ΤΕΧΝΟΛΟΓΙΑ ΛΟΓΙΣΜΙΚΟΥ
Χειμερινό Εξάμηνο
2016
SIP COMMUNICATOR

Design Document

Version:	1.0
Print Date:	
Release Date:	21/12/2016
Release State:	Αρχικό
Approval State:	Πρόχειρο
Approved by:	
Prepared by:	Φάνης Καραμπλιάς (el12174) Νικόλαος Καψούλης (el11110) Γρηγόρης Μπαλάσκας (el12005) Γεώργιος Πατσέας (el11103)
Reviewed by:	
Path Name:	
File Name:	SDD_g18.docx
Document No:	

Document Change Control

Version	Date	Authors	Summary of Changes

Document Sign-Off

Name (Position)	Signature	Date

Περιεχόμενα

1 Εισαγωγη

1.1 ΠΕΡΙΛΗΨΗ

1.2 Αναφορές

2 Βασικες Σχεδιαστικες Αποφασεις

2.1 Βαση Δεδομενων

2.1.1 Users

2.1.2 Calls

2.1.3 Block_List

2.1.4 Forward_List

2.2 Συμβατοτητα με την υπάρχουσα δομή και πρωτόκολλο

2.2.1 Ο Proxy server μεσολαβεί στη φραγή και την προώθηση

2.2.2 Επικοινωνία client με τη βάση δεδομένων

3 Αρχιτεκτονικη

3.1 ψηφιδικά διαγράμματα

3.2 παραταξιακά διαγράμματα

4 Λεπτομερή διαγραμματα κλασεων

4.1 Διαγράμματα κλάσεων σε UML

4.2 Λεπτομέρειες των μεθόδων

4.2.1 SipProxy

4.2.1.1 ProxyCall

4.2.1.2 Database Server

4.2.1.3 BlockingServer

4.2.1.4 ForwardingServer

4.2.1.5 TimeServer

4.2.1.6 Policy

4.2.1.7 Billing Server

4.2.1.8 Observer

4.2.1.9 CallStart

4.2.1.10 CallEnd

4.2.1.11 Proxy

4.2.2 SIP Communicator

4.2.2.1 GuiManager

4.2.2.2 AuthenticationSplash

4.2.2.3 RegisterUser

4.2.2.4 BlockingActions

4.2.2.5 ForwardingActions

4.2.2.6 PolicySelection

5 Διαγραμμα ΚΑΤΑΣΤΑΣΗΣ

6 ΑΝΟΙΚΤΑ ΖΗΤΗΜΑΤΑ

6.1 Ανεξέλεγτη χρέωση καλούντος

6.2 Ανίχνευση κύκλων στην προώθηση κλήσης

6.3 Σχεσιακό Μοντέλο Βάσης

6.4 Πολιτικές Χρέωσης

1 Εισαγωγή

1.1 Περίληψη

Σε αυτή την άσκηση εργαστηρίου ο σκοπός είναι να σχεδιάσουμε και να υλοποιήσουμε τρεις νέες λειτουργίες στο πρόγραμμα πελάτη SIP Communicatorκαι στο πρόγραμμα εξυπηρέτησης JAINSIP Proxy. Τα προγράμματα SIP Communicator και JAINSIPPROXY υλοποιούν το πρωτόκολλο SessionInitiationProtocol(SIP), το οποίο ορίζεται στο πρότυπο RFC3261[1]που έχει εκδοθεί από τον οργανισμό National Institute of Standards and Technology (NIST). Το πρωτόκολλο SIP επιτρέπει μεταξύ άλλων την εξέλιξη εφαρμογών επικοινωνίας μέσω Διαδικτύου. Μια τέτοια εφαρμογή είναι και η τηλεφωνία μέσω Διαδικτύου ή όπως είναι περισσότερο γνωστή σαν VoiceoverIP (VoIP).Για την άσκηση,οι νέες λειτουργίες που θα εστιάσουμε την προσοχή μας αφορούν τη

1. δυνατότητα προώθησης κλήσης (callforwarding), όπου όταν ο χρήστης Α καλέσει έναν χρήστη Β, ο Β μπορεί να προωθήσει την κλήση αυτήν σε ένα χρήστη Γ και αν αυτός απαντήσει να επικοινωνήσει με τον

Α. Επιτρέπεται μάλιστα και επαναπροώθηση από τον Γ κοκ. Ο μόνος περιορισμός μας είναι ότι απαγορεύονται οι κυκλικές προωθήσεις.

2. τη δυνατότητα περιορισμού εισερχομένων κλήσεων (callblocking), όπου αν ο χρήστης Α έχει βάλει τον χρήστη Β σε λίστα με ανεπιθύμητους, δε θα δέχεται κλήσεις από τον Β και ο Β θα βλέπει ότι ο Α δεν είναι διαθέσιμος και

3. την υποδομή χρέωσης κλήσεων (billing), όπου θα χρεώνεται μόνο ο χρήστης Α ο οποίος και ξεκίνησε την κλήση, και η χρέωση αυτή θα προστίθεται στο λογαριασμό του στο τέλος της κλήσης.

Ο πηγαίος κώδικας για τα προγράμματα SIP Communicator και JAINSIP Proxy διανέμονται σαν προγράμματα ανοικτής χρήσης (opensource).

Στο SDD αυτό αναλύουμε τα παρακάτω ζητήματα:

- Κεφάλαιο 2: Αναλύουμε και εξηγούμε τις βασικές σχεδιαστικές αοφάσεις που πάρθηκαν για την υλοποίση αυτήν της εργασίας
- Κεφάλαιο 3: Παρουσιάζουμε τα component diagrams, deployment diagrams και την αρχιτεκτονική του συστήματος. Έτσι μπορούμε αντίστοιχα, να κατανοήσουμε καλύτερα τη δομή του συστήματος και το πως αναλύεται και σε υπομονάδες, τη διάταξη των υπομονάδων και τη συνολική λογική στη σχεδιάση του συστήματος.
- Κεφάλαιο 4: Παρουσιάζουμε τα class diagrams και δίνουμε λεπτομέρειες για το τι κάνει κάθε μέθεδος, ώστε να γίνεται σε μεγάλο βαθμό κατανοητή η υλοποίσηση τεχνικά του συστήματος
- Κεφάλαιο 5: Παρουσιάζουμε τα state diagrams, όπου μπορούμε να δούμε και πως χρησιμοποιείται η κύρια κλάση
- Κεφάλαιο 6: Παρουσιάζουμε τα ανοιχτά ζητήματα πάνω στην εργασία μας. Αυτά είναι κομμάτια τα οποία δεν έχουν αποφασιστεί ακόμα, καθώς είτε χρειάζονται περαιτέρω έρευνα είτε επειδή απαιτείται να υλοποιηθούν κάποια άλλα κομμάτια πρώτα.

1.2 Resources - References

[1] RFC 3261 - SIP : Session Initiation Protocol : http://www.ietf.org/rfc/rfc3261.txt

[2] Eclipse: http://www.eclipse.org/downloads/index.php

[3] SIP Proxy : Από τον ιστότοπο του μαθήματος στο mycourses.ntua.gr

[4] SIP Communicator : Από τον ιστότοπο του μαθήματος στο mycourses.ntua.gr

[5]Απαιτήσεις Project: Project-Description-gr-v2.0-2016.pdf από mycourses.ntua.gr

[6] SRS Document: https://bitbucket.org/softengntua2016/softeng-sip-project-g18/SRS_g18.pdf

2 Βασικές Σχεδιαστικές Αποφάσεις

Οι παρακάτω αποφάσεις πάρθηκαν με πρώτο κριτήριο να καλυφθεί η πλήρης λειτουργικότητα του συστήματος. Επιπλέον κριτήριο ήταν να χρησιμοποιηθεί όσο καλύτερα γίνεται η υπάρχουσα δομή του πηγαίου κώδικα και οι δυνατότητες που δίνει με βάση τη δομή του, αλλά και να είναι ξεκάθαρες οι επεμβάσεις μας ώστε να μπορούν να είναι εύκολα κατανοητές αλλά και επεκτάσιμες σε ένα επόμενο στάδιο.

2.1 Βάση Δεδομένων

Απαραίτητη για τη λειτουργικότητα κρίθηκε το να μπορούμε να αποθηκεύουμε κάποια στοιχεία, γενικά για το σύστημα αλλά και για κάθε χρήστη συγκεκριμένα. Για αυτό κρίθηκε αναγκαία η υλοποίηση μίας βάσης δεδομένων. Επιλέξαμε να κάνουμε χρήση **σχεσιακής** βάσης δεδομένων (MySQL) διότι βοηθούσε στην κατηγοριοποίηση των στοιχείων ανά χρήστη. Τα δεδομένα είναι λίγα άρα η αποδοτικότητα επηρεάζεται ελάχιστα, οπότε βασικό κριτήριο επιλογής μας ήταν η εύκολη συμβατότητα με το υπάρχον σύστημα, η σταθερότητα μίας τόσο δοκιμασμένης λύσης και η εύκολη συσχέτιση δεδομένων μέσω αυτής.

Η βάση μας θα έχει τα εξής tables:

2.1.1 Users

Εδώ κρατάμε τα στοιχεία κάθε χρήστη

id (integer)
username (chars)
password (string) (κρυπτογράφηση?)
bill (float)

2.1.2 Calls

Στον πίνακα αυτό κρατάμε τα στοιχεία που είναι απαραίτητα για κάθε κλήση, ώστε να μπορούμε αναλόγως στο τέλος να υπολογίσουμε και το σημαντικό πεδίο της χρέωσης. Αυτό γίνεται αναλόγως με τη διάρκεια της κλήσης και το policy κάθε χρήστη.

id (integer)
caller (integer -> id του πίνακα users)
callee (integer -> id του πίνακα users)
start (float)
end (float)
cost (float)

2.1.3 Block_List

Ποιος χρήστης έχει μπλοκάρει ποιον.

blocker (integer -> id του πίνακα users)
blockee (integer -> id του πίνακα users)

2.1.4 Forward_List

Πίνακας που δείχνει σε ποιον προωθεί κάθε χρήστης τις κλήσεις του. Αν δεν τις προωθέι σε κανένα θα είναι null το forwardee.

forwarder (integer -> id του πίνακα users)
forwardee (integer -> id του πίνακα users)

2.2 Συμβατότητα με την υπάρχουσα δομή και πρωτόκολλο

Σε αυτό το σημείο καλούμαστε να αποφασίσουμε με βάση τον υπάρχοντα κώδικα και τη μορφή του πρωτοκόλλου, πως θα γίνουν κάποιες βασικές εργασίες του συστήματος μας.

2.2.1 Ο Proxy server μεσολαβεί στη φραγή και την προώθηση

Η προώθηση και το μποκάρισμα των κλήσεων γίνεται “server-side” όπως και επιτάσσει το πρωτόκολλο SIP. Ο χρήστης (client) στέλνει στο server με ποιον θέλει να μιλήσει και στη συνέχεια όλες οι διεργασίες γίνονται υπολογίζονται εκεί.

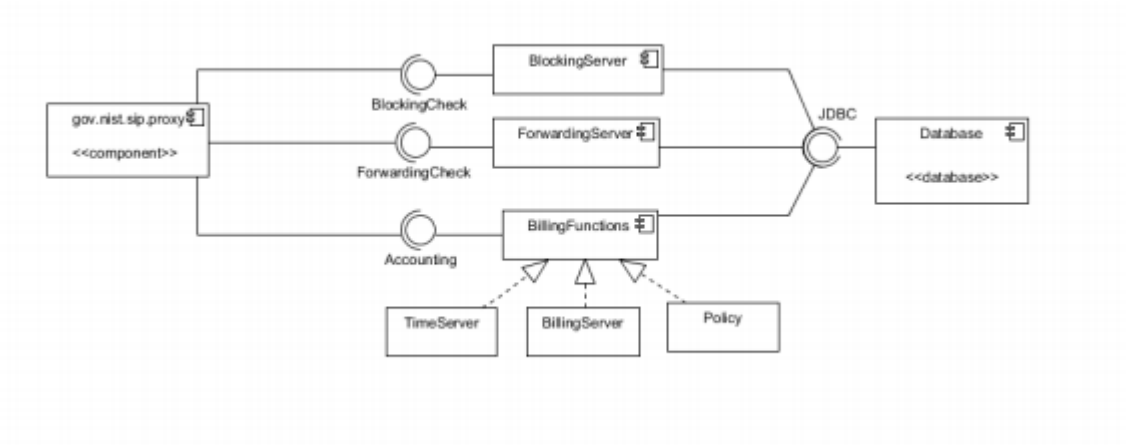
2.2.2 Επικοινωνία client με τη βάση δεδομένων

Για λόγους ασφαλείας, για οποιαδήποτε αλληλεπίδραση του client με τη βάση δεδομένων, επεμβαίνει στη μέση ο server. Έτσι, μιλάει ο client με το server και αντίστροφα, και ο server με τη database και αντίστροφα.

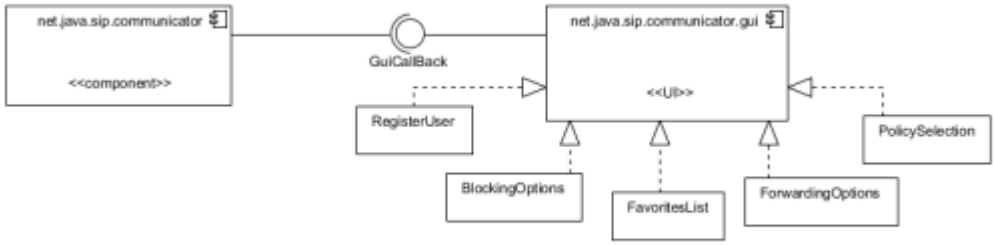
3 Αρχιτεκτονική

3.1 Ψηφιδικά Διαγράμματα

Στη σχεδίασή μας συμπεριλάβαμε δύο ψηφιδικά διαγράμματα, ένα για την πλευρά του Proxy(server side) και ένα για την πλευρά του Sip Communicator (client side). Πρώτα παρουσιάζουμε πρώτα το διάγραμμα για τον Proxy:



Σε αυτό το διάγραμμα επισημαίνουμε ότι χρησιμοποιούμε την αρχιτεκτονική pipe and filter. Συγκεκριμένα, ο Proxy προωθεί δεδομένα (push) στους BlockingServer, ForwardingServer και BillingFunctions και αυτοί κάνουν κάποιου είδους επεξεργασία και επιστρέφουν κατάλληλα μηνύματα στον Proxy. Στη συνέχεια παρουσιάζουμε το διάγραμμα από την πλευρά του Sip Communicator:

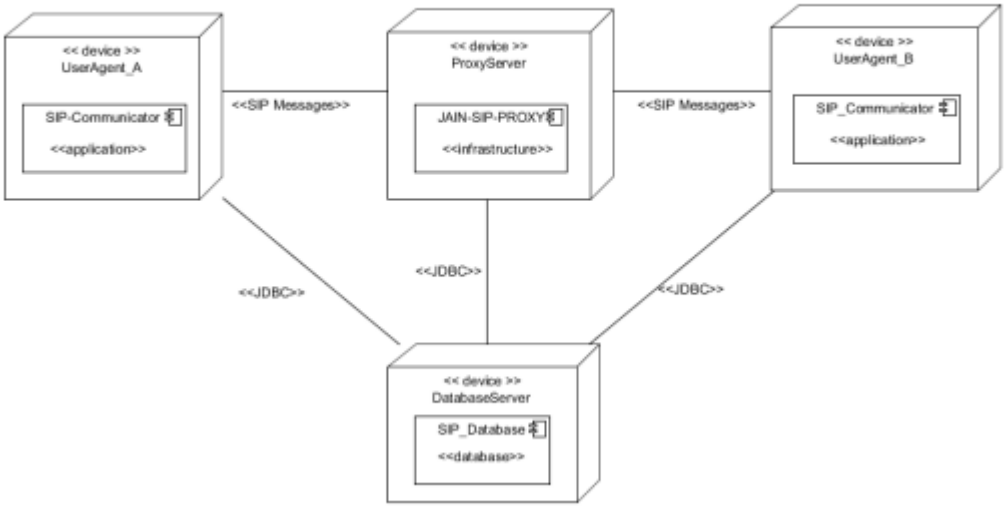


Ένα ακόμη αρχιτεκτονικό στυλ που χρησιμοποιούμε είναι αυτό του Information Hiding. Ο Sip Communicator εκτελεί απλώς ένα invite και στη

συνέχεια ο Proxy εκτελεί αρκετές λειτουργίες (έλεγχο για φραγή κλήσης, προώθηση κλήσης) και κατόπιν συνεχίζει με την εξυπηρέτηση της αίτησης του χρήστη. Αυτό γίνεται για να είναι απλός ο κώδικας πελάτη και όλη η λειτουργία μεταφέρεται στην πλευρά του server.

3.2 Παραταξιακά Διαγράμματα

Τα παραταξιακά διαγράμματα αναφέρονται στη γενική σχεδιαστική δομή του συστήματος. Παρουσιάζουν και μοντελοποιούν τη σχετική διάταξη και παράταξη των υπομονάδων κατά την εκτέλεση και λειτουργία του συστήματος. Στη δική μας περίπτωση κατασκευάσαμε το παρακάτω διάγραμμα:

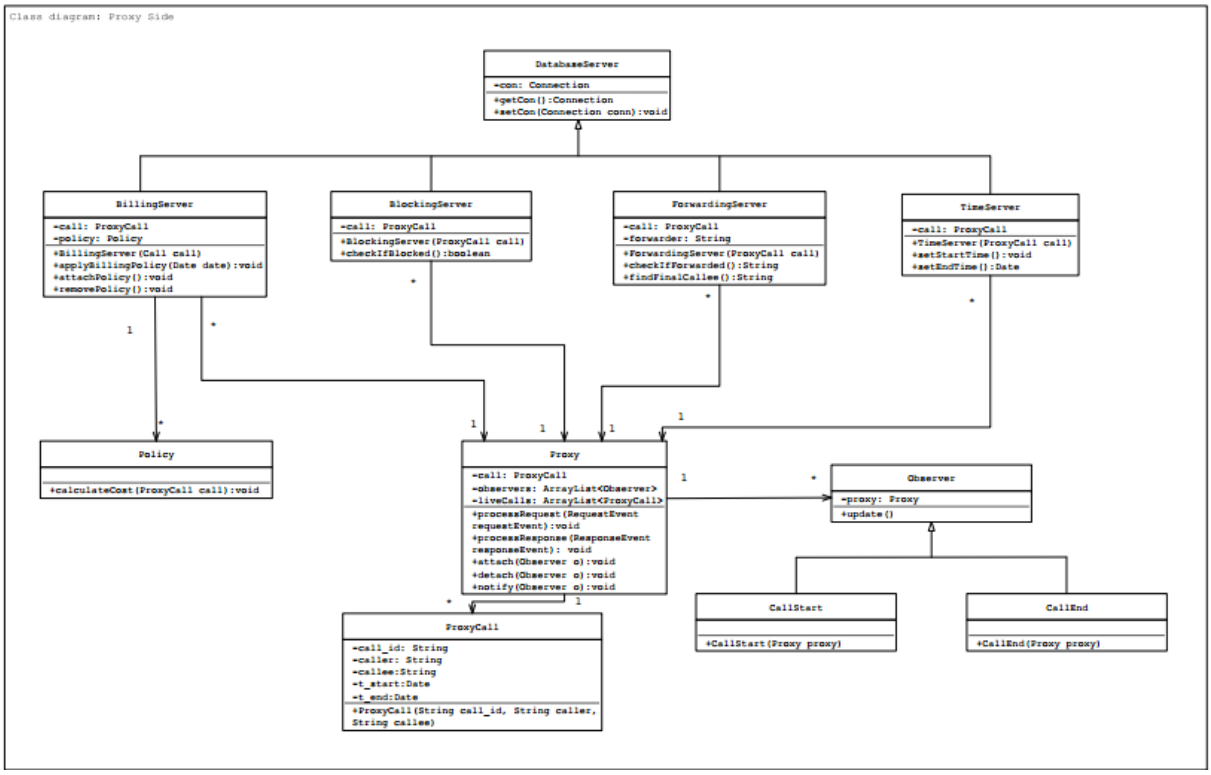


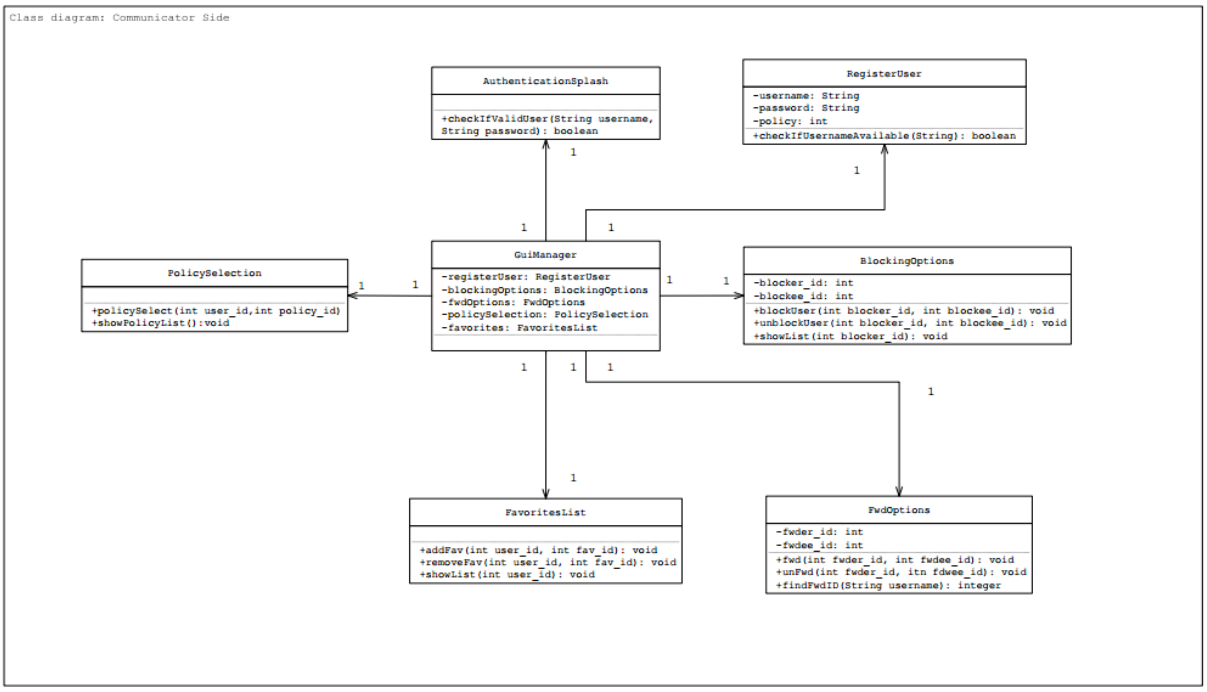
Από το παραπάνω διάγραμμα είναι φανερό ότι χρησιμοποιούμε το μοντέλο clientserver. Οι δύο χρήστες Α και Β επικοινωνούν μέσω του Proxy αρχικά μέχρι να γίνει η εγκατάσταση της σύνδεσης μεταξύ τους, ενώ από τη στιγμή που γίνει η σύνδεση το μοντέλο μετατρέπεται ουσιαστικά σε peer-to-peer.

4 Λεπτομερή διαγράμματα κλάσεων

4.1 Διαγράμματα κλάσεων σε UML

Παρακάτω Παρουσιάζουμε τα λεπτομερή διαγράμματα των κλάσεων που κατασκευάσαμε/τροποποιήσαμε για να υλοποιήσουμε τις ζητούμενες λειτουργίες:





4.2 Λεπτομέρειες των μεθόδων

4.2.1 SipProxy

4.2.1.1 ProxyCall

Τα αντικείμενα αυτής της κλήσης θα έχουν τις απαραίτητες πληροφορίες για την κοστολόγηση της κλήσης.

public ProxyCall(int caller_id, int callee_id, Date t_start, Date t_end)

Αρχικοποίησε το call_id //το id της κλήσης

Αρχικοποίησε το caller //ο καλών

Αρχικοποίησε το callee //ο καλούμενος

Αρχικοποίησε το t_start //χρονική στιγμή έναρξης κλήσης

Αρχικοποίησε το t_end //χρονική στιγμή λήξης κλήσης

4.2.1.2 Database Server

Υπερκλάση που πραγματοποιεί σύνδεση με τη βάση δεδομένων. Δημιουργεί ένα καινούριο αντικείμενο τύπου Connection ώστε να είναι δυνατή η σύνδεση με τη βάση.

4.2.1.3 BlockingServer

Επεκτείνει την DatabaseServer.

public BlockingServer(ProxyCall call)

Αρχικοποίησε το call

public boolean checkIfBlocked()

Έλεγξε στη βάση δεδομένων αν υπάρχει εγγραφή για την οποία ο callee μπλοκάρει τον caller.

Αν υπάρχει επέστρεψε true

Αλλιώς επέστρεψε false

4.2.1.4 ForwardingServer

Επεκτείνει την DatabaseServer.

public ForwardingServer(ProxyCall call)

Αρχικοποίησε το call

Αρχικοποίησε το forwarder με την τιμή του callee

public int checkIfForwarded()

Βρες στη βάση δεδομένων σε ποιον προωθεί ο callee

Αν επιστραφεί null επέστρεψε “”

Αλλιώς Επίστρεψε το username αυτού στον οποίο προωθείται η κλήση

public void findFinalCallee()

Φτιάξε ένα καινούριο HashSet fList

toUser = checkIfForwarded()

fList.add(forwarder)

forwarder = toUser

while(toUser != “”)

Θέσε το πεδίο callee του αντικειμένου call ίσο με την τιμή επιστροφής της checkIfForwarded

Δημιούργησε νέο BlockingServer με παράμετρο call

Αν checkIfBlocked() επέστρεψε BUSY

Αν περιέχεται στην fList ο toUser τότε ανιχνεύτηκε κύκλος και επομένως επίστρεψε BUSY

4.2.1.5 TimeServer

Επεκτείνει την DatabaseServer.

public TimeServer(ProxyCall call)

Αρχικοποίησε το call

public void setStartTime()

Εισήγαγε στη βάση δεδομένων νέα εγγραφή στον πίνακα κλήσεων με την τρέχουσα ώρα και ημερομηνία στα αντίστοιχα πεδία

public Date setEndTime()

Επίστρεψε την τρέχουσα ημερομηνία και ώρα ως τέλος της τρέχουσας κλήσης

4.2.1.6 Policy

Υλοποιεί το σχεδιαστικό μοτίβο συμπεριφοράς Strategy ώστε να υπολογίζεται το κόστος της κλήσης ανάλογα με την εκάστοτε πολιτική. Εξειδικεύεται σε υποκλάσεις, μία για κάθε πολιτική χρέωσης.

public void calculateCost(ProxyCall call)

4.2.1.7 Billing Server

Επεκτείνει την DatabaseServer.

public BillingServer(ProxyCall call)

Αρχικοποίησε το call

Βρες από τη βάση την πολιτική χρέωσης που έχει επιλέξει ο καλών, δημιούργησε ένα νέο αντικείμενο της αντίστοιχης κλάσης και κάνε attach αυτό το αντικείμενο στο πεδίο policy.

public void applyBillingPolicy(Date date)

Βρες το τρέχον ποσό για τον caller στη βάση δεδομένων

Κάλεσε την calculateCost για το policy

Ενημέρωσε τη βάση με τα στοιχεία ώρας τερματισμού και κόστους της κλήσης

public void attachPolicy(Policy policy)

Έλεγξε αν το πεδίο BillingServer.policy έχει αρχικοποιηθεί

Αν BillingServer .policy είναι null

Θέσε BillingServer.policy = policy

αλλιώς

Εκτέλεσε removePolicy()

Θέσε BillingServer.policy = policy public void removePolicy()

Θέσε BillingServer.policy = null

4.2.1.8 Observer

Η κλάση αυτή υλοποιεί το σχεδιαστικό μοτίβο συμπεριφοράς Observer. Προσάπτεται στην κλάση Proxy και χρησιμοποιείται ώστε να ενημερώνεται κατάλληλα η βάση δεδομένων όταν ληφθεί μήνυμα επιβεβαίωσης/αποχώρησης.

4.2.1.9 CallStart

Επεκτείνει τον Observer.

public void update()

Φτιάξε νέο αντικείμενο TimeServer

Κάλεσε την setStartTime

4.2.1.10 CallEnd

Επεκτείνει τον Observer.

public void update()

Φτιάξε νέο αντικείμενο TimeServer

Κάλεσε την setEndTime()

Φτιάξε νέο αντικείμενο BillingServer

Κάλεσε την applyBillingPolicy(date)

4.2.1.11 Proxy

Η κλάση αυτή και οι δύο μέθοδοι τροποποιήθηκαν ώστε να καλούνται οι νέες κλάσεις που κατασκευάστηκαν για την πραγματοποίηση των πρόσθετων λειτουργικοτήτων. Διατηρεί επιπλέον λίστα με δύο observers ώστε ανάλογα με το μήνυμα επιβεβαίωσης ή αποχώρησης που θα ληφθεί να καλείται η αντίστοιχη update του observer. Επίσης, διατηρεί λίστα liveCalls με τις κλήσεις σε εξέλιξη ώστε να μπορεί να βρει τα στοιχεία της κάθε κλήσης

public void processRequest(RequestEvent requestEvent)

Δημιούργησε νέο αντικείμενο call τύπου ProxyCall

Δημιούργησε νέο BlockingServer με παράμετρο call

Αν checkIfBlocked επίστρεψε BUSY

αλλιώς

Δημιούργησε νέο αντικείμενο Forwarding με παράμετρο call

Κάλεσε την findFinalCallee

Τροποποίησε το ToHeader του request ώστε το αίτημα να προωθηθεί στον κατάλληλο callee

public void processResponse(ResponseEvent responseEvent)

Βρες από τα headers του response τον caller και τον callee

Αν το response είναι τύπου INVITE και ο κωδικός απάντησης είναι 200 OK τότε

Φτιάξε ένα αντικείμενο τύπου ProxyCall

Βάλε στη λίστα liveCalls την κλήση

Κάλεσε τον observer που περιμένει έναρξη κλήσης

Αν το response είναι τύπου BYE τότε

Βρες την κλήση στη λίστα liveCalls

Κάλεσε τον observer που περιμένει τέλος κλήσης

Αφαίρεσε την κλήση από τη λίστα liveCalls

public void attach(Observer observer)

Βάλε το αντικείμενο observer στο τέλος της λίστας observers

public void detach(Observer observer)

Βγάλε το αντικείμενο observer από τη λίστα

public void notify(Observer observer)

Κάλεσε την update για το αντικείμενο observer

4.2.2 SIP Communicator

Όπως έχει ήδη αναφερθεί η επιπλέον λειτουργικότητα που απαιτείται για την επέκταση του Πρωτοκόλλου SIP στην πλευρά του User Agent περιορίζεται μόνο στη σχεδίαση κατάλληλων διεπαφών έτσι ώστε προσδιοριστούν οι Λίστες Φραγής, Προώθησης και Αγαπημένων Χρηστών και να αποθηκευτούν στη Βάση Δεδομένων. Επομένως, επεκτείναμε κατάλληλα το net.java.sip.communicator.gui , τροποποι- ώντας την κλάση GuiManager.java και προσθέτοντας τις απαραίτητες κλάσεις.

4.2.2.1 GuiManager

Προστέθηκαν τα κατάλληλα buttons/tabs για την Είσοδο , Εγγραφή , Προσδιορισμό χρηστών προς φραγή εισερχόμενων κλήσεων, προώθηση και Επιλογή Πακέτου Χρέωσης.

4.2.2.2 AuthenticationSplash

Εμπλουτίζουμε τη κλάση με τη μέθοδο CheckIfValidUser και αντίστοιχο «Login» button.

public boolean CheckIfValidUser(String username,String password)

Εκτελείται όταν ο χρήστης εισάγει τα στοιχεία (username,password) και πατήσει το «Login» button.

Έλεγξε αν το username και password που εισήγαγε ο χρήστης βρίσκονται στη Βάση Δεδομένων

Αν username και password είναι έγκυρα επίστρεψε true
αλλιώς επίστρεψε false

4.2.2.3 RegisterUser

Η κλάση RegisterUser εμφανίζει ένα παράθυρο JFrame για την πρώτη εγγραφή του χρήστη στην υπηρεσία. Μόλις ο χρήστης συμπληρώσει τα στοιχεία του και πατήσει «Submit» εκτελείται το actionperformed όπου συμβαίνουν τα εξής :

Αν checkIfUsernameAvailable(username) καταχώρησε τα στοιχεία χρήστη στον πίνακα Users της ΒΔ

αλλιώς εμφάνισε το μήνυμα «Το username χρησιμοποιείται ήδη από άλλο χρήστη»

public boolean checkIfUsernameAvailable (String username)

Έλεγξε αν το username που επέλεξε ο χρήστης υπάρχει στη ΒΔ

Αν το username υπάρχει επίστρεψε false

αλλιώς επίστρεψε true

4.2.2.4 BlockingActions

Η κλάση BlockingActions εμφανίζει ένα παράθυρο JFrame για τον προσδιορισμό των χρηστών για φραγή εισερχόμενων κλήσεων (Block/Unblock).

public void showList(int blocker_id)

Εμφανίζει την λίστα με τους υπάρχοντες χρήστες της υπηρεσίας και το αντίστοιχο button για τη διαχείρηση της φραγής εισερχομένων κλήσεων από τον εκάστοτε χρήστη. Δηλαδή :

Για κάθε χρήστη:

Αν βρίσκεται στην blockedList του συγκεκριμένου χρήστη

εμφάνισε «unblock» button

αλλιώς

εμφάνισε «block» button

Με το πάτημα του αντίστοιχου button εκτελείται το κατάλληλο actionperfomed και πραγματοποιείται η αντίστοιχη μέθοδος :

public void blockUser(int blocker_id, int blockee_id)

Εκτελείται με το πάτημα του «block» button και περιλαμβάνει τα εξής βήματα :

Αποθήκευσε την επιλογή του χρήστη εκτελώντας το κατάλληλο sql insert statement στον πίνακα BlockingList

Εμφάνισε κατάλληλο μήνυμα

public void unblockUser(int blocker_id, int blockee_id)

Εκτελείται με το πάτημα του «unblock» button και περιλαμβάνει τα εξής βήματα :

Αποθήκευσε την επιλογή του χρήστη εκτελώντας το κατάλληλο sql delete statement στον πίνακα blockingList

Εμφάνισε κατάλληλο μήνυμα

4.2.2.5 ForwardingActions

Η κλάση ForwardingActions εμφανίζει ένα παράθυρο JFrame για τον προσδιορισμό της προώθησης κλήσεων προς ένα άλλο χρήστη της υπηρεσία (Forward/UnForward)

Αν ο χρήστης έχει ενεργοποιημένη την προώθηση κλήσεων (υπάρχει εγγραφή στη Βάση Δεδομένων)

εμφάνισε «unforward» button

αλλιώς

εμφάνισε μήνυμα «Εισάγετε username χρήστη για προώθηση κλήσεων» text field για συμπλήρωση του username και button «forward»

Με το πάτημα του αντίστοιχου button εκτελείται το κατάλληλο actionperfomed και πραγματοποιείται η αντίστοιχη μέθοδος :

public void forwardUser(int fwder_id, int fwdee_id)

Εκτελείται με το πάτημα του «block» button και περιλαμβάνει τα εξής βήματα:

Μάθε τα ids ttων εμπλεκόμενων χρηστών (forwarder, forwardee) από αντίστοιχο session χρησιμοποιώντας την findForwardeeID

Αποθήκευσε την επιλογή του χρήστη εκτελώντας το κατάλληλο sql insert statement στο πίνακα ForwardingList

Εμφάνισε κατάλληλο μήνυμα

public void unforwardUser(int fwder_id, int fwdee_id)

Εκτελείται με το πάτημα του «unblock» button και περιλαμβάνει τα εξής βήματα:

Αποθήκευσε την επιλογή του χρήστη εκτελώντας το κατάλληλο sql delete statement

Εμφάνισε κατάλληλο μήνυμα

public int findForwardeeID(String username)

Επιστρέφει το id του forwardee με βάση το username.

Αν το username δεν υπάρχει στη Βάση Δεδομένων επίστρεψε -1

αλλιώς επίστρεψε user_id

4.2.2.6 PolicySelection

Η κλάση PolicySelection εμφανίζει ένα παράθυρο JFrame για τον προσδιορισμό της Πολιτικής Χρέωσης.

public void showPolicyList()

Εμφανίζει την λίστα με τις διαθέσιμες πολιτικές χρέωσης (πακέτα) και κατάλληλα buttons «Subscribe».

Με το πάτημα του button της αντίστοιχης πολιτικής χρέωσης εκτελείται το actionperfomed και πραγματοποιείται η μέθοδος :

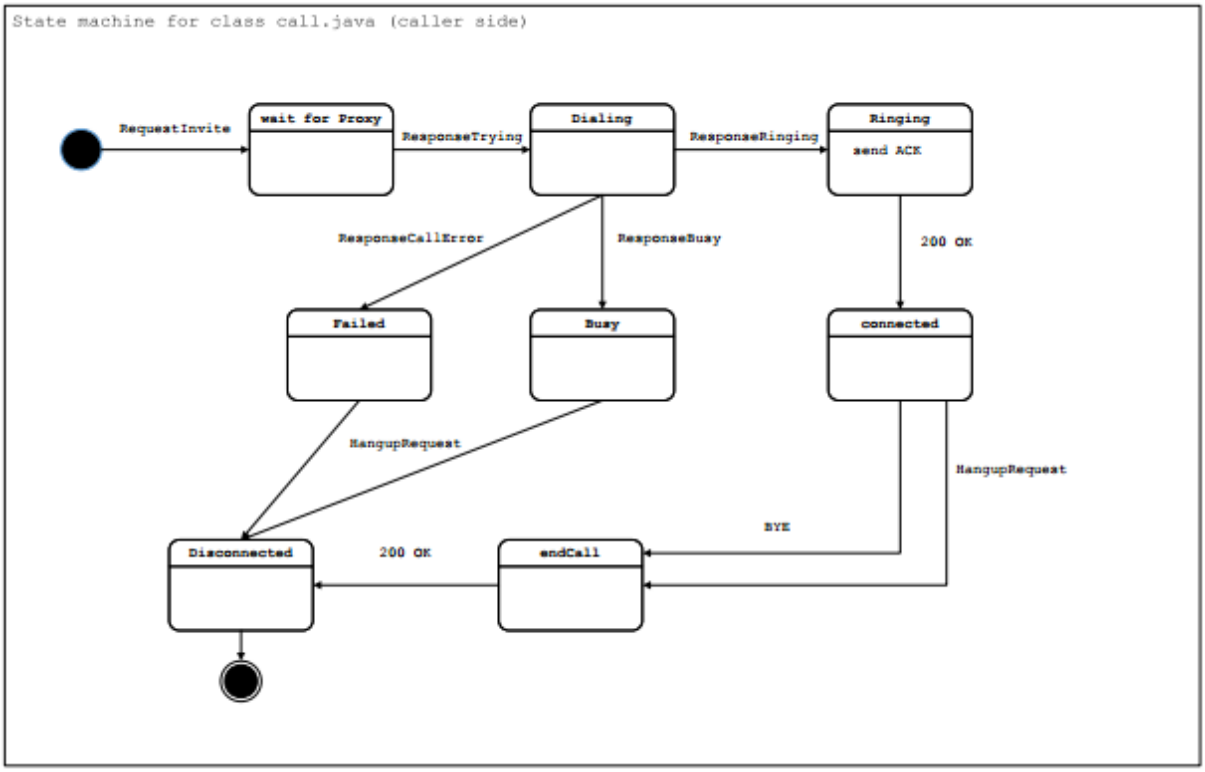
public void PolicySelection(int user_id,int policy_id)

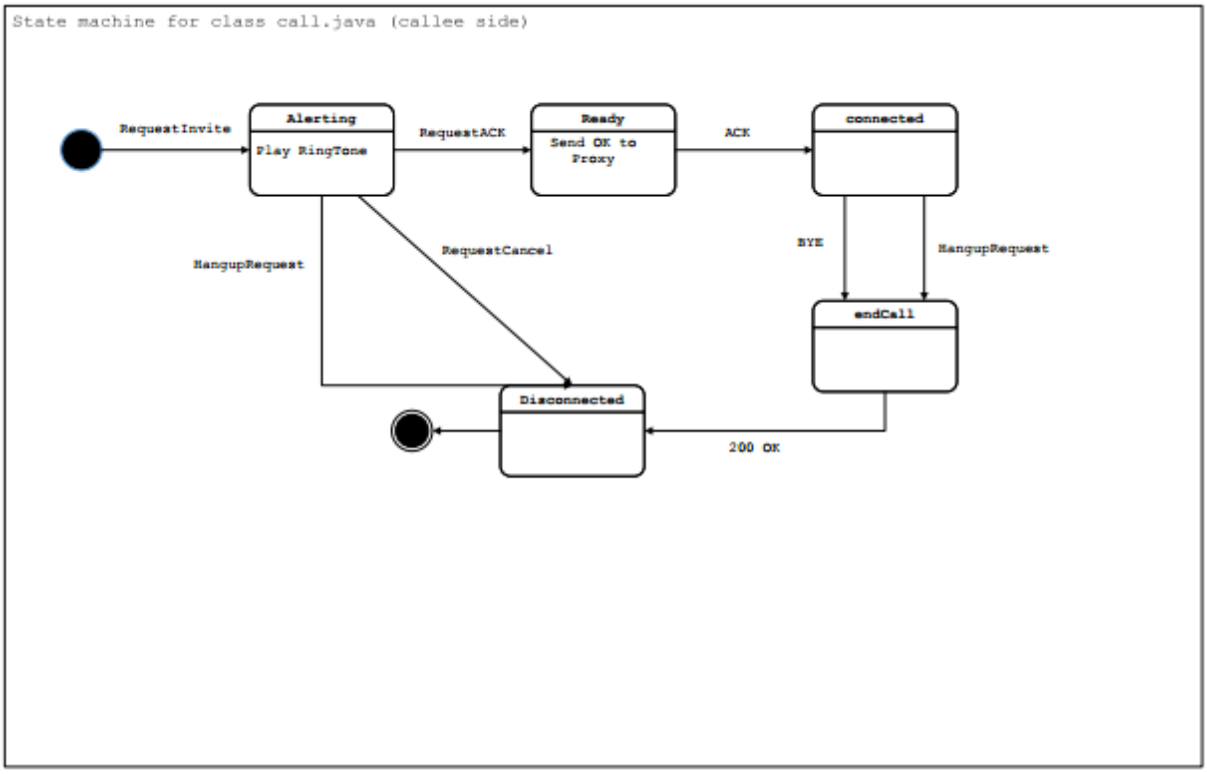
Αποθήκευσε την επιλογή του χρήστη εκτελώντας το κατάλληλο sql insert/update statement στο πίνακα Users

Εμφάνισε κατάλληλο μήνυμα

5 Διάγραμμα Κατάστασης

Παρακάτω παρατίθεται το διάγραμμα κατάστασης της κλάσης net.java.sip. communicator.sip.Call. Στο διάγραμμα αυτό παρουσιάζονται οι καταστάσεις που μπορεί να βρεθεί η κλάση αυτή καθώς και τα μηνύματα που σχετίζονται με τη μετάβαση από μια κατάσταση σε μια άλλη. Λόγω σχετικής ασυμμετρίας μεταξύ του caller και του callee δείχνουμε δύο διαγράμματα ένα για κάθε περίπτωση:





6 Ανοιχτά Ζητήματα

6.1 Ανεξέλεγκτη χρέωση καλούντος

Όταν ο Proxy δεν αντιληφθεί με σωστό τρόπο το τέλος της συνδιάλεξης (π.χ. Client crashes), πρέπει να γίνεται κάποιος έλεγχος, ώστε ο caller να μην χρεώνεται παραπάνω από όσο πρέπει.

6.2 Ανίχνευση κύκλων στην προώθηση κλήσης

Δεν προβλέπεται ακόμα υλοποίηση του τρόπου ελέγχου ύπαρξης κύκλων στη λειτουργία Forwarding και ο χειρισμός τέτοιων προβληματικών καταστάσεων όπως να επανερχόμαστε στον αρχικό χρήστη προς τον οποίο έγινε η κλήση με διαδοχικά Forwarding.

6.3 Σχεσιακό Μοντέλο Βάσης

Δεν έχει καθοριστεί πλήρως ακόμα (πεδία πινάκων, primary keys, indexes κλπ).

6.4 Πολιτικές Χρέωσης

Εκκρεμεί η ακριβής διατύπωση των προγραμμάτων χρέωσης που θα συμπεριλάβουμε και θα διαθέτουμε στους χρήστες για να επιλέξουν.