2.5.1 Δομή του REGISTER Request

Ένα REGISTER *request* έχει την ακόλουθη γενική μορφή:

```
REGISTER sip:141.29.36.134 SIP/2.0
Via: SIP/2.0/UDP 141.29.36.14:5060
To: Ioannis Papoutsis <sip:user1@chania.teicrete.gr>
From: Ioannis Papoutsis
<sip:user1@chania.teicrete.gr>;tag=e1101aa5b053b6b
Call-ID: 67e07d3ace63049321692e5a4ac9425d@chania.teicrete.gr
CSeq: 1079876078 REGISTER
Contact: Ioannis Papoutsis
<sip:user1@141.29.36.14:5060>;expires=900;methods="INVITE, ACK,
OPTIONS, BYE, CANCEL, REGISTER, REFER, SUBSCRIBE, NOTIFY, MESSAGE"
Max-Forwards: 70
Content-Length: 0
User-Agent: SCS/v3.1.12.33
```

Τα REGISTER *requests* προσθέτουν, αφαιρούν, ή ρωτούν για καταχωρήσεις-αντιστοιχίσεις. Ένα REGISTER *request* μπορεί να προσθέσει μια αντιστοίχηση μεταξύ μιας *address-of-record* και μίας ή περισσοτέρων *contact addresses*. Η διαδικασία εκ μέρους ενός συγκεκριμένου *address-of-record* μπορεί να πραγματοποιηθεί και από τρίτον κατάλληλα εξουσιοδοτημένο. Ο *client* μπορεί να αφαιρέσει προηγούμενες καταχωρήσεις, ή μπορεί να εκτελέσει αναζήτηση για να καθορίσει ποιες καταχωρήσεις αναφέρονται σε μια *address-of-record*.

Ένα REGISTER *request* δεν δημιουργεί κάποιο *dialog*. Ένας UAC μπορεί να συμπεριλάβει ένα *Route header field* βασισμένο σε μια *pre-existing* διαδρομή όπως περιγράφηκε προηγουμένως. Το *Record-Route header field* δεν έχει νόημα στα REGISTER *requests* ή *responses* και πρέπει να αγνοείται. Συγκεκριμένα ο UAC δεν

πρέπει να δημιουργεί νέα *route sets* βασιζόμενος στην παρουσία ή απουσία του *Record-Route header field* σε οποιοδήποτε *response* προς ένα REGISTER *request*.

Τα παρακάτω *header fields* εκτός του *Contact* πρέπει να συμπεριλαμβάνονται σε ένα REGISTER *request*. (το Contact *header field* είναι προαιρετικό)

Request-URI: Το *Request-URI* φέρει το όνομα του *domain* της *location service* για την οποία προορίζεται το *registration* (π.χ. "`sip:siemens.com`"). Η πληροφορία του χρήστη (*user info*) και ο χαρακτήρας "@" του SIP URI δεν πρέπει να δίδονται σε αυτό το πεδίο.

```
REGISTER sip:141.29.36.134 SIP/2.0
Via: SIP/2.0/UDP 141.29.36.14:5060
To: Ioannis Papoutsis <sip:user1@chania.teicrete.gr>
From: Ioannis Papoutsis
<sip:user1@chania.teicrete.gr>;tag=e1101aa5b053b6b
Call-ID: 67e07d3ace63049321692e5a4ac9425d@chania.teicrete.gr
CSeq: 1079876078 REGISTER
Contact: Ioannis Papoutsis
<sip:user1@141.29.36.14:5060>;expires=900;methods="INVITE, ACK,
OPTIONS, BYE, CANCEL, REGISTER, REFER, SUBSCRIBE, NOTIFY, MESSAGE"
Max-Forwards: 70
Content-Length: 0
User-Agent: SCS/v3.1.12.33
```

To: Το "*To*" *header field* περιέχει την *address-of-record* για την οποία θα δημιουργηθεί, τροποποιηθεί, ή ερωτηθεί το *registration*. Τα "*To*" και *request URI* τυπικά διαφέρουν αφού το πρώτο έχει και το *username*. Η *address-of-record* πρέπει να είναι μία SIP ή SIPS URI.

```
REGISTER sip:141.29.36.134 SIP/2.0
Via: SIP/2.0/UDP 141.29.36.14:5060
To: Ioannis Papoutsis <sip:user1@chania.teicrete.gr>
From: Ioannis Papoutsis
<sip:user1@chania.teicrete.gr>;tag=e1101aa5b053b6b
Call-ID: 67e07d3ace63049321692e5a4ac9425d@chania.teicrete.gr
CSeq: 1079876078 REGISTER
Contact: Ioannis Papoutsis
<sip:user1@141.29.36.14:5060>;expires=900;methods="INVITE, ACK,
OPTIONS, BYE, CANCEL, REGISTER, REFER, SUBSCRIBE, NOTIFY, MESSAGE"
Max-Forwards: 70
Content-Length: 0
User-Agent: SCS/v3.1.12.33
```

From: Το *From header field* περιέχει την *address-of-record* του ατόμου που είναι υπεύθυνο για το *registration*. Η τιμή αυτή ταυτίζεται με την τιμή στο "*To*" *header field*, εκτός αν το *request* είναι *third party registration* (*registration* από τρίτο).

```
REGISTER sip:141.29.36.134 SIP/2.0
Via: SIP/2.0/UDP 141.29.36.14:5060
To: Ioannis Papoutsis <sip:user1@chania.teicrete.gr>
From: Ioannis Papoutsis
<sip:user1@chania.teicrete.gr>;tag=e1101aa5b053b6b
```

```
Call-ID: 67e07d3ace63049321692e5a4ac9425d@chania.teicrete.gr
CSeq: 1079876078 REGISTER
Contact: Ioannis Papoutsis
<sip:user1@141.29.36.14:5060>;expires=900;methods="INVITE, ACK,
OPTIONS, BYE, CANCEL, REGISTER, REFER, SUBSCRIBE, NOTIFY, MESSAGE"
Max-Forwards: 70
Content-Length: 0
User-Agent: SCS/v3.1.12.33
```

Call-ID: Όλα τα *registrations* από τον ίδιο UAC θα πρέπει να χρησιμοποιούν την ίδια τιμή στο *Call-ID header field*, όταν τα *registrations* στέλνονται στον ίδιο *registrar*. Αν ο ίδιος *client* χρησιμοποιούσε διαφορετικές τιμές στο *Call-ID header field*, ο *registrar* δεν θα μπορούσε να διακρίνει την άφιξη κάποιου καθυστερημένου *request*.

```
REGISTER sip:141.29.36.134 SIP/2.0
Via: SIP/2.0/UDP 141.29.36.14:5060
To: Ioannis Papoutsis <sip:user1@chania.teicrete.gr>
From: Ioannis Papoutsis
<sip:user1@chania.teicrete.gr>;tag=e1101aa5b053b6b
Call-ID: 67e07d3ace63049321692e5a4ac9425d@chania.teicrete.gr
CSeq: 1079876078 REGISTER
Contact: Ioannis Papoutsis
<sip:user1@141.29.36.14:5060>;expires=900;methods="INVITE, ACK,
OPTIONS, BYE, CANCEL, REGISTER, REFER, SUBSCRIBE, NOTIFY, MESSAGE"
Max-Forwards: 70
Content-Length: 0
User-Agent: SCS/v3.1.12.33
```

CSeq: Η τιμή στο *CSeq header field* εγγυάται την σωστή σειρά των REGISTER *requests*. Ένας UA πρέπει αυξάνει την τιμή του *CSeq header field* κατά ένα για κάθε REGISTER *request* με το ίδιο *Call-ID*

```
REGISTER sip:141.29.36.134 SIP/2.0
Via: SIP/2.0/UDP 141.29.36.14:5060
To: Ioannis Papoutsis <sip:user1@chania.teicrete.gr>
From: Ioannis Papoutsis
<sip:user1@chania.teicrete.gr>;tag=e1101aa5b053b6b
Call-ID: 67e07d3ace63049321692e5a4ac9425d@chania.teicrete.gr
CSeq: 1079876078 REGISTER
Contact: Ioannis Papoutsis
<sip:user1@141.29.36.14:5060>;expires=900;methods="INVITE, ACK,
OPTIONS, BYE, CANCEL, REGISTER, REFER, SUBSCRIBE, NOTIFY, MESSAGE"
Max-Forwards: 70
Content-Length: 0
User-Agent: SCS/v3.1.12.33
```

Contact: Τα REGISTER *requests* μπορεί να περιέχουν ένα *contact header field* με μηδενική ή διάφορη του μηδενός τιμή.

Οι UAs δεν επιτρέπεται να στέλνουν νέα *registrations* (τα οποία περιέχουν νέα τιμή στο *Contact header field*) αν δεν λάβουν ένα *final response* από τον *registrar* για το προηγούμενο *request*. Αν δεν ληφθεί *final response* το νέο *request*

μπορεί να σταλεί μόλις περάσει το χρονικό όριο ισχύος του προηγούμενου *request* (*time out*).

Οι παρακάτω *Contact header* παράμετροι έχουν ειδική σημασία σε ένα REGISTER *request*.

```
REGISTER sip:141.29.36.134 SIP/2.0
Via: SIP/2.0/UDP 141.29.36.14:5060
To: Ioannis Papoutsis <sip:user1@chania.teicrete.gr>
From: Ioannis Papoutsis
<sip:user1@chania.teicrete.gr>;tag=e1101aa5b053b6b
Call-ID: 67e07d3ace63049321692e5a4ac9425d@chania.teicrete.gr
CSeq: 1079876078 REGISTER
Contact: Ioannis Papoutsis
<sip:user1@141.29.36.14:5060>;expires=900;methods="INVITE, ACK,
OPTIONS, BYE, CANCEL, REGISTER, REFER, SUBSCRIBE, NOTIFY, MESSAGE"
Max-Forwards: 70
Content-Length: 0
User-Agent: SCS/v3.1.12.33
```

Action: Η παράμετρος "*action*" έχει πάψει να χρησιμοποιείται από το RFC 2543 [11]. Οι UACs δεν θα πρέπει να την χρησιμοποιούν.

Expires: Η παράμετρος "*expires*" δείχνει την χρονική διάρκεια που επιθυμεί ο UA μια καταχώρηση ως έγκυρη. Η τιμή είναι ένας αριθμός που δείχνει δευτερόλεπτα. Αν αυτή η παράμετρος δεν παρέχεται, χρησιμοποιείται στην θέση της η τιμή από το *Expires header field*.

```
REGISTER sip:141.29.36.134 SIP/2.0
Via: SIP/2.0/UDP 141.29.36.14:5060
To: Ioannis Papoutsis <sip:user1@chania.teicrete.gr>
From: Ioannis Papoutsis
<sip:user1@chania.teicrete.gr>;tag=e1101aa5b053b6b
Call-ID: 67e07d3ace63049321692e5a4ac9425d@chania.teicrete.gr
CSeq: 1079876078 REGISTER
Contact: Ioannis Papoutsis
<sip:user1@141.29.36.14:5060>;expires=900;methods="INVITE, ACK,
OPTIONS, BYE, CANCEL, REGISTER, REFER, SUBSCRIBE, NOTIFY, MESSAGE"
Max-Forwards: 70
Content-Length: 0
User-Agent: SCS/v3.1.12.33
```

2.5.2 Επεξεργασία Καταχωρήσεων

Δημιουργία Καταχωρήσεων

Ένα REGISTER *request* που στέλνεται σε κάποιον *registrar* περιέχει την/τις *contact address(es)*. Η *address-of-record* περιέχεται στο "*To*" *header field* του REGISTER *request*.

Οι τιμές του *Contact header field* ενός *request* τυπικά αποτελούνται από SIP ή SIPS URIs οι οποίες αναγνωρίζουν συγκεκριμένα *endpoints* (`sip:john@pc11.`

siemens.com), ενώ μπορεί να χρησιμοποιηθεί οποιοδήποτε URI *scheme*. Ένας SIP UA μπορεί να επιλέξει την καταχώρηση τηλεφωνικών αριθμών (χρησιμοποιώντας την tel URL RFC 2806 [8]) ή την καταχώρηση *email addresses* (mailto URL RFC 2368 [13]) σαν επαφές για μια *address-of-record*.

Για παράδειγμα ο John με *address-of-record* "sip:john@pc11.siemens.com" θα έκανε *register* σε κάποιον SIP *registrar* στο *domain "siemens.com"*. Τα *registrations* θα ήταν στην διάθεση ενός *proxy server* στο παραπάνω *domain* για την δρομολόγηση των *requests* με παραλήπτη την *address-of-record* του John στο ανάλογο *endpoint*.

Μόλις ο *client* ολοκληρώσει τις καταχωρήσεις του σε κάποιον *registrar*, μπορεί να στείλει διαδοχικά *registrations* τα οποία περιέχουν νέες καταχωρήσεις ή αλλαγές στις υπάρχουσες καταχωρήσεις, αν αυτό είναι απαραίτητο. Ένα *2xx response* σε κάποιο REGISTER *request* θα περιέχει στο *Contact header field* μια **ολοκληρωμένη** λίστα των καταχωρήσεων που έχουν γίνει *registered* για μία *address-of-record* στον ίδιο *registrar*.

Αν η *address-of-record* στο "*To*" *header field* ενός REGISTER *request* είναι μία SIPS URI, τότε η τιμή στο *Contact header field* θα πρέπει επίσης να είναι SIPS URI. Οι *clients* θα πρέπει να κάνουν *register* τις *non*-SIPS URIs κάτω από μια SIPS *address-of-record* όταν η ασφάλεια του *resource* που αναπαριστάται από την *contact address* μπορεί να εγγυηθεί με άλλους τρόπους. Αυτό μπορεί να βρει εφαρμογή σε URIs που συμπεριλαμβάνουν άλλα πρωτόκολλα διαφορετικά από το SIP, ή σε SIP *devices* που χρησιμοποιούν μεθόδους ασφαλείας διαφορετικές από το TLS [21].

Θέτοντας το Expiration Interval των Contact Addresses

Όταν ένας *client* στέλνει ένα REGISTER *request*, έχει την δυνατότητα να προτείνει ένα *expiration interval* το οποίο δείχνει την διάρκεια για την οποία ο *client* θέλει να είναι έγκυρο το *request*.

Υπάρχουν δύο τρόποι με τους οποίους ο *client* μπορεί να δηλώσει το *expiration interval* για μια καταχώρηση. Ο πρώτος τρόπος γίνεται μέσω ενός *Expires header field* ενώ ο δεύτερος μέσω της παραμέτρου "*expires*" στο *Contact header field*. Ο τελευταίος επιτρέπει τον ορισμό *expiration intervals* ανά καταχώρηση, όταν δηλώνονται περισσότερες από μία αντιστοιχήσεις στο REGISTER *request*. Στον πρώτο τρόπο έχουμε την υπόδειξη ενός *expiration interval* για όλες τις τιμές του *Contact header field* που δεν περιέχουν την *expires* παράμετρο.

Αν κανένας από τους προαναφερθείς μηχανισμούς δεν συναντάται σε κάποιο REGISTER *request*, με αυτόν τον τρόπο ο *client* δηλώνει την επιθυμία του στον *server* να επιλέξει ο ίδιος το *expiration time*.

Κατάργηση των καταχωρήσεων (Registrations)

Τα *registrations* είναι *soft-state* και λήγουν αν δεν ανανεωθούν, μπορούν όμως και να εξαλειφθούν. Ένας *client* μπορεί να προσπαθήσει να επηρεάσει το *expiration interval* που έχει επιλεγεί από τον *registrar*. Έτσι ο UA έχει την δυνατότητα να ζητήσει την άμεση κατάργηση μιας καταχώρησης απλά καθορίζοντας

το *expiration interval* σε "0" σε κάποιο REGISTER *request*. Οι UAs θα πρέπει να υποστηρίζουν αυτόν τον μηχανισμό έτσι ώστε οι καταχωρήσεις να ακυρώνονται πριν λήξουν.

Η χρήση του " * " στο *Contact header field* επιτρέπει σε έναν UA να αφαιρέσει όλες τις αντιστοιχήσεις που είναι συνδεδεμένες με μια *address-of-record*, χωρίς να ξέρει την ακριβή τους τιμή.

Ανανέωση Καταχωρήσεων

Κάθε UA είναι υπεύθυνος για τις καταχωρήσεις που έχει δημιουργήσει στο παρελθόν. Μια καταχώρηση που έχει δημιουργηθεί από κάποιον UA μπορεί και να ανανεωθεί μονάχα από τον ίδιο UA.
Το *Response 200 (OK)* που προέρχεται από έναν *registrar* περιέχει μια λίστα από *Contact fields* απαριθμώντας όλες τις τρέχουσες καταχωρίσεις. Ο UA συγκρίνει κάθε *contact address* για να διαπιστώσει αν δημιούργησε την δικιά του. Αν η σύγκριση δώσει θετικό αποτέλεσμα ο UA θα ανανεώσει το *expiration time interval* σύμφωνα με την παράμετρο *expires*. Αν η σύγκριση δώσει αρνητικό αποτέλεσμα ανανεώνεται η τιμή του *Expires field*. Στη συνέχεια ο UA εκπέμπει ένα REGISTER *request* για κάθε καταχώρηση που του αναλογεί πριν λήξει το *Expiration interval*. Μπορούν, επίσης, να συνδυαστούν αρκετά *updates* σε ένα *request*.

Εκπέμποντας ένα Request

Μόλις ετοιμαστεί το request της λειτουργίας REGISTER, και βρεθεί ο προορισμός του μηνύματος, ο UAC ακολουθεί την διαδικασία που περιγράφτηκε παραπάνω ώστε να δοθεί το REGISTER στο *transaction layer*. Αν το *transaction layer* επιστρέψει κάποιο *time out error* επειδή δεν απαντήθηκε κάποιο *response* στο REGISTER, ο *client* δεν πρέπει να δοκιμάσει αμέσως το *registration* στον ίδιο *registrar*.

Μία άμεση προσπάθεια στον ίδιο *registrar* πιθανότατα θα έχει το ίδιο αποτέλεσμα (*time out*). Αντίθετα αν ο *client* περιμένει κάποιο εύλογο χρονικό διάστημα ώστε οι συνθήκες που δημιούργησαν το σφάλμα να εξαλειφθούν, είναι πιο πιθανό να εξυπηρετηθεί το αίτημα του, ενώ ταυτόχρονα μειώνεται το ανώφελο φορτίο στο δίκτυο. Ωστόσο το SIP δεν καθορίζει το χρονικό διάστημα που πρέπει να μεσολαβεί ανάμεσα στις διαδοχικές προσπάθειες.

Error Responses

Αν ένας UA λάβει κάποιο *response 423* (*Interval Too Brief*), μπορεί να δοκιμάσει ξανά το *registration* αφού θέσει το *expiration interval* όλων των *contact addresses* του REGISTER *request* ίσο ή μεγαλύτερο από το *expiration interval* της τιμής του *Min-Expiration header field* του *response 423*.

Συνολική αντιμετώπιση των REGISTER Requests

Ο *registrar* είναι ένας UAS που ανταποκρίνεται στα REGISTER *requests* με κάποιο *response* και διατηρεί μία λίστα με τις καταχωρήσεις-αντιστοιχήσεις στην οποία έχουν πρόσβαση οι *proxy* & *redirect servers* που ανήκουν στο ίδιο *domain*. Ο

registrar διαχειρίζεται τα *requests* σύμφωνα με όσα αναφέρονται στην ενότητα 2.2.2 με την διαφορά ότι δέχεται μονάχα REGISTER *requests*. Ένας *registrar* δεν μπορεί να δημιουργήσει κάποιο *6xx response*.

Ο *registrar* μπορεί να κάνει *redirect* τα REGISTER *requests* όταν κρίνεται απαραίτητο. Μία κοινή χρήση για έναν *registrar* θα ήταν να ακούει σε κάποιο *multicast interface* για REGISTER *requests* και να κάνει *redirection* στο αντίστοιχο *unicast interface* απατώντας με ένα *response 302* (*Moved Temporarily*).

Οι *registrars* πρέπει να αγνοούν το *Record-Rout header field* αν αυτό περιέχεται στο REGISTER *request*. Ακόμα θα πρέπει να μην συμπεριλαμβάνουν το *Record-Rout header field* σε οποιοδήποτε *response* που στέλνουν πίσω στον UAC.

Ένας *registrar* μπορεί να λάβει κάποιο *request* που διήλθε μέσω ενός *proxy* ο οποίος επεξεργάζεται τα REGISTER *requests* σαν *unknown requests* και προσθέτει μια τιμή στο *Record-Route header field*.

Ο *registrar* πρέπει να γνωρίζει (π.χ. μέσω του *configuration*) το σύνολο του / των *domain(s)* για το οποίο διατηρεί τις καταχωρήσεις-αντιστοιχίες. Τα REGISTER *requests* πρέπει να υφίστανται επεξεργασία από έναν *registrar* με την σειρά που καταφθάνουν, ανεξάρτητα από τα υπόλοιπα *registrations*.

Όταν λαμβάνεται ένα REGISTER *request*, ο *registrar* ακολουθεί τα παρακάτω βήματα:

1. Ο *registrar* ελέγχει το *request URI* ώστε να καθορίσει αν έχει πρόσβαση στις αντιστοιχήσεις-καταχωρήσεις του *domain* που προσδιορίζεται από το παραπάνω *header field*. Αν δεν έχει πρόσβαση και αν ο *server* λειτουργεί και ως *proxy*, ο *server* θα πρέπει να προωθήσει το *request* στο αντίστοιχο *domain* ακολουθώντας την γενική συμπεριφορά για τα *proxying messages*.
2. Ο *registrar* πρέπει να επεξεργαστεί τις τιμές στο *Require header field* όπως περιγράφηκε στο 2.2.2 ώστε να είναι σίγουρο πως ο *registrar* παρέχει οποιαδήποτε απαιτούμενα *extensions*.
3. Ο *registrar* θα πρέπει να πιστοποιήσει τον UAC. Μηχανισμοί για την πιστοποίηση των SIP *user agents* υπάρχουν αλλά δεν αποτελούν αντικείμενο μελέτης αυτής της εργασίας. Σε καμία περίπτωση δεν μπορεί να θεωρηθεί ότι η συμπεριφορά του *registration* υπερκαλύπτει τους γενικούς μηχανισμούς ασφαλείας του SIP. Αν δεν υπάρχουν διαθέσιμοι μηχανισμοί ασφαλείας ο *registrar* μπορεί να επιλέξει την *From address* σαν ισχύουσα ταυτότητα του *originator* του *request*.
4. Ο *registrar* θα πρέπει να προσδιορίσει αν ο *authenticated user* έχει δικαίωμα να αλλάξει τα *registrations* για την *address-of-record* που επιθυμεί. Για παράδειγμα ένας *registrar* μπορεί να διατηρεί μια *database* που αντιστοιχεί κάποιο *user name* σε μια λίστα από *addresses-of-record* για την οποία ο *user* έχει *authorization* να αλλάξει την καταχώρηση. Αν ο *authenticated user* δεν είναι εξουσιοδοτημένος να αλλάξει την καταχώρηση ο *registrar* θα απαντήσει με ένα *response 403* (*Forbidden*) και θα παρακάμψει τα επόμενα βήματα.

Σε αρχιτεκτονικές που παρέχουν *third-party registrations* (από τρίτους) κάποια οντότητα μπορεί να είναι υπεύθυνη για την ανανέωση των *registrations* που είναι συνδεδεμένα με πολλαπλές *addresses-of-record*.

5. Ο *registrar* υπολογίζει την *address-of-record* από το "*To*" *header field* του *request*. Αν η *address-of-record* δεν είναι έγκυρη για το *domain* του *request URI* ο *registrar* θα στείλει ένα *response 404* (*Not found*) και θα παρακάμψει τα υπόλοιπα βήματα. Το URI θα πρέπει να μετατραπεί σε κανονική μορφή. Για να γίνει κάτι τέτοιο πρέπει να αφαιρεθούν όλες οι παράμετροι του URI (συμπεριλαμβάνεται και η *user-param*) και οποιοιδήποτε *escaped* χαρακτήρες θα πρέπει να μετατραπούν στην *unescaped* μορφή τους. Το αποτέλεσμα χρησιμεύει σαν *index* της λίστας των καταχωρήσεων.
6. Ο *registrar* ελέγχει αν το *request* περιέχει το *Contact header field*, αν αυτό δεν περιέχεται εκτελεί το τελευταίο βήμα της λίστας. Αν το *Contact header field* είναι παρών ο *registrar* ελέγχει αν υπάρχει τιμή με τον χαρακτήρα " * " και κάποιο *Expires field*. Αν το *request* έχει επιπλέον *Contact header fields* ή *expiration time* διάφορο του μηδενός τότε το *request* είναι άκυρο. Ο *server* θα επιστρέψει ένα *response 400* (*Invalid Request*) και θα παρακάμψει τα υπόλοιπα βήματα. Αν δεν συμβεί τίποτα από τα παραπάνω ο *server* θα ελέγξει το *Call-ID* συμφωνεί με την αποθηκευμένη τιμή που διατηρεί για κάθε καταχώρηση. Αν οι τιμές δεν είναι ίδιες θα πρέπει να αφαιρέσει την καταχώρηση. Αν οι τιμές συμφωνούν θα πρέπει να αφαιρέσει την καταχώρηση μόνο αν το *Cseq* του *request* είναι υψηλότερο από την τιμή που είναι αποθηκευμένη για την καταχώρηση. Διαφορετικά το *update* ακυρώνεται και το *request* αποτυγχάνει.
7. Στη συνέχεια ο *registrar* επεξεργάζεται κάθε *contact address* στο *Contact header field*. Για κάθε *address* καθορίζει το *expiration interval* ως ακολούθως:

   - Αν η τιμή έχει κάποια *expires* παράμετρο αυτή η τιμή θα πρέπει να ληφθεί υπ' όψιν σαν *requested expiration*
   - Αν δεν υπάρχει τέτοιου είδους παράμετρος αλλά το *request* διαθέτει *Expires header field*, αυτή η τιμή θα πρέπει να ληφθεί σαν *requested expiration*.
   - Αν δεν υπάρχει τίποτα από τα παραπάνω μια *pre-configured default* τιμή θα πρέπει να χρησιμοποιηθεί σαν *requested expiration*.

   Ο *registrar* μπορεί να επιλέξει κάποιο *expiration interval* μικρότερο από αυτό που απαιτήθηκε από τον *client*. Αν το *expiration interval* είναι μεγαλύτερο του μηδενός, μικρότερο της μίας ώρας, και μικρότερο της *minimum default* τιμής του *registrar*, ο τελευταίος έχει την δυνατότητα να απορρίψει το *request* με ένα *response 423* (*Interval Too Brief*). Το *response* θα πρέπει να περιέχει ένα *Min-Expires header field* που δηλώνει το *minimum expiration interval* το οποίο ο *registrar* είναι πρόθυμος να δεχτεί.

   Επιτρέποντας σε ένα *registrar* να ορίζει το *registration interval* έχουμε προστασία από εξαιρετικά συχνά *registration refreshes*. Το *expiration interval* ενός *registration* χρησιμοποιείται συχνά στην δημιουργία των *services*. Ένα παράδειγμα θα μπορούσε να είναι μία υπηρεσία *follow-me*, όπου ο χρήστης μπορεί να είναι διαθέσιμος σε ένα *terminal* για μικρό χρονικό διάστημα. Γι' αυτό το λόγο οι *registrars* θα πρέπει να δέχονται *registrations* μικρής χρονικής διάρκειας. Για ένα *request* σε αυτού του είδους, τα *registrations*

μπορεί να γίνει *rejected* μονάχα όταν το *interval* είναι τόσο μικρό όπου τα *refreshs* θα υποβάθμιζαν την απόδοση του *registrar*.

Στη συνέχεια ο *registrar* ψάχνει την λίστα των επιτυχημένων καταχωρήσεων για κάθε διεύθυνση χωριστά χρησιμοποιώντας ένα σύνολο από κανόνες σύγκρισης. Αν η καταχώρηση δεν βρεθεί προστίθεται. Αν η καταχώρηση υπάρχει, ο *registrar* ελέγχει την τιμή του *Call-ID field*. Αν οι δύο τιμές είναι διαφορετικές (της υπάρχουσας καταχώρησης και του *request*) η καταχώρηση θα πρέπει να αφαιρεθεί αν το *expiration time* είναι 0 ενώ διαφορετικά θα πρέπει να ανανεωθεί. Αν όμως οι τιμές στο *Call-ID field* είναι ίδιες ο *registrar* θα συγκρίνει και την τιμή του *CSeq*. Αν η τιμή του *request* είναι υψηλότερη από της καταχώρησης θα συνεχίσει η διαδικασία ανανέωσης ή αφαίρεσης της καταχώρησης όπως προηγουμένως. Αν η τιμή δεν είναι μεγαλύτερη το *update* ακυρώνεται και το *request* αποτυγχάνει.

Αυτός ο αλγόριθμος εξασφαλίζει ότι τα *requests* που έρχονται καθυστερημένα και σε λάθος σειρά από τον ίδιο UA, θα αγνοούνται.

Κάθε καταχώρηση καταγράφει τις τιμές των *Call-ID* και *CSeq* του *request*.

Οι ανανεώσεις των καταχωρήσεων θα πρέπει να δημοσιεύονται (π.χ. στους *proxy* ή *redirect servers*) όταν όλες οι προσθήκες των ανανεώσεων έχουν πραγματοποιηθεί με επιτυχία για κάποιο *request*. Αν για κάποιο λόγο κάποια καταχώρηση αποτύχει (π.χ. λόγω της *back-end database*) τότε το *request* θα πρέπει να αποτύχει και όλες οι καταχωρήσεις που έγιναν εξαιτίας του *request* θα πρέπει να ακυρωθούν.

8. Ο *registrar* θα επιστρέψει ένα *response 200* (OK). Το *response* θα πρέπει να περιέχει τιμές στο *Contact header field* όπου θα απαριθμούνται όλες οι καταχωρήσεις. Κάθε τιμή *Contact* θα πρέπει να συνοδεύεται από μία παράμετρο "*expires*" η οποία υποδεικνύει το *expiration interval* που έχει επιλεχτεί από τον *registrar*. Το *response* θα πρέπει να περιέχει και ένα *Date header field*.

2.6 Λειτουργία Ερώτησης Δυνατοτήτων (OPTIONS)

Η SIP λειτουργία OPTIONS επιτρέπει σε ένα UA να αντλήσει πληροφορίες από έναν άλλο UA ή *proxy server* για τις δυνατότητες – υπηρεσίες που προσφέρει. Με αυτό τον τρόπο είναι δυνατό ένας *client* να ανακαλύψει πληροφορίες για τις παρεχόμενες λειτουργίες, *content types*, *extensions*, *codecs*, π.χ. χωρίς «*ringing*» στο *other party*. Για παράδειγμα, πριν ο *client* εισάγει ένα *Require header field* μέσα σε έναν INVITE (αν δεν είναι σίγουρος αν ο προοριζόμενος UAS παρέχει τη συγκεκριμένη επιλογή), μπορεί να ελέγξει τον UAS με μια λειτουργία OPTIONS και να διαπιστώσει αν η επιλογή επιστρέφεται σε ένα *Supported header field*. Όλοι οι UAs πρέπει να υποστηρίζουν την λειτουργία OPTIONS.

Ο στόχος του OPTIONS *request* διευκρινίζεται από το *request URI*, το οποίο μπορεί να δείχνει σε έναν άλλο UA ή SIP *server*. Αν το OPTIONS απευθύνεται σε έναν *proxy server*, το *request URI* τίθεται χωρίς *user part*, όπως ακριβώς συμβαίνει σε ένα *Register request*.

Εναλλακτικά, ένας *server* που λαμβάνει ένα OPTIONS *request* με τιμή 0 στο *Max-Forwards header field* μπορεί να ανταποκρίνεται στο *request* ανεξάρτητα από το *request URI*. Αυτή η συμπεριφορά είναι κοινή με το HTTP/1.1 και μπορεί να χρησιμοποιηθεί σαν δυνατότητα *trace route* για τον έλεγχο των δυνατοτήτων ξεχωριστών *hop servers* στέλνοντας μια σειρά από OPTIONS *requests* με αύξουσες *Max-Forwards* τιμές.

Σαν αποτέλεσμα του τρόπου λειτουργίας ενός UA, το *transaction layer* μπορεί να επιστρέψει ένα *time out error* αν στο OPTIONS δεν υπάρξει απάντηση. Αυτό θα μπορούσε να σημαίνει ότι ο στόχος είναι *unreachable* και έτσι *unavailable*.

Ένα OPTIONS *request* μπορεί να σταλεί σαν μέρος ενός ήδη *established dialog* για τον έλεγχο των *peer on* δυνατοτήτων οι οποίες πρόκειται να χρησιμοποιηθούν αργότερα στο ίδιο *dialog*.

2.6.1 Δημιουργία του OPTIONS request

Το OPTION *request* κατασκευάζεται με βάση τους κανόνες που έχουν οριστεί προηγουμένως για τα SIP *requests*.

Στα OPTIONS μπορεί να υπάρχει ένα *Contact header field*.

Ένα *Accept header field* θα πρέπει να περιέχεται ώστε να υποδεικνύεται ο τύπος του *message body* όπου ο UAC επιδιώκει να δεχτεί στο *response*. Τυπικά, εδώ τίθεται το *format* το οποίο χρησιμοποιείται για να περιγράψει τις *media* δυνατότητες του UA, π.χ. το SDP (*application/sdp*).

Το *response* σε ένα OPTIONS *request* υποτίθεται ότι απευθύνεται στο *request URI* του *original request*. Ωστόσο μονάχα όταν τα OPTIONS αποτελούν κομμάτι ενός *established dialog* εγγυώνται ότι τα μελλοντικά *requests* θα ληφθούν από τον *server* που δημιούργησε το OPTIONS *response*.

Παράδειγμα:

```
OPTIONS sip:userA@chania.teicrete.gr SIP/2.0
Via: SIP/2.0/UDP pc33.siemens.com;branch=z9hG4bKhjhs8ass877
Max-Forwards: 70
To: <sip: userA@chania.teicrete.gr >
From: UserB <sip:userB@siemens.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 63104 OPTIONS
Contact: <sip:userB@pc33. siemens.com>
Accept: application/sdp
Content-Length: 0
```

2.6.2 Επεξεργασία ενός OPTIONS request

Το *response* σε ένα OPTIONS κατασκευάζεται χρησιμοποιώντας τους βασικούς κανόνες για ένα SIP *response* που έχουν αναφερθεί προηγουμένως. Ο

κωδικός του *response* θα πρέπει να είναι ίδιος με αυτόν που θα χρησιμοποιούταν αν το *request* ήταν INVITE. Έτσι ένα *200* (OK) θα σταλεί αν ο UAS είναι έτοιμος να δεχτεί κλήση, ένα *486* (*Busy here*) θα σταλεί αν ο UAS είναι *busy* κ.τ.λ. Με αυτόν τον τρόπο επιτρέπεται στα OPTIONS *requests* να χρησιμοποιούνται για να καθορίζουν την βασική κατάσταση ενός UAS, η οποία μπορεί να χρησιμοποιηθεί σαν ένδειξη αν ο UAS δεχτεί ένα INVITE *request*.

Ένα OPTIONS *request* που λαμβάνεται μέσα σε κάποιο *dialog*, δημιουργεί ένα *response 200 (OK)* το οποίο είναι πανομοιότυπο με αυτό που θα είχε κατασκευαστεί έξω από το *dialog* και δεν έχει καμία επίδραση στο *dialog*.

Η χρήση των OPTIONS παρουσιάζει περιορισμούς εξαιτίας των διαφορών που υπάρχουν στον χειρισμό των OPTIONS και INVITES *requests* από τους *proxy servers*. Ενώ ένα *forked* INVITE θα είχε σαν απάντηση πολλαπλά *responses 200 (OK)*, ένα *forked* OPTIONS θα είχε σαν αποτέλεσμα ένα μοναδικό *response* αφού οι *proxies* το μεταχειρίζονται σαν *non*-INVITE *request*.

Αν το *response* σε ένα OPTIONS *request* δημιουργείται από έναν *proxy server*, ο *proxy* επιστρέφει ένα *200 (OK)* καταγράφοντας τις δυνατότητες του *server*. Αυτό το *response* δεν περιέχει *message body*.

Τα πεδία *Allow, Accept, Accept-Encoding, Accept-Language*, και *Supported* (βλέπε κατώτερο παράδειγμα) πρέπει να είναι παρών όταν σε ένα *response 200 (OK)* προς ένα OPTIONS *request*. Αν το *response* δημιουργείται από έναν *proxy*, το *Allow header field* θα πρέπει να παραλειφθεί εφόσον είναι ασαφές λόγω της άγνωστης μεθόδου που χρησιμοποιεί ο *proxy*. Το *Contact header field* μπορεί να είναι παρών σε ένα *response 200 (OK)* και να έχει την ίδια σημασιολογία όπως σε ένα *response 3xx*, δηλαδή να καταγράψει ένα σύνολο από εναλλακτικά ονόματα και μεθόδους για την προσπέλαση του χρήστη.

Τέλος ο τύπος του *message body* που μπορεί να σταλεί καθορίζεται από το *Accept header field* στο OPTIONS *request*. (με προεπιλεγμένη τιμή την *application/sdp* αν το *Accept header field* δεν είναι παρών).

Παράδειγμα:

```
SIP/2.0 200 OK
   Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKhjhs8ass877
   ;received=192.0.2.4
   To: <sip:carol@chicago.com>;tag=93810874
   From: Alice <sip:alice@atlanta.com>;tag=1928301774
   Call-ID: a84b4c76e66710
   CSeq: 63104 OPTIONS
   Contact: <sip:carol@chicago.com>
   Contact: <mailto:carol@chicago.com>
   Allow: INVITE, ACK, CANCEL, OPTIONS, BYE
   Accept: application/sdp
   Accept-Encoding: gzip
   Accept-Language: en
   Supported: foo
   Content-Type: application/sdp
   Content-Length: 274
```

(SDP not shown)

ΚΕΦΑΛΑΙΟ: 3 ΕΦΑΡΜΟΓΗ ΤΟΥ SIP ΠΡΩΤΟΚΟΛΛΟΥ ΣΕ ΠΡΑΓΜΑΤΙΚΟ ΠΕΡΙΒΑΛΛΟΝ ΛΕΙΤΟΥΡΓΙΑΣ

3.1 Εισαγωγή

Στα πλαίσια παρουσίασης των δυνατοτήτων του πρωτοκόλλου SIP θα γίνουν πραγματικές πειραματικές συνδέσεις μεταξύ UAs. Κατά την διάρκεια αυτών των συνδέσεων θα υπάρχει ενεργό κάποιο πρόγραμμα που θα καταγράφει (*capture*) τα πακέτα που θα διακινούνται. Με την μελέτη και καταγραφή του SIP περιεχομένου των πακέτων θα φανούν όλα τα μηνύματα που ανταλλάσσονται σε ένα SIP *session* και αναλύθηκαν στο Κεφάλαιο 2.

Για την όλη υλοποίηση υπήρχαν διαθέσιμοι μια σειρά από UA Clients, όπως ο SCS-Client v1.0.0 της Siemens Switzerland Ltd [22] είτε ο Helmsman® User Agent v3.0.9 της Ubiquity [24].

Σαν SIP *proxy servers* μπορεί να χρησιμοποιηθούν ο SCS Proxy Server v1.0.0.2 της Siemens Switzerland Ltd ο οποίος μπορεί να λειτουργήσει σαν *stand-alone application* ή σαν *service* σε περιβάλλον Windows NT. Επίσης μπορεί να χρησιμοποιηθεί ο *online* SIP **Proxy/Registrar Server** που παρέχεται από το SIP Center [25] και το MySIP.ch [23] (για την χρήση του SIP Proxy/Registrar Server που παρέχεται από το SIP Center απαιτείται registration στο site).

3.2 Επιλογή και Υλοποίηση SIP Client

Οι UA *clients* μπορούν, γενικά, να είναι κάποιο *SIP-phone* (ζιπίφωνο) (όπως το Cisco **SIP IP Phone 7960** [26] ή το **Siemens Optiset 100 phone** βλέπε σχήμα 3.1).

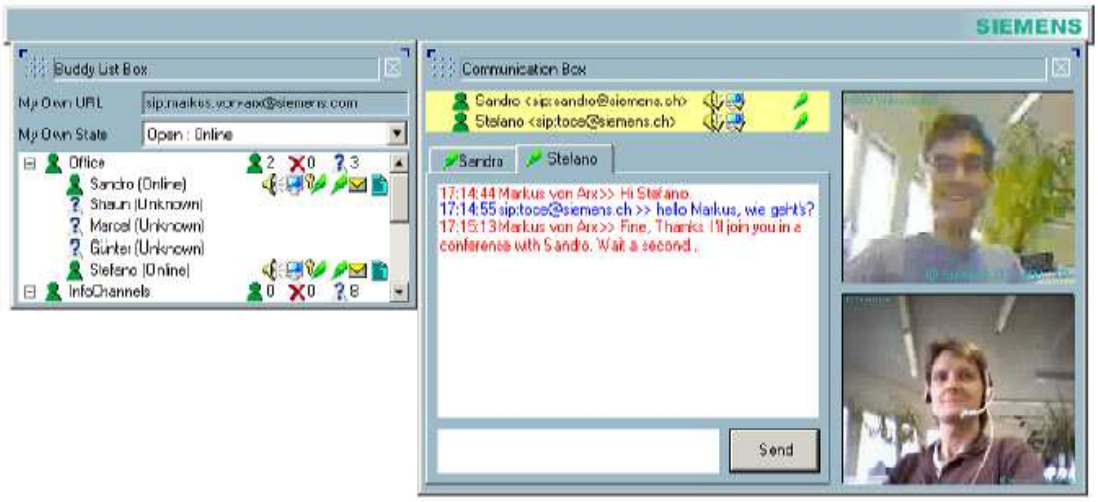
Σχήμα 3.1

Εναλλακτικά, την λειτουργία του SIP UA *client* μπορούν να επιτελέσουν τα *soft-phones* που προέρχονται από εταιρίες όπως η Siemens, Ubiquity, Cisco , Ericsson, Microsoft.

Τα χαρακτηριστικά και οι δυνατότητες των πιο σημαντικών *soft-phones* αναλύονται παρακάτω:

Siemens SCS-Client v1.0.0

Ο SIP Communication System Client αποτελεί ένα πρόγραμμα επικοινωνίας πολυμέσων. Υποστηρίζει πλήρως το πρωτόκολλο SIP (παρέχοντας επίσης τα *Presence* και *IM extensions*), ενώ θα μπορούσε να χαρακτηριστεί ως μια από τις πιο πλήρεις λύσεις στον τομέα των πολυμέσων καλύπτοντας την σύγκλιση υπηρεσιών εικόνας, ήχου , μεταφοράς αρχείων , *chat* , *instant messaging*. Ιδιαίτερη σημασία έχει το γεγονός ότι ο SCS-client παρέχει *sessions* τύπου *point-to-point* αλλά και τύπου *conference*. Επίσης μπορεί να συνεργαστεί μαζί με άλλους *clients* (π.χ. **Microsoft Messenger**). Η παρακάτω εικόνα του σχήματος 3.2 αποτελεί *screenshot* του SCS-Client κατά την διάρκεια κάποιου SIP *session*.



Σχήμα 3.2

Τεχνικές προδιαγραφές του SCS Client όπως αυτές έχουν γνωστοποιηθεί από το datasheet της εφαρμογής:

• Session Initiation Protocol (SIP), RFC 3261
• INFO Μέθοδος, RFC 2976
• SIP Specific Event Notification, RFC 3265
• Draft-ietf-sip-content indirect- mech-00
• Draft ietf-sip-message-07
• Session Description Protocol (SDP), RFC 2327
• Real Time Protocol (RTP), RFC 1889
• Running on Windows NT, 2000 and XP
• Local ad hoc conferencing up to five participants (audio, video, chat)
• Video: H.263, H.261 (planned), QCIF, CIF
• Audio Codecs: G.711, G.723, GSM (planned), G.729 (planned)
• Media trix SIP stack

Ubiquity Helmsman® User Agent (HUA)

Ο Helmsman® User Agent 3.0 αποτελεί τον πιο πρόσφατο SIP *Client* της εταιρίας Ubiquity [24]. Ο HUA δίνει την δυνατότητα σε έναν προσωπικό υπολογιστή να δημιουργεί και να λαμβάνει κλήσεις φωνής (ή αλλιώς *voice calls* ή **VoIP** ή *Internet Calls*). Επιπρόσθετα στις λειτουργίες αρχικοποίησης και λήψης *call sessions*

ο HUA προσφέρει πληθώρα από χαρακτηριστικά όπως παρακολούθηση κλήσης (*call monitor*) , και διαχείριση ταυτόχρονα πολλαπλών κλήσεων. Ο SIP UA *client* της Ubiquity είναι εξαιρετικά παραμετροποιήσιμος, έτσι ώστε η αλλαγή των προσωπικών στοιχείων κάθε χρήστη αλλά και οι διαθέσιμες επιλογές δικτύου αλλάζουν πολύ εύκολα (όπως δείχνουν τα παρακάτω *screenshots* των σχημάτων 3.3 και 3.4).

Σχήμα 3.3

Σχήμα 3.4

Θα πρέπει να σημειωθεί πως για την χρήση του Helmsman User Agent απαιτείται άδεια χρήσης η οποία γίνεται προσπελάσιμη πολύ εύκολα στέλνοντας *e-mail* στην ηλεκτρονική διεύθυνση της Ubiquity [uasupport@sipcenter.com].

Οι ελάχιστες απαιτήσεις του προγράμματος από τον υπολογιστή στον οποίο εγκαθίσταται είναι

- Προ-εγκατεστημένη έκδοση της *Java* που να συμπίπτει ή να είναι νεότερη από την JRE 1.3
- Κάρτα ήχου που να συνεργάζεται με τον UA

Τέλος θα πρέπει να σημειωθεί ότι ο HUA μπορεί να επικοινωνήσει *peer-to-peer* χωρίς την ύπαρξη κάποιου *proxy server*. Σε αυτή τη περίπτωση ο UA αυτόματα δημιουργεί μια *contact address* της μορφής

```
sip:user@<host IP address>
```

Οποιοσδήποτε επιθυμεί να καλέσει, θα πρέπει να χρησιμοποιήσει την παραπάνω SIP διεύθυνση.

3.3 Επιλογη και Υλοποίηση SIP Proxy Server

Ο SIP *proxy server* είναι λογισμικό που τρέχει πάνω σε κάποια πλατφόρμα (κυρίως Solaris) και σκοπός του είναι να παρέχει υπηρεσίες δρομολόγησης κλήσεων (*call routing*) σε ένα δίκτυο IP τηλεφωνίας. Χρησιμοποιώντας πολλούς *proxy servers* άμεσα συνδεδεμένους μεταξύ τους οι διάφοροι πάροχοι μπορούν να προσφέρουν μεγάλης κλίμακας και αρκετά αξιόπιστα δίκτυα φωνής βασισμένα στη μεταγωγή πακέτου.

Παραδείγματα διαθέσιμων λύσεων για SIP *Proxy Servers* αποτελούν τα κάτωθι:

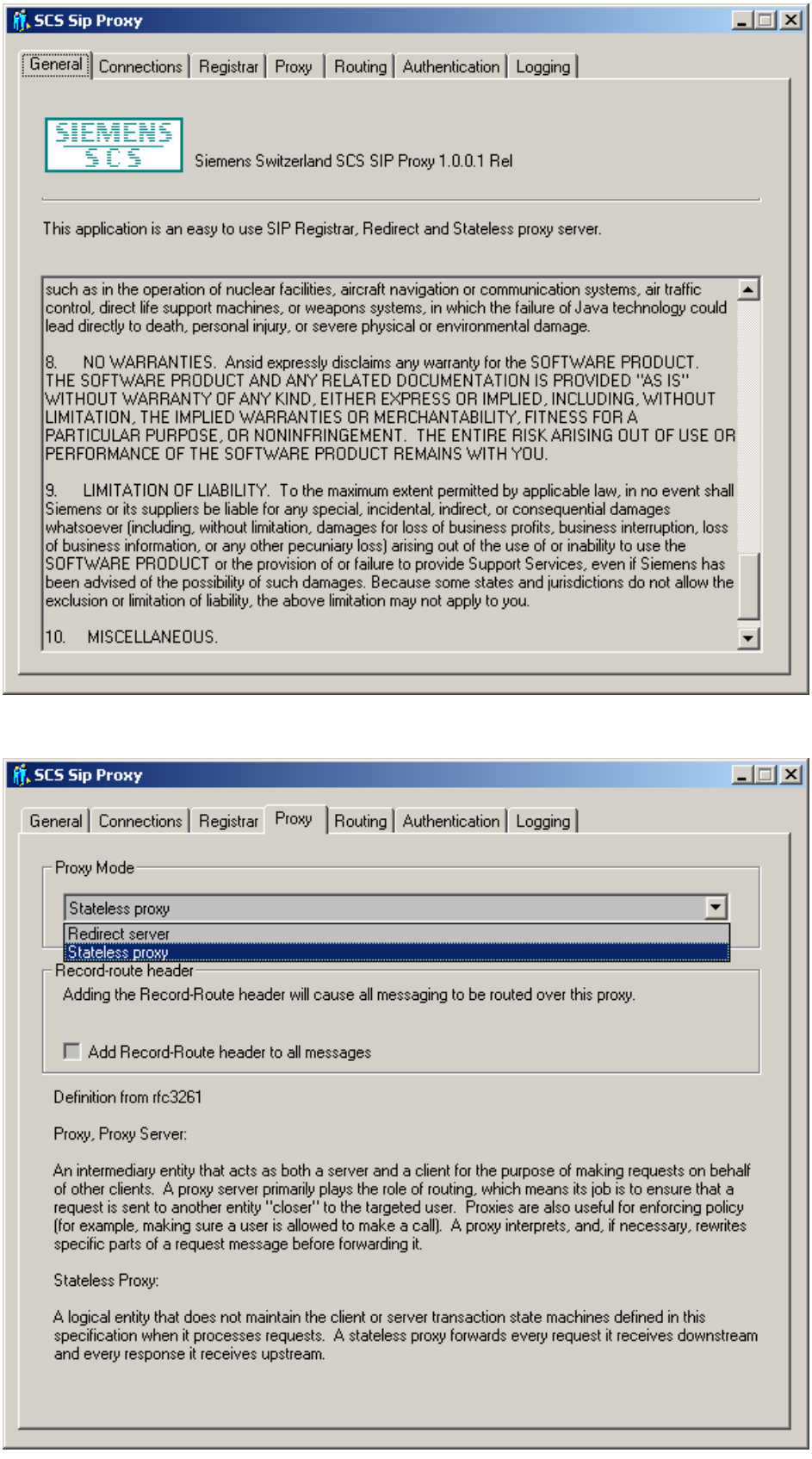
Siemens SCS-Proxy V1.0.0.1

Ο SCS-Proxy (βλέπε σχήμα 3.5) προέρχεται από την Siemens Switzerland Ltd [22] (όπως και ο αντίστοιχος *client*). Η δοκιμαστική έκδοση μπορεί να βρεθεί στον δικτυακό τόπο της Siemens Switzerland που αφορά το SIP και είναι www.mysip.ch [23]. Ο συγκεκριμένος *proxy server* διαθέτει και έναν *registrar server*. Έτσι όταν ζητηθεί από τον *proxy* ένα *request* για κάποιον SIP *user*, ο *proxy* είναι σε θέση να κάνει *query* την *database* που διατηρεί ο *registrar*. Ο συγκεκριμένος *proxy* μπορεί να λειτουργήσει είτε ως *stateless* είτε ως *redirect server*.

Επιπλέον θα πρέπει να σημειωθεί ένα ιδιαίτερο χαρακτηριστικό που διαθέτει ο *proxy* της Siemens, ότι δηλαδή δέχεται ειδικές οδηγίες ώστε να δρομολογεί *requests* που προορίζονται για κάποιο *domain* σε μια προεπιλεγμένη διεύθυνση. Αυτή η προεπιλεγμένη διεύθυνση θα πρέπει να οδηγεί στον *proxy server* για το προαναφερθέν *domain* ώστε να μην πέφτουν στο κενό οι κλήσεις (για την ακρίβεια τα requests).

Υπάρχει επίσης η δυνατότητα της δρομολόγησης όλων των SIP μηνύματων που ανταλλάσσουν μεταξύ τους οι δυο *Clients* μέσω του SIP *Proxy server*. Κάνοντας, λοιπόν, *click* στην επιλογή `<Add Record-Route header to all messages>` η οποία βρίσκεται στην καρτέλα *Proxy*, δηλώνεται στον *proxy* πως θα πρέπει να διατηρήσει τον εαυτό του στην διαδρομή που ακολουθούν όλα τα SIP μηνύματα. Αυτό γίνεται όπως έχει αναφερθεί στο 2 Κεφάλαιο χρησιμοποιώντας ένα *route header field*.

Τέλος, ένα εξίσου χρήσιμο χαρακτηριστικό που παρέχει ο SIP *Proxy server* της Siemens Switzerland Ltd [22] είναι το *IP packet logging*. Με αυτή τη λειτουργία επιτυγχάνεται ο έλεγχος του SIP περιεχομένου των μηνυμάτων που φθάνουν ή στέλνονται.


SCS Sip Proxy
General
Connections
Registrar
Proxy
Routing
Authentication
Logging
SIEMENS
SCS
Siemens Switzerland SCS SIP Proxy 1.0.0.1 Rel
This application is an easy to use SIP Registrar, Redirect and Stateless proxy server.
such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines, or weapons systems, in which the failure of Java technology could lead directly to death, personal injury, or severe physical or environmental damage.
8. NO WARRANTIES. Ansid expressly disclaims any warranty for the SOFTWARE PRODUCT. THE SOFTWARE PRODUCT AND ANY RELATED DOCUMENTATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OR MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. THE ENTIRE RISK ARISING OUT OF USE OR PERFORMANCE OF THE SOFTWARE PRODUCT REMAINS WITH YOU.
9. LIMITATION OF LIABILITY. To the maximum extent permitted by applicable law, in no event shall Siemens or its suppliers be liable for any special, incidental, indirect, or consequential damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or any other pecuniary loss) arising out of the use of or inability to use the SOFTWARE PRODUCT or the provision of or failure to provide Support Services, even if Siemens has been advised of the possibility of such damages. Because some states and jurisdictions do not allow the exclusion or limitation of liability, the above limitation may not apply to you.
10. MISCELLANEOUS.
SCS Sip Proxy
General
Connections
Registrar
Proxy
Routing
Authentication
Logging
Proxy Mode
Stateless proxy
Redirect server
Stateless proxy
Record-route header
Adding the Record-Route header will cause all messaging to be routed over this proxy.
Add Record-Route header to all messages
Definition from rfc3261
Proxy, Proxy Server:
An intermediary entity that acts as both a server and a client for the purpose of making requests on behalf of other clients. A proxy server primarily plays the role of routing, which means its job is to ensure that a request is sent to another entity "closer" to the targeted user. Proxies are also useful for enforcing policy (for example, making sure a user is allowed to make a call). A proxy interprets, and, if necessary, rewrites specific parts of a request message before forwarding it.
Stateless Proxy:
A logical entity that does not maintain the client or server transaction state machines defined in this specification when it processes requests. A stateless proxy forwards every request it receives downstream and every response it receives upstream.

Σχήμα 3.5

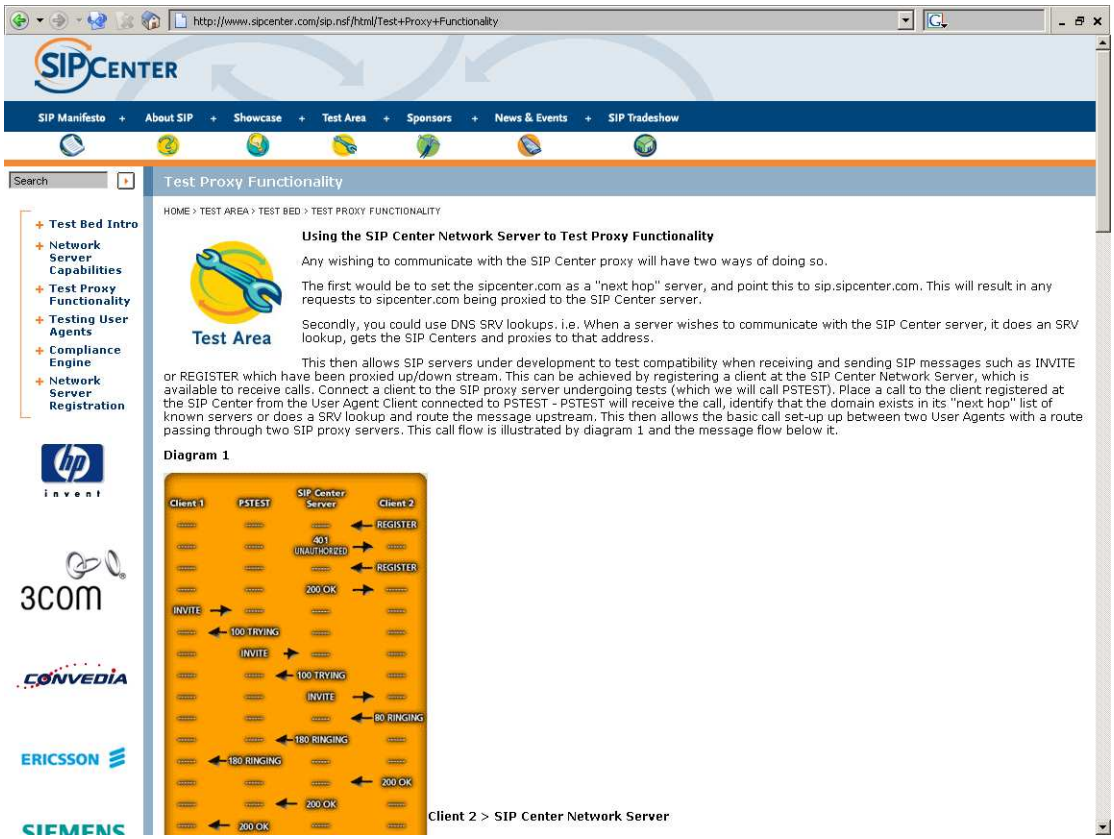
Public SIP Proxy στο Sip.sipcenter.com

Ο SIP *Proxy/Registrar server* του www.sipcenter.com (βλέπε σχήμα 3.6) είναι διαθέσιμος από οποιαδήποτε *public* IP διεύθυνση. Πρόκειται για το *hostname* του SIP *proxy server* που προσφέρει το SIP Center [25]. Για να μπορέσει κανείς να πειραματιστεί με αυτόν τον *proxy* θα πρέπει να δημιουργήσει μια καταχώρηση στην διεύθυνση που αναφέρεται παραπάνω δίνοντας τα στοιχεία του και ένα *email* στο οποίο θα αποσταλούν το όνομα χρήστη (*username*) και ο κωδικός για την χρήση της σηγκεκριμένης υπηρεσίας. Αντίθετα με τον προηγούμενο *sip proxy server*, σε αυτόν δεν υπάρχει η δυνατότητα να κάνει κανείς *register* χρησιμοποιώντας ένα οποιοδήποτε SIP URI. Η αποδεκτή SIP URI είναι της παρακάτω μορφής:

`sip:`**`username`**`@sipcenter.com`

Όπως γίνεται κατανοητό για να μπορέσει να λειτουργήσει ένας *sip proxy server* είναι αναγκαία ή ύπαρξη ενός *Registrar*. Ο *registrar* του SIP Center [25] βρίσκεται επίσης στο [`sip.sipcenter.com`].

Αν και θα παρουσίαζαν αρκετό ενδιαφέρον, δεν υπάρχουν διαθέσιμες πληροφορίες για το *hardware* που φιλοξενεί αυτούς τους servers.



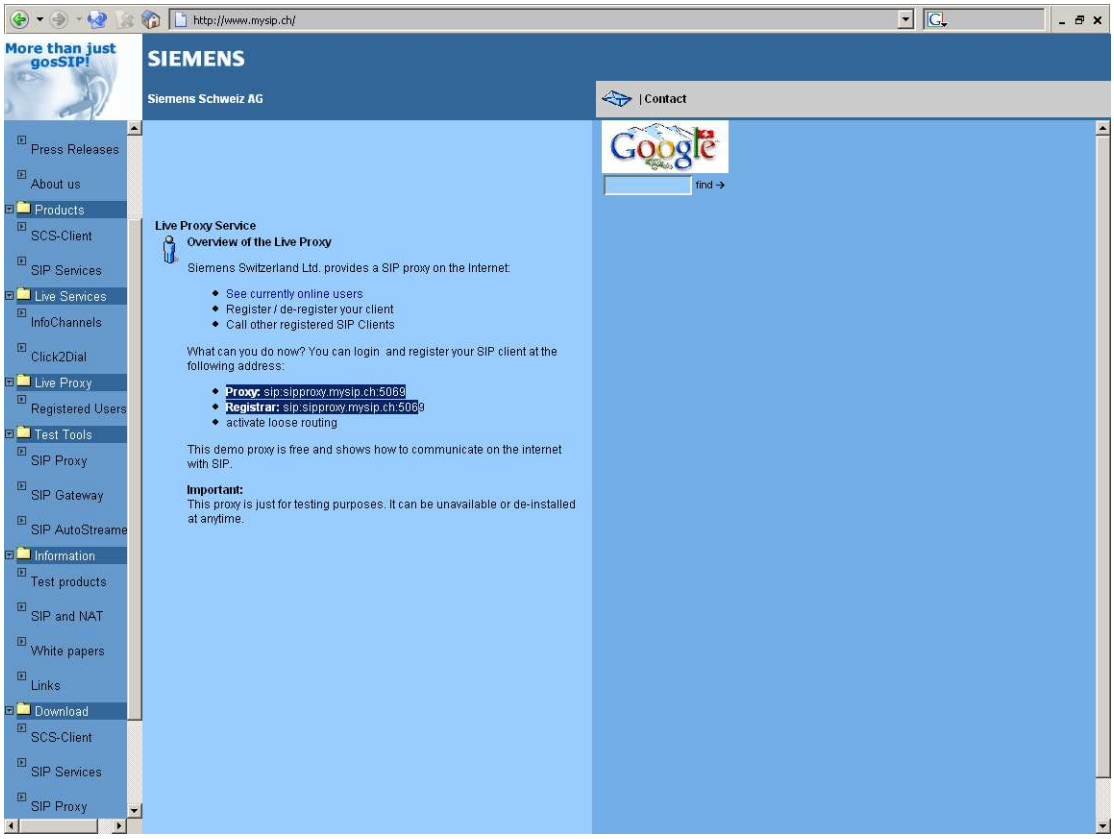
Σχήμα 3.6

Live Proxy – MySIP.ch

Ενναλακτική λύση διαθέσιμου online SIP *Proxy/Registrar server* αποτελεί και ο **Live Proxy** του *MySIP.ch* [23] (βλέπε σχήμα 3.7). Εδώ δεν υπάρχουν διαδικασίες καταχώρησης ενώ οι ρυθμίσεις που απαιτούνται από τους *Clients* είναι*:*

- **Proxy:** sip:sipproxy.mysip.ch:5069
- **Registrar:** sip:sipproxy.mysip.ch:5069

Θα πρέπει να σημειωθεί πως η χρήση της υπηρεσίας δεν μπορεί να θεωρείται δεδομένη και μπορεί να ανασταλεί χωρίς καμία προειδοποίηση.



Σχήμα 3.7

3.4 Διάταξη Υλοποίησης και Εφαρμογής SIP Πρωτοκόλλου

Τα κριτήρια επιλογής εξοπλισμού, με βάση τα οποία έχει προκύψει η παρακάτω διάταξη (βλέπε σχήμα 3.9) ήταν:

- Ανεξαρτησία από διαθεσιμότητα σύνδεσης στα sites μέσω Internet
- Λόγοι ευελιξίας
- Μη διαθεσιμότητα SIP phones
- Δυνατότητα εξοικείωσης με πληθώρα σεναρίων επικοινωνίας μέσω χρήσης του SIP πρωτοκόλλου

Απαιτείται, λοιπόν, η ύπαρξη 3 ή 4 υπολογιστών (PC) με λειτουργικό Windows XP ή Windows 2000/NT συνδεδεμένοι σε ένα *Ethernet* δίκτυο. Προαιρετικά το *Ethernet* δίκτυο μπορεί να έχει σύνδεση στο Ίντερνετ. Οι ελάχιστες απαιτήσεις που θα πρέπει να πληρούν τα PCs περιγράφονται από τον παρακάτω πίνακα (βλέπε σχήμα 3.8). Για να γίνουν οι κατάλληλες ρυθμίσεις απαιτούνται ασφαλώς δικαιώματα χρήσης Διαχειριστή Συστήματος.

Platform Type	Software	Operating System Features	CPU	Memory (MB)	Free Disk (MB)
Intel Processor or equivalent	SIP User Agent 2.0	Windows NT version 4 Workstation and Server	Pentium II 400	128	50

Σχήμα 3.8

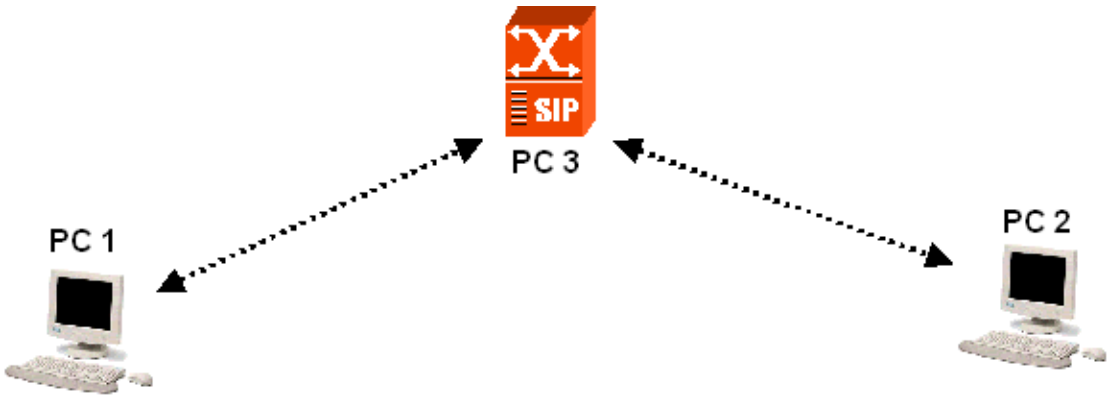
Επιπλέον, οι εφαρμογές που χρησιμοποιούνται στην πειραματική διαδικασία είναι βασισμένες στην τεχνολογία JAVA. Έτσι λοιπόν για να λειτουργήσουν είναι απαραίτητο να είναι προεγκατεστημένη η εικονική μηχανή ή το Software Development Kit της JAVA.

Software Version/Specification	Software Dependency
SIP User Agent release 2.0	JRE 1.3

Σχήμα 3.9

Εξετάστηκαν τα κάτωθι σενάρια υλοποίησης:

Α. Σύνδεση δύο SIP *Clients* σε έναν SIP *Proxy* (βλέπε σχήμα 3.10)
Β. Σύνδεση δύο SIP *Clients* μέσω δύο SIP *Proxies* (βλέπε σχήμα 1.3)



Σχήμα 3.10

PC 1: Σε αυτόν τον υπολογιστή θα τρέχει το *Software* του *SCS-Client v1.0.0*
PC 2: Σε αυτόν τον υπολογιστή θα τρέχει το *Software* του *Helmsman® User Agent*
PC 3: Σε αυτόν τον υπολογιστή θα τρέχει το *Software* του *SCS-Proxy V1.0.0.1*

Θεωρείται πως όλοι οι υπολογιστές ανήκουν στο ίδιο *domain*. Αν το *domain* είναι το chania.teicrete.gr τότε τα SIP URIs θα έχουν την παρακάτω μορφή

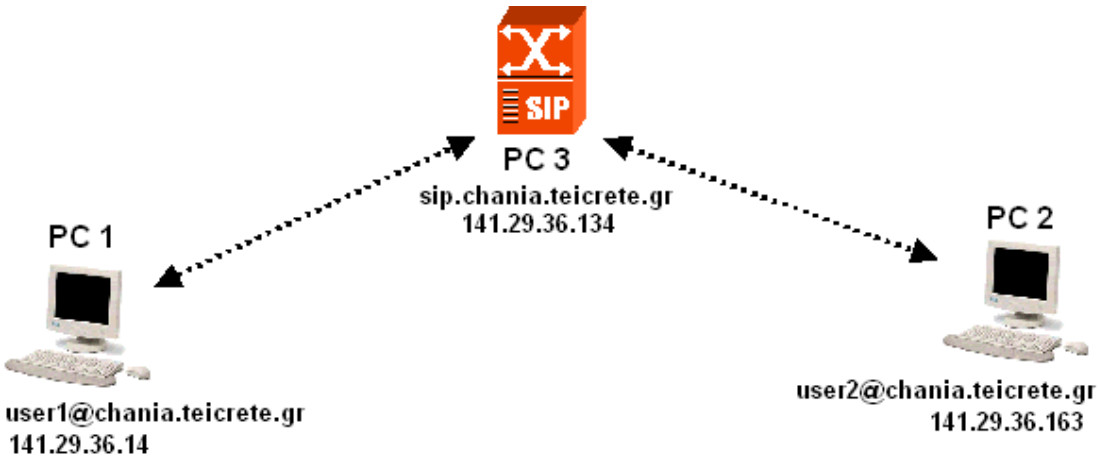
`sip:`**`username`**`@chania.teicrete.gr`

Στο σενάριο Α παρουσιάζεται η διασύνδεση των 2 SIP users:

`PC 1` `: sip:`**`user1`**`@chania.teicrete.gr`

`PC 2` `: sip:`**`user2`**`@chania.teicrete.gr`

Και έτσι προκύπτει το ακόλουθο σχήμα 3.11.



Σχήμα 3.11

Για να επιτευχθεί επικοινωνία στο SIP πρωτόκολλο θα πρέπει να γίνουν οι παρακάτω ρυθμίσεις στους UAs:

```
Proxy server        : sip.chania.teicrete.gr:5060

Registrar           : sip.chania.teicrete.gr:5060

Public SIP address : sip:usern@chania.teicrete.gr,
```

όπου usern αντικαθίσταται από user1 ή user2 ανάλογα με την περίπτωση.

Σχήμα 3.12

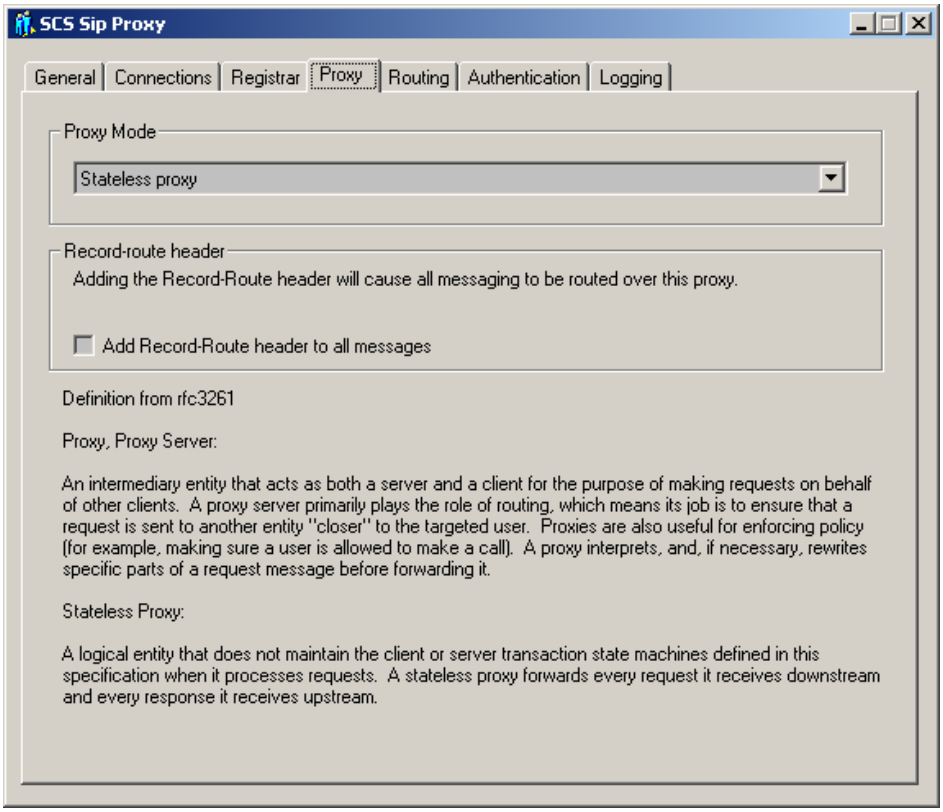
Οι ρυθμίσεις του SIP *proxy server* αφορούν στον τρόπο λειτουργίας του. Θα πρέπει λοιπόν να οριστεί το *domain* για το οποίο θα γίνονται δεκτά τα *registration*. Αυτό θα είναι ίδιο με το *domain* των SIP URIs δηλαδή

```
Domain name : chania.teicrete.gr
```

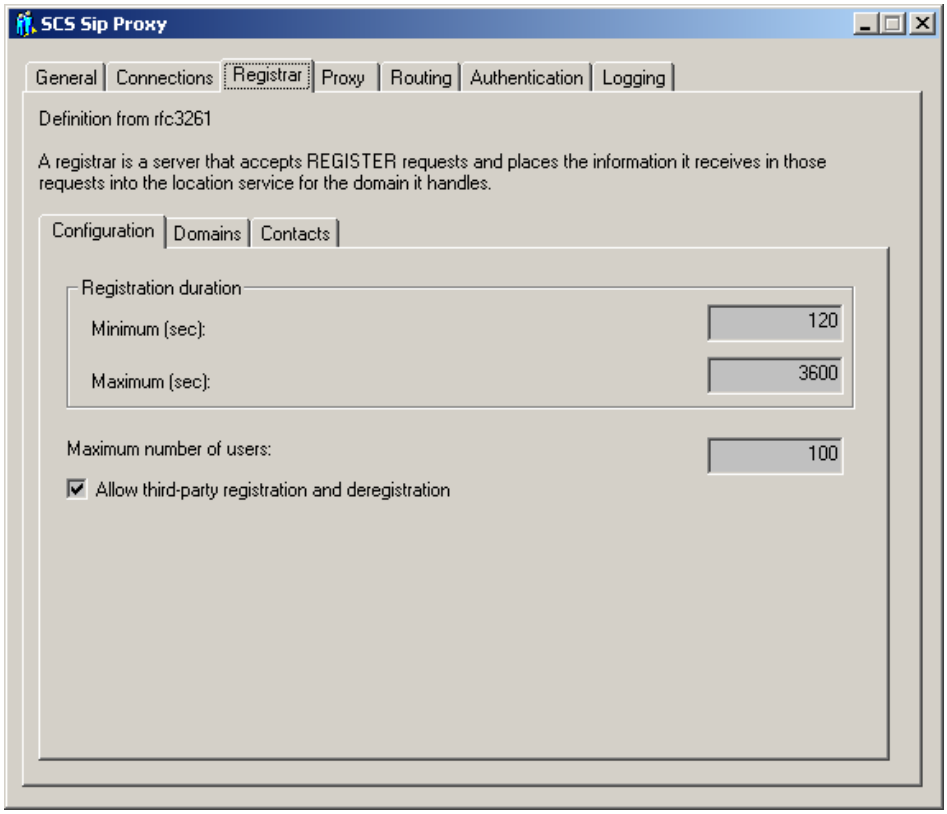
Επίσης ειδικές ρυθμίσεις μπορούν να δοθούν για το χρονικό όριο που θα διαρκούν τα *register requests*. Κάτι πολύ σημαντικό είναι η λειτουργία του *proxy* (*stateless* / *redirect*). Σε αυτήν την εφαρμογή ο *proxy server* θα λειτουργήσει ως *stateless proxy*. Τα screenshots των σχημάτων 3.13, 3.14, 3.15 δίνουν σαφή εικόνα για τις ρυθμίσεις που πρέπει να γίνουν.

Όπως παρατηρήθηκε δεν έγινε χρήση συνδυασμού *username* και *password* για το *registration* που εκτελεί ο κάθε UA *client* στον *Registrar* του sip.chania.teicrete.gr. Αυτό υποδεικνύει την έλλειψη ασφάλειας στο SIP δίκτυο. Ωστόσο, αυτό το γεγονός δεν πρέπει να αποτελεί λόγο ανησυχίας καθώς ο SIP *proxy server* της Siemens παρέχεται δωρεάν μονάχα για πειραματισμό και έρευνα.

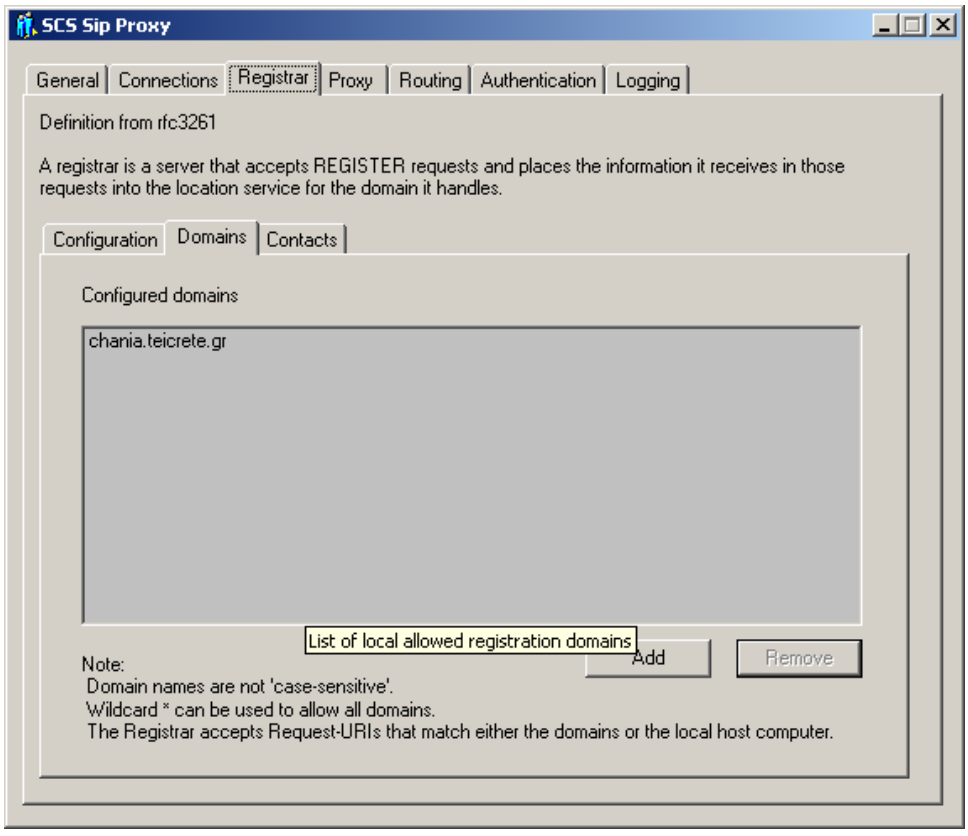
Η λειτουργία του *server* ως *stateless proxy*, το χρονικό περιθώριο μεταξύ των *registrations*, και το *domain* για το οποίο επιτρέπονται τα *registrations* εικονίζονται στα *sceenshots* των σχημάτων 3.13, 3.14, 3.15 .



Σχήμα 3.13



Σχήμα 3.14



Σχήμα 3.15

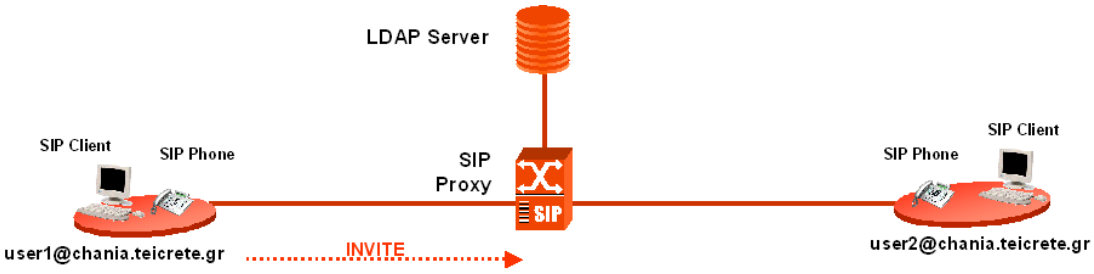
3.5 Αναλυτική Διαδιακασία Εφαρμογής SIP Πρωτοκόλλου

Η διεξαγωγή μιας συνόδου (αποκατάσταση σύνδεσης, επικοινωνία, αποσύνδεση) με χρήση του SIP πρωτοκόλλου κατά το σενάριο Α βασίζεται στην διάταξη υλοποίησης του σχήματος 3.11.

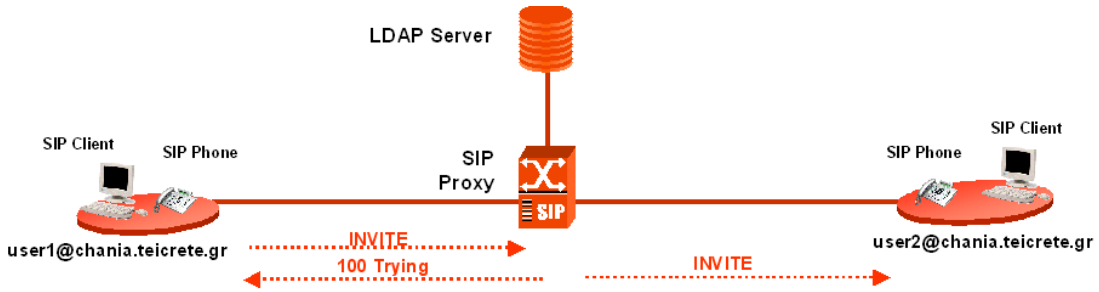
Στην πειραματική διάταξη έχουμε το σενάριο όπου ο χρήστης με την *sip address* `sip:user1@chania.teicrete.gr` καλεί τον χρήστη με *sip address* `sip:user2@chania.teicrete.gr`, επιτυγχάνεται επικοινωνία και στη συνέχεια ο *userB* προκαλεί τον τερματισμό της συνόδου. Η συνεπακόλουθη ανταλλαγή μηνυμάτων απεικονίζεται στο σχήμα 3.15, αναλυτική καταγραφή των οποίων ακολουθεί:

220	9.556201	141.29.36.14	141.29.36.134	SIP	Request: REGISTER sip:141.29.36.134
221	9.557827	141.29.36.134	141.29.36.14	SIP	Status: 200 OK (1 bindings)
745	33.120900	141.29.36.14	141.29.36.134	SIP/SDP	Request: INVITE sip:user2@chania.teicrete.gr, with session description
746	33.139687	141.29.36.134	141.29.36.14	SIP	Status: 100 Trying
753	33.280083	141.29.36.134	141.29.36.14	SIP/SDP	Status: 180 Ringing, with session description
897	39.935571	141.29.36.134	141.29.36.14	SIP/SDP	Status: 200 OK, with session description
899	39.964597	141.29.36.14	141.29.36.134	SIP	Request: ACK sip:user2@141.29.36.163:5060
1420	44.343543	141.29.36.163	141.29.36.14	SIP	Request: BYE sip:user1@141.29.36.14:5060
1421	44.354226	141.29.36.14	141.29.36.163	SIP	Status: 200 OK
1429	44.843486	141.29.36.163	141.29.36.14	SIP	Request: BYE sip:user1@141.29.36.14:5060
1430	44.843875	141.29.36.14	141.29.36.163	SIP	Status: 200 OK

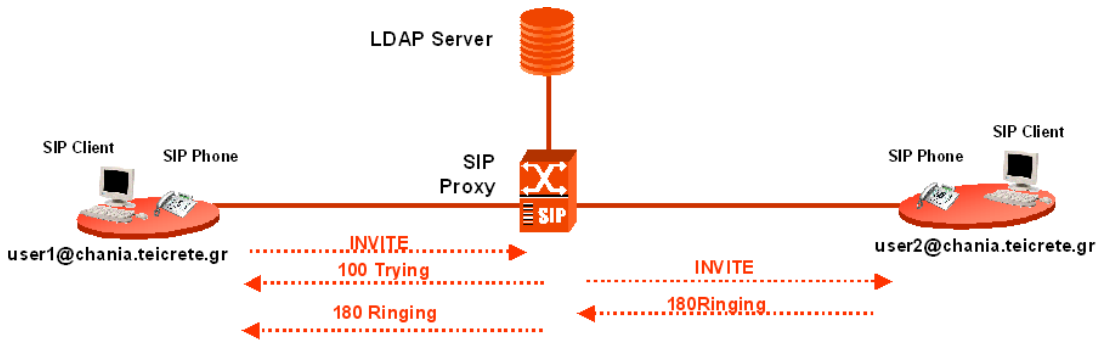
Σχήμα 3.15



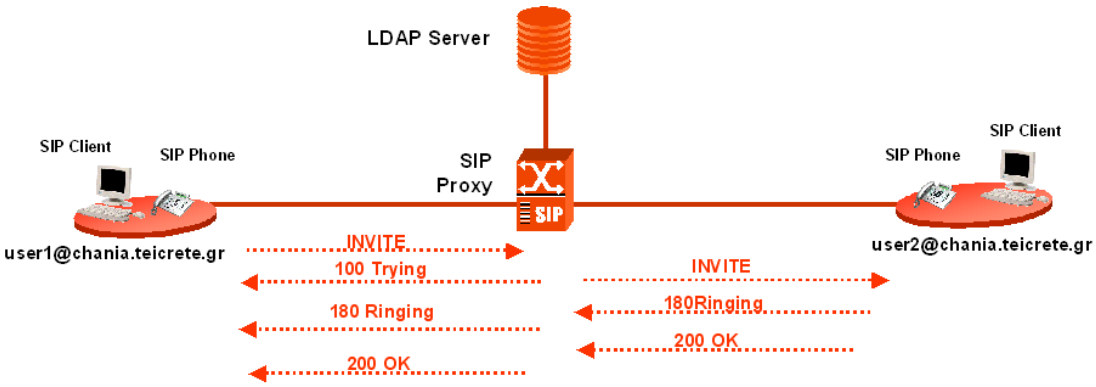

```
INVITE sip:user2@chania.teicrete.gr SIP/2.0
Via: SIP/2.0/UDP 141.29.36.14:5060
To: sip:user2@chania.teicrete.gr
From:Ioannis Papoutsis
<sip:user1@chania.teicrete.gr>;tag=0b787bccc46ed13
Call-ID: 9931f74b4cc1a3c@chania.teicrete.gr
CSeq: 1980991524 INVITE
Max-Forwards: 70
Allow: INVITE, ACK, OPTIONS, BYE, CANCEL, REGISTER, REFER,
SUBSCRIBE, NOTIFY, MESSAGE
Content-Type: application/sdp
Content-Length: 267
Contact: Ioannis Papoutsis <sip:user1@141.29.36.14:5060>
User-Agent: SCS/v3.1.12.33
```



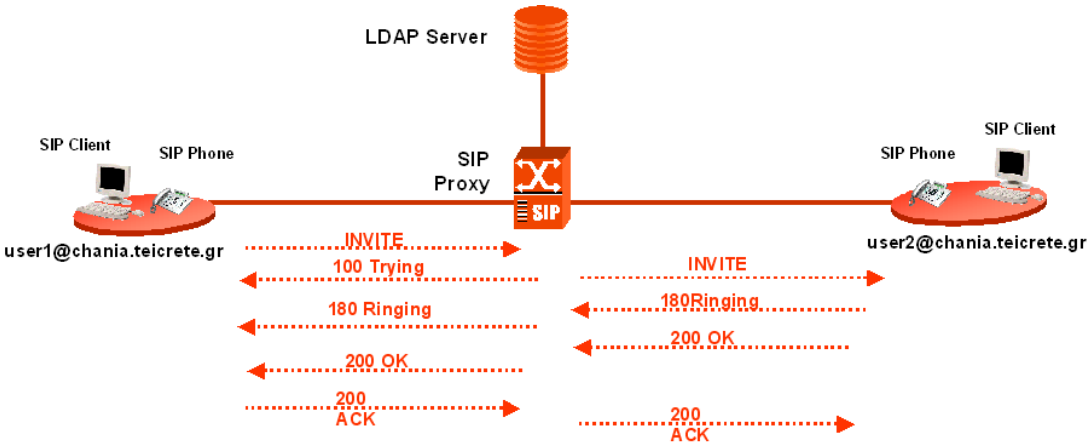

```
SIP/2.0 100 Trying
To: sip:user2@chania.teicrete.gr
From: Ioannis Papoutsis
<sip:user1@chania.teicrete.gr>;tag=0b787bccc46ed13
CSeq: 1980991524 INVITE
Call-ID: 9931f74b4cc1a3c@chania.teicrete.gr
Via: SIP/2.0/UDP 141.29.36.14:5060
Content-Length: 0
```



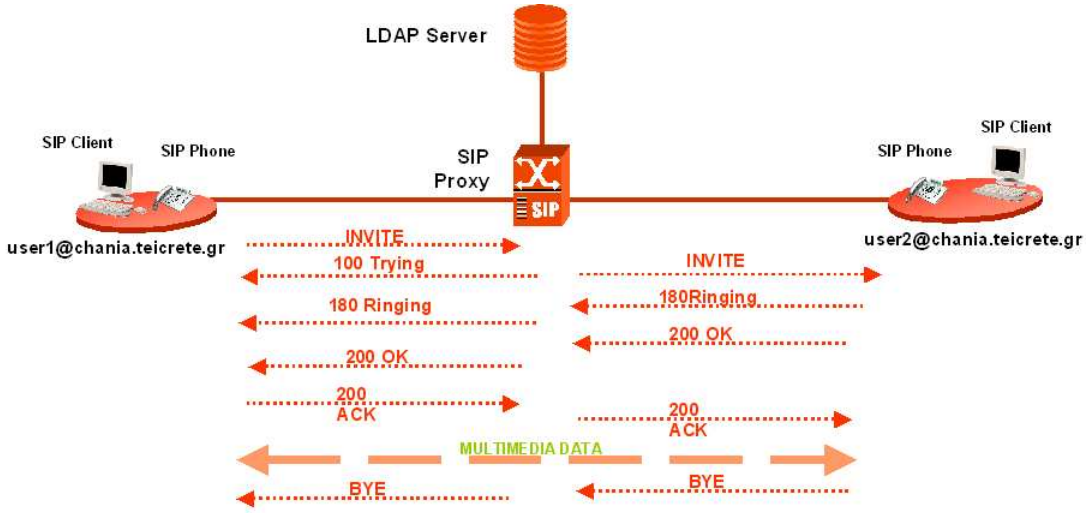

```
SIP/2.0 180 Ringing
To: sip:user2@chania.teicrete.gr;tag=957343957
From: Ioannis Papoutsis
<sip:user1@chania.teicrete.gr>;tag=0b787bccc46ed13
CSeq: 1980991524 INVITE
Call-ID: 9931f74b4cc1a3c@chania.teicrete.gr
Via: SIP/2.0/UDP 141.29.36.14:5060
Content-Type: application/sdp
Contact: sip:user2@141.29.36.163:5060
Content-Length: 0
```



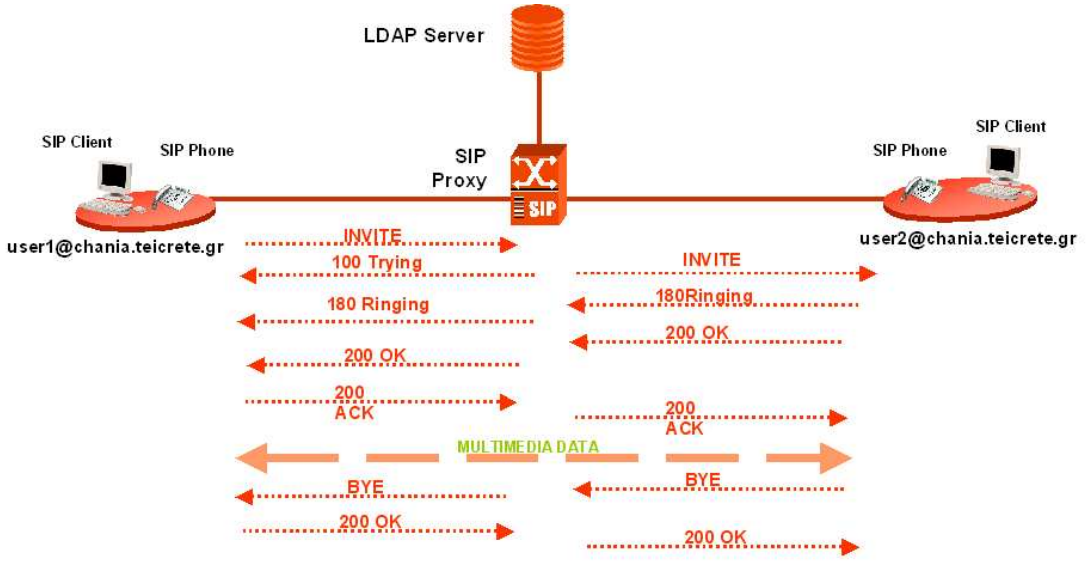

```
SIP/2.0 200 OK
To: sip:user2@chania.teicrete.gr;tag=957343957
From: Ioannis Papoutsis
<sip:user1@chania.teicrete.gr>;tag=0b787bccc46ed13
CSeq: 1980991524 INVITE
Call-ID: 9931f74b4cc1a3c@chania.teicrete.gr
Via: SIP/2.0/UDP 141.29.36.14:5060
Content-Type: application/sdp
Contact: sip:user2@141.29.36.163:5060
Content-Length: 121
```




```
ACK sip:user2@141.29.36.163:5060 SIP/2.0
Via: SIP/2.0/UDP 141.29.36.14:5060
To: sip:user2@chania.teicrete.gr;tag=957343957
From: Ioannis Papoutsis
<sip:user1@chania.teicrete.gr>;tag=0b787bccc46ed13
Call-ID: 9931f74b4cc1a3c@chania.teicrete.gr
CSeq: 1980991524 ACK
Max-Forwards: 70
Content-Length: 0
Contact: Ioannis Papoutsis <sip:user1@141.29.36.14:5060>
User-Agent: SCS/v3.1.12.33
```




```
BYE sip:user1@141.29.36.14:5060 SIP/2.0
Call-ID: 9931f74b4cc1a3c@chania.teicrete.gr
Content-Type: application/sdp
To:  Ioannis Papoutsis
<sip:user1@chania.teicrete.gr>;tag=0b787bccc46ed13
From: sip:user2@chania.teicrete.gr;tag=957343957
CSeq: 1 BYE
Content-Length: 0
Via: SIP/2.0/UDP
141.29.36.163:5060;branch=8D1D24A313C4000000FDA38C4562-1*0
```




```
SIP/2.0 200 OK
Call-ID: 9931f74b4cc1a3c@chania.teicrete.gr
CSeq: 1 BYE
From: sip:user2@chania.teicrete.gr;tag=957343957
To: Ioannis Papoutsis
<sip:user1@chania.teicrete.gr>;tag=0b787bccc46ed13
```

```
Via: SIP/2.0/UDP
141.29.36.163:5060;branch=8D1D24A313C4000000FDA38C4562-1*0
Content-Length: 0
User-Agent: SCS/v3.1.12.33
```

Μια διαφοροποίση μεταξύ εφαρμογής του SIP πρωτοκόλλου και της συμβατικής τηελφωνίας αποτελεί το γεγονός ότι αν η πλατφόρμα επικοινωνίας ήταν το PSTN, τότε θα έπρεπε απλά ο *user1* να γνωρίσει τον τηλεφωνικός αριθμό του *user2*. Το τηλεφωνικό κέντρο γνωρίζει σε ποιο *port* βρίσκεται συνδεδεμένος ο τηλεφωνικός αριθμός του *user2* και αντίστοιχα δρομολογεί την κλήση. Η μεγάλη διαφορά στο SIP είναι ότι ο κάθε χρήστης δεν έχει μια σταθερή τοποθεσία. Έτσι όταν κάποιος χρήστης κάνει *log on* σε κάποιο τερματικό θα πρέπει να δηλώνει σε κάποιον *server* πως είναι διαθέσιμος να δεχτεί κλήσεις (κάτι σαν την διαδικασία *location update* στην κινητή τηλεφωνία). Αυτός ο *server* είναι ο *proxy/registrar server*, και η μόνη διεύθυνση που πρέπει να γνωρίζει σε αυτή την περίπτωση ο *client* είναι αυτή του *proxy/registrar server*.

3.6 Θέματα Περαιτέρω Μελέτης

Υπάρχουν πολλά θέματα με τα οποία μπορεί να ασχοληθεί κανείς για να δώσει συνέχεια σε αυτή τη πτυχιακή εργασία, όπως :

- Λειτουργία του SIP πρωτοκόλλου σε συνεργασία με έναν PSTN Gateway και τα σχετικά προκύπτοντα ζητήματα αριθμοδότησης
- Μελέτη του SIP πρωτοκόλλου στην επικοινωνία αυτού με τον έξω κόσμο όταν στο τοπικό δίκτυο τρέχει NAT
- Υλοποίηση (προγραμματισμός σε C++) εφαρμογής που να παίζει τον ρόλο ενός SIP client
- Υλοποίηση (προγραμματισμός σε C++) εφαρμογής που να παίζει τον ρόλο ενός Signaling Gateway και να υποστιρίζει το SIP αλλά και το πρωτόκολλο H.323

Γενικότερα, αξίζει να αναφερθεί ο ρόλος που το SIP πρωτόκολλο θα διαδραματίσει στην εξέλιξη των επικοινωνιών στο άμεσο μέλλον. Ενδεικτικά της χρήσης του αποτελούν τα στοιχεία του πίνακα του σχήματος 3.16 και το γεγονός ότι τα αναμενόμενα έσοδα που θα προκύψουν από υπηρεσίες σχετιζόμενες με το SIP πρωτόκολλο θα ανέλθουν από 67,31 εκαταμύρια ευρώ το 2003 στα 2857 εκατομύρια ευρώ το 2007.
(πηγή πληροφοριών: Analysys Research)

	2002	2003	2004	2005	2006	2007
Sip connected phones	0.000	0.164	0.394	0.912	1.876	3.817
Sip fixed messaging	12.93	15.91	20.11	26.47	28.39	31.83
Mobile	0.000	0.271	1.778	5.953	12.77	20.96
Total Sip Clients in use	12.93	16.34	22.28	33.33	43.04	56.61

Σχήμα 3.16

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] RFC 1889 - RTP: A Transport Protocol for Real-Time Applications

[2] RFC 2326 - Real Time Streaming Protocol (RTSP)

[3] RFC 3015 - Megaco Protocol Version 1.0

[4] RFC 2327 - SDP: Session Description Protocol

[5] RFC 2822 - Internet Message Format

[6] RFC 2616 - Hypertext Transfer Protocol

[7] RFC 2396 - Uniform Resource Identifiers (URI)

[8] RFC 2806 - URLs for Telephone Calls

[9] RFC 3261 - SIP: Session Initiation Protocol

[10] RFC 2046 - Multipurpose Internet Mail Extensions (MIME)

[11] RFC 2543 - SIP: Session Initiation Protocol

[12] http://www.astalavista.box.sk

[13] RCF 2368 - The mailto URL scheme

[14] http://www.acm.org/sigmm/MM97/papers/malpani/ qsbmm97.html

[15] RCF 3305 - Uniform Resource Identifiers

[16] RCF 819 - Domain naming convention for Internet applications

[17] http://c2.com/cgi/wiki?BackusNaurForm

[18] RCF 1341 - MIME (Multipurpose Internet Mail Extensions)

[19] RCF 2068 - Hypertext Transfer Protocol -- HTTP/1.1

[20] RCF 2311 - S/MIME Version 2 Message Specification

[21] RFC 2716 - PPP EAP TLS Authentication Protocol

[22] http://www.siemens.ch

[23] http://www.mysip.ch

[24] http://www.ubiquity.net

[25] http://www.sipcenter.com

[26] http://www.cisco.com