ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Διαχείριση Αδειών Λογισμικού σε Κατανεμημένα Συστήματα

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

ΕΛΕΝΗΣ Β. ΔΗΜΗΤΡΟΠΟΥΛΟΥ

Επιβλέπουσα : Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Αθήνα, Μάρτιος 2008

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Διαχείριση Αδειών Λογισμικού σε Κατανεμημένα Συστήματα

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

ΕΛΕΝΗΣ Β. ΔΗΜΗΤΡΟΠΟΥΛΟΥ

Επιβλέπουσα : Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 11η Μαρτίου 2008.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Θεοδώρα Βαρβαρίγου	Εμμανουήλ Πρωτονοτάριος	Βασίλειος Λούμος
Καθηγήτρια Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2008

(Υπογραφή)

.....

ΕΛΕΝΗ Β. ΔΗΜΗΤΡΟΠΟΥΛΟΥ

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

© 2008 – Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου.

Περίληψη

Σκοπός της διπλωματικής εργασίας που παρουσιάζεται στο παρόν έγγραφο ήταν η ανάπτυξη ενός αξιόπιστου, ασφαλούς και ευέλικτου συστήματος παροχής άδειας, βασισμένου στο εξυπηρετούμενο μοντέλο παροχής άδειας και ικανού να εφαρμοστεί σε κατανεμημένα συστήματα.

Η έρευνα στο πεδίο της παροχής άδειας χρήσης λογισμικού, είναι πολύ διαδεδομένη στις μέρες μας, δεδομένου ότι οι εταιρίες πώλησης λογισμικού επενδύουν στο πεδίο αυτό με σκοπό να εξασφαλίσουν την κυκλοφορία μόνο γνήσιων αντιγράφων και να επιτύχουν μέγιστη ασφάλεια στους μηχανισμούς αδειδότησης.

Ο μηχανισμός που αναπτύχθηκε, βασίζεται στη χρήση μιας διαδικτυακής υπηρεσίας και ενός δαίμονα – εξυπηρετητή, στοιχεία τα οποία παρεμβαίνουν επιτυγχάνοντας την παροχή άδειας λογισμικού, με τρόπο ευέλικτο, αξιόπιστο και ασφαλές. Η αίτηση και η χορήγηση της άδειας, πραγματοποιούνται διαδικτυακά, με κλήση εξ' αποστάσεως της διαδικτυακής υπηρεσίας.

Η υλοποίηση του συστήματος, πραγματοποιήθηκε με χρήση τεχνολογιών αιχμής και σύγχρονων και διαδεδομένων πλατφορμών. Συγκεκριμένα έγινε χρήση του Apache Tomcat Server, του Apache HTTP Server, της Java ως γλώσσας προγραμματισμού, ενώ έγινε ανάπτυξη JSP και PHP δυναμικών σελίδων για τις γραφικές διεπαφές των διαχειριστών.

Λέξεις Κλειδιά: <<Παροχή Άδειας, FLEXlm Server, Διαδικτυακή Υπηρεσία, Axis2, License Daemon, License Service, Grid, Java, JSP, PHP, Apache Tomcat server, Apache HTTP Server>>

Abstract

The aim of the thesis presented in this document was the development of a reliable, safe and flexible license management system, based on the served licensing model, and capable of being applied to distributed systems.

The research in the field of software licensing, is in high development nowadays, taking as granted that the software vendors, invest a high budget in this field having the goal to assure the distribution of exclusively original copies and to achieve the maximum safety of the licensing mechanisms.

The mechanism developed, is based on the use of a web service and a daemon – server, components which intervene, achieving software licensing, via a flexible, reliable and safe procedure. The request and the granting are being held via internet, with a remote call of a web service.

The system implementation has been developed with the use of modern technology and platforms. More precisely, the choice of Apache Tomcat Server, Apache HTTP Server and Java as a programming language, has been made, while the development of the dynamic pages for the graphic interfaces of administration has been made with the JSP and PHP technologies.

Keywords: <<Licensing, FLEXlm Server, Web Service, Axis2, License Daemon, License Service, Grid, Java, JSP, PHP, Apache Tomcat server, Apache HTTP Server>>

Ευχαριστίες

Θα ήθελα να ευχαριστήσω την Καθηγήτρια Βαρβαρίγου Θεοδώρα για την ανάθεση της διπλωματικής εργασίας, την παρακολούθηση και παροχή κατευθύνσεων για την υλοποίηση της.

Επίσης, δεν θα ήθελα να παραλείψω τη συμβολή του Κυριαζή Δημοσθένη διδάκτορα του Εργαστηρίου Τηλεπικοινωνιών, που με τις πολύτιμες συμβουλές του, συνέβαλε στην επιτυχή ολοκλήρωση αυτής της διπλωματικής εργασίας.

.

Θα ήθελα επιπλέον να κάνω ιδιαίτερη αναφορά στον Κατσαρό Γρηγόριο, επιστημονικό συνεργάτη του Εργαστηρίου Τηλεπικοινωνιών και πολύ καλό φίλο, που με την αμέριστη υπομονή, συμπαράσταση, την καθοδήγηση και τις συμβουλές του, συνέβαλλε στην επιτυχή ολοκλήρωση αυτού του έργου.

Επιπρόσθετα, θα ήθελα να ευχαριστήσω την πολύ καλή μου φίλη Κιλτένη Κωνσταντίνα, διπλωματούχο Ηλεκτρολόγο Μηχανικό και Μηχανικό Υπολογιστών, για την βοήθειά της, την υπέροχη συνεργασία μας, την τεράστια υπομονή της, την εμπιστοσύνη της καθώς και για την στήριξη της καθ' όλη τη διάρκεια εκπόνησης της διπλωματικής εργασίας και γενικότερα.

Θα ήθελα ακόμα να ευχαριστήσω θερμά, την πολύ καλή μου φίλη Αυγουστή Δήμητρα, για τη σημαντικότατη προσφορά της σε κάθε μου βήμα, όντας πάντα δίπλα μου, έτοιμη να μοιραστεί τις αγωνίες μου και να με βοηθήσει με όλους τους τρόπους, καθώς και τα αδέρφια μου Νίκο και Τάνια για την πολύτιμη συμπαράστασή τους με κάθε τρόπο.

Τέλος, οφείλω κυρίως να ευχαριστήσω τους γονείς μου για την στήριξη, την υπομονή, την κατανόησή τους, καθώς και για όλα όσα μου προσέφεραν αυτά τα χρόνια, μοιραζόμενοι τα όνειρα μου.

Πίνακας περιεχομένων

1 ΕΙΣΑΓΩΓΗ 14

1.1 ΑΝΤΙΚΕΙΜΕΝΟ ΤΗΣ ΔΙΠΛΩΜΑΤΙΚΗΣ 14
1.2 ΟΡΓΑΝΩΣΗ ΤΟΥ ΕΓΓΡΑΦΟΥ 14

2 ΠΕΡΙΓΡΑΦΗ ΘΕΜΑΤΟΣ 16

2.1 ΔΙΑΔΙΚΑΣΙΑ ΠΑΡΟΧΗΣ ΆΔΕΙΑΣ 16
2.1.1 Τι είναι η Διαδικασία Παροχής Άδειας Λογισμικού 16
2.1.2 Ιστορική Αναδρομή 16
2.1.3 Ο ρόλος του Διαδικτύου 19
2.2 ΔΙΑΔΙΚΤΥΑΚΕΣ ΥΠΗΡΕΣΙΕΣ 22
2.2.1 Τι είναι η Διαδικτυακή Υπηρεσία 22
2.2.2 Χαρακτηριστικά μιας διαδικτυακής υπηρεσίας 23
2.3 ΤΕΧΝΟΛΟΓΙΑ AXIS 2 24
2.3.1 Τι είναι ο axis2 24
2.3.2 Δημιουργία, τοποθέτηση και προσπέλαση διαδικτυακών υπηρεσιών 24
2.3.3 Τα γενικά χαρακτηριστικά του AXIS2 26
2.4 FLEXLM ΕΦΑΡΜΟΓΕΣ 28
2.4.1 Τι είναι ο FLEXlm 28
2.4.2 Συστατικά του FLEXlm 29
2.4.2.1 Συστατικά του σύγχρονου μοντέλου 29
2.4.2.2 Συστατικά του μοντέλου κλειδωμένου σε κόμβο 30
2.4.3 Περιγραφή των Συστατικών του Σύγχρονου Μοντέλου 30
2.4.4 Εργαλεία διαχείρισης του FLEXlm 34
2.4.5 Συνοπτική Περιγραφή της διαδικασίας απόκτησης άδειας στο Σύγχρονο Μοντέλο 35
2.5 ΥΠΗΡΕΣΙΟΣΤΡΑΦΗΣ ΑΡΧΙΤΕΚΤΟΝΙΚΗ –SERVICE ORIENTED ARCHITECTURE 36
2.5.1 Τι είναι SOA? 36
2.5.2 Γιατί SOA; 39
2.5.3 Τα πλεονεκτήματα της SOA; 40
2.6 ΠΛΕΓΜΑ-GRID 40
2.6.1 Τι είναι? 40
2.6.2 Βασικά Χαρακτηριστικά και Πλεονεκτήματα του Grid 42
2.6.2.1 Βασικά Χαρακτηριστικά 42
2.6.2.2 Πλεονεκτήματα 43
2.6.3 Grid Υπηρεσία και Σύγκριση με την Διαδικτυακή Υπηρεσία 45
2.6.3.1 Τι είναι GRID υπηρεσία ? 45
2.6.4 Παροχή Άδειας στο Grid 46

3 ΑΝΑΛΥΣΗ ΚΑΙ ΣΧΕΔΙΑΣΗ 47

3.1 ΠΕΡΙΓΡΑΦΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ 47
3.1.1. Σκοπός της νέας αρχιτεκτονικής 47
3.1.2. Η υπάρχουσα αρχιτεκτονική ως βάση της νέας 48
3.1.2.1 Περιγραφή των οντοτήτων και των στοιχείων της νέας αρχιτεκτονικής 49
3.2 ΠΕΡΙΓΡΑΦΗ ΛΕΙΤΟΥΡΓΙΩΝ 50
3.2.1 Λειτουργία υπάρχουσας αρχιτεκτονικής 50
3.2.2 Λειτουργία νέας αρχιτεκτονικής 52
3.2.3 Λειτουργικότητα νέας αρχιτεκτονικής 54

4 ΥΛΟΠΟΙΗΣΗ 55

4.1 ΠΛΑΤΦΟΡΜΕΣ ΚΑΙ ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΑ ΕΡΓΑΛΕΙΑ 55

4.1.1 JAVA 55

4.1.1.1 Πρωτεύοντες στόχοι 56

4.1.1.2 Γιατί Java ? 56

4.1.2 Γραφικό Περιβάλλον (User Interface) 57

4.1.2.1 Java Server Pages (JSP) 57

4.1.2.2 PHP 57

4.1.3 Δικτυακός Εξυπηρετητής - Web Server 58

4.1.3.1 Tomcat 58

4.1.3.2 AXIS 58

4.1.3.3 Apache HTTP Server 61

4.2 ΛΕΠΤΟΜΕΡΕΙΕΣ ΥΛΟΠΟΙΗΣΗΣ 61

4.2.1 Υλοποίηση συστήματος – Συγγραφή κώδικα 62

4.2.1.1 License Daemon 63

4.2.1.2 License Service 65

5 ΈΛΕΓΧΟΣ 67

5.1 ΟΔΗΓΟΣ ΕΓΚΑΤΑΣΤΑΣΗΣ 67

5.1.1 Εγκατάσταση Περιβάλλοντος 67

5.1.1.1 Εγκατάσταση Περιβάλλοντος Εκτέλεσης Java (JRE) 67

5.1.1.2 Εγκατάσταση του Tomcat 67

5.1.1.3 Εγκατάσταση του Axis2 68

5.1.1.4 Εγκατάσταση του Apache HTTP Server 68

5.1.1.5 Εγκατάσταση της PHP 68

5.1.2 Εγκατάσταση νέου συστήματος 69

5.1.2.1 Εγκατάσταση License Daemon και εγκατάσταση των JSP σελίδων 69

5.1.2.2 Εγκατάσταση License Service και εγκατάσταση των PHP σελίδων 70

5.2 ΑΝΑΛΥΤΙΚΗ ΠΑΡΟΥΣΙΑΣΗ ΕΛΕΓΧΟΥ 71

5.2.1 Εκτέλεση License Daemon 71

5.2.1.1 Μέσω γραμμής εντολών 71

5.2.1.2 Μέσω παραθυριακής γραφικής διεπαφής 72

5.2.1.3 Μέσω JSP σελίδων 77

5.2.2 Εκτέλεση License Service 80

5.2.3 Συνολική Εκτέλεση 90

5.2.4 Μετρήσεις, Γραφήματα και Συμπεράσματα 95

6 ΕΠΙΛΟΓΟΣ 98

6.1 ΣΥΝΟΨΗ ΚΑΙ ΣΥΜΠΕΡΑΣΜΑΤΑ 98

6.1.1 Σύνοψη 98

6.1.2 Συμπεράσματα 98

6.2 ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ 99

6.2.1 Δυνατές Επεκτάσεις-Εκκρεμότητες 99

7 ΒΙΒΛΙΟΓΡΑΦΙΑ 101

ΠΑΡΑΡΤΗΜΑ 102

LICENSE DAEMON 102

CreateLog.java 102

LicenseDaemon.java 104

LicenseDaemonAux.java 107

LicenseDaemonHandler.java 109

LicenseHandler2.java 112

LicenseServiceClient.java 114

NetworkInfo.java 117

Simple.java *118*
Simple2.java *119*
UserInterface.java *119*
LICENSE SERVICE 122
LicenseService.java *122*
JSP 125
index.jsp *125*
media.jsp *130*
medium.jsp *131*
ServerSettings.jsp *133*
WrongPasswords.jsp *139*
PHP 141
index.php *141*
LMUtilities.php *143*
result.php *144*
SetServers.php *145*

1 Εισαγωγή

Στο παρόν κεφάλαιο γίνεται αρχικά προσέγγιση του αντικειμένου της διπλωματικής εργασίας και παρουσιάζεται η πορεία που ακολουθήθηκε για την εκπόνησή της και την ολοκλήρωση του συστήματος λογισμικού που αναπτύχθηκε στα πλαίσια αυτής. Τέλος γίνεται ανάλυση της οργάνωσης του παρόντος τόμου της διπλωματικής εργασίας σε κεφάλαια και περιγραφή του περιεχομένου των κεφαλαίων.

1.1 Αντικείμενο της διπλωματικής

Η ανάγκη για των εταιριών παροχής λογισμικού να εξασφαλίσουν τη διανομή των προϊόντων τους μόνο σε νόμιμους χρήστες καθώς και η εξέλιξη της διαδικασίας παροχής άδειας, η οποία πλέον στις περισσότερες των περιπτώσεων πραγματοποιείται με την επικοινωνία με κάποιον εξυπηρετητή παροχής άδειας, οδήγησε στην αύξηση του επιστημονικού ενδιαφέροντος στον τομέα αυτό, ώστε να δημιουργηθούν συστήματα παροχής άδειας, όσο το δυνατόν ασφαλέστερα. Σε αυτό το πλαίσιο η διπλωματική εργασία που παρουσιάζεται στο παρόν έγγραφο, έχει ως σκοπό την υλοποίηση ενός μηχανισμού παροχής άδειας μέσω εξυπηρετητή ο οποίος θα παρέχει αξιοπιστία στην επικοινωνία, ευελιξία στη διαχείριση, μεγαλύτερη ασφάλεια στον πάροχο άδειας από την ήδη υπάρχουσα. Ο μηχανισμός αυτός θα αναπτυχθεί σύμφωνα με τις προδιαγραφές του FLEXlm Server και των FLEXlm εφαρμογών με χρήση διαδικτυακών υπηρεσιών ώστε η διαδικασία να πραγματοποιείται διαδικτυακά και ώστε να δημιουργηθεί μια βάση για τη δημιουργία μιας υπηρεσίας παροχής άδειας στο υπηρεσιοστραφές περιβάλλον Grid. Ο μηχανισμός αυτός θα μπορεί να εφαρμοστεί σε κατανεμημένα συστήματα και θα παρεμβαίνει στην αρχιτεκτονική τόσο από την πλευρά του τελικού χρήστη, όσο και από την πλευρά του εξυπηρετητή. Επίσης, θα παρέχει δυνατότητες διαχείρισης, ρύθμισης των παραμέτρων λειτουργίας και παρακολούθησης της χρήσης, από τους τελικούς χρήστες, του συστήματος. Τέλος, το σύστημα θα ελεγχθεί ώστε να επιβεβαιωθεί η εγκυρότητα του, τα πλεονεκτήματα που προσφέρει χωρίς αισθητές απώλειες σε απόδοση καθώς και οι πολλαπλές δυνατότητες του.

Η ανάπτυξη αυτής της υπηρεσίας θα ακολουθήσει την παρακάτω διαδικασία:
- Εξοικείωση με τις σχετικές τεχνολογίες
- Σχεδιασμός του μηχανισμού
- Συγγραφή κώδικα υπηρεσίας
- Συγγραφή κώδικα διασυνδέσεων χρήστη (User Interfaces)
- Εξαγωγή συμπερασμάτων και συλλογή αποτελεσμάτων

1.2 Οργάνωση του εγγράφου

Το παρόν έγγραφο αποτελείται από εφτά κεφάλαια και ένα παράρτημα. Στα κεφάλαια αυτά παρουσιάζεται αναλυτικά και σε βάθος ο σχεδιασμός, η υλοποίηση, η λειτουργία και ο έλεγχος της εφαρμογής.

Το κεφάλαιο 2 έχει τίτλο «Περιγραφή Θέματος» και τοποθετεί το θέμα ως προς την επιστημονική περιοχή στην οποία ανήκει.
Το κεφάλαιο 3 έχει τίτλο «Ανάλυση και Σχεδίαση» και αφορά το τμήμα της σχεδίασης του συστήματος λογισμικού. Σε αυτό το κεφάλαιο δίνεται ο στόχος της εργασίας, το πρόβλημα που αυτή καλείται να αντιμετωπίσει και γίνεται προσπάθεια να περιγραφεί η αρχιτεκτονική και η λειτουργία του συστήματος με την παρουσίαση διαγραμμάτων της δομής του και περιπτώσεων χρήσης του.
Το κεφάλαιο 4 με τίτλο «Υλοποίηση», αναφέρεται στα εργαλεία που χρησιμοποιήθηκαν για την υλοποίηση του μηχανισμού καθώς και επεξηγηματικές πληροφορίες σχετικά με αυτή.
Το κεφάλαιο 5 έχει τίτλο «Έλεγχος» και περιλαμβάνει τον Οδηγό Εγκατάστασης του συστήματος και τα αποτελέσματα χρήσης του.
Τέλος, το κεφάλαιο 6 έχει τίτλο «Επίλογος» και περιλαμβάνει τη σύνοψη της διπλωματικής εργασίας, τα συμπεράσματα που εξήχθησαν με την ολοκλήρωσή της καθώς και μελλοντικές επεκτάσεις του μηχανισμού που υλοποιήθηκε.
Το κεφάλαιο 7 περιλαμβάνει τη Βιβλιογραφία ενώ το Παράρτημα αναλυτικά τον κώδικα που υλοποιήθηκε.

2 Περιγραφή Θέματος

Στο κεφάλαιο αυτό παρουσιάζεται η έννοια και η ιστορική ανάδρομη της παροχής άδειας και παρουσιάζονται οι σχετικές με το θέμα και το σύστημα τεχνολογίες.

2.1 Διαδικασία Παροχής Άδειας

2.1.1 Τι είναι η Διαδικασία Παροχής Άδειας Λογισμικού

Πρόκειται για μία συμφωνία μεταξύ του παρόχου του λογισμικού και του τελικού χρήστη του λογισμικού. Η άδεια χρήσης επιτρέπει στον τελικό χρήστη να χρησιμοποιήσει ένα ή περισσότερα αντίγραφα του λογισμικού σε περιπτώσεις όπου, η χρήση χωρίς άδεια επιφέρει ποινικές ευθύνες.

Με τον όρο «χορήγηση άδειας χρήσης λογισμικού» εννοούμε ότι ο πνευματικός ιδιοκτήτης επιτρέπει σε χρήστες να πραγματοποιήσουν ορισμένες ενέργειες, που εκείνος έχει καθορίσει, πάνω στο προϊόν. Τέτοιες ενέργειες περιλαμβάνουν την εκτέλεση, την αντιγραφή, την τροποποίηση και την διανομή του λογισμικού.
Πλέον, σχεδόν όλες οι εφαρμογές λειτουργούν συχνότερα με άδειες χρήσης παρά με τη αγοραπωλησία. Υπάρχει μεγάλη ποικιλία στους διαφορετικούς τύπους αδειών χρήσης λογισμικού. Κάποιοι βασίζονται στον αριθμό των μηχανημάτων στα οποία η αδειοδοτημένη εφαρμογή θα εκτελείται ενώ άλλοι βασίζονται στον αριθμό των χρηστών που την χρησιμοποιούν. Οι περισσότερες άδειες χρήσης για προσωπικούς υπολογιστές επιτρέπουν την εκτέλεση της εκάστοτε εφαρμογής σε ένα μόνο μηχάνημα καθώς και τη αντιγραφή της μόνο για περιπτώσεις αποθήκευσης. Κάποιες άδειες ωστόσο επιτρέπουν τη εκτέλεση της εφαρμογής σε διαφορετικούς υπολογιστές υπό την προϋπόθεση ότι τα αντίγραφα δεν εκτελούνται ταυτόχρονα. [EC]

2.1.2 Ιστορική Αναδρομή

Πριν την εμφάνιση των δικτύων, στα τέλη της δεκαετίας 1960-1970, όλοι οι υπολογιστές ήταν μεμονωμένες και αυτοδύναμες μονάδες. Ο τυπικός τρόπος για αντιγραφή δεδομένων και εφαρμογών από έναν υπολογιστή σε έναν άλλο, ή για διανομή εφαρμογών σε έναν υπολογιστή, γινόταν με κάποιο μέσο φυσικής αποθήκευσης στο οποίο αποθηκεύονταν τα δεδομένα ή οι εφαρμογές. Τα μοντέλα παροχής άδειας χρήσης λογισμικού σε μεμονωμένους υπολογιστές είχαν ισχύ μόνο στις ίδιες μεμονωμένες μονάδες.

Άδεια με συστελλόμενο περιτύλιγμα (Shrink-wrapped)

Το πρώτο μοντέλο που χρησιμοποιήθηκε για να εκφράσει την διανομή αδειών λογισμικού, ονομάστηκε μοντέλο *συστελλόμενο περιτύλιγμα (*Shrink-wrapped). Σε αυτό το μοντέλο η εφαρμογή αποθηκεύεται σε ένα μέσο αποθήκευσης (δισκέτα, CD) το οποίο είναι περιτυλιγμένο από χαρτί ή καλύπτεται από πλαστικό ή βρίσκεται σε κάποιο κουτί, και παραδίδεται στο χρήστη σε πακέτο. Η συσκευασία, περιλαμβάνει εκτυπωμένους τους όρους χρήσης τους οποίους ο χρήστης αποδέχεται αυτόματα με το άνοιγμα της συσκευασίας. Για εφαρμογές με πολυάριθμους όρους χρήσης, το κείμενο με τους όρους βρισκόταν εντός της συσκευασίας.

Αυτές οι άδειες χρήσης οδήγησαν στις *ηλεκτρονικές άδειες με συστελλόμενο περιτύλιγμα.* Αυτές τις άδειες τις αποδέχονται οι χρήστες απαντώντας ένα ναι ή όχι στην ερώτηση που τους υποβάλλει η εφαρμογή στην έναρξη της, αν αποδέχονται τους όρους χρήσης.

Εφόσον οι μεμονωμένοι υπολογιστές δεν είναι συνδεδεμένοι σε τοπικό δίκτυο, ή στο διαδίκτυο, δεν υπάρχει καμία δυνατότητα για παροχή αδειών χρήσης λογισμικού εξ αποστάσεως. Το μοντέλο «άδεια με συστελλόμενο περιτύλιγμα» παρέχει την δυνατότητα στον χρήστη να χρησιμοποιήσει το λογισμικό χωρίς κανένα περιορισμό, ενώ η άδεια δεν λήγει ποτέ αφού το προϊόν αγοράστηκε. Οι κάτοχοι της άδειας μπορούν εφ' όρου ζωής να χρησιμοποιούν την εφαρμογή, ωστόσο δεν είναι κάτοχοι της, αλλά κάτοχοι της άδειας χρήσης της. Απαγορεύεται η αντιγραφή της εφαρμογής, η διανομή της και η μεταπώληση της. Παρόλα αυτά, στο μοντέλο αυτό, δεν υπάρχει κάποια μέθοδος παρεμπόδισης του χρήστη να αντιγράψει, να διανείμει ή να εγκαταστήσει την εφαρμογή σε άλλους υπολογιστές. Έτσι, ένας μεγάλος αριθμός χρηστών εκούσια ή ακούσια, δεν τηρούσαν τους όρους χρήσης.

Αυτό συνετέλεσε και ακόμα συντελεί στα υψηλά ποσοστά πειρατείας στη βιομηχανία λογισμικού και στις πολύ μεγάλες οικονομικές απώλειες των ιδιοκτητριών εταιριών. Το ηλεκτρονικό μοντέλο «άδεια με συστελλόμενο περιτύλιγμα» συνδυάζεται σήμερα με τεχνικές που αποτρέπουν τους χρήστες να αντιγράψουν το περιεχόμενο του αυθεντικού CD.

Σήμερα το μοντέλο αυτό τείνει να αντικατασταθεί από την παροχή άδειας με κλειδί προϊόντος, με συνδρομή και άλλες μεθόδους καταμέτρησης για εταιρικούς πελάτες. Το λογισμικό προς κατανάλωση, που απευθύνεται σε μεμονωμένους χρήστες, όπως για παράδειγμα τα παιχνίδια στον υπολογιστή, ακόμη χρησιμοποιεί το μοντέλο με συστελλόμενο περιτύλιγμα.

Μοντέλο παροχής άδειας με σειριακό αριθμό

Αρκετές εφαρμογές λογισμικού, κατά την διάρκεια της εγκατάστασης τους, απαιτούν την εισαγωγή ενός σειριακού αριθμού. Ο σειριακός αριθμός είναι συνήθως ένας συνδυασμός από γράμματα και αριθμούς, και εφόσον εισαχθεί σωστά, η εγκατάσταση συνεχίζεται. Αρχικά, ο σειριακός αριθμός ήταν τυπωμένος στις οδηγίες εγκατάστασης που συνόδευαν την εφαρμογή. Η εφαρμογή επιβεβαίωνε ότι είχε εισαχθεί ένας έγκυρος σειριακός αριθμός αλλά εφόσον δεν υπήρχε καμία σύνδεση προς άλλους υπολογιστές ή προς μία κεντρική βάση δεδομένων αδειών χρήσης, δεν ήταν δυνατό να εξασφαλιστεί ότι η εφαρμογή θα εκτελεστεί μόνο σε μία συσκευή ή να ανακληθεί μία άδεια σε μία παράνομα εγκατεστημένη εφαρμογή. Αυτό επέτρεψε την παράνομη αντιγραφή και διανομή λογισμικού. Η παροχή άδειας χρήσης με σειριακό αριθμό χρησιμοποιείται ακόμη, αλλά εξαιτίας της ραγδαίας εξάπλωσης του Διαδικτύου και της κεντρικής διαχείρισης μιας βάσης δεδομένων με άδειες λογισμικού, το

μοντέλο αυτό έχει την δυνατότητα να εξασφαλίσει ότι ένας σειριακός αριθμός αντιστοιχεί σε συγκεκριμένο αριθμό συσκευών.

Μοντέλο παροχής άδειας Περιορισμένου Χρόνου

Μία συγκεκριμένη χρονική περίοδος μπορεί να χρησιμοποιηθεί για να περιορίσει τις άδειες χρήσης. Μπορεί να χρησιμοποιηθεί για να ορίσει την δοκιμαστική (demo ή trial) έκδοση μιας εφαρμογής στην οποία ο χρήστης μπορεί να κάνει χρήση της εφαρμογής χωρίς κανένα κόστος σε συγκεκριμένο χρονικό διάστημα. Το διάστημα αυτό είναι συνήθως 30 ημέρες. Η χρονική περίοδος μπορεί ακόμη να ορίσει μία περίοδο χρήσης με άδεια κατόπιν αγοραπωλησίας, για παράδειγμα ένα ή δύο χρόνια. Το μοντέλο παροχής άδειας Περιορισμένου Χρόνου μπορεί να εφαρμοσθεί σε μεμονωμένους υπολογιστές εφόσον την εγκυρότητα την επιβεβαιώνει το ρολόι του συστήματος. Με τη λήξη του συγκεκριμένου χρονικού διαστήματος, γινόταν ανάκληση της άδειας. Οι υλοποιήσεις του μοντέλου αυτού, γρήγορα απέβησαν διαβλητές εφόσον ρυθμίζοντας το ρολόι του συστήματος σε ημερομηνία και ώρα που η άδεια ίσχυε, η εφαρμογή μπορούσε να χρησιμοποιηθεί ακόμη και όταν είχε λήξει η άδεια της. Σήμερα οι άδειες χρήσης περιορισμένου χρόνου, είναι σε θέση να εντοπίσουν τυχόν ρύθμιση του ρολογιού του συστήματος. Σε τέτοια περίπτωση, οι άδειες ανακαλούνται.

Μοντέλο παροχής άδειας βασισμένο στα Χαρακτηριστικά

Ένας άλλος τρόπος για παροχή άδειας χρήσης λογισμικού σε μεμονωμένους υπολογιστές, είναι οι εκδόσεις των αδειών χρήσης βασισμένων στα χαρακτηριστικά. Μία απλή εφαρμογή κυκλοφορεί σε πολλές διαφορετικές εκδόσεις, η καθεμία εκ των οποίων διαθέτει διαφορετικό επίπεδο χαρακτηριστικών. Αυτά τα σύνολα χαρακτηριστικών απευθύνονται σε χρήστες με διαφορετικές ανάγκες και απαιτήσεις σε απόδοση. Οι διάφορες εκδόσεις διαφέρουν στην τιμή και διασφαλίζουν ότι προχωρημένοι χρήστες θα πληρώσουν για να χρησιμοποιήσουν τα χαρακτηριστικά που τους ενδιαφέρουν, ενώ επιτρέπει στους απλούς χρήστες να αποφύγουν να πληρώσουν για χαρακτηριστικά που δεν τους ενδιαφέρουν και δεν ξέρουν πώς να τα χρησιμοποιήσουν.

Μια δεύτερη επιλογή είναι η εφαρμογή να συμπεριληφθεί σε ένα μεμονωμένο πακέτο. Κατά την διάρκεια της εγκατάστασης, ο χρήστης παροτρύνεται να επιλέξει ποια έκδοση της εφαρμογής επιθυμεί να εγκατασταθεί. Αυτές οι εκδόσεις συνήθως περιλαμβάνουν την απλούστατη ή αρχάρια έκδοση, η οποία περιλαμβάνει το ελάχιστο δυνατό σύνολο βασικών χαρακτηριστικών, την κανονική έκδοση και την προχωρημένη ή εξειδικευμένη έκδοση που περιέχει επιπλέον χαρακτηριστικά τα οποία μόνο εξειδικευμένοι χρήστες αναζητούν.

Το μοντέλο αυτό χρησιμοποιείται ευρύτατα σήμερα και συχνά συνδυάζεται με κάποιο άλλο μοντέλο παροχής άδειας έτσι ώστε να εξασφαλίζει την παρεμπόδιση της αντιγραφής καθώς και των πολλαπλών εγκαταστάσεων.

Μοντέλο παροχής άδειας με βάση της Τοποθεσία

Το μοντέλο παροχής άδειας με βάση την Τοποθεσία εισήχθη για να κάνει το λογισμικό περισσότερο προσιτό οικονομικά στις μεγάλες εταιρίες και τα εκπαιδευτικά ιδρύματα. Αυτό το μοντέλο παρέχει άδειες σε όλους τους υπολογιστές μέσα σε μια εταιρία ή εκπαιδευτικό ίδρυμα ή σε πρόσωπα που σχετίζονται, για εγκατάσταση και χρήση της εφαρμογής ή συνόλου εφαρμογών από των ίδιο πωλητή.

Μοντέλο παροχής άδειας βασισμένο στο Υλικό: Dongle

Η διασφάλιση χρήσης λογισμικού σε μεμονωμένους υπολογιστές μπορεί να επιτευχθεί χρησιμοποιώντας μία μικρή συσκευή που ονομάζεται *dongle*. Το dongle περιέχει έναν αριθμό κλειδί που η εφαρμογή χρειάζεται να επιβεβαιώνει κάθε φορά που εκτελείται. Το dongle πρέπει να εισαχθεί στην κατάλληλη θύρα του υπολογιστή για να ξεκινήσει η εφαρμογή την εκτέλεση της.

Εικόνα 1: Dongle

Μοντέλο παροχής άδειας .NET

Το μοντέλο παροχής άδειας με .NET εισήχθη το 2002 ως τμήμα της προγραμματιστικής πλατφόρμας της Microsoft με όνομα .NET Framework development και υλοποιείται σε μεμονωμένη συσκευή χωρίς τοπικό ή παγκόσμιο δίκτυο και χωρίς διαδίκτυο. Η βασική του υλοποίηση που παρέχεται μαζί με την πλατφόρμα επιβεβαιώνει την εγκυρότητα της άδειας ελέγχοντας την τιμή της εγγραφής κατά την εκτέλεση. Ωστόσο οι προγραμματιστές μπορούν να το επεκτείνουν να υποστηρίζει περισσότερα μοντέλα επεκτείνοντας τον διαχειριστή αδειών. Η διαθεσιμότητα μιας εφαρμογής σε αυτό το μοντέλο απαιτεί την εισαγωγή στον κώδικα της, κατάλληλων στοιχείων, γεγονός που αποτελεί μειονέκτημα μιας και αποτελεί επιπλέον εργασία για τον προγραμματιστή ενώ υπάρχει η πιθανότητα ασυμβατότητας του κώδικα.

2.1.3 Ο ρόλος του Διαδικτύου

Οι τεχνικές παροχής άδειας λογισμικού εξελίχθηκαν ακολουθώντας την ανάπτυξη του λογισμικού, του υλικού και των δικτύων. Παρά την ανάπτυξη αυτή, ο κύριος σκοπός της παροχής άδειας, παραμένει ο ίδιος: *η προστασία μιας εφαρμογής από εκούσια ή ακούσια κακή χρήση*. Οι τεχνικές παροχής άδειας, παράλληλα με την διαφύλαξη της χρήσης του λογισμικού, στοχεύουν στην παροχή ενός εύκολου και ευέλικτου τρόπου είτε απόκτησης της εφαρμογής, είτε πληρωμής της χρήσης της, είτε επιβολής των όρων χρήσης της στους χρήστες .

Συγκεκριμένα συστήματα σε διαφορετικά περιβάλλοντα διαχειρίζονται τη διαδικασία παροχής άδειας με διαφορετικούς τρόπους και κάθε περιβάλλον έχει το δικό του τρόπο παραγωγής, έγκρισης και διανομής της άδειας. Οι τεχνικές παροχής άδειας χρήσης λογισμικού εξαρτώνται από τον τύπο της εφαρμογής και τους τελικούς χρήστες, από το κόστος της εφαρμογής, από το υλικό και το δικτυακό περιβάλλον στο οποίο εκτελείται η εφαρμογή και από τους δράστες που συμμετέχουν στη διαδικασία παροχής και διανομής της άδειας.

Η εμφάνιση του Διαδικτύου

Με την εμφάνιση των τοπικών δικτύων(Local Area Networks-LAN), οι περισσότερες επιχειρήσεις και εκπαιδευτικά ιδρύματα συνδέουν υπολογιστές, εξυπηρετητές, εκτυπωτές και άλλες συσκευές χρησιμοποιώντας ένα τοπικό δίκτυο. Τα τοπικά δίκτυα κατέστησαν δυνατούς νέους τρόπους διανομής της εφαρμογής και της άδειας χρήσης της, δεδομένου ότι ο κάθε κόμβος μπορεί να αποκτήσει λογισμικό και άδεια από οποιονδήποτε άλλο εντός δικτύου. Επίσης οι εφαρμογές στα τοπικά δίκτυα μπορούν να βρίσκονται σε ένα μόνο μηχάνημα και μπορούν να προσπελάζονται από οποιαδήποτε άλλη συσκευή εντός δικτύου.

Η ιδέα ότι οι κόμβοι εντός τοπικού δικτύου μπορούν να προσπελάζονται από έναν κεντρικό εξυπηρετητή ο οποίος χειρίζεται τις άδειες χρήσης, οδήγησε σε καινούργιες αρχιτεκτονικές. Τα δίκτυα επίσης, κατέστησαν δυνατούς, νέους τρόπους ταυτοποίησης των συσκευών μέσω της διεύθυνσης IP, του ονόματος συσκευής ή μέσω της μοναδικής διεύθυνσης MAC του προσαρμογέα δικτύου τους. Επιπρόσθετα, οι χρήστες ενός δικτύου έχουν μοναδικά ονόματα τα οποία μπορούν να χρησιμοποιηθούν ως ένας τρόπος ταυτοποίησης.

Αυτά τα χαρακτηριστικά μιας δικτυακής υποδομής συνετέλεσαν στην διαφοροποίηση των μεθόδων με τις οποίες ανεξάρτητοι χρήστες και επιχειρήσεις ή ομάδες αυτών αποκτούν άδεια χρήσης.

Τα μοντέλα παροχής άδειας χρήσης που χρησιμοποιούνται σήμερα σε εφαρμογές και η λειτουργία τους προϋποθέτει την ύπαρξη διαδικτύου, είναι το «σύγχρονο μοντέλο» και το μοντέλο «κλειδωμένο στον κόμβο».

Μοντέλο παροχής άδειας χρήσης με σύγχρονο τρόπο (concurrent license model)

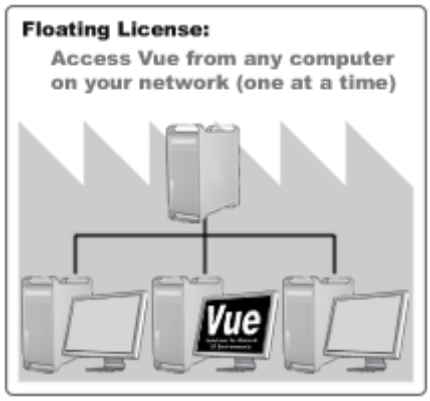
Πριν να συνδεθούν οι υπολογιστές μέσω τοπικών δικτύων, κάθε υπολογιστής σε μία εταιρία έπρεπε να αποκτήσει άδεια χρήσης λογισμικού ανεξάρτητα από τους υπολοίπους, ακόμα και αν σε δεδομένη χρονική στιγμή μόνο λίγοι από τους υπολογιστές χρησιμοποιούσαν τη συγκεκριμένη εφαρμογή. Αυτή η μέθοδος παροχής άδειας είχε ως αποτέλεσμα οι επιχειρήσεις να πληρώνουν για περισσότερες άδειες από όσες χρησιμοποιούσαν στην πραγματικότητα. Για να αποφευχθεί αυτό σε ένα τοπικό δίκτυο, εισήχθη το σύγχρονο μοντέλο.

Μια επιχείρηση μπορεί να αγοράσει ένα συγκεκριμένο αριθμό αδειών και να εγκαταστήσει την εφαρμογή σε περισσότερους υπολογιστές από όσες είναι οι άδειες που αγόρασε. Η προϋπόθεση για την λειτουργία της εφαρμογής είναι ότι ο αριθμός της ταυτόχρονης χρήσης της εφαρμογής στο δίκτυο, να μην ξεπερνά τον αριθμό των αδειών που έχουν αγοραστεί.

Όλες οι άδειες που έχουν αγορασθεί, αποθηκεύονται σε έναν εξυπηρετητή και πριν από τη χρήση της εφαρμογής κάθε πελάτης ζητά μια άδεια από τον εξυπηρετητή. Η άδεια χορηγείται στον πελάτη, εφόσον ο αριθμός των αδειών που έχουν δεσμευτεί από άλλους χρήστες τη δεδομένη στιγμή είναι μικρότερος από τον συνολικό αριθμό των αποθηκευμένων αδειών. Με αυτό τον τρόπο, οι άδειες "ρέουν" στο δίκτυο, αφού κάθε υπολογιστής εντός δικτύου μπορεί να τις δεσμεύσει.

Συχνά το σύγχρονο μοντέλο, απαιτεί από την εφαρμογή να επιβεβαιώσει, κατά τη διάρκεια της εκτέλεσής της, τη χρήση της άδειας και να την αποδεσμεύσει, επιστρέφοντας τη στον εξυπηρετητή μόλις διακόψει τη λειτουργία της, προκειμένου να μπορέσει να τη

χρησιμοποιήσει κάποιος άλλος χρήστης. Για το λόγο αυτό, το σύγχρονο μοντέλο απαιτεί TCP/IP συνδέσεις, καθ' όλη τη διάρκεια εκτέλεσης της εφαρμογής. Αν χαθεί η επικοινωνία και η χρήση της άδειας δεν έχει επιβεβαιωθεί, η άδεια χρήσης ανακαλείται και η εκτέλεση της εφαρμογής διακόπτεται. Η απαίτηση για συνεχείς TCP/IP συνδέσεις δημιουργεί προβλήματα απόδοσης στον εξυπηρετητή, καθώς ο αριθμός των ταυτόχρονων συνδέσεων μπορεί να είναι πολύ μεγάλος σε κάποιες επιχειρήσεις.



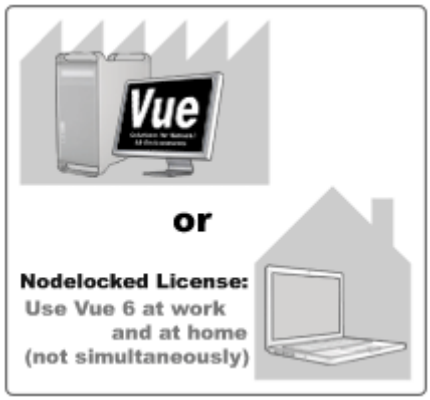
Εικόνα 2: Σύγχρονο Μοντέλο

Μοντέλο παροχής άδειας κλειδωμένης σε κόμβο (node-locked license model)

Σε ένα δίκτυο οι άδειες μπορούν επίσης να είναι κλειδωμένες σε ένα συγκεκριμένο κόμβο-συσκευή, και μόνο αυτός ο κόμβος (ή κόμβοι) μπορεί να χρησιμοποιήσει την εφαρμογή. Αυτό το μοντέλο είναι γνωστό ως μοντέλο παροχής άδειας κλειδωμένης σε κόμβο.

Μία τέτοια άδεια περιλαμβάνει πληροφορίες για την ταυτότητα του κόμβου, για παράδειγμα την στατική διεύθυνση IP του, το όνομα υπολογιστή του ή την διεύθυνση MAC της κάρτας δικτύου του.

Αυτό το μοντέλο παροχής άδειας εξασφαλίζει ότι η εφαρμογή θα εκτελεσθεί μόνο στον συγκεκριμένο κόμβο-συσκευή. Η εκτέλεση κλειδωμένου σε κόμβο λογισμικού δεν απαιτεί την ύπαρξη εξυπηρετητή παροχής άδειας. Παρόλα αυτά, αυτό το μοντέλο εφαρμόζεται μόνο σε τοπικά δίκτυα εφόσον με τη σύνδεση σε αυτά. οι υπολογιστές αποκτούν μοναδικά χαρακτηριστικά στα οποία κλειδώνονται οι άδειες. [Dus05]



Εικόνα 3: Μοντέλο Κλειδωμένο σε Κόμβο

2.2 Διαδικτυακές Υπηρεσίες

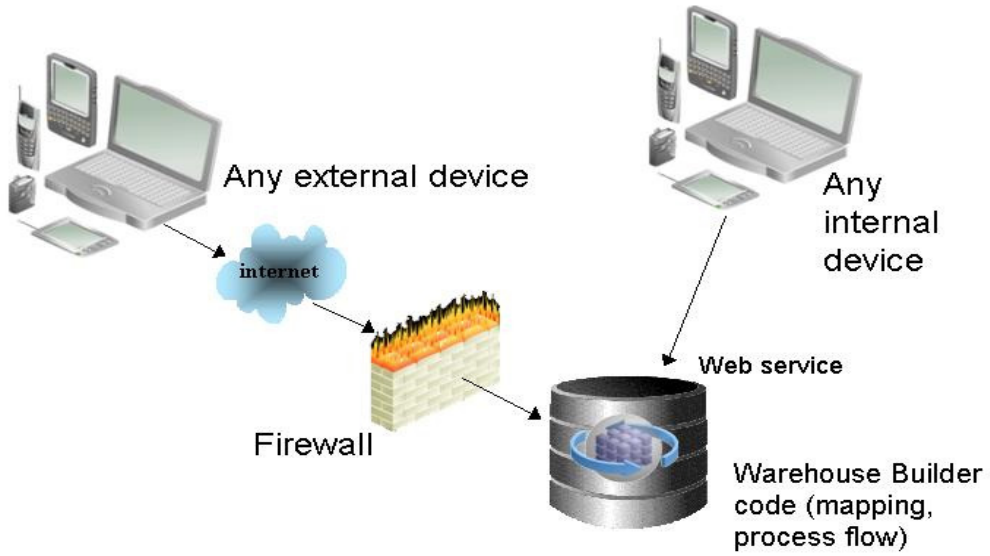
2.2.1 Τι είναι η Διαδικτυακή Υπηρεσία

Ο οργανισμός W3C(World Wide Web Consortium) ορίζει την διαδικτυακή υπηρεσία ως "ένα σύστημα λογισμικού σχεδιασμένο να υποστηρίζει διασυστηματική διαδραστικότητα μέσω ενός δικτύου". Με τον όρο διασυστηματική, εννοείται η λειτουργία ανεξαρτήτως συστήματος - πλατφόρμας.

Οι διαδικτυακές υπηρεσίες είναι συχνά απλά Web APIs (application programming interface) που μπορούν να προσπελαστούν μέσω κάποιου δικτύου, όπως για παράδειγμα το διαδίκτυο, και μπορούν να εκτελεστούν σε ένα απομονωμένο σύστημα που φιλοξενεί τις δεδομένες υπηρεσίες.

Ο ορισμός που δίνει ο οργανισμός W3C, αναφέρει την ύπαρξη πολλών διαφορετικών συστημάτων. Ωστόσο κατά την συνήθη χρήση μιας διαδικτυακής υπηρεσίας οι όροι που αναφέρονται, είναι ο πελάτης και ο εξυπηρετητής που επικοινωνούν με μηνύματα XML έτσι όπως προβλέπεται από το πρωτόκολλο SOAP.

Οι λειτουργίες που υποστηρίζει ο εξυπηρετητής περιγράφονται από την Γλώσσα Περιγραφής Διαδικτυακών Υπηρεσιών (Web Services Description Language (WSDL)). Η περιγραφή αυτή είναι αναγνώσιμη από μηχανή. Την περιγραφή αυτή, δεν την απαιτεί κάποιος ακροδέκτης του πρωτοκόλλου *SOAP*, αλλά αποτελεί ένα προαπαιτούμενο για αυτοματοποιημένη δημιουργία κώδικα από την πλευρά του πελάτη σε Java και .NET SOAP πλαίσια (πλαίσια όπως το Spring και ο Apache CXF αποτελούν εξαιρέσεις). Κάποιοι οργανισμοί όπως το WS-I, περιλαμβάνουν τόσο το SOAP όσο και το WSDL στον ορισμό τους για την διαδικτυακή υπηρεσία.



Εικόνα 4: Διαδικτυακές Υπηρεσίες

2.2.2 Χαρακτηριστικά μιας διαδικτυακής υπηρεσίας

Δεν υπάρχει κανένα έγγραφο που να περιλαμβάνει όλα τα χαρακτηριστικά των διαδικτυακών υπηρεσιών, μιας και αυτά είναι πρότυπα συναρτησιακά στοιχεία. Συγχρόνως όμως, δεν υπάρχει ένα σταθερό σύνολο χαρακτηριστικών που τις διέπουν. Υπάρχει ένα σταθερό σύνολο προδιαγραφών πυρήνα το οποίο όμως αλλάζει, ανάλογα με την επιλογή της τεχνολογίας. Το σύνολο αυτό περιλαμβάνει:

1)Πρωτόκολλο SOAP

Επεκτάσιμη μορφή μηνύματος βασισμένη σε XML, που περιέχει δεσμούς με άλλα πρωτόκολλα. Τα πρωταρχικά πρωτόκολλα είναι το HTTP και το HTTPS, αν και δεσμοί για άλλα, όπως το SMTP και XMPP, έχουν ήδη υλοποιηθεί.

2)Γλώσσα περιγραφής διαδικτυακών υπηρεσιών (WSDL)

Οι διαπροσωπίες των διαδικτυακών υπηρεσιών περιγράφονται από μια γλώσσα με XML μορφή σύμφωνα με τις λεπτομέρειες των δεσμών τους με συγκεκριμένα πρωτόκολλα. Η WSDL τυπικά χρησιμοποιείται για να δημιουργήσει κώδικα εξυπηρετητή και πελάτη και για να τους διαμορφώσει.

3) Universal Description Discovery and Integration (UDDI)

Πρωτόκολλο για εντοπισμό και ανάδειξη των μεταδεδομένων που σχετίζονται με διαδικτυακές υπηρεσίες. Επιτρέπει στις εφαρμογές να εντοπίσουν τις υπηρεσίες είτε κατά την φάση σχεδιασμού τους είτε κατά την φάση εκτέλεσης.

Προκειμένου να βελτιωθεί η διασυστηματικότητα των διαδικτυακών υπηρεσιών εκδόθηκαν πολλά προφίλ με βασικότερο το βασικό προφίλ.

Ένα προφίλ είναι ένα σύνολο προδιαγραφών πυρήνα (SOAP, WSDL, ...) σε συγκεκριμένη έκδοση (SOAP 1.1, UDDI 2, ...) μαζί με κάποιες πρόσθετες απαιτήσεις που καθορίζουν πλήρως τη λειτουργία του πυρήνα.

Το *βασικό προφίλ WS-I* (συχνά αναφέρεται και ως BP), παρέχει την δυνατότητα λειτουργίας των διαδικτυακών υπηρεσιών ανεξαρτήτου υπολογιστικής πλατφόρμας, με τρόπο διασυστηματικό δηλαδή. Το βασικό προφίλ χρησιμοποιεί την WSDL για να περιγράψει τις υπηρεσίες.

Για να επεκταθεί η λειτουργικότητα των διαδικτυακών υπηρεσιών έχουν αναπτυχθεί ή είναι υπό ανάπτυξη ορισμένες προδιαγραφές. Αυτές οι προδιαγραφές αναφέρονται ως WS-*, ορισμένες εκ των οποίων αναφέρονται παρακάτω:

WS-Ασφάλεια

Καθορίζει πώς η XML Κωδικοποίηση και η XML Υπογραφή στο SOAP εξασφαλίζουν ασφαλή ανταλλαγή μηνυμάτων.

WS-Αξιοπιστία

Ένα πρότυπο πρωτόκολλο από το OASIS για αξιόπιστη ανταλλαγή μηνυμάτων μεταξύ δύο διαδικτυακών υπηρεσιών.

WS-Αξιόπιστη Ανταλλαγή Μηνυμάτων

Ένα πρωτόκολλο για αξιόπιστη ανταλλαγή μηνυμάτων μεταξύ δύο διαδικτυακών υπηρεσιών, το οποίο διένειμε η Microsoft σε συνεργασία με τις BEA και IBM και το οποίο προτυποποιεί η OASIS .

WS-Διευθυνσιοδότηση

Ένας τρόπος για να περιγραφεί η διεύθυνση του παραλήπτη (και του αποστολέα) ενός μηνύματος, εντός του ίδιου του SOAP μηνύματος.

WS-Δοσοληψία

Ένας τρόπος για χειρισμό των δοσοληψιών [WB08]

2.3 Τεχνολογία Axis 2

2.3.1 Τι είναι ο axis2

Ο Apache Axis2 είναι ένας πυρήνας για διαδικτυακές υπηρεσίες. Πρόκειται για τον ευρύτατα χρησιμοποιούμενο Apache Axis1, ο οποίος σχεδιάστηκε και κατασκευάστηκε από την αρχή, με καινούργια χαρακτηριστικά. Ο Apache Axis2 παρέχει την δυνατότητα στον χρήστη του, να προσθέσει διαπροσωπίες διαδικτυακών υπηρεσιών σε διαδικτυακές εφαρμογές αλλά μπορεί να λειτουργήσει και ως ένας ξεχωριστός εξυπηρετητής εφαρμογών (server application). Υλοποιήσεις του Axis2 είναι διαθέσιμες σε γλώσσα Java και C++. Αναπτύχθηκε υπό την αιγίδα του οργανισμού Apache Software Foundation.

Ο Apache Axis είναι ένα πλαίσιο ανοιχτού λογισμικού, για διαδικτυακές υπηρεσίες, το οποίο βασίζεται στην γλώσσα xml. Αποτελείται από δύο υλοποιήσεις εξυπηρετητών SOAP (μία σε γλώσσα Java και μία σε C++), από διάφορα εργαλεία και από APIs που εξυπηρετούν την δημιουργία και την τοποθέτηση των διαδικτυακών υπηρεσιών. Χρησιμοποιώντας τον Apache Axis, οι προγραμματιστές είναι σε θέση να δημιουργήσουν δομημένες υπολογιστικές εφαρμογές, οι οποίες λειτουργούν σε οποιοδήποτε υπολογιστικό σύστημα. [WAX208]

2.3.2 Δημιουργία, τοποθέτηση και προσπέλαση διαδικτυακών υπηρεσιών

Στην περίπτωση χρήσης της Java έκδοσης του Axis, υπάρχουν δύο τρόποι να εκτεθεί ο κώδικας Java ως διαδικτυακή υπηρεσία. Ο πρώτος τρόπος, που είναι και πιο εύκολος, είναι η χρήση των Axis JWS (Java Web Service) αρχείων. Ο δεύτερος τρόπος περιλαμβάνει τοποθέτηση των διαδικτυακών υπηρεσιών με τον τυπικό πλέον, τρόπο. Η τυπική τοποθέτηση δίνει την πρόσθετη δυνατότητα στον χρήστη να καθορίζει τα τμήματα του κώδικα που επιθυμεί να θέσει ως διαδικτυακές υπηρεσίες.

Στην συνέχεια παρουσιάζονται συνοπτικά οι δύο τρόποι δημιουργίας, τοποθέτησης και προσπέλασης των διαδικτυακών υπηρεσιών.

1ος τρόπος

Δημιουργία διαδικτυακών υπηρεσιών με χρήση των JWS αρχείων

Τα JWS αρχεία είναι αρχεία πηγαίου κώδικα Java και περιέχουν τις κλάσεις Java που θα τεθούν ως Διαδικτυακές Υπηρεσίες. Η διαφορά τους με τα τυπικά αρχεία java είναι η επέκταση του αρχείου(.jws αντί για .java) καθώς και ότι τοποθετούνται ως πηγαίος κώδικας (.java) κι όχι ως μεταφρασμένα αρχεία κλάσεων (.class). Η δημιουργία λοιπόν περιλαμβάνει την δημιουργία του κατάλληλου JWS αρχείου.

Τοποθέτηση διαδικτυακών υπηρεσιών με χρήση των JWS αρχείων

Εφόσον τεθεί το Axis2 servlet στον εξυπηρετητή, η τοποθέτηση πλέον γίνεται με απλή αντιγραφή των αρχείων jws στον φάκελο του Axis στον εξυπηρετητή. Αυτό ισχύει σε περίπτωση που χρησιμοποιείται ο Apache Tomcat ως υποδοχέας. Σε περίπτωση που χρησιμοποιείται κάποιος άλλος, η τυπική δημιουργία αρχείου WAR είναι απαραίτητη.

Πρόσβαση στις διαδικτυακές υπηρεσίες που έχουν δημιουργηθεί με JWS αρχεία

Οι διαδικτυακές υπηρεσίες υλοποιημένες με JWS αρχεία, είναι προσβάσιμες μέσω της διαδικτυακής διεύθυνσης τους, π.χ. "http://localhost:8080/axis/.../file.jws". Αν χρησιμοποιείται διαφορετικός υποδοχέας από τον Apache Tomcat , η διαδικτυακή διεύθυνση μπορεί να είναι διαφορετική.

2ος τρόπος

Τυπική δημιουργία διαδικτυακών υπηρεσιών με χρήση Axis Archive αρχείων (.aar)

Ο 2^{ος} τρόπος δημιουργίας απαιτεί την ύπαρξη ενός συγκεκριμένου περιγραφέα τοποθέτησης που ονομάζεται WSDD (Web Service Deployment Descriptor). Μπορεί να χρησιμοποιηθεί για να προσδιορίσει ποιές πηγές μπορούν να τεθούν ως διαδικτυακές υπηρεσίες.

Η δημιουργία της διαδικτυακής υπηρεσίας περιλαμβάνει την δημιουργία ενός Axis Archive αρχείου (.aar) το οποίο περιέχει το μεταφρασμένο αρχείο (.class) του αρχείου Java της Διαδικτυακής Υπηρεσίας, καθώς και ένα αρχείο XML .

Τυπική τοποθέτηση διαδικτυακών υπηρεσιών

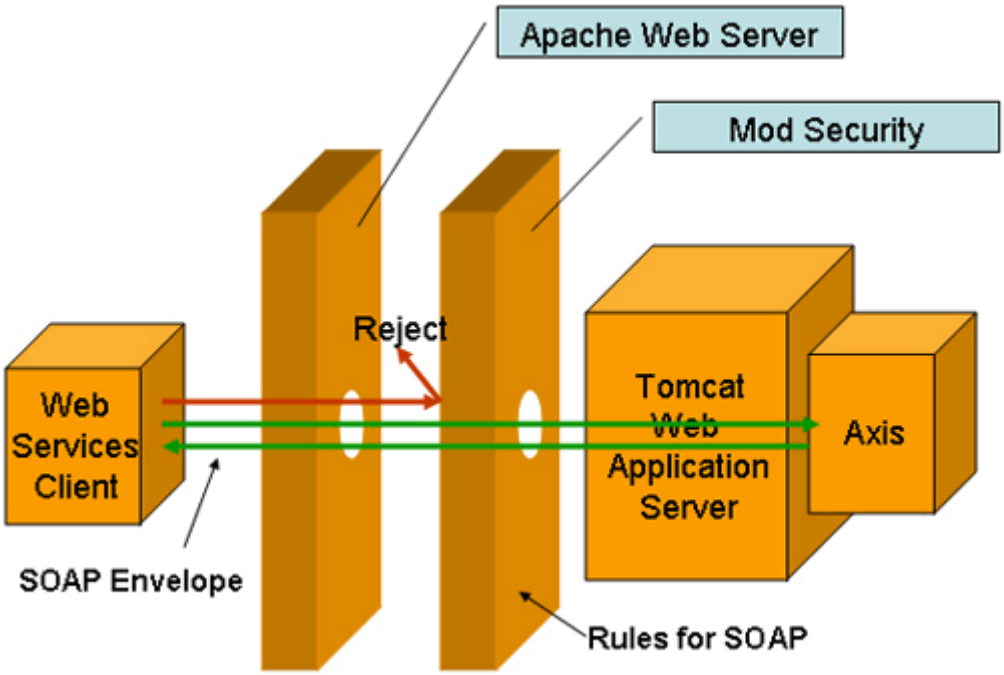
Η τυπική τοποθέτηση περιλαμβάνει την αντιγραφή του αρχείου .aar που αντιπροσωπεύει την διαδικτυακή εφαρμογή στο φάκελο υπηρεσιών του υποδοχέα.

Πρόσβαση στις τυπικές διαδικτυακές υπηρεσίες

Οι διαδικτυακές υπηρεσίες είναι προσβάσιμες μέσω της διαδικτυακής διεύθυνσης τους, π.χ. "http://localhost:8080/axis/.../name_of_aar". Αν χρησιμοποιείται διαφορετικός υποδοχέας από τον Apache Tomcat, η διαδικτυακή διεύθυνση μπορεί να είναι διαφορετική.

Αυτόματη παραγωγή WSDL κώδικα

Όταν τεθεί μία διαδικτυακή υπηρεσία, κατά την προσπέλαση της δημιουργείται αυτόματα ένα αρχείο WSDL με την κατάληξη ?WSDL. [WAX08]



Εικόνα 5: Axis 2

2.3.3 Τα γενικά χαρακτηριστικά του AXIS2

Ο Axis2 ως πυρήνας Διαδικτυακών Υπηρεσιών τηρεί τις βασικές προϋποθέσεις για την λειτουργία τους, όπως η υποστήριξη βασικών πρωτοκόλλων επικοινωνίας, η υποστήριξη του βασικού προφίλ WS 1.1 κτλ.

Παρακάτω παρατίθεται ο πίνακας γενικών χαρακτηριστικών 1. Πέρα από τον πυρήνα Axis2 αποτυπώνεται επίσης η υποστήριξη ή μη, και άλλων πυρήνων διαδικτυακών υπηρεσιών. [SC08]

ΓΕΝΙΚΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ AXIS2

Χαρακτηριστικό	**Axis 1.x**	**Axis 2**	**Celtix**	**Glue**	**JBoss WS**	**XFire (1.2)**	**GlassFish**	**OracleAS 10g**
Συνέπεια ως προς το Basic Profile 1.1 (WS-I)	X	X	X	X	X	X	X	X
Εύκολη Δημιουργία Υπηρεσιών από POJOs	X	X	X	X	X	X	X	X

Ανοιχτός Κώδικας	X	X	X		X	X	X	
RPC-Κωδικοποίηση	X			X	X		X	X
Υποστήριξη Spring	X	X	X			X	X	
Υποστήριξη REST		X					X	X
IDEA πρόσθετα		X			X	X	X	Ant
Eclipse πρόσθετα		X			X	X	X	Ant
NetBeans πρόσθετα							X	Ant
JDeveloper								X
Hot Deployment		X			X	?	X	X
Soap 1.1	X	X	X	X	X	X	X	X
Soap 1.2	X	X		X	X	X	X	X
Streaming XML (StAX based)		X				X	X	
WSDL 1.1 ->Code (Client)	X	X	X	X	X	X	X	X
WSDL 1.1 ->Code (Server)	X	X	X	X	X	X	X	X
WSDL 2.0 ->Code (Client)		X	?	?	?	?		?
WSDL2.0 ->Code (Server)		X	?	?	?	?		?
Ασύγχρονη Επικοινωνία από πλευρά Πελάτη	X	X					X	With BPEL
Ασύγχρονη Επικοινωνία από πλευρά Εξυπηρετητή	X	X					X	With BPEL

Δημιουργία Κώδικα με συγκεκριμένη Πολιτική		X						

Πίνακας 1: Γενικά Χαρακτηριστικά του Axis2

2.4 FLEXlm εφαρμογές

2.4.1 Τι είναι ο FLEXlm

Το FLEXlm είναι η πιο διαδεδομένη μηχανή παροχής άδειας που χρησιμοποιείται στη βιομηχανία λογισμικού. Ανήκει στην κατηγορία μηχανών παροχής άδειας μέσω διαδικτύου. Η δοσοληψία αδειών χρήσης λογισμικού συνεπώς, πραγματοποιείται σε διαδικτυακό επίπεδο, και το χρησιμοποιούν πολλές εταιρίες λογισμικού για να ελέγχουν την χρήση που γίνεται στα προϊόντα τους.

Το FLEXlm προήλθε από τη συνεργασία δύο εταιριών: της GLOBEtrotter Software και της Highland Software το 1988. Το 1994 η εταιρία GLOBEtrotter αγόρασε τα δικαιώματα της Highland πάνω στο FLEXlm προϊόν. Η Highland συνέχισε να είναι ο μεταπωλητής του τρίτου μέρους του λογισμικού. Η GLOBEtrotter αγοράστηκε από την Macrovision το 2000. Η Macrovision με τη σειρά της, μετονόμασε το προϊόν από FLEXlm σε FLEXnet. Το προϊόν FLEXlm , το εμπορεύεται η Macrovision με όνομα FLEXnet Publisher.

Σε γενικές γραμμές είναι ένας διαχειριστής παροχής άδειας λογισμικού υπό την ευθύνη της εταιρίας Macrovision, ο οποίος υλοποιεί πλήρως όλες τις απαραίτητες λειτουργίες για την διαχείριση των αδειών και χρησιμοποιείται συχνά σε επιχειρησιακά περιβάλλοντα για να παρέχει τις άδειες σε πολλαπλούς τελικούς χρήστες ενός λογισμικού.

Όπως αναφέραμε και στην ενότητα 2.1, ο τρόπος παροχής αδειών χρήσης ενός λογισμικού ποικίλλει. Μιλώντας ωστόσο για παροχή άδειας μέσω διαδικτύου, οι τρόποι είναι δύο:

- η επιχείρηση έχει αγοράσει ένα σύνολο αδειών που δεσμεύονται δυναμικά στα υπολογιστικά συστήματα κάθε φορά που εκείνα το ζητήσουν (μιλώντας με FLEXlm όρους, πρόκειται για άδειες που «ρέουν»). Έτσι, όταν ο χρήστης ζητήσει άδεια για να χρησιμοποιήσει το λογισμικό, δεσμεύει μία από το σύνολο της εταιρίας, ενώ όταν τερματίσει τη λειτουργία του, την αποδεσμεύει. Με αυτόν τον τρόπο για παράδειγμα, μια εταιρία που έχει αγοράσει 50 άδειες μπορεί να υποστηρίξει χρήση του λογισμικού σε εκατοντάδες περιστασιακούς χρήστες με την προϋπόθεση ότι οι ταυτόχρονοι χρήστες του λογισμικού δεν ξεπερνούν τους 50.
- η άδεια χρήσης ενός λογισμικού μπορεί να συνδέεται με συγκεκριμένο υπολογιστικό μηχάνημα (μιλώντας με όρους FLEXlm, node-locked άδεια) επιτρέποντας την χρήση του λογισμικού μόνο στο συγκεκριμένο μηχάνημα (κόμβος σε δίκτυο).

Το παραπάνω μοντέλα υποστηρίζονται πλήρως από το FLEXlm.

Το σημαντικότερο πλεονέκτημα του FLEXlm είναι ότι η υποστήριξη που παρέχει στο παραπάνω μοντέλο και συγκεκριμένα η υποστήριξη του στο πρώτο τρόπο παροχής. Επιτρέπει στις άδειες λογισμικού να είναι διαθέσιμες, να «ρέουν» όπως συνήθως λέγεται, οπουδήποτε στο δίκτυο, αντί να είναι δεσμευμένες σε συγκεκριμένα υπολογιστικά μηχανήματα. Η δυνατότητα αυτή ευνοεί τόσο τους χρήστες όσο και τους διαχειριστές παροχής άδειας. Από την μια πλευρά, οι χρήστες μπορούν να κάνουν αποτελεσματικότερη χρήση αδειών εφόσον τις μοιράζονται στο δίκτυο. Από την άλλη, οι διαχειριστές μπορούν να ελέγχουν ποιός χρησιμοποιεί την εφαρμογή και ποιά είναι τα μηχανήματα στα οποία είναι διαθέσιμες οι άδειες.

2.4.2 Συστατικά του FLEXlm

Τα στοιχεία από τα οποία αποτελείται το FLEXlm, οργανώνονται με βάση το μοντέλο παροχής άδειας που χρησιμοποιείται από τον πωλητή λογισμικού.

Τα δύο μοντέλα παροχής άδειας διαφέρουν ως προς την απαίτηση τους ή όχι, για εξυπηρετητή παροχής άδειας.

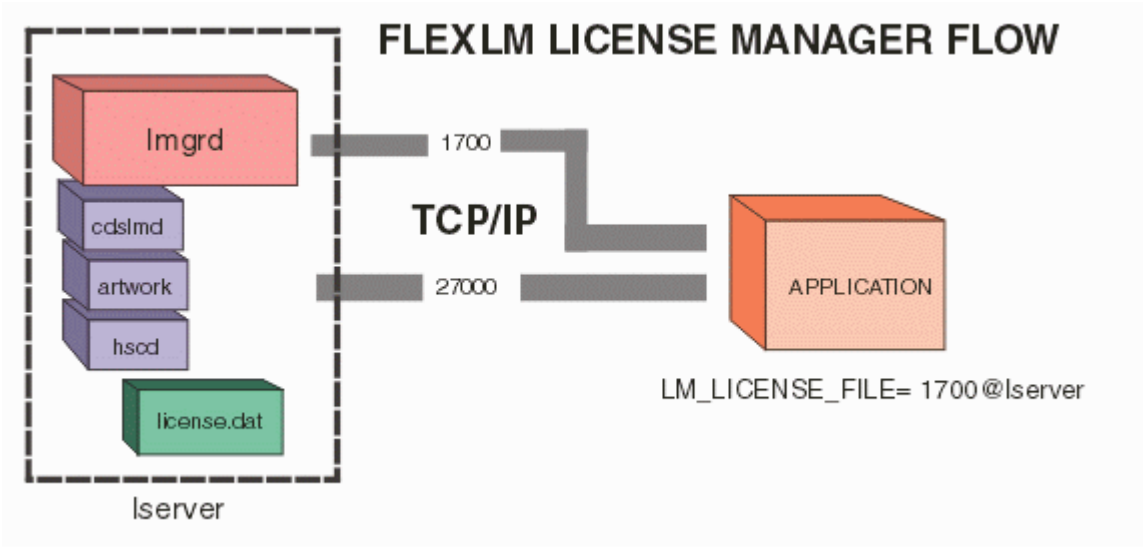
Συγκεκριμένα:

- Σύγχρονο(concurrent) μοντέλο: Οι άδειες παρέχονται από έναν εξυπηρετητή παροχής άδειας. Αυτό συνήθως αποκαλείται και εξυπηρετούμενο μοντέλο παροχής άδειας. Τα αρχεία άδειας (license files) που παρέχονται από τον πωλητή λογισμικού και περιλαμβάνουν τις γραμμές SERVER, VENDOR και κατ' επιλογήν USE_SERVER, απαιτούν έναν εξυπηρετητή παροχής άδειας.
- Μοντέλο κλειδωμένο σε κόμβο (node locked): Οι άδειες δεν παρέχονται από τον εξυπηρετητή παροχής άδειας αλλά είναι διαθέσιμες στην εφαρμογή άμεσα. Αυτό συνήθως ονομάζεται και μη εξυπηρετούμενο μοντέλο παροχής άδειας.

2.4.2.1 Συστατικά του σύγχρονου μοντέλου

Για το σύγχρονο μοντέλο, τα απαιτούμενα στοιχεία του FLEXlm είναι τέσσερα:

- Ο δαίμονας διαχείρισης άδειας(License Manager Daemon - lmgrd)
- Ο δαίμονας-πωλητής(Vendor Daemon) ο οποίος μαζί με το lmgrd αποτελούν τον εξυπηρετητή παροχής άδειας(License Server)
- Το αρχείο παροχής άδειας (License File)
- Η εφαρμογή που διαθέτει FLEXlm άδεια, μαζί με τη στατική βιβλιοθήκη πελάτη.



Εικόνα 6: Συστατικά FLEXlm

Επιπρόσθετα με αυτά τα τέσσερα στοιχεία, υπάρχουν τέσσερα προαιρετικά στοιχεία:

- Το Debug Log File που δημιουργείται και γράφεται από το lmgrd
- Το Report Log File που δημιουργείται και γράφεται από τον Vendor Daemon για χρήση από το FLEXnet Manager.
- Το αρχείο με τις επιλογές διαχείρισης του τελικού χρήστη (End-User Administration Options File)- αρχείο που δημιουργείται και διατηρείται από τον τελικό χρήστη.

Τα στοιχεία που αποτελούν τον License Server βρίσκονται σε ένα μηχάνημα στο δίκτυο αλλά μπορούν εναλλακτικά να βρίσκονται στο ίδιο μηχάνημα με την εφαρμογή-πελάτη. Τα τρία προαιρετικά αρχεία: Debug Log, Administration Options, και Report Log ρυθμίζονται από τον τελικό χρήστη της εφαρμογής ή αντίστοιχα από τον διαχειριστή του συνόλου των τελικών χρηστών.

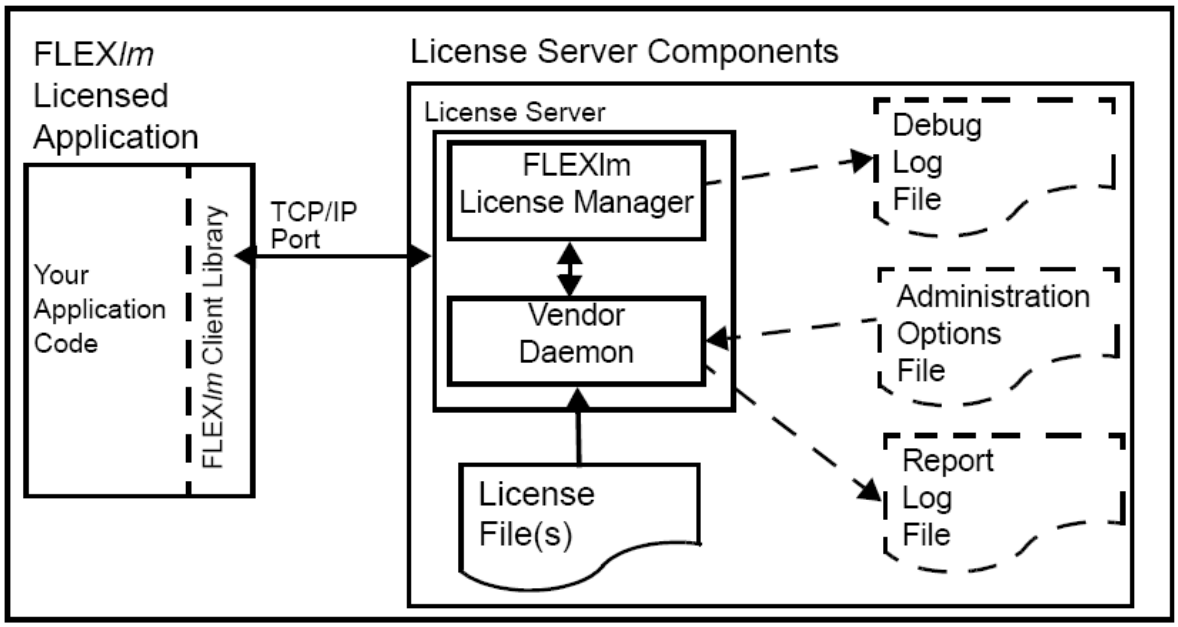
2.4.2.2 Συστατικά του μοντέλου κλειδωμένου σε κόμβο

Για το μοντέλο κλειδωμένο σε κόμβο, τα απαιτούμενα στοιχεία του FLEXlm είναι δύο:

1. Η εφαρμογή που διαθέτει FLEXlm άδεια, μαζί με τη στατική βιβλιοθήκη πελάτη.
2. Το αρχείο παροχής άδειας(License File)

2.4.3 Περιγραφή των Συστατικών του Σύγχρονου Μοντέλου

Στο σχήμα 7 απεικονίζονται όλα τα στοιχεία που συμμετέχουν στο σύγχρονο μοντέλο παροχής άδειας.



Εικόνα 7: Σύσταση FLEXlm

Δαίμονας Διαχείρισης Αδειών (License Manager Daemon, lmgrd)

Τι είναι ?

Είναι μία διεργασία της Macrovision που εκτελείται διαρκώς στον FLEXlm License Server.

Λειτουργία

Η πρώτη επαφή των FLEXlm εφαρμογών πραγματοποιείται με τον δαίμονα διαχείρισης αδειών (lmgrd). Εκείνος καθορίζει ποιος είναι ο κατάλληλος δαίμονας-πωλητής (Vendor Daemon) που θα τις εξυπηρετήσει. Μετά από αυτόν τον καθορισμό, οι εφαρμογές συνδέονται κατευθείαν με τον Vendor Daemon που τους αντιστοιχεί χωρίς να επέμβει ξανά ο lmgrd.

Δαίμονας-Πωλητής Αδειών (License Vendor Daemon)

Τι είναι ?

Κάθε εταιρία λογισμικού που παρέχει προϊόντα της με μοντέλο παροχής άδειας μέσω διαδικτύου, έχει κυκλοφορήσει εφαρμογές συμβατές με το FLEXlm. Η εταιρία είναι λοιπόν ο επίσημος πωλητής του λογισμικού της . Οι άδειες που παρέχονται στους τελικούς χρήστες που διαθέτουν αυτές τις εφαρμογές, δίνονται κατ' αίτηση, από μία διεργασία που εκτελείται στον License Server. Η διεργασία αυτή αντιστοιχεί σε έναν πωλητή-εταιρία, και όσες εταιρίες λογισμικού υποστηρίζονται από τον εκάστοτε FLEXlm License Server, τόσες διεργασίες εκτελούνται.

Για παράδειγμα ένας FLEXlm License Server που υποστηρίζει την FLEXlm έκδοση της εφαρμογής "foo" της εταιρίας "boo", διαθέτει μία διεργασία υλοποιημένη από την "boo".

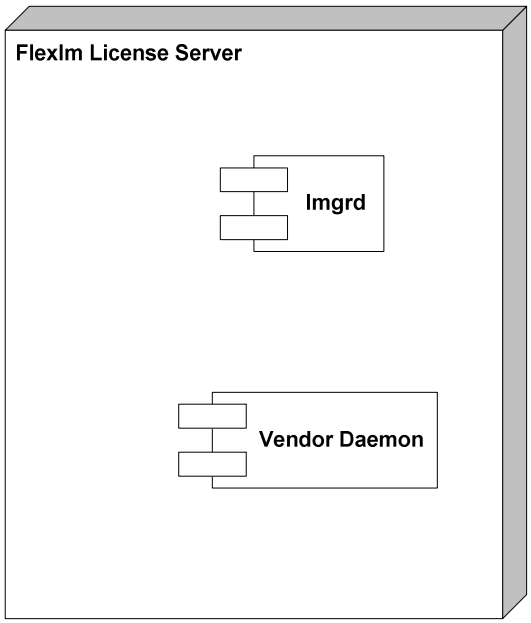
Αυτή η διεργασία ονομάζεται Δαίμονας-Πωλητής Αδειών ή Vendor Daemon.

Λειτουργία

Ο Vendor Daemon αναλαμβάνει την επικοινωνία με τις εφαρμογές που αντιστοιχούν σε αυτόν (π.χ. ο Vendor Daemon της “boo” θα αναλάβει τα αιτήματα των εφαρμογών “foo”) . Εφόσον ο lmgrd έχει γνωστοποιήσει στις εφαρμογές ποιός είναι ο Vendor Daemon που θα τις εξυπηρετήσει, οι εφαρμογές στη συνέχεια επικοινωνούν με τον Vendor Daemon και εκείνος επιτρέπει ή όχι την εκτέλεση τους. Η επικοινωνία πραγματοποιείται μέσω TCP/IP τεχνολογίας.

Ο Vendor Daemon καταγράφει ανά πάσα στιγμή, πόσες άδειες χρησιμοποιούνται και ποιός τις έχει δεσμεύσει.

Αν ο Vendor Daemon τερματιστεί για οποιοδήποτε λόγο, όλοι οι χρήστες χάνουν τις άδειες τους (παρόλο που αυτό δε σημαίνει ότι οι εφαρμογές σταματούν να εκτελούνται εκείνη τη στιγμή). Οι χρήστες συνήθως επανακτούν την άδεια τους αυτόματα, όταν ο lmgrd εκκινήσει ξανά τον Vendor Daemon, ενώ οι εφαρμογές μπορεί να τερματιστούν αν ο Vendor Daemon δεν επανεκκινηθεί μέσα σε δεδομένο χρονικό διάστημα.



Εικόνα 8: Σύσταση FLEXlm Server

Τα δύο παραπάνω στοιχεία - συστατικά συνιστούν τον License Server, όπως φαίνεται και στην εικόνα 8.

Η FLEXlm-εφαρμογή και ο FLEXlm License Server μπορούν να τρέχουν σε ξεχωριστά μηχανήματα στο δίκτυο. Η διαμόρφωση της επικοινωνίας μεταξύ της FLEXlm-εφαρμογής και του Vendor Daemon είναι ανεξάρτητη του είδους της εφαρμογής επιτρέποντας την ύπαρξη δικτύων οποιασδήποτε μορφής. Αυτό σημαίνει ότι ο License Server και ο υπολογιστής που εκτελεί μια εφαρμογή μπορεί να είναι διαφορετικά υπολογιστικά μηχανήματα που μπορούν να έχουν εγκατεστημένα διαφορετικά λειτουργικά συστήματα(WINDOWS και UNIX για παράδειγμα).

Αρχείο Παροχής Άδειας (License File)

Τι είναι ?

Οι πληροφορίες σχετικές με την παροχή άδειας αποθηκεύονται σε ένα αρχείο κειμένου το οποίο ονομάζεται License File. Το License File δημιουργείται από τον Vendor Daemon, ενώ το επεξεργάζεται και το εγκαθιστά ο lmgrd.

Λειτουργία

Περιλαμβάνει πληροφορίες για τα μηχανήματα του lmgrd και του Vendor Daemon και τουλάχιστον μία γραμμή δεδομένων (που ονομάζεται FEATURE ή INCREMENT γραμμή) για κάθε προϊόν που έχει άδεια χρήσης. Κάθε γραμμή FEATURE περιλαμβάνει ένα κλειδί άδειας ή μια υπογραφή που εξαρτάται από τα δεδομένα αυτής της γραμμής, τα hostid που προσδιορίζονται στη SERVER γραμμή(-ές), και άλλα δεδομένα που αφορούν τον πωλητή. Οι περισσότερες εφαρμογές έχουν μια προβλεπόμενη τοποθεσία για το License File.

Παράδειγμα

```
SERVER my_server 17007ea8 1700
VENDOR sampled
FEATURE f1 sampled 1.000 01-jan-2005 10 SIGN=9BFAC0316462
FEATURE f2 sampled 1.000 01-jan-2005 10 SIGN=1B9A308CC0F7
```

Η γραμμή SERVER ορίζει ως License Server εκείνον με όνομα “my_server”, με αναγνωριστικό “17007ea8” και θύρα στην οποία ακούει, την 1700.

Η γραμμή VENDOR ορίζει ως Vendor Daemon εκείνον με όνομα sampled.

Οι γραμμές FEATURE περιγράφουν την άδεια που απαιτείται για το προϊόν. Εδώ, υπάρχουν διαθέσιμες 10 άδειες για την εφαρμογή f1 (όνομα που ορίζει ο Vendor Daemon) με έκδοση 1.000, που θα εξυπηρετηθούν από τον Vendor Daemon με όνομα sampled. Η ημερομηνία λήξης των αδειών είναι η 1/1/2005 και η υπογραφή πιστοποιεί το γνήσιο της άδειας.

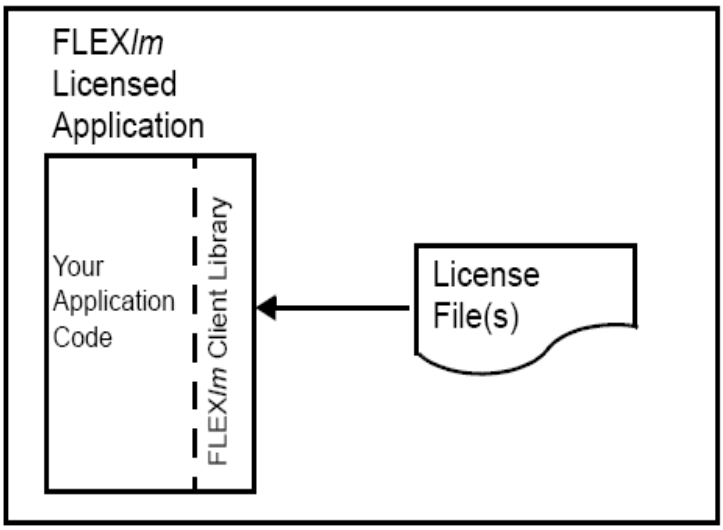
Η FLEXlm Εφαρμογή

Τι είναι ?

Η εφαρμογή σε συνδυασμό μια FLEXlm βιβλιοθήκη που διαθέτει ο τελικός χρήστης, ορισμένη να εκτελείται εφόσον αποκτήσει άδεια μέσω διαδικτύου.

Λειτουργία

Ο κώδικας της εφαρμογής είναι συνδεδεμένος με την FLEXlm βιβλιοθήκη του πελάτη η οποία παρέχει την επικοινωνία με τον License Server. Κατά τη διάρκεια της εκτέλεσης, η εφαρμογή επικοινωνεί με το Vendor Daemon για να ζητήσει άδεια.



Εικόνα 9: FLEXlm εφαρμογή

Αρχείο Καταγραφής Σφαλμάτων (Debug Log File)

Το αρχείο καταγραφής σφαλμάτων περιλαμβάνει μηνύματα που αφορούν την κατάσταση του συστήματος αλλά και τυχόν σφάλματα που εμφανίστηκαν κατά την εκτέλεση. Τα μηνύματα αυτά είναι πολύ χρήσιμα για την εύρεση των σφαλμάτων στον κώδικα του License Server. Κάποια από τα περιεχόμενα του, αφορούν το lmgrd και κάποια τους Vendor Daemon.

Αρχείο Επιλογών του Τελικού Χρήστη (End-User Administration Options File)

Το αρχείο επιλογών του τελικού χρήστη αρχείο, επιτρέπει στον τελικό χρήστη της εφαρμογής, να ελέγξει διάφορες λειτουργικές παραμέτρους του FLEXlm. Ο τελικός χρήστης ο οποίος επεξεργάζεται αυτό το αρχείο είναι ο διαχειριστής στο σύνολο των τελικών χρηστών μιας εταιρίας για παράδειγμα. Οι παράμετροι του FLEXlm που μπορεί να ρυθμίσει, είναι μόνο εκείνες που ανήκουν στην δικαιοδοσία του την οποία ορίζει ο Vendor Daemon.

Αρχείο Καταγραφής Αναφορών (Report Log File)

Το αρχείο καταγραφής αναφορών περιλαμβάνει πληροφορίες για την χρήση των προϊόντων και δημιουργείται από τον Vendor Daemon.

2.4.4 Εργαλεία διαχείρισης του FLEXlm

Ο FLEXlm Server παρέχει κάποια εργαλεία στο διαχειριστή, ώστε να τον βοηθήσει να ρυθμίσει τη διαδικασία παροχής άδειας στο διαδίκτυο.

Οι λειτουργίες κάποιων από αυτών είναι οι εξής:

lmdiag: επιτρέπει τη διάγνωση προβλημάτων όταν δεν μπορεί να ελεγχθεί μια άδεια. Αν ο έλεγχος επιτύχει, το lmdiag το υποδεικνύει, ενώ αν αποτύχει υποδεικνύει το λόγο για τον οποίον απέτυχε.

lmdown: το εργαλείο lmdown επιτρέπει την απενεργοποίηση των επιλεγμένων FLEXlm Servers.

lmhostid: το εργαλείο lmhostid επιστρέφει το FLEXlm hostid της παρούσας πλατφόρμας.

lmreread: το εργαλείο lmreread προκαλεί τον lmgrd να ξαναδιαβάσει το license file και να εκκινήσει καινούργιους vendor daemon αν έχουν προστεθεί. Επιπλέον, όσοι vendor daemons τρέχουν ήδη, καλούνται να ξαναδιαβάσουν το license file για να ελέγξουν οποιεσδήποτε αλλαγές έχουν γίνει στις ρυθμίσεις τους.

lmstat: Το εργαλείο lmstat βοηθά στην παρατήρηση της κατάστασης όλης της δραστηριότητας που αφορά την παροχή άδειας, συμπεριλαμβανομένων και των daemons που εκτελούνται αλλά και των licenses.

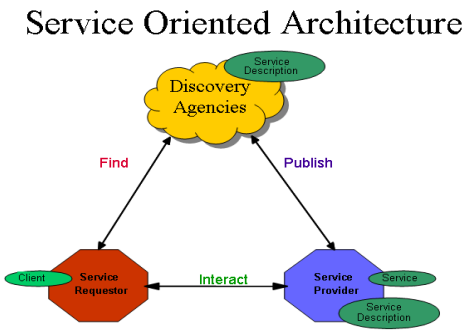
2.4.5 Συνοπτική Περιγραφή της διαδικασίας απόκτησης άδειας στο Σύγχρονο Μοντέλο

Η διαδικασία απόκτησης της άδειας εκτέλεσης μιας FLEXlm εφαρμογής περιλαμβάνει τα ακόλουθα βήματα:

i. Η FLEXlm βιβλιοθήκη του πελάτη εντοπίζει το license file, το οποίο περιλαμβάνει το όνομα και το αναγνωριστικό του FLEXlm License Server καθώς και τον αριθμό της TCP/IP θύρας στην οποία ακούει ο lmgrd (όπως προαναφέραμε ο lmgrd είναι τμήμα του License Server).
ii. Η FLEXlm-εφαρμογή συνδέεται με τον lmgrd στην παραπάνω θύρα. Ο lmgrd αναγνωρίζει τον Vendor Daemon στον οποίο πρέπει απευθυνθεί η εφαρμογή.
iii. Το lmgrd εντοπίζει ποιό μηχάνημα και ποιά TCP/IP θύρα αντιστοιχεί στον Vendor Daemon και γνωστοποιεί τα στοιχεία του στη FLEXlm-εφαρμογή. (Σημ: ο lmgrd έχει καθορίσει εξαρχής ποιό μηχάνημα εκτελεί ποιόν Vendor Daemon, εδώ τον εντοπίζει απλώς)
iv. Η FLEXlm-εφαρμογή πραγματοποιεί σύνδεση με τον δεδομένο Vendor Daemon και του στέλνει την αίτηση για παροχή άδειας.
v. Ο Vendor Daemon ελέγχει αν υπάρχουν διαθέσιμες άδειες και στέλνει στην FLEXlm-εφαρμογή την αποδοχή ή την άρνηση του στην αίτηση της.
vi. Η FLEXlm βιβλιοθήκη του πελάτη, δέχεται ή αρνείται την άδεια. [Mac04]

2.5 Υπηρεσιοστραφής Αρχιτεκτονική –Service Oriented Architecture

2.5.1 Τι είναι SOA?



Εικόνα 10: SOA

Η Υπηρεσιοστραφής Αρχιτεκτονική (Service-Oriented-Architecture - SOA) αποτελεί μια πολύ ενδιαφέρουσα αρχιτεκτονική ιδέα. Βασίζεται στη συνεργασία σύνθετων στοιχείων (στοιχεία ενσωματωμένα μέσα σε άλλα στοιχεία). Η συνεργασία είναι τέτοια ώστε να μην περιορίζει τα επιμέρους στοιχεία να αποτελούν μέρος ενός άλλου συστήματος ή να επιτελέσουν μια άλλη λειτουργία. Ο στόχος της SOA είναι να ελαχιστοποιήσει τις περιττές εξαρτήσεις ανάμεσα σε στοιχεία και σε συστήματα λογισμικού αλλά και να διατηρήσει τη λειτουργικότητα του. [GCR08]

Τι εννοούμε με τον όρο υπηρεσίες;

Οι υπηρεσίες, όμως και τα υπόλοιπα στοιχεία που συνιστούν ένα υπολογιστικό σύστημα, είναι ανεξάρτητες κατασκευαστικές οντότητες. Η διαφορά όμως έγκειται στο ότι οι υπηρεσίες διαθέτουν έναν αριθμό μοναδικών χαρακτηριστικών που τους επιτρέπουν να θεωρούνται τμήμα υπηρεσιοστραφούς αρχιτεκτονικής.

Ένα τέτοιο χαρακτηριστικό είναι η παντελής αυτονομία μιας υπηρεσίας από τις άλλες. Αυτό σημαίνει ότι κάθε υπηρεσία είναι υπεύθυνη για συγκεκριμένη λειτουργία το οποίο με τη σειρά του σημαίνει ότι η κάθε υπηρεσία έχει έναν πολύ περιορισμένο επιχειρησιακό στόχο.

Ο σχεδιασμός, στην πορεία του, οδηγεί στην δημιουργία μεμονωμένων μονάδων συνδεδεμένων μεταξύ τους μέσω ενός κοινού πρωτοκόλλου επικοινωνιών που χρησιμοποιούνε. Λόγω της ανεξαρτησίας που τους παρέχει το πρωτόκολλο η προγραμματιστική λογική που ενθυλακώνουν, δεν χρειάζεται να συμμορφωθεί με συγκεκριμένη πλατφόρμα ή τεχνολογία. Είναι πλήρως ανεξάρτητες.

Οι διαδικτυακές υπηρεσίες ως υπηρεσίες

Το πιο διαδεδομένο και πιο επιτυχές είδος υπηρεσιών είναι οι διαδικτυακές υπηρεσίες (βλέπε κεφ. 2.2)

Αυτό το είδος υπηρεσίας έχει τις εξής δύο λειτουργικές απαιτήσεις:

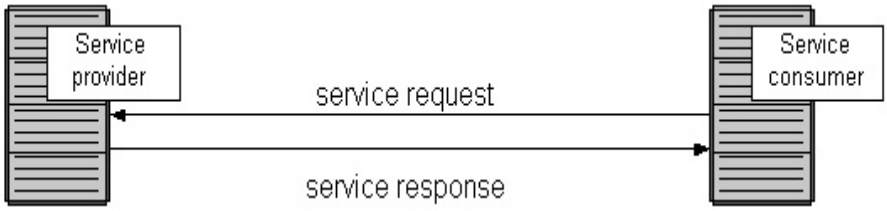
- Επικοινωνεί με Internet πρωτόκολλα (πιο σύνηθες είναι το HTTP)
- Στέλνει και λαμβάνει δεδομένα με τη μορφή *XML* αρχείων

Πρακτικά η SOA, αποτελεί έναν τρόπο σχεδίασης ενός συστήματος λογισμικού έτσι ώστε αυτό, να μπορεί να παρέχει υπηρεσίες είτε σε εφαρμογές που αποτελούν τον τελικό χρήστη είτε σε άλλες υπηρεσίες μέσω δημοσιευμένων διεπαφών.

Σε πολλές περιπτώσεις, οι υπηρεσίες παρέχουν έναν καλύτερο τρόπο για να εκθέτουν τις λειτουργίες μιας επιχείρησης και συνεπώς ένα άριστο τρόπο για αναπτύσσουν εφαρμογές οι οποίες αντιπροσωπεύουν εργασίες μιας επιχείρησης.

Η SOA δεν είναι απλά μια αυθαίρετη έννοια, είναι μια αρχιτεκτονική, εξαιρετικά σημαντική αυτή την εποχή εξαιτίας της αναδυόμενης χρήσης της τεχνολογίας των Διαδικτυακών Υπηρεσιών(Web Services).

Η ιδέα της SOA και η λειτουργία της φαίνονται και στο σχήμα 11:



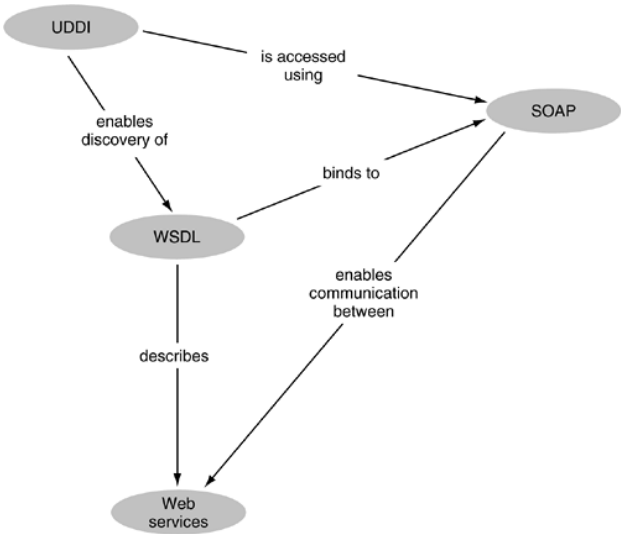
Εικόνα 11: Υπηρεσία SOA και Συνδέσεις [GCR08]

- *Υπηρεσία(Service)* — Μια λογική οντότητα. Ο ρόλος της προσδιορίζεται από μία ή περισσότερες δημοσιευμένες διεπαφές.
- *Παραγωγός Υπηρεσίας (Service Producer)* — Η οντότητα λογισμικού η οποία υλοποιεί μια προδιαγραφή της υπηρεσίας.
- *Καταναλωτής Υπηρεσίας(Service Consumer)* — Η οντότητα λογισμικού η οποία καλεί έναν παραγωγό υπηρεσίας. Παραδοσιακά αυτή η οντότητα ονομάζεται "πελάτης". Ωστόσο ένας καταναλωτής υπηρεσίας μπορεί να είναι μια εφαρμογή-τελικός χρήστης ή μια άλλη υπηρεσία. [ASK02]

Αυτή η απλή απεικόνιση μιας SOA παρουσιάζει έναν καταναλωτή υπηρεσίας να στέλνει μια αίτηση στον παραγωγό υπηρεσίας. Στη συνέχεια ο παραγωγός υπηρεσίας αποκρίνεται στον καταναλωτή υπηρεσίας.

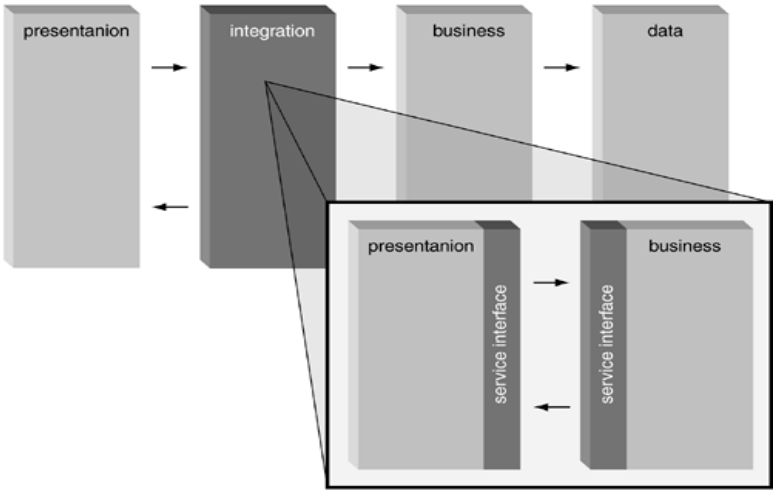
Τόσο η διεπαφή της αίτησης όσο και της απόκρισης που επακολουθεί, ορίζονται με έναν τρόπο ο οποίος είναι κατανοητός και από τον παραγωγό και από τον καταναλωτή υπηρεσίας. Ένας συνήθης μηχανισμός για την υλοποίηση αυτής της διεπαφής, είναι η χρήση των διαδικτυακών υπηρεσιών(Web Services). Στην πραγματικότητα οι διαδικτυακές υπηρεσίες είναι το βασικό συστατικό οποιασδήποτε SOA υλοποίησης.

Το κλειδί με μια υλοποίηση SOA είναι οι υπάρχουσες εφαρμογές να υποστηρίζουν μη συνεκτικές συνδεδεμένες υπηρεσίες. Η σχέση των ήδη υπαρχουσών τεχνολογιών για την υποστήριξη οποιασδήποτε SOA υλοποίησης φαίνεται στην εικόνα 12. [GCR08]



Εικόνα 12: SOA ως Σύνολο Διαδικτυακών Υπηρεσιών

Άλλες τεχνολογίες που βασίζονται στην XML όπως είναι η γλώσσα WSDL και το πρωτόκολλο SOAP συνδυάζονται για να περιγράψουν, να συνδυάσουν και να διευκολύνουν την επικοινωνία για αυτές τις διαδικτυακές υπηρεσίες. Επιπλέον, μια τεχνολογία όπως η UDDI μπορεί να χρησιμοποιηθεί σαν διανομέας υπηρεσιών(service broker), ο οποίος διευκολύνει τον εντοπισμό επιθυμητών υπηρεσιών.

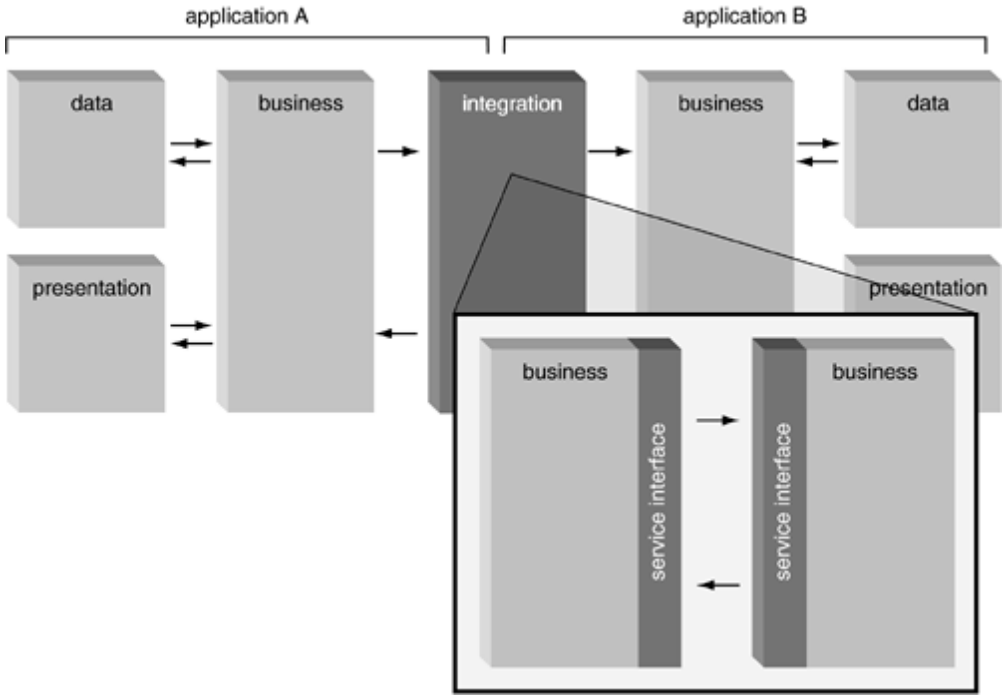


Εικόνα 13: μια λογική αναπαράσταση μιας SOA(για μια μοναδική πολυεπίπεδη αρχιτεκτονική).

Η εικόνα 13 απεικονίζει, πώς η SOA μεταβάλλει την υπάρχουσα πολυεπίπεδη αρχιτεκτονική εισάγοντας ένα λογικό επίπεδο το οποίο με τη χρήση υπαρχουσών προγραμματιστικών διεπαφών (που παρέχονται από διαδικτυακές υπηρεσίες), εγκαθιστά ένα κοινό σημείο ενσωμάτωσης.

Αυτό το επίπεδο που εισάγει η υπηρεσία αποτελεί τη βάση ενός νέου μοντέλου το οποίο μπορεί να επεκταθεί υπό τη σκοπιά μιας μοναδικής εφαρμογής για να ενοποιήσει

διαφορετικές πλατφόρμες σε ένα ανοιχτό διασυστηματικό περιβάλλον. Όταν οι διαδικτυακές υπηρεσίες χρησιμοποιούνται σε εφαρμογές που προσπελάζονται από ολόκληρο υπολογιστικό σύστημα (cross-application) (όπως στο σχήμα 14 λογική SOA ενσωμάτωσης), θέτουν τους εαυτούς ως μέρος της υποδομής μιας επιχείρησης.



Εικόνα 14: Λογική αναπαράσταση SOA ενσωμάτωσης

Καθώς η χρήση των υπηρεσιών διαφοροποιείται, η σημασία της ασφάλειας και των κλιμακούμενων απαιτήσεων διαμορφώνεται ανάλογα. Τα καλοσχεδιασμένα υπηρεσιοστραφή περιβάλλοντα θα προσπαθήσουν να υλοποιήσουν αυτές τις προκλήσεις με αυθαίρετη υποδομή, αντί για τις συνηθισμένες λύσεις που αφορούν συγκεκριμένες εφαρμογές. [Erl04]

2.5.2 Γιατί SOA;

Διαφορές της SOA από την υπάρχουσα αρχιτεκτονική

- Η SOA είναι ένα εναλλακτικό μοντέλο απέναντι στα πλέον παραδοσιακά αντικειμενοστραφή μοντέλα στα οποία οι μονάδες χαρακτηρίζονται από μεγάλο βαθμό αλληλεξάρτησης, όπως το Common Object Request Broker Architecture (CORBA) και Distributed Component Object Model (DCOM). Στη SOA μεμονωμένες υπηρεσίες μπορούν να κατασκευαστούν με αντικειμενοστραφείς μεθόδους ωστόσο η γενική αρχιτεκτονική είναι υπηρεσιοστραφής .
- Η SOA σήμερα διαφέρει στη χρήση μιας πιο γενικής, πρότυπης γλώσσας διεπαφής με όνομα XML (eXtensible Markup Language) που χρησιμοποιείται στην Γλώσσα Περιγραφής Διαδικτυακών Υπηρεσιών (WSDL) , συγκριτικά με

την παλιότερη Γλώσσα Ορισμού Διεπαφής (IDL) που συναντάται στην CORBA.

- Η SOA βασίζεται στη λογική της χρήσης υπηρεσιών που επικοινωνούν μέσω πρωτοκόλλων όπως το SOAP, επιτρέποντας μία αρχιτεκτονική με τις λιγότερες εξαρτήσεις. Σε παλαιότερα μοντέλα, τα αντικείμενα εξαρτιόνταν πολύ το ένα από το άλλο λόγω της χρήσης πολύ συγκεκριμένων προτύπων επικοινωνίας που επέβαλε ο εκάστοτε πωλητής.
- Το σύστημα της IBM με όνομα MQseries, είναι ακόμα ένα μηνυματοστραφές ενδιάμεσο λογισμικό αλλά στερείται του οργανωμένου ορισμού της ροής εργασίας που απαιτείται για τη συνεργασία των υπηρεσιών. Η SOA υποστηρίζει την σύνθεση σύνθετων υπηρεσιών από απλές προκειμένου να βρεθεί λύσει σε ένα επιχειρησιακό πρόβλημα με ένας εύκολο τρόπο τοποθέτησης και χρήσης.

2.5.3 Τα πλεονεκτήματα της SOA;

- Παρέχει ανεξαρτησία στην τοποθεσία: Οι υπηρεσίες δεν χρειάζεται πλέον να σχετίζονται με ένα συγκεκριμένο σύστημα σε ένα συγκεκριμένο δίκτυο.
- Η επικοινωνία είναι ανεξάρτητη του πρωτοκόλλου. Έτσι, η διεπαφή της προσφέρει την δυνατότητα επαναχρησιμοποίησης του κώδικα.
- Προσαρμόζεται εύκολα ενώ αποκρίνεται γρήγορα στις μεταβαλλόμενες επιχειρησιακές απαιτήσεις.
- Απλουστεύει την ανάπτυξη εφαρμογών, επιτρέπει τοποθέτηση της εφαρμογής εν ώρα εκτέλεσης και προσφέρει καλύτερη διαχείριση υπηρεσιών.
- Οι μονάδες χαρακτηρίζονται από τον ελάχιστο βαθμό εξάρτησης. Έτσι, η ενσωμάτωση γίνεται εύκολη συνθέτοντας εφαρμογές, διεργασίες, ή πιο πολύπλοκες υπηρεσίες από άλλες λιγότερο πολύπλοκες.
- Παρέχει δυνατότητα ταυτοποίησης των καταναλωτών της υπηρεσίας καθώς και όλη τη λειτουργικότητα ασφαλείας μέσω διεπαφών των υπηρεσιών σε αντίθεση με τους μηχανισμούς αλληλεξάρτησης που χρησιμοποιούν οι άλλες αρχιτεκτονικές.
- Επιτρέπει στους καταναλωτές των υπηρεσιών (π.χ. Web Services) να βρίσκουν και να συνδέονται με διαθέσιμες Υπηρεσίες δυναμικά. [Zul08]

2.6 Πλέγμα-GRID

2.6.1 Τι είναι?

Η τεχνολογία grid (πλέγμα) μπορεί να σημαίνει κάτι διαφορετικό για τον καθένα. Στη μεγάλη πλειοψηφία της παρουσιάζεται ως το ανάλογο των ηλεκτρικών δικτύων όπου οι χρήστες (ή οι ηλεκτρικές συσκευές) έχουν πρόσβαση στον ηλεκτρισμό μέσω υποδοχών στους τοίχους χωρίς να γνωρίζουν ή να ενδιαφέρονται για το πού ή πώς ο ηλεκτρισμός παράγεται στην πραγματικότητα.

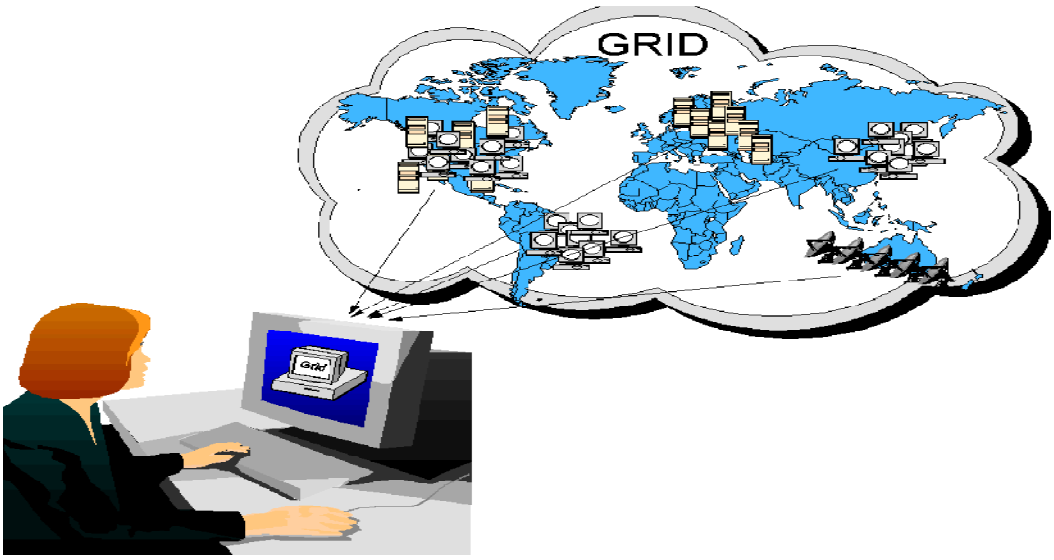
Στην τεχνολογία grid, η χρήση των υπολογιστών επεκτείνεται και ο κάθε χρήστης (ή η κάθε εφαρμογή-πελάτης) έχει πρόσβαση σε υπολογιστικούς πόρους (επεξεργαστές, αποθήκευση

δεδομένων, εφαρμογές δεδομένων και ούτω καθεξής) που χρειάζεται. Η πρόσβαση από τον χρήστη γίνεται ειτε γνωρίζει ειτε όχι πού βρίσκονται αυτοί οι πόροι ή ποιές είναι οι υπάρχουσες τεχνολογίες, το υλικό, το λειτουργικό σύστημα και ούτω καθεξής, που χρησιμοποιεί.

Η τεχνολογία grid μπορεί να θεωρηθεί ως μία διαδικασία προσπαθώντας να επιτύχουμε ένα κοινό σκοπό, ενσωματώνοντας σε αυτή την κατεύθυνση, πολλαπλές τεχνολογίες και λύσεις.

Η λεγόμενη "εικονικοποίηση", δηλαδή η χρήση κατανεμημένων υπολογιστικών τεχνολογικών υποδομών οι οποίες εξελίσσονται υπό την υποστήριξη εφαρμογών από διαφορετικούς πόρους και οργανισμούς είναι το βασικό χαρακτηριστικό αυτής της "διαδικασίας" και εφαρμόζεται στις τεχνολογίες, στις πλατφόρμες και στους οργανισμούς. Αυτού του είδους η "εικονικοποίηση" είναι εφικτή μόνο με ανοιχτά και κοινά πρότυπα ώστε οι εφαρμογές να έχουν πρόσβαση σε όλους τους απαραίτητους πόρους. Ένα τέτοιο περιβάλλον που παρέχει τη δυνατότητα για διαμοιρασμό και πρόσβαση πόρων μέσω ενός κατανεμημένου και ετερογενούς περιβάλλοντος δεν απαιτεί μόνο την τεχνολογία για την "εικονικοποίηση" των πόρων αλλά και την τεχνολογία και τα πρότυπα στους τομείς της δρομολόγησης, της ασφάλειας, της χρέωσης, της διαχείρισης συστημάτων και ούτω καθεξής.

Η τεχνολογία grid συμπεριλαμβάνει ένα αναπτυσσόμενο σύνολο ανοιχτών προτύπων για διαδικτυακές υπηρεσίες και διεπαφές οι οποίες κάνουν τις υπηρεσίες, ή τους υπολογιστικούς πόρους, διαθέσιμους στο Διαδίκτυο. Πολύ συχνά οι τεχνολογίες grid χρησιμοποιούνται σε ομογενείς συστοιχίες και μπορούν να προσθέσουν αξία σε αυτές τις συστοιχίες βοηθώντας για παράδειγμα, στη δρομολόγηση ή στην παροχή πόρων στη συστοιχία. Ο όρος grid, και οι σχετικές με αυτό τεχνολογίες, έχουν εφαρμογή σε όλο το φάσμα αυτό. Αν εστιάσουμε την προσοχή μας σε κατανεμημένες υπολογιστικές λύσεις, θα μπορούσαμε να ορίσουμε το grid ως διανεμημένοι υπολογισμοί μέσω εικονικών πόρων. Στόχος είναι να δημιουργήσουμε την ψευδαίσθηση ενός απλού αλλά μεγάλου και δυναμικού εικονικού υπολογιστή μέσα από ένα σύνολο συνδεδεμένων (και πιθανότατα ετερογενών) συστημάτων που μοιράζονται διάφορους πόρους. [JBF+05]



Εικόνα 15: GRID – Πλέγμα

2.6.2 Βασικά Χαρακτηριστικά και Πλεονεκτήματα του Grid

2.6.2.1 Βασικά Χαρακτηριστικά

Σύμφωνα με την υπάρχουσα βιβλιογραφία πάνω στο θέμα «Τεχνολογία Grid» , δέκα είναι τα βασικά χαρακτηριστικά της τεχνολογίας . Ένα πλέγμα-grid διαθέτει:

– *Μεγάλη Κλίμακα*: Το grid πρέπει να είναι σε θέση να λειτουργεί με έναν μεγάλο αριθμό πηγών ο οποίος μπορεί να κυμαίνεται από δεκάδες μέχρι εκατομμύρια. Εγείρεται λοιπόν ένα πολύ σοβαρό πρόβλημα: όσο το grid μεγαλώνει, ο βαθμός απόδοσης του μειώνεται.

– *Γεωγραφική Κατανομή*: Οι πηγές του grid μπορούν να βρίσκονται σε απομακρυσμένες τοποθεσίες.

–*Ετερογένεια*: Ένα grid διαχειρίζεται πηγές υλικού όπως και πηγές λογισμικού. Αυτές ενδέχεται να διαφέρουν πλήρως: πηγές για δεδομένα, αρχεία, στοιχεία λογισμικού, προγράμματα, αισθητήρες, επιστημονικά εργαλεία, συσκευές απεικόνισης, προσωπικά ψηφιακά μέσα οργάνωσης, υπολογιστές, υπερυπολογιστές και δίκτυα.

– *Διαχείριση πηγών*: Οι πηγές μέσα στο a grid ανήκουν σε διαφορετικούς οργανισμούς που επιτρέπουν την πρόσβαση σε άλλους οργανισμούς (π.χ. χρήστες). Έτσι οι εφαρμογές μπορούν να χρησιμοποιήσουν μη τοπικές πηγές με υψηλή απόδοση και με μειωμένο κόστος.

– *Πολλαπλή διαχείριση*: Κάθε οργανισμός μπορεί να εφαρμόζει διαφορετικές πολιτικές ασφαλείας και διαχείρισης κάτω τις οποίες επιτρέπεται η προσπέλαση και χρήση των ιδιόκτητων πόρων τους. Ως αποτέλεσμα, το μέχρι τώρα μείζον θέμα ασφαλείας των δικτύων εμπλέκεται περισσότερο μιας και τώρα θα πρέπει να συμπεριλάβει και τις εκάστοτε πολιτικές.

– *Συντονισμός των Πηγών*: Οι πόροι σε ένα grid πρέπει να συντονίζονται προκειμένου να παρέχουν συγκεντρωτικές υπολογιστικές δυνατότητες.

– *Διαφανής Πρόσβαση*: Ένα grid πρέπει να αντιμετωπίζεται ως ένας μεμονωμένος εικονικός υπολογιστής.

– *Αξιόπιστη Πρόσβαση*: Ένα grid πρέπει να εξασφαλίζει υψηλή ποιότητα στις υπηρεσίες που παρέχει. Η ανάγκη για αξιόπιστη υπηρεσία είναι βασική, μιας και οι χρήστες απαιτούν διαβεβαιώσεις για την ποιότητα των αποτελεσμάτων (Quality of Service (QoS)).

– *Συνεπής πρόσβαση*: Ένα grid πρέπει να κατασκευαστεί με συγκεκριμένες υπηρεσίες, πρωτόκολλα και διεπαφές έτσι ώστε να αποκρύπτεται η ετερογένεια των πηγών και να επιτρέπεται όμως ταυτόχρονα, η κλιμάκωση του. Χωρίς αυτά τα δεδομένα, δεν θα ήταν εφικτή η ανάπτυξη των εφαρμογών και η διεισδυτική χρήση τους.

– *Διεισδυτική Πρόσβαση*: Το grid πρέπει να παρέχει πρόσβαση στου διαθέσιμους πόρους προσαρμοζόμενο σε ένα δυναμικό περιβάλλον στο οποίο η αστοχία των πόρων είναι κάτι σύνηθες. Δεν υπαινίσσεται ότι οι πηγές βρίσκονται παντού ή ότι είναι παγκοσμίως διαθέσιμες αλλά ότι το grid θα πρέπει να προσαρμόσει την συμπεριφορά του έτσι ώστε να πετύχει την καλύτερη δυνατή απόδοση από τους διαθέσιμους πόρους. [BDG04]

2.6.2.2 Πλεονεκτήματα

Τα βασικότερα πλεονεκτήματα της τεχνολογίας Grid είναι:

- **Εκμετάλλευση υποαπασχολούμενων πόρων:**
  Ένα από τα βασικά πλεονεκτήματα της τεχνολογίας grid είναι η εκτέλεση μιας υπάρχουσας εφαρμογής σε ένα διαφορετικό μηχάνημα. Το μηχάνημα στο οποίο εκτελείται κανονικά η εφαρμογή, μπορεί να είναι απασχολημένο εξαιτίας μεγάλης δραστηριότητας. Επομένως η εν λόγω εφαρμογή θα μπορούσε να εκτελεστεί σε κάποιο ανενεργό μηχάνημα που βρίσκεται κάπου αλλού. Αν λάβουμε υπ' όψιν ότι στους περισσότερους οργανισμούς υπάρχουν μεγάλες ποσότητες πόρων που υπολειτουργούν, ότι τα περισσότερα επιτραπέζια μηχανήματα είναι απασχολημένα για λιγότερο από το 5% του χρόνου σε μια μέρα εργασίας κι ότι σε κάποιους οργανισμούς, ακόμα και οι εξυπηρετητές μπορεί να είναι ανενεργοί, είναι προφανές ότι η τεχνολογία grid μπορεί να παρέχει ένα πλαίσιο για την εκμετάλλευση αυτών των υποαπασχολούμενων πόρων και έτσι έχει τη δυνατότητα να αυξήσει ουσιαστικά τη χρήση των πόρων αυτών.

- **Η δυνατότητα παράλληλης χρήσης της κεντρικής μονάδας επεξεργασίας:**
  Η δυνατότητα για μαζική παράλληλη χρήση της κεντρικής μονάδας επεξεργασίας σε συνδυασμό με τις καθαρά επιστημονικές ανάγκες, δίνει νέα ώθηση στις βιομηχανίες όπως η βιοιατρική, η μοντελοποίηση της οικονομικής επιστήμης, η εκμετάλλευση του πετρελαίου, τα κινούμενα σχέδια και πολλές άλλες. Το κοινό χαρακτηριστικό όλων αυτών είναι ότι οι εφαρμογές έχουν γραφεί ώστε να χρησιμοποιούν αλγόριθμους οι οποίοι μπορούν να χωριστούν σε κομμάτια που εκτελούνται ανεξάρτητα. Μια εφαρμογή που εκτελείται σε περιβάλλον grid και απασχολεί εντατικά την κεντρική μονάδα επεξεργασίας μπορεί να θεωρηθεί σαν πολλές μικρότερες εφαρμογές καθεμία από τις οποίες εκτελείται σε διαφορετικό μηχάνημα στο grid. Όταν αυτές οι εφαρμογές δεν χρειάζεται να επικοινωνούν μεταξύ τους, η συνολική εργασία γίνεται όλο και πιο αποδοτική. Μια πλήρως αποδοτική εφαρμογή, για παράδειγμα, θα ολοκληρώσει την εκτέλεσή της στο ένα δέκατο του χρόνου που θα έκανε αν χρησιμοποιούσε δέκα φορές τον αριθμό των επεξεργαστών. Η δημιουργία εφαρμογών που υπακούουν σε αλγορίθμους και ο διαμοιρασμός των εργασιών σε διαφορετικά μηχάνημα μπορεί να αποτελέσει μια δύσκολη και επίπονη διαδικασία και συχνά απαιτεί μαθηματικό και προγραμματιστικό ταλέντο. Οι νέες εφαρμογές που απαιτούν μεγάλη υπολογιστική ισχύ για την εκτέλεσή τους, κατασκευάζονται σήμερα για παράλληλη επεξεργασία και γίνονται πιο εύκολα διαθέσιμες σε grid περιβάλλον.

- **Εικονικοί πόροι και εικονικοί οργανισμοί για συνεργασία:** Μια άλλη δυνατότητα της τεχνολογίας grid είναι η παροχή ενός περιβάλλοντος συνεργασίας σε ένα ευρύτερο ακροατήριο. Στο παρελθόν, η κατανεμημένη αρχιτεκτονική είχε υποσχεθεί αυτή τη συνεργασία και την πέτυχε μέχρι ενός σημείου. Η τεχνολογία grid μπορεί να επεκτείνει αυτές τις δυνατότητες της κατανεμημένης αρχιτεκτονικής σε ένα ακόμη πιο ευρύ ακροατήριο, προσφέροντας πρότυπα εξέχουσας σημασίας τα οποία διευκολύνουν ακόμα και τα πιο ετερογενή συστήματα να συνεργαστούν ώστε να αποτελέσουν ένα μεγάλο εικονικό υπολογιστικό σύστημα το οποίο προσφέρει μια ποικιλία πόρων όπως φαίνεται στο σχήμα 14. Οι χρήστες του grid μπορούν να οργανωθούν δυναμικά σε έναν αριθμό εικονικών οργανισμών, ο καθένας από τους οποίους διαθέτει διαφορετική πολιτική. Ο διαμοιρασμός ξεκινάει με αρχεία, αλλά δεν περιορίζεται σε αυτά, εφόσον περιλαμβάνει κι άλλες πηγές

όπως συσκευές, λογισμικό, υπηρεσίες, άδειες και ούτω καθεξής.

- **Πρόσβαση σε επιπλέον πόρους:** Όπως αναφέρθηκε ήδη, εκτός από τους πόρους που αφορούν την αποθήκευση και την κεντρική μονάδα επεξεργασίας, το grid μπορεί να παρέχει πρόσβαση και σε άλλους πόρους επίσης. Οι επιπλέον πόροι μπορούν να παρέχονται σε επιπλέον ποσότητες ή/και με επιπλέον δυνατότητες. Για παράδειγμα, αν ένας χρήστης πρέπει να αυξήσει το συνολικό εύρος ζώνης του στο Διαδίκτυο για να υλοποιήσει μια μηχανή αναζήτησης εξόρυξης δεδομένων, η εργασία μπορεί να χωριστεί σε μηχανήματα grid που συνδέονται ανεξάρτητα στο Διαδίκτυο. Με αυτό τον τρόπο, η συνολική δυνατότητα αναζήτησης πολλαπλασιάζεται. Επίσης κάποια μηχανήματα μπορεί να έχουν ακριβό λογισμικό το οποίο χρειάζονται οι χρήστες. Οι εργασίες των χρηστών λοιπόν μπορούν να αποστέλλονται σε τέτοια μηχανήματα εκμεταλλευόμενες πληρέστερα τις άδειες χρήσης λογισμικού.

- **Ισοζυγισμός των πόρων**: Η τεχνολογία grid διαχειρίζεται ένα μεγάλο αριθμό πόρων που αποτελείται από ξεχωριστά μηχανήματα τα οποία αποτελούν ένα μεγάλο σύστημα. Για τις εφαρμογές που υποστηρίζουν τεχνολογία grid, το grid μπορεί να προσφέρει ένα ισοζυγισμό των πόρων δρομολογώντας τις διεργασίες σε μηχανήματα όπου η χρήση είναι χαμηλή. Αυτό το χαρακτηριστικό μπορεί να είναι ανεκτίμητο κατά το χειρισμό περιστασιακού υπερβολικού φορτίου δραστηριότητας σε μια μεγάλη επιχείρηση.

- **Αξιοπιστία:** Κάποια υπολογιστικά συστήματα χρησιμοποιούν ακριβό λογισμικό προκειμένου να αυξήσουν την αξιοπιστία τους. Κατασκευάζονται με τη χρήση ολοκληρωμένων με πλεονάζοντα κυκλώματα, τα οποία στηρίζονται σε καλύτερα αποτελέσματα και περιλαμβάνουν λογική ώστε να επιτύχουν επαρκή αποκατάσταση από ένα σύνολο βλαβών του υλικού. Τα μηχανήματα επίσης χρησιμοποιούν διπλούς επεξεργαστές ώστε αν ένας από αυτούς πάθει βλάβη να μπορεί να αντικατασταθεί χωρίς να κλείσει ο άλλος. Οι παροχές ισχύος και τα συστήματα αερισμού διπλασιάζονται. Τα συστήματα λειτουργούν σε ειδικές συνθήκες ισχύος οι οποίες μπορούν να εκκινήσουν γεννήτριες αν διακοπεί το ρεύμα. Όλα αυτά συντελούν σε ένα αξιόπιστο σύστημα, αλλά με μεγάλο κόστος εξαιτίας του διπλασιασμού κάποιων στοιχείων. Στο μέλλον, θα δούμε μια επιπλέον προσέγγιση που αφορά την αξιοπιστία και βασίζεται στο λογισμικό και στο υλικό. Το grid είναι απλά η αρχή μιας τέτοιας τεχνολογίας. Τα συστήματα σε ένα grid μπορεί να είναι σχετικά φθηνά και γεωγραφικά διασκορπισμένα. Έτσι, αν υπάρξει κάποια βλάβη ισχύος ή άλλου είδους, σε μια περιοχή, τα υπόλοιπα τμήματα του grid δεν είναι πιθανό να επηρεαστούν. Το λογισμικό διαχείρισης του grid μπορεί να δρομολογήσει ξανά τις εργασίες σε άλλα μηχανήματα στο grid όταν εντοπιστεί μια βλάβη. Σε πραγματικές, κρίσιμες συνθήκες, πολλαπλά αντίγραφα σημαντικών εργασιών μπορεί να εκτελούνται σε διαφορετικά μηχανήματα στο grid. Τα αποτελέσματά τους μπορούν να ελέγχονται για οποιοδήποτε είδος αστάθειας, όπως βλάβες υπολογιστών, βλάβες δεδομένων ή παραποίηση τους.

- **Διαχείριση:** Ο σκοπός της «εικονικοποίησης» των πόρων στο grid καθώς και της διαχείρισης ετερογενών συστημάτων θα δημιουργήσει νέες ευκαιρίες για την καλύτερη διαχείριση μιας μεγαλύτερης και περισσότερο κατανεμημένης τεχνολογίας διαδικτύου. Θα ήταν πιο εύκολο να «εικονικοποιήσουμε» τις δυνατότητες και τη χρήση διευκολύνοντας τα τμήματα της τεχνολογίας διαδικτύου να ελέγχουν τις δαπάνες για τους υπολογιστικούς πόρους μέσω ενός μεγαλύτερου οργανισμού. Το grid προσφέρει διαχείριση των προτεραιοτήτων ανάμεσα σε διαφορετικά ερευνητικά προγράμματα. Στο

παρελθόν το κάθε πρόγραμμα μπορεί να ήταν υπεύθυνο για τους δικούς του διαδικτυακούς πόρους και τα σχετικά έξοδα. Συχνά αυτοί οι πόροι μπορεί να υπολειτουργούν ενώ ένα άλλο πρόγραμμα χρειάζεται περισσότερους πόρους εξαιτίας απρόσμενων γεγονότων. Με το grid είναι πιο εύκολο να διαχειριστεί κανείς τέτοιες καταστάσεις.

2.6.3 Grid Υπηρεσία και Σύγκριση με την Διαδικτυακή Υπηρεσία

2.6.3.1 Τι είναι GRID υπηρεσία ?

Μια διεπαφή υπηρεσίας που σχετίζεται με ένα πόρο grid είναι γνωστή σαν Grid υπηρεσία. Τόσο οι πόροι όσο και η κατάστασή τους ελέγχονται και διαχειρίζονται μέσω υπηρεσιών Grid σε περιβάλλον Grid. Μια υπηρεσία Grid μπορεί να απαιτεί πρόσβαση σε περισσότερους από έναν πόρους ή αντίθετα. Είναι επίσης πιθανό πολλαπλές Grid υπηρεσίες να έχουν πρόσβαση στον ίδιο πόρο ή μια Grid υπηρεσία να μπορεί να δημιουργεί ένα νέο στιγμιότυπο ενός πόρου κάθε φορά που αυτή καλείται.

Πολλαπλοί πόροι grid μπορεί να απαιτούν να αλληλεπιδρούν μεταξύ τους και να ενσωματώνονται ανάλογα με τις επιχειρησιακές απαιτήσεις. Είναι πολύ πιο πιθανό οι πόροι να φιλοξενούνται σε τεχνολογικά ετερογενές περιβάλλον. Επομένως απαιτείται ένα πλαίσιο το οποίο να αφαιρεί την υλοποίηση πόρων που σχετίζονται με συγκεκριμένο περιβάλλον, από το περιβάλλον ανταλλαγής μηνυμάτων μέσω υπηρεσιών Grid. Ένα τέτοιο πλαίσιο παρέχει η SOA.

Ως επακόλουθο η ευέλικτη SOA, η οποία στηρίζεται σε ανοιχτά πρότυπα θα έκανε πολύ πιο εύκολη την ενσωμάτωση ετερογενών πόρων και πολλαπλών επιπέδων της αρχιτεκτονικής grid. Μια τέτοια αρχιτεκτονική θα μας διευκόλυνε να επιτύχουμε διανεμημένο διαμοιρασμό πόρων σε ετερογενείς και δυναμικά εικονικούς οργανισμούς, όπως η τεχνολογία grid.

Το Παγκόσμιο Φόρουμ του Grid (Global Grid Forum - GGF) έχει υιοθετήσει μια αρχιτεκτονική βασισμένη σε αρχές SOA (Open Grid Services Architecture - OGSA) η οποία παρέχει ένα πλαίσιο για την υλοποίηση του Grid.

Όλοι οι πόροι (φυσικοί ή λογικοί) σε ένα OGSA-grid μοντέλο προτυποποιούνται σαν υπηρεσίες Grid. Αυτές οι υπηρεσίες Grid είναι κατασκευασμένες στην κορυφή μιας πολυεπίπεδης SOA τεχνολογίας βασισμένης σε διαδικτυακές υπηρεσίες. Αυτό διευκολύνει μια Grid υπηρεσία να χρησιμοποιήσει τις δυνατότητες του μοντέλου ανταλλαγής μηνυμάτων των διαδικτυακών υπηρεσιών. Πολλά από τα πρότυπα των διαδικτυακών υπηρεσιών εξελίχθηκαν ώστε να παρέχουν ασφαλείς και αξιόπιστες δοσοληψίες.

2.6.3.2 Σύγκριση Grid και Διαδικτυακών υπηρεσιών

Παρόλο που οι Grid υπηρεσίες υλοποιούνται με τη χρήση τεχνολογίας διαδικτυακών υπηρεσιών, υπάρχει μια βασική διαφορά μεταξύ μιας Grid υπηρεσίας και μιας διαδικτυακής υπηρεσίας.

Μια διαδικτυακή υπηρεσία επιτελεί το ρόλο της εύρεσης και της κλήσης σταθερών υπηρεσιών. Ένα ευέλικτο κείμενο γραμμένο σε γλώσσα περιγραφής των διαδικτυακών υπηρεσιών (Web Service Description Language – WSDL) δείχνει στην τοποθεσία που φιλοξενεί τη διαδικτυακή υπηρεσία. Μια Grid υπηρεσία επιτελεί το ρόλο της διαχείρισης ενός εικονικού πόρου και της κατάστασής του. Το grid αποτελεί ένα δυναμικό περιβάλλον. Έτσι, μια Grid υπηρεσία μπορεί να είναι μεταβλητή αντί για σταθερή.

Μια Grid υπηρεσία μπορεί να δημιουργηθεί δυναμικά και να καταστραφεί σε αντίθεση με μια διαδικτυακή υπηρεσία, η οποία συχνά θεωρείται διαθέσιμη αν το αντίστοιχο WSDL αρχείο της είναι προσβάσιμο στον πελάτη. Οι διαδικτυακές υπηρεσίες επίσης συνήθως ζουν περισσότερο χρόνο από τους πελάτες τους. Αυτό έχει σημαντικές επιρροές για το τον τρόπο που διαχειρίζονται οι Grid υπηρεσίες, που παίρνουν όνομα, ανακαλύπτονται και χρησιμοποιούνται. Το OGSA μοντέλο έχει υιοθετήσει ένα σχεδιαστικό μοτίβο για τη δημιουργία μεταβλητών Grid υπηρεσιών βασισμένες σε πρωτόκολλα grid, χρησιμοποιώντας WSDL. [JBF+05]

2.6.4 Παροχή Άδειας στο Grid

Δεδομένου ότι το περιβάλλον Grid αποτελεί μια ολοένα και περισσότερο χρησιμοποιούμενη τεχνολογία, η ένταξη σε αυτό εφαρμογών δημοφιλών στους περισσότερους χρήστες κρίνεται απαραίτητη. Οι περισσότερες ωστόσο δημοφιλείς εφαρμογές δεν διατίθενται δωρεάν και η χρήση τους απαιτεί τη σύναψη συμφωνίας ανάμεσα στην εταιρία – ιδιοκτήτρια της εφαρμογής και τον τελικό χρήστη. Προκειμένου λοιπόν να ενταχθούν στο Grid περιβάλλον, θα πρέπει να υλοποιηθεί ένας μηχανισμός αδειοδότησης τους από την αντίστοιχη εταιρία.

Δεδομένου επίσης ότι, η αρχιτεκτονική του Grid είναι υπηρεσιοστραφής (SOA), ο μηχανισμός αυτός πρέπει να αποτελεί Grid υπηρεσία, καθιστώντας τη χρήση web services για την υλοποίηση του, μονόδρομο.

Στα κεφάλαια που ακολουθούν, περιγράφεται εκτενώς ο μηχανισμός αδειοδότησης που υλοποιήθηκε ο οποίος βασίζεται στη χρήση web services. Ένας τέτοιος μηχανισμός θα μπορούσε να τροποποιηθεί κατάλληλα ώστε να εξυπηρετήσει την παραπάνω λειτουργία.

3 Ανάλυση και σχεδίαση

Στο κεφάλαιο αυτό θα γίνει παρουσίαση της νέας αρχιτεκτονικής καθώς και των λειτουργιών που αυτή προσφέρει. Θα περιγραφούν επίσης τα καινούργια χαρακτηριστικά τα οποία προσφέρει σε σύγκριση με την υπάρχουσα, καθώς και τα καινούργια εργαλεία που διατίθενται για την εκκίνηση του συστήματος.

3.1 Περιγραφή Αρχιτεκτονικής

Στην παράγραφο αυτή παρουσιάζονται οι οντότητες και τα στοιχεία που αποτελούν τόσο την παλιά όσο και τη νέα αρχιτεκτονική, καθώς και οι διαφορές τους, τόσο στην αρχιτεκτονική όσο και στη λειτουργία.

3.1.1. Σκοπός της νέας αρχιτεκτονικής

Τα μοντέλα παροχής άδειας μέσω εξυπηρετητή όπως παρουσιάστηκαν στο κεφάλαιο 2, εδραιώνουν την επικοινωνία μεταξύ πελάτη και εξυπηρετητή σε χαμηλό επίπεδο (επίπεδο μεταφοράς TCP), με αποτέλεσμα να μην είναι δυνατός ο έλεγχος αυτής της επικοινωνίας καθώς και η όποια παρέμβαση σε αυτήν, ενώ επιπλέον, ο εξυπηρετητής αφήνει εκτεθειμένες όλες τις θύρες του ώστε να έχουν πρόσβαση οι εφαρμογές – πελάτες.

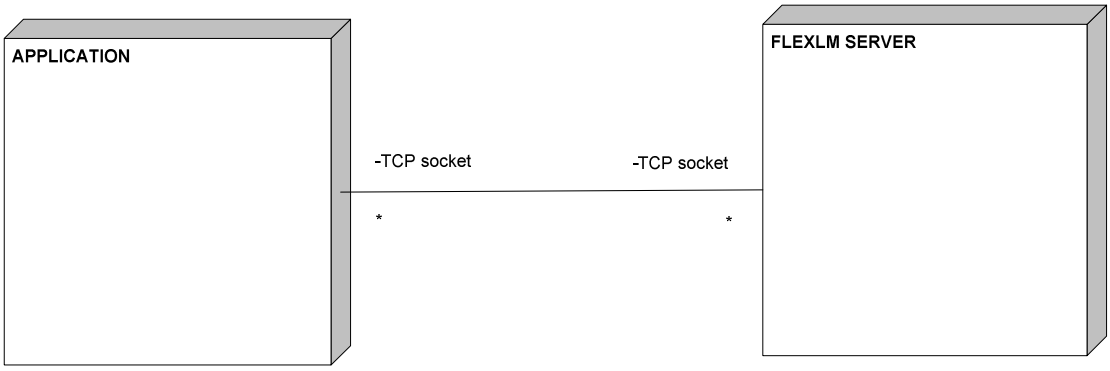
Έχοντας ως στόχο να εξαλειφθούν αυτές οι ατέλειες και κατ' επέκταση να δημιουργηθεί ένα πιο αξιόπιστο, ασφαλές, ευέλικτο και τεχνολογικά αναβαθμισμένο σύστημα παροχής άδειας, σχεδιάστηκε και υλοποιήθηκε το μοντέλο παροχής άδειας του οποίου η αρχιτεκτονική περιγράφεται στο παρόν κεφάλαιο.

Η αξιοπιστία του νέου συστήματος παροχής άδειας, εξασφαλίζεται μεταφέροντας την επικοινωνία στο επίπεδο εφαρμογής εφόσον η χορήγηση άδειας πραγματοποιείται πλέον διαδικτυακά. Στην ήδη υπάρχουσα αρχιτεκτονική προστίθενται στοιχεία (components) υλοποιημένα με τεχνολογίες αιχμής γεγονός που προσδίδει επιπλέον, ευελιξία στη διαχείριση και ευκολία στον έλεγχο της επικοινωνίας. Με την προσθήκη αυτών των στοιχείων επιτρέπεται στον πάροχο αδειών λογισμικού να αναβαθμίσει το επίπεδο ασφαλείας του εφόσον επιτρέπει πρόσβαση μόνο σε μία θύρα του.

3.1.2. Η υπάρχουσα αρχιτεκτονική ως βάση της νέας

υπάρχουσα αρχιτεκτονική

Η υπάρχουσα αρχιτεκτονική περιλαμβάνει δύο οντότητες: την εφαρμογή- πελάτη και τον FLEXlm εξυπηρετητή (σχήμα 16).

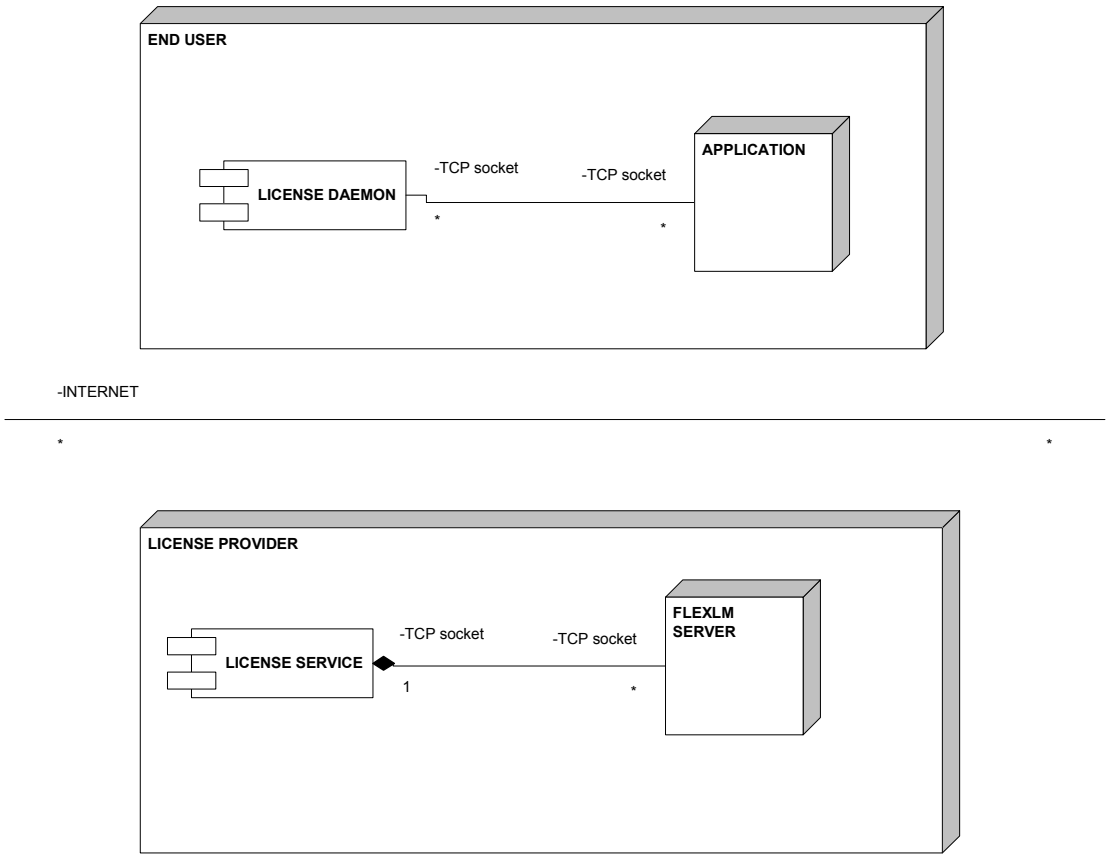


Εικόνα 16: Διάγραμμα Στοιχείων (Component Diagram) της υπάρχουσας αρχιτεκτονικής

Οι δύο οντότητες επικοινωνούν μεταξύ τους στο επίπεδο μεταφοράς TCP. Η σύνδεση πραγματοποιείται με TCP sockets.

νέα αρχιτεκτονική

Η νέα αρχιτεκτονική περιλαμβάνει δύο οντότητες: τον End User και τον License Provider (σχήμα 17).



Εικόνα 17: Διάγραμμα Στοιχείων (Component Diagram) της νέας αρχιτεκτονικής

Η οντότητα End User επικοινωνεί με την οντότητα License Provider στο επίπεδο εφαρμογής χρησιμοποιώντας το πρωτόκολλο SOAP.

3.1.2.1 Περιγραφή των οντοτήτων και των στοιχείων της νέας αρχιτεκτονικής

Όπως αναφέρθηκε και παραπάνω, οι οντότητες που περιλαμβάνει η νέα αρχιτεκτονική είναι δύο: ο End User και ο License Provider.

Η οντότητα End User αποτελείται από τα στοιχεία, application (εφαρμογή-πελάτης) και License Daemon ενώ η οντότητα License Provider αποτελείται από τα στοιχεία License Service και FLEXlm Server. Ακολουθεί αναλυτική περιγραφή των στοιχείων και των ρόλων τους κατά οντότητα.

<u>End User</u>

εφαρμογή-πελάτης:

Πρόκειται για εφαρμογή λογισμικού της οποίας η εκτέλεση πραγματοποιείται εφόσον της έχει δοθεί άδεια εκτέλεσης από τον αρμόδιο FLEXlm εξυπηρετητή. Η ταυτότητα του εξυπηρετητή (IP διεύθυνση ή hostid) είναι γνωστή στην εφαρμογή μέσω ενός αρχείου κειμένου που αυτή διαθέτει και στο οποίο αναγράφεται. Αν και ο σωστός όρος που θα

περιέγραφε την παραπάνω εφαρμογή θα απαιτούσε τον χαρακτηρισμό της ως «συμβατή με το FLEXlm», για λόγους ευκολίας θα αναφέρεται παντού στο έγγραφο ως εφαρμογή-πελάτης.

License Daemon:

Ο License Daemon είναι ένας εξυπηρετητής ο οποίος περιμένει αιτήσεις από την εφαρμογή-πελάτη σε όποιες θύρες έχει ρυθμιστεί από το διαχειριστή του. Μόλις λάβει αίτηση, περιμένει αιτήσεις από την ίδια εφαρμογή πελάτη σε μία δεύτερη θύρα. Όλες τις αιτήσεις που λαμβάνει τις προωθεί στον FLEXlm Server μέσω του License Service. Τέλος, συμμετέχει στη διαδικασία προώθησης, μέσω του License Service, των απαντήσεων του FLEXlm Server στην εφαρμογή. Η διαχείριση του License Daemon γίνεται με τρεις δυνατούς τρόπους, ώστε να δίνεται ευελιξία στο χρήστη. Επομένως υπάρχει η δυνατότητα διαχείρισης, μέσω γραμμής εντολών, μέσω παραθυριακής γραφικής διεπαφής και μέσω δυναμικών σελίδων διαδικτύου.

License Provider

License Service:

Το στοιχείο License Service είναι μια διαδικτυακή υπηρεσία η οποία καλείται από τον License Daemon ο οποίος και της προωθεί όλες τις αιτήσεις της εφαρμογής. Το License Service συνδέεται με τον FLEXlm Server και του προωθεί με τη σειρά του τις αιτήσεις που έλαβε από τον License Daemon, δηλαδή αυτές που είχαν προέλθει από την εφαρμογή. Το License Service είναι επίσης υπεύθυνο για τη λήψη των απαντήσεων του FLEXlm Server και για την προώθησή τους στον License Daemon. Η διαχείριση του License Service, γίνεται μέσω δυναμικών σελίδων διαδικτύου.

FLEXlm Server:

Εξυπηρετητής εξοπλισμένος με λογισμικό της εταιρίας Macrovision Corporation του οποίου η αποκλειστική λειτουργία είναι η αδειοδότηση εφαρμογών συμβατών με αυτόν, η παροχή δηλαδή στις εφαρμογές, των αδειών χρήσης του λογισμικού τους (βλ. Κεφάλαιο 2).

3.2 Περιγραφή Λειτουργιών

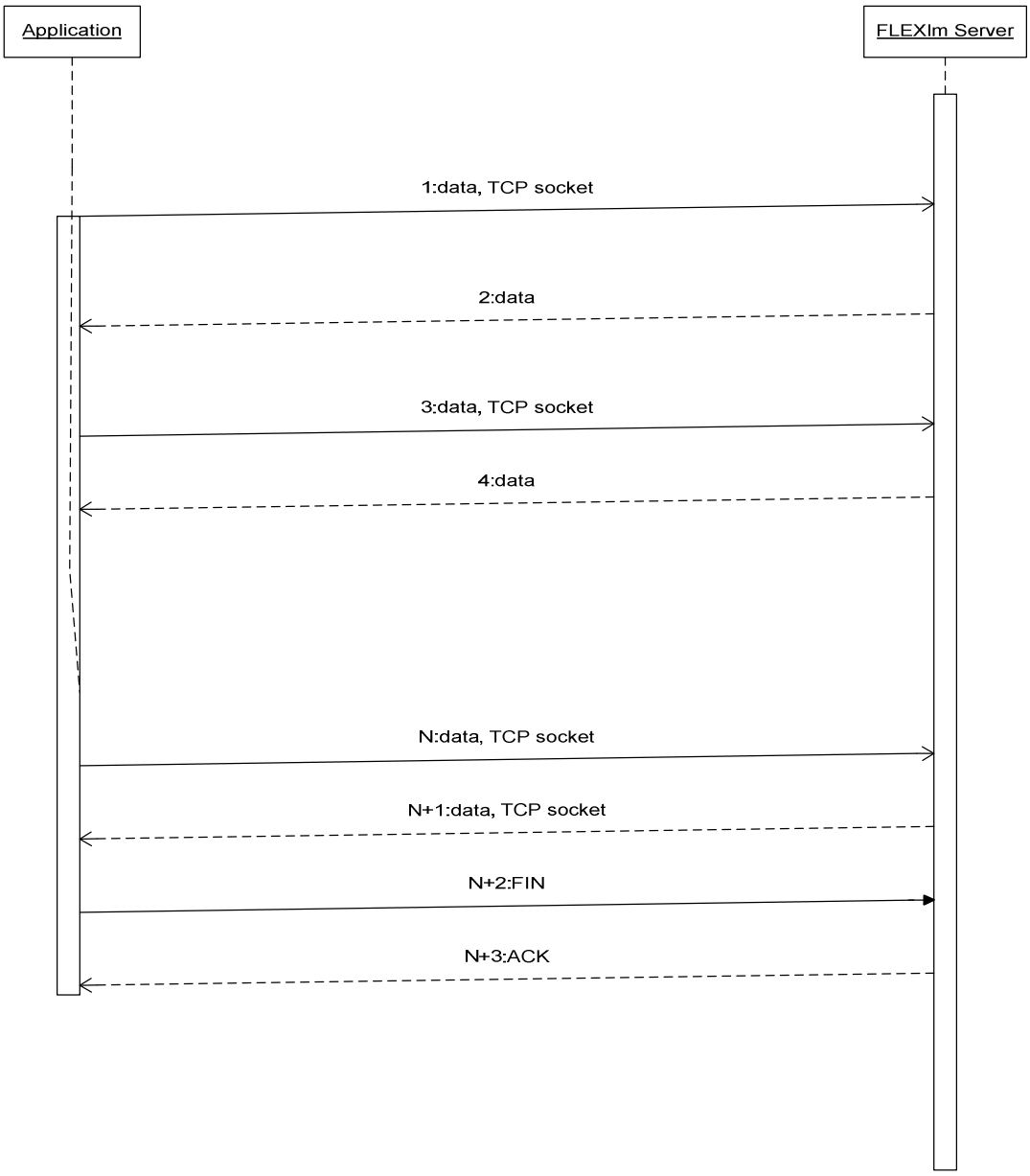
Δεδομένου ότι η νέα αρχιτεκτονική στηρίζεται στην υπάρχουσα κρίθηκε αναγκαίο να περιγραφεί αρχικά η λειτουργία της υπάρχουσας και στην συνέχεια η λειτουργία της νέας, προκειμένου ο αναγνώστης να κατανοήσει εις βάθος τους λόγους σχεδίασης.

3.2.1 Λειτουργία υπάρχουσας αρχιτεκτονικής

Η απόκτηση της άδειας χρήσης του λογισμικού στην υπάρχουσα αρχιτεκτονική, ακολουθεί την παρακάτω διαδικασία.

Η εφαρμογή πελάτης διαβάζει το license file από το οποίο καθορίζεται ο FLEXlm server με τον οποίο θα επιχειρήσει να συνδεθεί. Συγκεκριμένα, διαβάζει την IP του server και την προς σύνδεση θύρα.

Ο FLEXlm server βρίσκεται σε διαρκή αναμονή αιτήσεων προς όλες τις θύρες του. Η εφαρμογή ξεκινά την εκτέλεση της με την δημιουργία ενός socket με παραμέτρους την IP και την θύρα του server, που διάβασε από το license file. Η εφαρμογή αποστέλλει ένα μήνυμα στην θύρα. Στην συνέχεια, λαμβάνει στην απόκριση του μηνύματος την δεύτερη προς σύνδεση θύρα του server. Δημιουργεί δεύτερο socket με παραμέτρους την IP του ίδιου FLEXlm Server και την δεύτερη θύρα και παραμένει σε διαρκή επικοινωνία με τον εξυπηρετητή όσο εκτελείται η εφαρμογή, αποστέλλοντας και λαμβάνοντας μηνύματα. Η διαδικασία που περιγράφηκε απεικονίζεται στο ακολουθιακό διάγραμμα της εικόνας 18.



Εικόνα 18: Ακολουθιακό Διάγραμμα (Sequence Diagram) της υπάρχουσας αρχιτεκτονικής

3.2.2 Λειτουργία νέας αρχιτεκτονικής

Η απόκτηση της άδειας χρήσης του λογισμικού ακολουθεί παραπλήσια διαδικασία με την διαφορά ότι μεταξύ της εφαρμογής και του FLEXlm server παρεμβάλλονται δύο νέα στοιχεία: ο License Daemon και το License Service.

Η εφαρμογή πελάτης διαβάζει το license file από το οποίο καθορίζεται ο εικονικός πλέον εξυπηρετητής με τον οποίο θα συνδεθεί, δηλαδή ο License Daemon. Συγκεκριμένα, διαβάζει την IP και την προς σύνδεση θύρα.

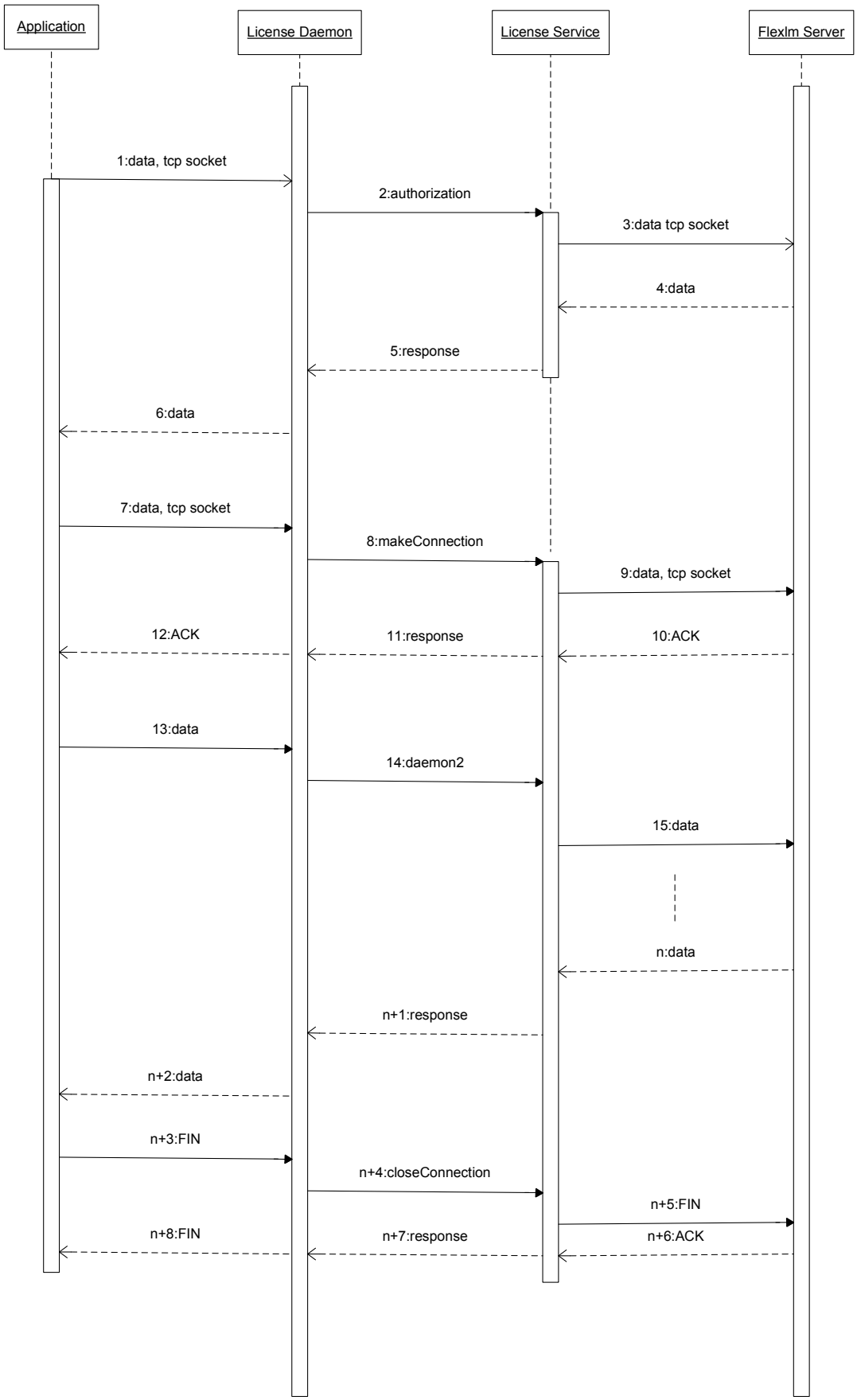
Ο License Daemon βρίσκεται σε διαρκή αναμονή αιτήσεων στις θύρες στις οποίες έχει ορίσει ο διαχειριστής του. Η εφαρμογή ξεκινά την εκτέλεση της με την δημιουργία ενός socket με παραμέτρους την IP και την θύρα του License Daemon, που διάβασε από το license file. Δημιουργούνται τα αντίστοιχα streams εισόδου / εξόδου μεταξύ των δύο στοιχείων, για την ανάγνωση και την εγγραφή αντίστοιχα των μηνυμάτων που ανταλλάσσονται. Η εφαρμογή αποστέλλει ένα μήνυμα στον License Daemon. Ο License Daemon με τη σειρά του καλεί το License Service με παράμετρο το μήνυμα και την θύρα.

Το License Service διαβάζει από ένα αρχείο τους διαθέσιμους FLEXlm Servers και επιχειρεί σύνδεση (δημιουργία socket με την προαναφερθείσα θύρα) με τον πρώτο εξ' αυτών . Εφόσον δεν είναι επιτυχής, επιλέγει τον επόμενο και ακολουθείται η ίδια διαδικασία μέχρι να επιτευχθεί σύνδεση. Στη συνέχεια, αφού δημιουργηθούν τα αντίστοιχα streams εισόδου / εξόδου, αποστέλλει το μήνυμα στον FLEXlm Server. Η διεύθυνση διαδικτύου (URL) του License Service προσδιορίζεται από το διαχειριστή του License Daemon εφάπαξ, πριν αυτός τεθεί σε αναμονή.

Ο FLEXlm Server απαντά στο μήνυμα που έλαβε, με ένα νέο μήνυμα στο οποίο περιέχεται η δεύτερη προς σύνδεση θύρα. Το μήνυμα αυτό, επιστρέφεται αυτούσιο μέσω του License Service στον License Daemon και στη συνέχεια στην εφαρμογή. Η σύνδεση τόσο μεταξύ εφαρμογής και License Daemon όσο και μεταξύ License Service και FLEXlm Server, διακόπτεται αφού πραγματοποιηθεί η ανταλλαγή των προαναφερθέντων μηνυμάτων.

Στη συνέχεια, η εφαρμογή δημιουργεί εκ νέου socket με παράμετρο την IP του License Daemon, που είναι ήδη γνωστή από το license file, και τη δεύτερη θύρα. Ο License Daemon καλεί το License Service με παράμετρο τη δεύτερη θύρα, το οποίο δημιουργεί socket με τον FLEXlm Server που έχει προσδιοριστεί προηγουμένως. Η εφαρμογή για να εκτελεστεί προϋποθέτει την ανταλλαγή αριθμού μηνυμάτων με τον FLEXlm Server στη δεύτερη θύρα κάτι που επιτυγχάνεται επαναλαμβάνοντας ισάριθμες φορές την ίδια διαδικασία που περιγράφηκε προηγουμένως για την ανταλλαγή μηνυμάτων στην πρώτη θύρα. Η μόνη διαφορά έγκειται στο ότι οι συνδέσεις μεταξύ εφαρμογής - License Daemon και License Service-FLEXlm Server παραμένουν ενεργές έως ότου διακοπεί η εκτέλεση της εφαρμογής (εκούσια διακοπή της εφαρμογής από τον χρήστη της, πρόβλημα σύνδεσης στο διαδίκτυο, διακοπή λειτουργίας του License Daemon ή του License Service ή του FLEXlm Server).

Η διαδικασία αυτή παρουσιάζεται συνοπτικά στο ακολουθιακό διάγραμμα της εικόνας 19 που ακολουθεί:



Εικόνα 19: Ακολουθιακό Διάγραμμα (Sequence Diagram) της νέας αρχιτεκτονικής

3.2.3 Λειτουργικότητα νέας αρχιτεκτονικής

Όπως αναφέρθηκε στην παράγραφο 3.2.1 η υπάρχουσα αρχιτεκτονική στερείται ασφάλειας, ευελιξίας και δυνατότητας ελέγχου της επικοινωνίας. Στον τομέα της ασφάλειας, τόσο η IP του FLEXlm Server όσοι και οι θύρες του είναι εκτεθειμένες στις εφαρμογές – πελάτες. Στον τομέα της ευελιξίας, ο FLEXlm Server επικοινωνεί με την εφαρμογή – πελάτη σε χαμηλό επίπεδο ενώ ο έλεγχος αυτής της επικοινωνίας είναι αδύνατος.

Η νέα αρχιτεκτονική παρουσιάζει, όπως αναφέρθηκε στην ενότητα 3.2.2, δύο καινούργιες οντότητες τον End User, που περιλαμβάνει τον εξυπηρετητή License Daemon και την εφαρμογή-πελάτη, και τον License Provider, που περιλαμβάνει το License Service και τον FLEXlm Server. Ο FLEXlm Server περικλείεται λοιπόν στον License Provider και δεν αποτελεί ξεχωριστή οντότητα. Ο FLEXlm Server επικοινωνεί αποκλειστικά και μόνο με το License Service με αποτέλεσμα να είναι αναγκαία γνωστή η ταυτότητα του μόνο στο License Service και σε καμία εφαρμογή-πελάτη. Με τον τρόπο αυτό εξασφαλίζεται η ασφάλεια του FLEXlm Server εφόσον η διεύθυνση IP του, δεν είναι πλέον εκτεθειμένη αφού δεν αναγράφεται στο license file της εφαρμογής. Παράλληλα, η οντότητα License Provider αφήνει εκτεθειμένη μόνο την θύρα 8080 ενώ πρόσβαση στις θύρες του FLEXlm Server έχει μόνο το License Service. Επιπλέον, ο License Provider είναι γνωστός μέσω της URL του License Service και όχι πλέον μέσω της IP του FLEXlm Server. Η επικοινωνία πραγματοποιείται πλέον στο επίπεδο εφαρμογής (διαδίκτυο), ενώ είναι δυνατός ο πλήρης έλεγχος της.

Έτσι λοιπόν γίνεται εφικτό, ένα αναβαθμισμένο και καινοτόμο σύστημα παροχής άδειας λογισμικού ικανό να αποτελέσει έναν νέο μηχανισμό που να ανταποκρίνεται στις απαιτήσεις που αναφέρθηκαν.

4 Υλοποίηση

Στο κεφάλαιο 4, παρουσιάζονται οι τεχνολογίες που χρησιμοποιήθηκαν και η περιγραφή του κώδικα που υλοποιήθηκε.

4.1 Πλατφόρμες και προγραμματιστικά εργαλεία

Για την ορθή, εγκατάσταση, λειτουργία και έλεγχο του συστήματος που υλοποιήθηκε, έγινε χρήση διαφόρων τεχνολογιών και πλατφόρμων. Οι τεχνολογίες και οι πλατφόρμες αυτές, καθώς και οι λόγοι που οδήγησαν στην επιλογή τους παρουσιάζονται διεξοδικά στις παραγράφους που ακολουθούν και είναι συνοπτικά οι εξής: Java, JSP, PHP, Axis2, Apache Tomcat Server, Apache HTTP Server.

4.1.1 JAVA

Η JAVA είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού που σχεδιάστηκε αρχικά από τη Sun Microsystems και κυκλοφόρησε το 1995 σαν ένα στοιχείο του πυρήνα της Java πλατφόρμας της Sun. Η σύνταξη της Java προέρχεται κατά ένα μεγάλο μέρος από τη γλώσσα C και τη C++ αλλά έχει πιο απλό αντικειμενοστραφές μοντέλο και λιγότερες ευκολίες χαμηλού επιπέδου. Οι Java εφαρμογές μεταγλωττίζονται τυπικά σε δυαδικό κώδικα ο οποίος μπορεί να εκτελεστεί σε οποιαδήποτε εικονική μηχανή της Java (Java Virtual Machine – JVM) ανεξάρτητα από την αρχιτεκτονική του υπολογιστή.

Ο δημιουργός της Java, James Gosling υποσχέθηκε μια γλώσσα "Write Once, Run Anywhere" (WORA), που σημαίνει ότι οποιοσδήποτε κώδικας γράφεται σε γλώσσα Java θα μπορεί να εκτελεστεί οπουδήποτε χωρίς αλλαγές. Έτσι παρέχεται σε όλες τις δημοφιλείς πλατφόρμες περιβάλλον εκτέλεσης χωρίς καθόλου κόστος. Η Java έγινε αμέσως δημοφιλής, αφού πολλοί φυλλομετρητές πλοήγησης σύντομα ενσωμάτωσαν την ικανότητα να εκτελούν ασφαλείς Java εφαρμογές (applets) σε ιστοσελίδες.

Οι αρχικοί μεταγλωττιστές της Java, οι εικονικές μηχανές και βιβλιοθήκες κλάσεων αναπτύχθηκαν από τη Sun το 1995. Από το Μάιο του 2007, σε συνεργασία με τις προδιαγραφές της Java Community Process, η Sun έκανε διαθέσιμο το μεγαλύτερο μέρος της τεχνολογίας της ως ελεύθερο λογισμικό υπό το GNU General Public License.

4.1.1.1 Πρωτεύοντες στόχοι

Υπήρχαν τέσσερις πρωτεύοντες στόχοι κατά τη δημιουργία της γλώσσας Java:

i. Έπρεπε να χρησιμοποιεί την μεθοδολογία του αντικειμενοστραφούς προγραμματισμού.
ii. Έπρεπε να επιτρέπει το ίδιο πρόγραμμα να μπορεί να εκτελεστεί σε πολλά υπολογιστικά συστήματα.
iii. Έπρεπε να περιλαμβάνει εργαλεία υποστήριξης για τη χρήση των δικτύων υπολογιστών.
iv. Έπρεπε να έχει σχεδιαστεί για να εκτελεί κώδικα από απομακρυσμένες πηγές με ασφάλεια.
v. Έπρεπε να είναι εύκολη στη χρήση επιλέγοντας ό,τι θεωρούνταν καλό στοιχείο των άλλων αντικειμενοστραφών γλωσσών.

4.1.1.2 Γιατί Java ?

Όλες οι προδιαγραφές με τις οποίες σχεδιάστηκε η Java, υποδηλώνουν τα πλεονεκτήματα που απορρέουν από τη χρήση της καθώς και τους λόγους για τους οποίους επιλέχθηκε ως γλώσσα υλοποίησης κατά την παρούσα ερευνητική εργασία.

Πιο αναλυτικά, εξαιτίας της κατασκευής της, ως αντικειμενοστραφούς γλώσσας παρέχει απλότητα στο σχεδιασμό, διαχωρισμό του κώδικα σε ξεχωριστές μονάδες, ευκολία στην πραγματοποίηση μικρών αλλαγών στον κώδικα χωρίς να επηρεαστεί η λειτουργία του, δυνατότητα επέκτασης του κώδικα προσθέτοντας νέες λειτουργίες χωρίς ιδιαίτερη δυσκολία και χωρίς μετατροπή του ήδη υπάρχοντος, ευκολία στη διόρθωση λαθών και τέλος δυνατότητα χρήσης των αντικειμένων σε διαφορετικά προγράμματα. Όλα αυτά τα χαρακτηριστικά παρείχαν τη δυνατότητα δημιουργίας ευέλικτου και επεκτάσιμου κώδικα.

Η ανεξαρτησία πλατφόρμας, σημαίνει ότι τα προγράμματα που είναι γραμμένα σε Java πρέπει να εκτελούνται με τον ίδιο τρόπο σε οποιαδήποτε πλατφόρμα υλικού ή λειτουργικού συστήματος. Επομένως, εξασφαλίζεται με αυτό τον τρόπο ότι η υλοποίηση που περιγράφεται εκτενώς στο έγγραφο, θα έχει εφαρμογή και θα μπορεί να εκτελεστεί σε οποιοδήποτε μηχάνημα ανεξαρτήτως λειτουργικού συστήματος καθώς και θα επιτρέπει τη συνεργασία διαφορετικών μηχανημάτων μεταξύ τους χωρίς επιπλοκές κάτι που είναι ιδιαίτερα σημαντικό αφού όπως φαίνεται και από την περιγραφή της αρχιτεκτονικής του συστήματος είναι απαραίτητη αυτή η συνεργασία.

Επίσης, η Java, από την αρχή σχεδιάστηκε για εφαρμογές που θα λειτουργούσαν μέσω του διαδικτύου. Εξαιτίας αυτού έχει καταστεί ως μία από τις βασικότερες γλώσσες προγραμματισμού για την ανάπτυξη εφαρμογών και υπηρεσιών διαδικτύου (web services). Έχει πολλά πλεονεκτήματα όσον αφορά την γρήγορη και ασφαλή διαδικτυακή επικοινωνία, κάτι απολύτως αναγκαίο στην υπηρεσία παροχής άδειας όπως αυτή περιγράφηκε.

Η έκδοση που χρησιμοποιήθηκε είναι η Java 2 Platform, Standard Edition 6.0 Update 2,(http://java.sun.com/javaee/downloads/index.jsp) και η πλατφόρμα ανάπτυξης ήταν το Eclipse, ελεύθερο λειτουργικό που διατίθεται στο διαδίκτυο (http://www.eclipse.org/).

Εναλλακτικά της Java θα μπορούσε να χρησιμοποιηθεί κάποια άλλη γλώσσα προγραμματισμού όπως C++ ή ακόμα και άλλη πλατφόρμα ανάπτυξης όπως .NET . Η πρώτη εναλλακτική λύση της C++ βρίσκεται πολύ κοντά σε λογική υλοποίησης με την JAVA, αφού είναι και αυτή αντικειμενοστραφής γλώσσα αλλά δεν είναι τόσο ευέλικτη και προσανατολισμένη σε δικτυακές εφαρμογές. Από την άλλη η επιλογή μιας πλατφόρμας ανάπτυξης όπως το .NET θα μας διευκόλυνε σε κάποια σημεία ανάπτυξης αφού έχει σχεδιαστεί για δικτυακές εφαρμογές, όμως μας περιορίζει αρκετά στα υπολογιστικά συστήματα που υποστηρίζει αφού είναι προσανατολισμένη κυρίως για Windows περιβάλλοντα.

4.1.2 Γραφικό Περιβάλλον (User Interface)

Για τη διαχείριση της εφαρμογής παροχής άδειας, τη λειτουργία της και την παρατήρηση των αποτελεσμάτων, είναι απαραίτητη η ύπαρξη γραφικών διεπαφών.

Επιλέχθηκε να παρέχεται η δυνατότητα διαχείρισης της εφαρμογής του License Daemon με τρεις διαφορετικούς τρόπους όπως περιγράφονται στο κεφάλαιο 3. Η παραθυριακή γραφική διεπαφή υλοποιήθηκε σε γλώσσα Java για τους λόγους που περιγράφηκαν προηγουμένως, ενώ οι δυναμικές σελίδες διαδικτύου, υλοποιήθηκαν σε τεχνολογία JSP.

Για τη διαχείριση του License Service επιλέχθηκε η δημιουργία δυναμικών σελίδων βασισμένων σε τεχνολογία PHP, ώστε να είναι εύκολη η απομακρυσμένη πρόσβαση από οποιονδήποτε στο δίκτυο με την κατάλληλη πάντα ταυτοποίηση (authentication).

4.1.2.1 Java Server Pages (JSP)

Η τεχνολογία των Java Server Pages (JSP) επιτρέπει στους σχεδιαστές ιστοσελίδων τη γρήγορη ανάπτυξη δυναμικών ιστοσελίδων, που συντηρούνται εύκολα και είναι πλούσιες σε περιεχόμενο. Ως μέλος της οικογένειας της τεχνολογίας Java, η τεχνολογία JSP διευκολύνει τη γρήγορη ανάπτυξη εφαρμογών διαδικτύου που είναι ανεξάρτητες πλατφόρμας. Η τεχνολογία JSP ξεχωρίζει τη διεπαφή χρήστη από τη δημιουργία του περιεχομένου, διευκολύνοντας τους σχεδιαστές να αλλάξουν πλήρως την εμφάνιση της σελίδας, χωρίς να αλλάζουν το δυναμικό υπόβαθρο. Όλα τα πλεονεκτήματα της Java καθώς και η αυτόματη μεταγλώττιση είναι αυτά που κάνουν την τεχνολογία JSP δημοφιλή και μία από τις επιλογές για την ανάπτυξη των δυναμικών σελίδων της εφαρμογής.

4.1.2.2 PHP

Η PHP είναι μια από τις πιο δημοφιλείς γλώσσες σχεδιασμού ιστοσελίδων από τη μεριά του εξυπηρετητή (server side), ειδικά σχεδιασμένη για ανάπτυξη δυναμικών ιστοσελίδων και η οποία μπορεί να ενσωματωθεί στην HTML.

Η PHP προσφέρει πολλά πλεονεκτήματα, αφού είναι σταθερή, γρήγορη, ασφαλής, εύκολη στη χρήση και ανοιχτού κώδικα. Η PHP δεν χρησιμοποιεί πολλούς από τους πόρους του

συστήματος επομένως εκτελείται γρήγορα χωρίς να καθυστερεί τις άλλες διαδικασίες και αποδίδει καλά με άλλο λογισμικό.

4.1.3 Δικτυακός Εξυπηρετητής - Web Server

Ένας Web Server είναι ένα πρόγραμμα που δέχεται αιτήσεις σε πρωτόκολλο HTTP (HTTP Requests) από τους πελάτες, γνωστούς ως φυλλομετρητές ιστού και τους εξυπηρετεί στέλνοντας HTTP αποκρίσεις μαζί με περιεχόμενα δεδομένων.

4.1.3.1 Tomcat

Ο Apache Tomcat είναι ένας δικτυακός υποδοχέας, ή εξυπηρετητής εφαρμογών που αναπτύχθηκε από την Apache Software Foundation (ASF). Ο Tomcat υλοποιεί τις προδιαγραφές του Java Servlet και των Java Server Pages (JSP) όπως αυτές προσδιορίζονται από τη Sun Microsystems, παρέχοντας ένα περιβάλλον για εκτέλεση Java κώδικα σε συνεργασία με έναν δικτυακό εξυπηρετητή (web server). Διαθέτει εργαλεία για τη ρύθμιση των παραμέτρων του και για τη διαχείριση του, αλλά μπορεί επίσης να ρυθμιστεί με επεξεργασία των αρχείων ρυθμίσεων τα οποία συνήθως έχουν XML μορφή. Ο Tomcat περιλαμβάνει τον δικό του εσωτερικό HTTP εξυπηρετητή.

Θεωρείται από τους πλέον αξιόπιστους Web Servers και χρησιμοποιείται ευρέως στην ανάπτυξη δικτυακών υπηρεσιών. Στην παρούσα εφαρμογή χρησιμοποιήθηκε η έκδοση 6.0.16 η οποία παρέχεται ελεύθερα στο διαδίκτυο μέσω του συνδέσμου http://tomcat.apache.org.

4.1.3.2 AXIS

Ο Axis2 είναι ένας πυρήνας διαδικτυακών υπηρεσιών ο οποίος τηρεί τις βασικές προϋποθέσεις για την λειτουργία τους.

Μια καινούργια αρχιτεκτονική για το Axis2 παρουσιάστηκε τον Αύγουστο του 2004, στο Colombo της Sri Lanka. Η καινούργια αρχιτεκτονική στην οποία βασίζεται το Axis2, είναι πιο ευέλικτη, πιο αποδοτική και έχει εναρμονιστεί λειτουργικά πλήρως σε σχέση με την αρχιτεκτονική του Axis1.x. Ορισμένα εργαλεία του Axis 1.x, όπως π.χ. χειριστές (handlers) κτλ., διατηρήθηκαν και στην καινούργια αρχιτεκτονική.

Ο Apache Axis2 δεν υποστηρίζει μόνο το SOAP 1.1 και SOAP 1.2, αλλά παρέχει επίσης υποστήριξη για το ευρέως δημοφιλές REST style of Web services. Η ίδια υλοποίηση μπορεί να παρέχει μια διαπροσωπία τύπου WS-* όπως και τύπου REST/POX ταυτόχρονα.

Ο Apache Axis2 είναι πρότυπος , πιο αποτελεσματικός, και χρησιμοποιεί περισσότερο την γλώσσα XML απ' όσο η προηγούμενη έκδοση. Είναι προσεκτικά σχεδιασμένος για να υποστηρίζει την εύκολη ενσωμάτωση πρόσθετων μονάδων, οι οποίες συνεισφέρουν στην ασφάλεια και στην αξιοπιστία του συστήματος. Τα πρόσθετα αυτά είναι διαθέσιμα ή υπό κατασκευή και περιλαμβάνουν:

- WS-Αξιόπιστη Ανταλλαγή Μηνυμάτων - Υπό τη υποστήριξη του Apache Sandesha2
- WS-Συντονισμός και WS-Ατομική Δοσοληψία - Υπό τη υποστήριξη του Apache Kandula2
- WS-Ασφάλεια - Υπό τη υποστήριξη του Apache Rampart
- WS-Διευθυνσιοδότηση - Μονάδα που περιλαμβάνεται στον πυρήνα του Axis2

Ο Apache Axis2 έχει κατασκευαστεί πάνω στον Apache AXIOM, ένα μοντέλο υψηλής απόδοσης βασισμένο στην XML.

Ο Axis2 συνοδεύεται από πολλά νέα χαρακτηριστικά, από ενισχυμένες λειτουργίες και από ειδικές επιχειρηματικές υλοποιήσεις. Πρόκειται για χαρακτηριστικά που τον καθιστούν καλύτερο σε σχέση με τους υπόλοιπους πυρήνες όπως φαίνεται και στον πίνακα 2:

i. Ταχύτητα – Ο Axis2 χρησιμοποιεί το δικό του μοντέλο αντικειμένου και το δικό του StAX (Streaming API for XML) αναλυτή κειμένου με αποτέλεσμα να παρέχει στους χρήστες του, μεγαλύτερη ταχύτητα από προηγούμενες εκδόσεις του Apache Axis.
ii. Μικρό μέγεθος απαιτούμενης μνήμης- Ο Axis2 σχεδιάστηκε από κάτω προς τα πάνω προσπαθώντας διαρκώς να κρατήσει τις απαιτήσεις σε μνήμη σε χαμηλά επίπεδα.
iii. AXIOM – Ο Axis2 συνοδεύεται από το δικό του ελαφρύ μοντέλο αντικειμένου, AXIOM, για επεξεργασία μηνυμάτων , το οποίο είναι επεκτάσιμο, έχει υψηλή απόδοση και είναι εύχρηστο στον προγραμματιστή.
iv. Εύκολη και Άμεσα Αναγνωρίσιμη Τοποθέτηση – Ο Axis2 είναι εφοδιασμένος με την δυνατότητα τοποθέτησης διαδικτυακών υπηρεσιών και χειριστών ενώ το σύστημα είναι σε λειτουργία. Με άλλα λόγια, νέες διαδικτυακές υπηρεσίες μπορούν να προστεθούν στο σύστημα, χωρίς να διακόψει την λειτουργία του ο εξυπηρετητής. Αρκεί απλά, η τοποθέτηση του αρχείου της επιθυμητής διαδικτυακής υπηρεσίας στον φάκελο υπηρεσιών και το μοντέλο εγκατάστασης θα εγκαταστήσει αυτόματα την υπηρεσία και την κάνει διαθέσιμη προς χρήση.
v. Ασύγχρονες Διαδικτυακές Υπηρεσίες – Ο Axis2 πλέον υποστηρίζει ασύγχρονες διαδικτυακές υπηρεσίες και ασύγχρονη κλήση διαδικτυακών υπηρεσιών χρησιμοποιώντας ασύγχρονες εφαρμογές-πελάτες και ασύγχρονες μεταφορές.
vi. Υποστήριξη MEP- Ο Axis2 πλέον παρέχει την ευελιξία για υποστήριξη των Προτύπων Ανταλλαγής Μηνυμάτων (Message Exchange Patterns (MEPs)) με εσωτερική υποστήριξη για τα βασικά MEPs όπως καθορίζονται στο WSDL 2.0.
vii. Ευελιξία – Η αρχιτεκτονική του Axis2 δίνει στον προγραμματιστή τη πλήρη ελευθερία να εισάγει επεκτάσεις στο σύστημα για τυπική επεξεργασία των επικεφαλίδων, για τη διαχείριση του συστήματος, και για οτιδήποτε άλλο μπορεί να φανταστεί κανείς.
viii. Σταθερότητα – Ο Axis2 προσδιορίζει ένα σύνολο από δημοσιευμένες διαπροσωπίες οι οποίες αλλάζουν σχετικά αργά, συγκριτικά με τα υπόλοιπα συστατικά του Axis.
ix. Ανάπτυξη προσανατολισμένη στα στοιχεία - Μπορεί κανείς εύκολα να ορίσει επαναχρησιμοποιούμενα δίκτυα των χειριστών προκειμένου να υλοποιήσει γνωστές τεχνικές επεξεργασίας των εφαρμογών του ή να τις μοιραστεί με τους συνεργάτες του.
x. Πλαίσιο Μεταφορών – Υπάρχει πλέον μια ξεκάθαρη ιδέα για την ενοποίηση και την χρήση των μεταφορών μεταξύ πρωτοκόλλων (π.χ.

αποστολείς και ακροατές για SOAP πάνω σε διάφορα πρωτόκολλα όπως SMTP, FTP, μηνυματοστραφές ενδιάμεσο λογισμικό κτλ), και ο πυρήνας είναι πλήρως ανεξάρτητος των μεταφορών.

xi. Υποστήριξη WSDL - Το Axis2 υποστηρίζει τις εκδόσεις 1.1 και 2.0, της Web Service Description Language, η οποία επιτρέπει στο χρήστη να έχει πρόσβαση σε απομακρυσμένες υπηρεσίες και επίσης να εξάγει αυτόματα αναγνώσιμες από μηχανές, περιγραφές της διαδικτυακής υπηρεσίας.

xii. Πρόσθετα – Αρκετές προδιαγραφές των Διαδικτυακών Υπηρεσιών ενσωματώθηκαν συμπεριλαμβανομένου του WSS4J για ασφάλεια (Apache Rampart), του Sandesha για αξιόπιστη ανταλλαγή μηνυμάτων, του Kandula το οποίο είναι ενθυλάκωση των WS-Συντονισμός, WS-ΑτομικήΔοσοληψία and WS-ΕπιχειρηματικήΔραστηριότητα.

xiii. Δυνατότητα Σύνθεσης και Επεκτασιμότητα:- Υπάρχουν πολλές μονάδες που αυξάνουν την δυνατότητα για σύνθεση και επέκταση. Οι μονάδες αυτές, υποστηρίζουν την δυνατότητα σύνθεσης καθώς και τις καινούργιες προδιαγραφές WS -*, με έναν απλό και ξεκάθαρο τρόπο. Η τοποθέτηση τους ωστόσο δεν είναι άμεσα αναγνωρίσιμη, εφόσον αλλάζουν την όλη συμπεριφορά του συστήματος. [SC08]

Σύγκριση με τεχνολογίες ίδιου σκοπού

Παρακάτω (πίνακας 2) παρατίθενται διάφοροι πυρήνες Διαδικτυακών Υπηρεσιών καθώς και τα αποτελέσματα σύγκρισης τους, που οδήγησαν στην επιλογή του Axis2.

- Axis 1.x.
- Celtic
- Glue
- JBossWS
- Xfire
- Metro@GlassFish
- OracleAS
- CXF (Είναι ο συνδυασμός δύο προηγούμενων εργασιών, του Celtix και του Xfire.)

Το axis2 περιλαμβάνει όλα τα βασικά χαρακτηριστικά ενός πυρήνα Διαδικτυακών Υπηρεσιών. Ωστόσο τα χαρακτηριστικά που τον ξεχώρισαν από τους υπόλοιπους πυρήνες και τον κατέστησαν κατάλληλο για την ζητούμενη υλοποίηση είναι τα εξής [SC08]:

Χαρακτηριστικό	**Axis 1.x**	**Axis 2**	**Celtix**	**Glue**	**JBossWS**	**XFire**	**GlassFish**	**OracleAS**
Δημιουργία Κώδικα με συγκεκριμένη Πολιτική		***X***						
WS-Επιχειρηματική Δραστηριότητα		***X***						

WS-Μεταφορά		***X***						
Υποστήριξη WSDL 2.0		***X***						

Πίνακας 2: Πλεονεκτήματα του Axis2

4.1.3.3 Apache HTTP Server

Ο Apache HTTP Server, ο οποίος συχνά αποκαλείται απλά Apache, είναι ένας web server γνωστός για το ρόλο κλειδί που έπαιξε στην αρχική ανάπτυξη του παγκόσμιου ιστού. Ο Apache ήταν η πρώτη βιώσιμη εναλλακτική στον Web Server Netscape Communications Corporation (γνωστός σήμερα ως Sun Java System Web Server), και έχει εξελιχθεί ώστε να ανταγωνιστεί άλλους Web Servers βασισμένους σε Unix όσον αφορά τη λειτουργικότητα και τις επιδόσεις.

Ο Apache αναπτύχθηκε και συντηρήθηκε από μια ανοιχτή κοινωνία αναλυτών υπό το πρίσμα της Apache Software Foundation. Η εφαρμογή είναι διαθέσιμη για μια ποικιλία λειτουργικών συστημάτων, όπως Unix, FreeBSD, Linux, Solaris, Novell NetWare, Mac OS X και Microsoft Windows. Ο Apache χαρακτηρίζεται ως ελεύθερο λογισμικό και ως λογισμικό ανοιχτού κώδικα.

4.2 Λεπτομέρειες υλοποίησης

Η υλοποίηση του μηχανισμού παροχής άδειας από έναν FLEXlm Server σε μια εφαρμογή μέσω μιας διαδικτυακής υπηρεσίας, όπως έχει ήδη αναφερθεί αποτελείται από διάφορα στοιχεία που τον συνθέτουν στο σύνολο του, χρησιμοποιώντας διαφορετικές τεχνολογίες, με σκοπό την προσέγγιση όλων των απαιτούμενων προδιαγραφών, την επιτέλεση των απαραίτητων λειτουργιών και τη διευκόλυνση του εκάστοτε χρήστη/διαχειριστή στην χρήση και επεξεργασία του.

Συνεπώς ο μηχανισμός αποτελείται από τα εξής στοιχεία:

i. License Components

- License Daemon: πρόγραμμα υλοποιημένο σε γλώσσα java και εγκατεστημένο σε κάποιο μηχάνημα στο υποδίκτυο της εφαρμογής- πελάτη. Η υπηρεσία αυτή αποτελεί έναν εξυπηρετητή, ο οποίος βρίσκεται σε αναμονή των αιτήσεων των εφαρμογών που βρίσκονται εγκατεστημένες σε οποιονδήποτε υπολογιστή στο δίκτυο αρκεί να διαθέτουν έγκυρο license file (όπως αυτό περιγράφεται στο κεφάλαιο 2) το οποίο να περιλαμβάνει στις SERVER γραμμές του, την IP του μηχανήματος που είναι εγκατεστημένος ο

License Daemon. Στη συνέχεια ο License Daemon επιμελείται την προώθηση των αιτήσεων από τις εφαρμογές στον FLEXlm Server μέσω του License Service ώστε να τους παρέχει την άδεια χρήσης, εφόσον την δικαιούνται. Ο License Daemon έχει σχεδιαστεί ώστε να επιτρέπει την πρόσβαση από οποιονδήποτε υπολογιστή (με οποιαδήποτε IP), είναι δηλαδή όπως θα λέγαμε "allow-all". Προκειμένου να λειτουργεί σωστά ο License Daemon και να εξυπηρετεί τις έγκυρες εφαρμογές-πελάτες, πρέπει όλες οι θύρες του μηχανήματος να είναι ανοιχτές και να μην εμποδίζεται η πρόσβαση σε αυτές από κάποιο τείχος προστασίας. Έτσι ο FLEXlm, μπορεί να διαθέτει τείχος προστασίας το οποίο επιτρέπει συνδέσεις μόνο από τις γνωστές προς αυτόν IP των License Service που γνωρίζει, ώστε να προστατεύεται από μη επιτρεπτή πρόσβαση, ενώ η IP του δεν είναι γνωστή στην εφαρμογή - πελάτη, αφού το μόνο που χρειάζεται να γνωρίζει αυτή, είναι η IP ή το hostid του License Daemon.

- License Service: web service υλοποιημένο σε γλώσσα java και εγκατεστημένο σε κάποιο μηχάνημα στο υποδίκτυο του FLEXlm Server. Η υπηρεσία αυτή καλείται από τον License Daemon, κάθε φορά που αυτός δέχεται αίτηση από μια εφαρμογή-πελάτη και την προωθεί στον FLEXlm Server προκειμένου αυτός να παράσχει την άδεια για την εκτέλεση της εφαρμογής. Το μηχάνημα του License Service μπορεί να διαθέτει τείχος προστασίας το οποίο επιτρέπει την σύνδεση μόνο στη θύρα 8080 και πρόσβαση από οποιαδήποτε IP, μηχανήματος επιθυμεί ο διαχειριστής του. Τη διαδικτυακή διεύθυνση του License Service τη γνωρίζει μόνο ο License Daemon (ο διαχειριστής του).

Τόσο ο License Daemon όσο και το License Service επικοινωνούν με την εφαρμογή-πελάτη και με το FLEXlm αντίστοιχα καθ' όλη τη διάρκεια της εκτέλεσης της εφαρμογής.

ii. Γραφικές Διεπαφές (Graphic Interfaces)

- License Daemon Interface:
  i. Παραθυριακή γραφική διεπαφή χρήστη, υλοποιημένη σε γλώσσα java, για τη διαχείριση του License Daemon. Μέσω αυτής της γραφικής διεπαφής, ο διαχειριστής έχει τη δυνατότητα, να ρυθμίσει, να εκκινήσει και να τερματίσει τη λειτουργία του License Daemon, καθώς και να παρατηρεί την εξέλιξη της διαδικασίας παροχής άδειας, όπως περιγράφεται στο κεφάλαιο 5.
  ii. Δυναμικές σελίδες διαδικτύου υλοποιημένες με τεχνολογία JSP (Java Server Page), για τη διαχείριση του License Daemon. Μέσω των σελίδων αυτών ο διαχειριστής μπορεί να εκκινήσει τη λειτουργία του License Daemon και να θέσει τις απαραίτητες παραμέτρους όπως αυτές παρουσιάζονται στο κεφάλαιο 5.

- License Service Interface: δυναμικές σελίδες διαδικτύου υλοποιημένες σε PHP κώδικα, για τη διαχείριση του License Service. Μέσω αυτής της γραφικής διεπαφής, ο διαχειριστής έχει τη δυνατότητα, να επιτηρεί και να τροποποιεί το αρχείο που περιέχει τους προς σύνδεση FLEXlm Servers, να ελέγχει τη χρήση του/των FLEXlm Servers έχοντας πρόσβαση στο αρχείο καταγραφής του License Service και τέλος να διαχειρίζεται κάποια από τα εργαλεία που παρέχονται από τον FLEXlm Server (lmutilities).

4.2.1 Υλοποίηση συστήματος – Συγγραφή κώδικα

Το σύστημα παροχής άδειας όπως υλοποιήθηκε περιλαμβάνει δύο στοιχεία, τον License Daemon και το License Service. Ο σχεδιασμός αυτός παρείχε τη δυνατότητα να εξασφαλιστεί η απαραίτητη ασφάλεια του FLEXlm Server από ανεπιθύμητη πρόσβαση αφού, η IP διεύθυνση του FLEXlm διατηρείται γνωστή μόνο στο License Service και ο License Provider αφήνει εκτεθειμένη μόνο μία θύρα του. Επίσης, στόχος ήταν η διαδικασία παροχής άδειας να μην εκτελείται πλέον στο επίπεδο μεταφοράς TCP, αλλά στο επίπεδο εφαρμογής, κάτι που προσφέρει ασφαλέστερη και πιο αξιόπιστη επικοινωνία. Λαμβάνοντας υπ' όψιν επιπλέον και το στόχο η υπηρεσία παροχής άδειας να παρέχεται με μια απλή κλήση εξ' αποστάσεως μιας διαδικτυακής υπηρεσίας είναι εμφανές πώς επιλέχθηκαν οι τεχνολογίες και οι πλατφόρμες που παρουσιάστηκαν νωρίτερα καθώς και ο σχεδιασμός του συστήματος, όσον αφορά το προγραμματιστικό μέρος, ο οποίος παρουσιάζεται συνοπτικά στη συνέχεια.

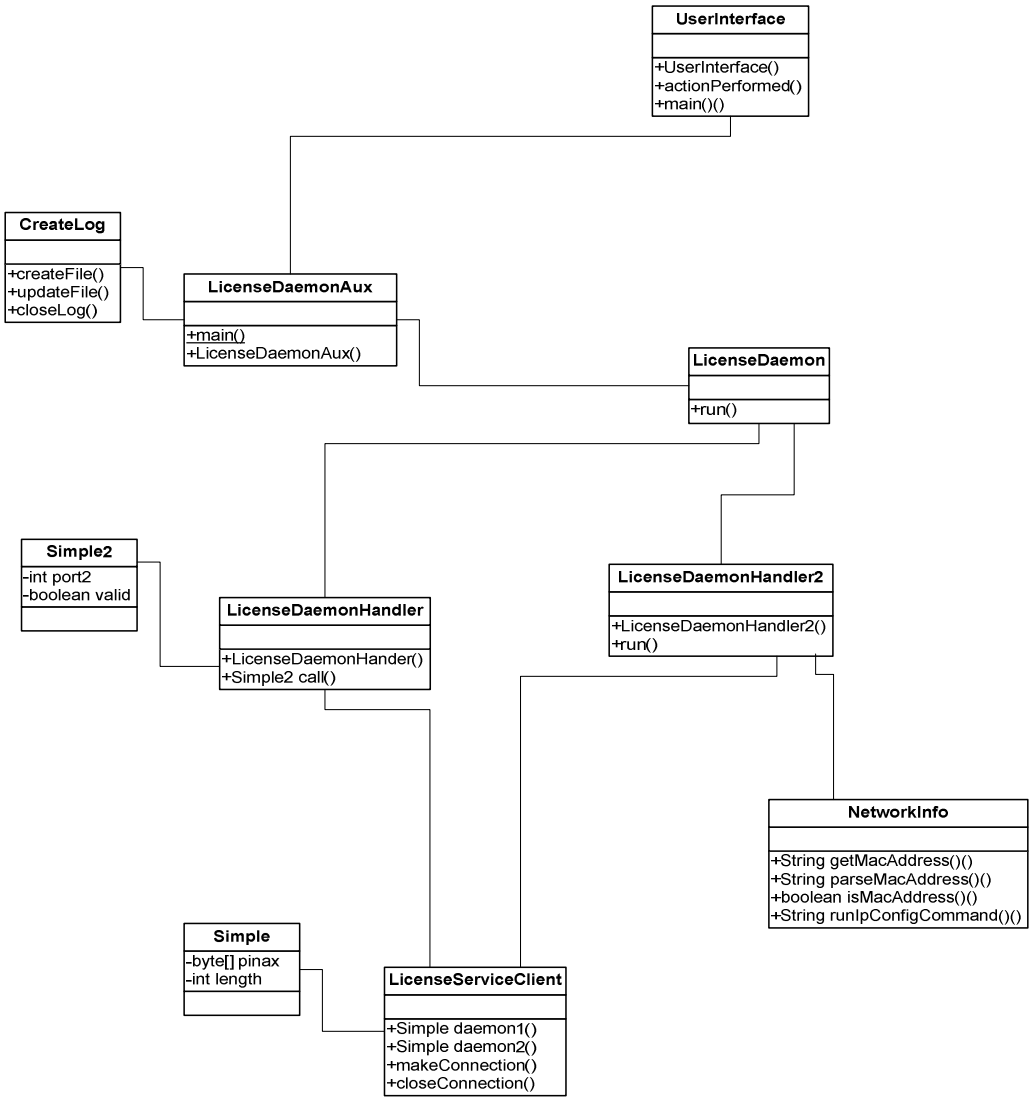
4.2.1.1 License Daemon

Τα απαραίτητα αρχεία-κλάσεις για την υλοποίηση του License Daemon, αποτελούν έναν εξυπηρετητή, ο οποίος δέχεται τις αιτήσεις από τις εφαρμογές και στη συνέχεια τις προωθεί στο License Service. Οι κλάσεις αυτές είναι οι εξής: LicenseDaemonAux.java, LicenseDaemon.java, LicenseDaemonHandler.java, LicenseDaemonHandler2.java, LicenseServiceClient.java.

Για την εκκίνηση του LicenseDaemon μέσω γραφικής διεπαφής χρήστη, υλοποιήθηκε η κλάση UserInterface.java.

Τη λειτουργία του License Daemon και τον έλεγχο της, υποβοηθούν οι εξής κλάσεις: CreateLog.java, NetworkInfo.java, Simple.java, Simple2.java.

Οι κλάσεις αυτές παρουσιάζονται συνοπτικά στο σχήμα 20.



Εικόνα 20: Διάγραμμα Κλάσεων (Class Diagram) του License Daemon

Πιο αναλυτικά η κάθε κλάση επιτελεί τη λειτουργία που περιγράφεται παρακάτω:

- **UserInterface.java:** Η κλάση αυτή είναι υπεύθυνη για την εμφάνιση της γραφικής διεπαφής από την οποία ο διαχειριστής συμπληρώνει τα ορίσματα, εκκινεί τον License Daemon και παρακολουθεί την όλη διαδικασία παροχής άδειας στις εφαρμογές. Με τον όρο ορίσματα εννοείται η διαδικτυακή διεύθυνση του License Service καθώς και η/οι θύρα/ες στις οποίες θα ακούει ο License Daemon.

- **LicenseDaemonAux.java:** Στιγμιότυπο της κλάσης αυτής μπορεί να δημιουργηθεί, είτε κατά την κλήση της main() μεθόδου της, από την αντίστοιχη JSP σελίδα ή από τη γραμμή εντολών, είτε από την κλήση της μεθόδου της LicenseDaemonAux από την UserInterface.java κατά την εκκίνηση του License Daemon μέσω γραφικής διεπαφής χρήστη. Η κλάση αυτή, δημιουργεί νήματα (threads) της LicenseDaemon, ανάλογα με τα ορίσματα που έχει λάβει, είτε μέσω γραμμής εντολών, είτε μέσω γραφικής διεπαφής χρήστη, είτε μέσω των JSP σελίδων. Επίσης μεταφέρει σε αυτά τη διαδικτυακή διεύθυνση του License Service που θα κληθεί, έτσι όπως αυτή

έχει οριστεί από το διαχειριστή.

- **LicenseDaemon.java:** Τα νήματα της κλάσης αυτής, αφού δημιουργήσουν sockets που ακούν σε όσες από τις θύρες έχουν οριστεί από το διαχειριστή, βρίσκονται σε διαρκή αναμονή των αιτήσεων από τις εφαρμογές – πελάτες, σε αυτές τις θύρες. Κάθε νήμα, μόλις λάβει μια αίτηση στη θύρα όπου ακούει δημιουργεί ένα νήμα της LicenseDaemonHandler.java μεταφέροντας σε αυτήν τη διαδικτυακή διεύθυνση του LicenseService και τη θύρα. Για κάθε αίτηση λοιπόν δημιουργείται διαφορετικό νήμα του LicenseDaemonHandler. Στη συνέχεια, το νήμα του LicenseDaemon περιμένει την απάντηση από τον εκάστοτε LicenseDaemonHandler η οποία περιέχει μία δεύτερη θύρα και δημιουργεί νέο socket που να ακούει σε αυτήν. Στην δεύτερη θύρα αναμένει τις επόμενες αιτήσεις της εφαρμογής – πελάτη ώστε να ολοκληρωθεί η αδειοδότησή της. Όταν λάβει το πρώτο μήνυμα από την εφαρμογή στην νέα θύρα, δημιουργεί ένα νήμα της κλάσης LicenseDaemonHandler2.java μεταφέροντας σε αυτήν εκ νέου τη διαδικτυακή διεύθυνση του License Service και τη νέα θύρα.

- **LicenseDaemonHandler.java:** Η κλάση αυτή, αφού δημιουργήσει streams ανάγνωσης/εγγραφής για την επικοινωνία με την εφαρμογή – πελάτη, διαβάζει το μήνυμα που αποστέλλεται από αυτή και το προωθεί στην κλάση LicenseServiceClient.java, καλώντας τη μέθοδο της, daemon1(), η οποία περιγράφεται παρακάτω. Στη συνέχεια, περιμένει την απάντηση από την daemon1() και την προωθεί στην εφαρμογή πελάτη. Στην απάντηση αυτή περιέχεται η θύρα του vendor daemon (βλ. κεφάλαιο 2) στην οποία θα επιχειρήσει να συνδεθεί η εφαρμογή - πελάτης ώστε να λάβει την άδεια. Η θύρα αυτή επιστρέφεται μέσω της LicenseDaemonHandler.java στην LicenseDaemon.java ώστε να δημιουργήσει το δεύτερο socket στο οποίο θα αναμένει αιτήσεις από την εφαρμογή – πελάτη, όπως περιγράφηκε προηγουμένως.

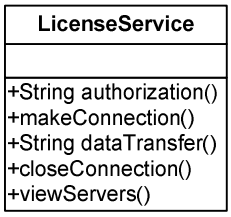
- **LicenseDaemonHandler2.java:** Η κλάση αυτή, αφού δημιουργήσει streams ανάγνωσης/εγγραφής για την επικοινωνία με την εφαρμογή – πελάτη, καλεί τη μέθοδο makeConnection() της κλάσης LicenseServiceClient.java η οποία περιγράφεται παρακάτω, ώστε να επιτευχθεί σύνδεση με τον FLEXlm Server. Με το που επιτευχθεί αυτή η σύνδεση, ξεκινά μια διαδικασία όπου γίνονται συνεχείς κλήσεις της μεθόδου daemon2() της LicenseServiceClient() ώστε να μεταφέρονται τα μηνύματα από και προς την εφαρμογή – πελάτη. Η διαδικασία αυτή συνεχίζεται έως ότου η εφαρμογή τερματίσει τη λειτουργία της, οπότε και καλείται η μέθοδος closeConnection() της LicenseServiceClient.java, η οποία περιγράφεται παρακάτω.

- **LicenseServiceClient.java:** Η κλάση αυτή περιλαμβάνει τέσσερις μεθόδους, κάθε μία υπεύθυνη για την κλήση της αντίστοιχης μεθόδου του License Service. Οι μέθοδοι αυτές είναι οι εξής: daemon1(), makeConnection(), daemon2(), closeConnection() και καλούν, μεταφέροντας τα κατάλληλα ορίσματα, τις μεθόδους authorization(), makeConnection(), dataTransfer(), closeConnection() του LicenseService αντίστοιχα, οι οποίες επιστρέφουν τις απαντήσεις των μηνυμάτων που λαμβάνονται από τον FLEXlm Server.

4.2.1.2 License Service

Το στοιχείο License Service αποτελείται από μία μόνο κλάση τη LicenseService.java, όπως παρουσιάζεται και στο διάγραμμα κλάσης της εικόνας 21, με τέσσερις βασικές και μια

βοηθητική μέθοδο και είναι υπεύθυνο για την αποστολή των αιτήσεων στον FLEXlm Server, αφού κληθεί από τον License Daemon καθώς και για την επιστροφή των αποκρίσεων του FLEXlm Server στον License Daemon.



Εικόνα 21: Διάγραμμα Κλάσεων (Class Diagram) του License Daemon

Οι μέθοδοι της LicenseService.java που περιγράφονται αναλυτικά παρακάτω είναι οι εξής: authorization(), makeConnection(), dataTransfer(), closeConnection() και viewServers() η βοηθητική μέθοδος.

- **viewServers():** Λαμβάνει από ένα αρχείο τους προς σύνδεση FLEXlm Servers ώστε το στοιχείο LicenseService να επιχειρήσει να συνδεθεί με τον πρώτο εξ' αυτών και στη συνέχεια σε περίπτωση που δεν είναι διαθέσιμος με τον επόμενο και ούτω καθεξής.

- **authorization():** Με την κλήση της μεθόδου αυτής, αφού επιλεγεί ο προς σύνδεση FLEXlm Server, δημιουργούνται socket και streams μεταξύ τους και αποστέλλεται το μήνυμα στην πρώτη θύρα, όπως αυτή έχει οριστεί στις παραμέτρους κλήσης της. Με τη λήψη της απόκρισης από τον FLEXlm Server, το μήνυμα επιστρέφεται στον License Daemon και διακόπτεται η σύνδεση με την πρώτη θύρα

- **makeConnecton():** Με την κλήση της μεθόδου αυτής, δημιουργείται socket με τη δεύτερη θύρα του FLEXlm Server.

- **dataTransfer():** Με την κλήση της μεθόδου αυτής, δημιουργούνται streams ανάγνωσης/εγγραφής από το socket που δημιουργήθηκε με την κλήση της makeConnection() για την ανταλλαγή μηνυμάτων με τον FLEXlm Server. Στη συνέχεια αποστέλλεται το μήνυμα της εφαρμογής – πελάτη στον FLEXlm Server, λαμβάνεται η απόκριση και προωθείται στον License Daemon.

- **closeConnection() :** Με την κλήση της μεθόδου αυτής καταγράφεται το αρχείο LicenseServiceHistory.log όπου φαίνεται ποιος υπολογιστής (ποια MAC address) συνδέθηκε με τον FLEXlm Server, σε ποια θύρα του, καθώς και πότε έγινε η σύνδεση και η αποσύνδεση.

5 Έλεγχος

Στο παρόν κεφάλαιο παρουσιάζεται ο οδηγός εγκατάστασης των τεχνολογιών και των πλατφόρμων που χρησιμοποιήθηκαν, παραδείγματα εκκίνησης και εκτέλεσης του συστήματος και τα αποτελέσματα εκτέλεσης του.

5.1 Οδηγός Εγκατάστασης

Για την ορθή λειτουργία του συστήματος παροχής άδειας, τόσο από την πλευρά του End User τόσο και από την πλευρά του License Provider, απαιτείται η εγκατάσταση του περιβάλλοντος, δηλαδή των πλατφόρμων και των τεχνολογιών που παρουσιάστηκαν στο κεφάλαιο 4, καθώς και η εγκατάσταση των στοιχείων License Daemon και License Service και των δυναμικών σελίδων διαχείρισής τους. Στις παραγράφους 5.1.1 και 5.1.2 που ακολουθούν, δίνεται ένας οδηγός εγκατάστασης όλων αυτών των στοιχείων.

5.1.1 Εγκατάσταση Περιβάλλοντος

5.1.1.1 Εγκατάσταση Περιβάλλοντος Εκτέλεσης Java (JRE)

Το αρχείο για την εγκατάσταση του JDK και του JRE βρίσκεται στην τοποθεσία http://java.sun.com/. Η έκδοση που χρησιμοποιήθηκε είναι η 1.6.0_04.

Σε περιβάλλον Windows ο χρήστης εκτελεί το αρχείο exe και ακολουθεί τις οδηγίες εγκατάστασης επιλέγοντας την εγκατάσταση και του jre. Σε περιβάλλον Linux ο χρήστης εκτελεί το αρχείο bin και ακολουθεί τις οδηγίες εγκατάστασης επιλέγοντας την εγκατάσταση και του jre.

Για την ορθή λειτουργία του περιβάλλοντος ο χρήστης υποχρεούται να θέσει κάποιες μεταβλητές περιβάλλοντος:

JAVA_HOME = η διαδρομή στο δίσκο όπου βρίσκεται εγκατεστημένο το jdk

JRE_HOME = διαδρομή στο δίσκο όπου βρίσκεται εγκατεστημένο το jre

PATH = other_path_variables;%JAVA_HOME%\bin

CLASSPATH = %JAVA_HOME%\ jre\lib\ext;% JRE_HOME%\lib\ext;

5.1.1.2 Εγκατάσταση του Tomcat

Το αρχείο για την εγκατάσταση του Tomcat διατίθεται δωρεάν στο διαδίκτυο στη διεύθυνση (http://tomcat.apache.org). Στην παρούσα εφαρμογή χρησιμοποιήθηκε η έκδοση 6.0.16.

Σε Windows περιβάλλον, απλά εκτελούμε το αρχείο που κατεβάσαμε από την ιστοσελίδα και ακολουθούμε τις οδηγίες. Για την εγκατάσταση σε Linux περιβάλλον, αρκεί να αντιγράψουμε τον κατάλογο που κατεβάσαμε από την ιστοσελίδα, στην τοποθεσία που επιθυμούμε να το εγκαταστήσουμε. Για την εκκίνηση του server εκτελούμε απλά το αρχείο startup.sh στον κατάλογο bin.

Αφού εκκινήσουμε τον tomcat μπορούμε να επισκεφθούμε την διεύθυνση http://127.0.0.1:8080/ και να επιβεβαιώσουμε την σωστή εγκατάσταση του.

Καλό είναι να ορίσουμε και την μεταβλητή συστήματος *CATALINA_HOME* που θα δείχνει το φάκελο εγκατάστασης.

CATALINA_HOME= η διαδρομή στο δίσκο όπου βρίσκεται εγκατεστημένος ο Tomcat

5.1.1.3 Εγκατάσταση του Axis2

Το Axis2 προσφέρεται στο διαδίκτυο δωρεάν (http://ws.apache.org/axis2/).

Για την εγκατάσταση του αρκεί να αντιγράψουμε το αρχείο axis2.war, της διανομής που κατεβάσαμε από το διαδίκτυο, και να το τοποθετήσουμε μέσα στον κατάλογο webapps του tomcat server. Αφού εκκινήσουμε τον tomcat μπορούμε να επισκεφθούμε την διεύθυνση http://127.0.0.1:8080/axis2/, και να επιβεβαιώσουμε την σωστή εγκατάσταση του προγράμματος αυτού. Η έκδοση που χρησιμοποιήσαμε εμείς ήταν η Axis2 1.3. Για την ορθή λειτουργία του Axis2 απαιτείται η δημιουργία της μεταβλητής του συστήματος :

AXIS2_HOME = η διαδρομή στο δίσκο όπου βρίσκεται εγκατεστημένο το Axis2

5.1.1.4 Εγκατάσταση του Apache HTTP Server

Για την εγκατάσταση του Apache HTTP Server ο χρήστης επισκέπτεται την τοποθεσία: http://httpd.apache.org/download.cgi και κατεβάζει ανάλογα με το λειτουργικό του σύστημα το αντίστοιχο αρχείο εγκατάστασης. Η έκδοση που χρησιμοποιήθηκε είναι η 2.2.4. Ο χρήστης ακολουθεί τις οδηγίες εγκατάστασης και αφού τον εκκινήσει μπορεί να επισκεφτεί την διεύθυνση http://127.0.0.1:80 για να ελέγξει αν έγινε σωστά η εγκατάσταση.

5.1.1.5 Εγκατάσταση της PHP

Για την εγκατάσταση της PHP ο χρήστης επισκέπτεται την τοποθεσία http://gr.php.net/downloads.php και κατεβάζει την δυαδική διανομή ανάλογα με το λειτουργικό του σύστημα. Η έκδοση που χρησιμοποιήθηκε είναι η 5.5.2. Στην συνέχεια εξάγει το περιεχόμενο της διανομής σε έναν φάκελο με όνομα PHP. Στον φάκελο αυτό, μετονομάζει το αρχείο με όνομα php-ini-recommend σε php.ini. Κατόπιν, ανοίγει το αρχείο

php.ini και θέτει τις βιβλιοθήκες που θα χρειαστούν για τη φόρτωση των php σελίδων, ορίζοντας τη διαδρομή στο δίσκο όπου βρίσκονται αυτές σύμφωνα με τις οδηγίες και βγάζοντας τα σχόλια από τις αντίστοιχες γραμμές. Για την ορθή λειτουργία της PHP απαιτείται η προσθήκη στη μεταβλητή του συστήματος PATH της διαδρομής της PHP στο δίσκο.

PATH = other_path_variables;%PHP_HOME%

όπου PHP_HOME η διαδρομή στο δίσκο όπου βρίσκεται η PHP.

Προκειμένου ο APACHE HTTP SERVER να μπορεί να φορτώνει PHP σελίδες χρειάζονται οι εξής προσθήκες στο αρχείο httpd.conf:

LoadModule php5_module PHP_HOME/php5apache2_2.dll

PHPIniDir "PHP_HOME"

AddType application/x-httpd-php .php

Αφού επανεκκινήσει τον APACHE HTTP SERVER ο χρήστης ελέγχει τη σωστή εγκατάσταση, προβάλλοντας οποιαδήποτε PHP σελίδα έχει τοποθετήσει στον κατάλληλο φάκελο του εξυπηρετητή, επισκεπτόμενος τη διεύθυνση http://127.0.0.1:80/path_to_php_file/file.php.

5.1.2 Εγκατάσταση νέου συστήματος

5.1.2.1 Εγκατάσταση License Daemon και εγκατάσταση των JSP σελίδων

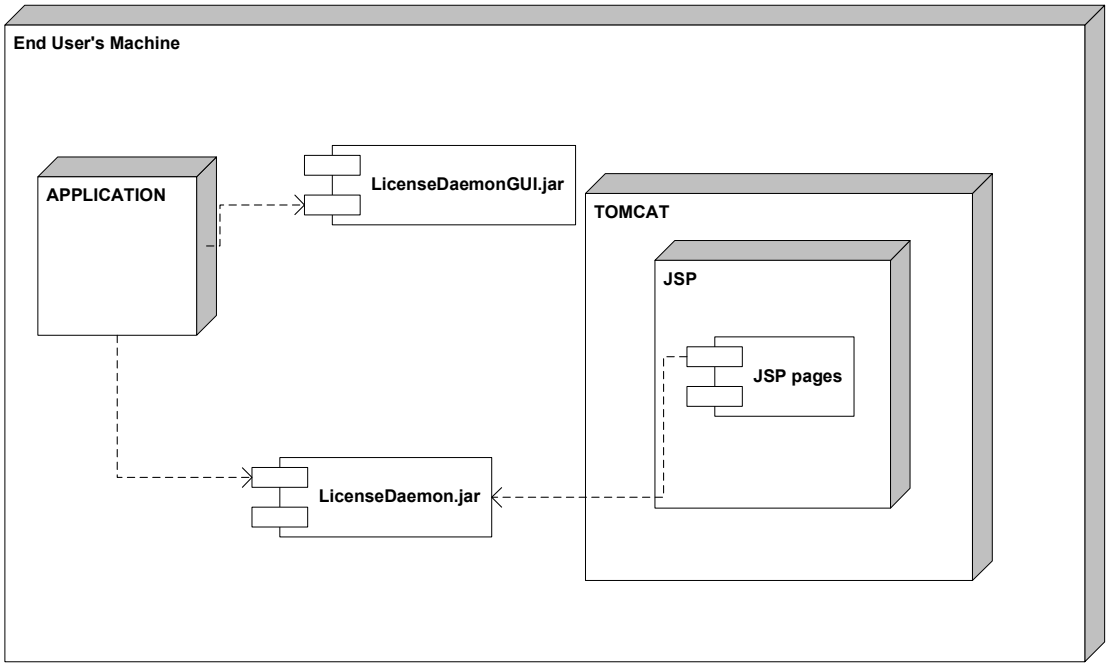
Ο διαχειριστής του License Daemon διαθέτει το αρχείο LicenseDaemon.zip το οποίο περιλαμβάνει τα εξής: τα αρχεία LicenseDaemon.bat, LicenseDaemon.jar, LicenseDaemonGui.bat, LicenseDaemonGUI.jar, τον φάκελο lib και το φάκελο LicenseDaemonSettings.

Ο διαχειριστής του License Daemon εξάγει τα αρχεία LicenseDaemon.bat, LicenseDaemon.jar, LicenseDaemonGui.bat, LicenseDaemonGUI.jar, και τον φάκελο lib σε οποιονδήποτε φάκελο επιθυμεί.

Ο φάκελος LicenseDaemonSettings που περιέχει τις JSP σελίδες τοποθετείται στον φάκελο CATALINA_HOME\webapps\ του tomcat και προσπελάζεται από την διεύθυνση http://127.0.0.1:8080/LicenseDaemonSettings/.

Ο φάκελος lib καθώς και το LicenseDaemon.jar τοποθετούνται στον φάκελο CATALINA_HOME\bin\.

Στο σχήμα 22 φαίνεται το αντίστοιχο deployment diagram.



Εικόνα 22: Deployment Diagram του τελικού χρήστη

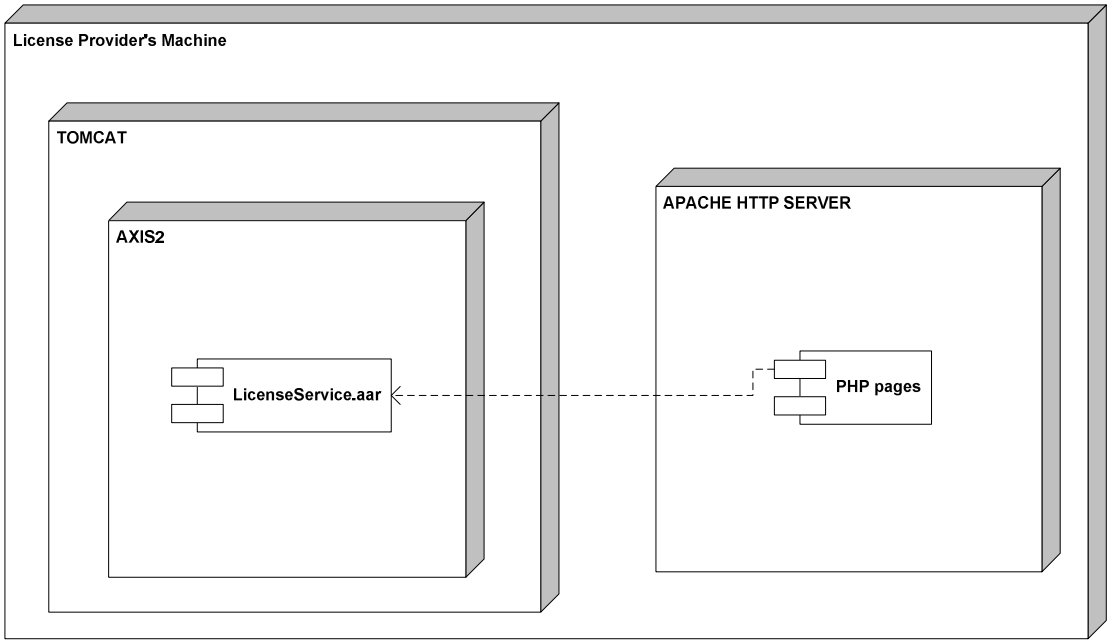
5.1.2.2 Εγκατάσταση License Service και εγκατάσταση των PHP σελίδων

Ο διαχειριστής του License Service διαθέτει το αρχείο LicenseService.zip το οποίο περιλαμβάνει τα εξής: τα αρχεία LicenseService.aar και AvailableServers.txt και το φάκελο LicenseServiceTools.

Ο διαχειριστής του License Service αντιγράφει το LicenseService.aar στον φάκελο CATALINA_HOME\webapps\axis2\WEB-INF\services του APACHE TOMCAT SERVER. Το αρχείο AvailableServers.txt τοποθετείται στον φάκελο C:\Documents and Settings.

Ο φάκελος LicenseServiceTools τοποθετείται μέσα στον φάκελο htdocs του Apache HTTP Server του μηχανήματος. Οι PHP επομένως προσπελάζονται από την διεύθυνση http://127.0.0.1/LicenseServiceTools/.

Στο σχήμα 23 φαίνεται το αντίστοιχο deployment diagram.



Εικόνα 23: Deployment Diagram του License Provider

5.2 Αναλυτική παρουσίαση ελέγχου

5.2.1 Εκτέλεση License Daemon

Όπως έχει αναφερθεί και στο κεφάλαιο 3, οι τρόποι εκτέλεσης του License Daemon είναι τρεις:

1. Μέσω γραμμής εντολών
2. Μέσω παραθυριακής γραφικής διεπαφής
3. Μέσω JSP σελίδων

5.2.1.1 Μέσω γραμμής εντολών

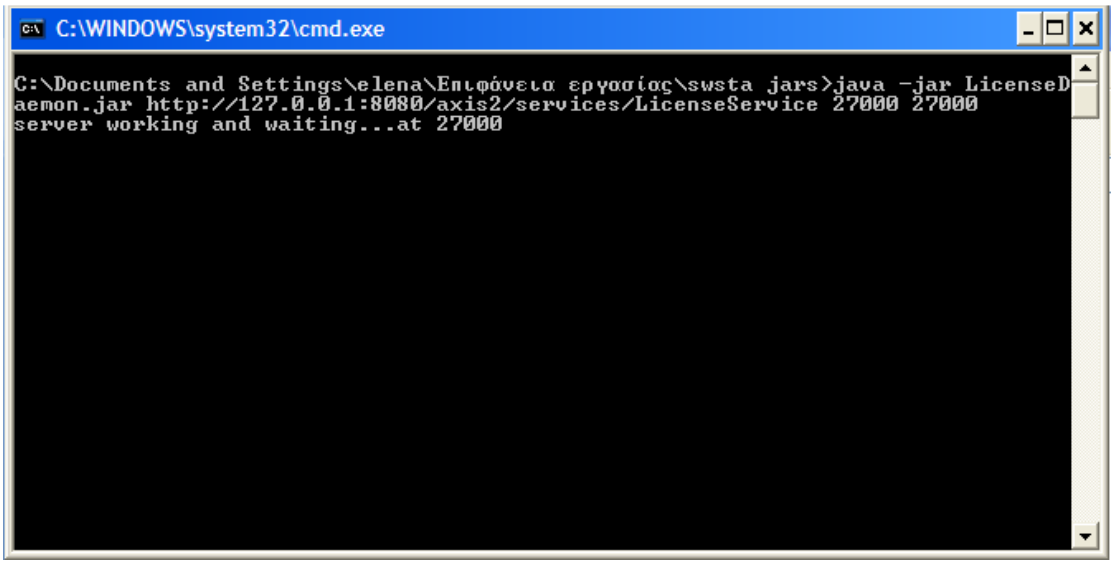
Ο διαχειριστής του License Daemon διαθέτει το LicenseDaemon.bat. Ανοίγοντας το αρχείο με οποιονδήποτε επεξεργαστή κειμένου παραμετροποιεί την ακόλουθη γραμμή:

java –jar LicenseDaemon.jar parameter1 parameter2 parameter3

Όπου

1. parameter1: η διαδικτυακή διεύθυνση του License Service
2. parameter2: το κάτω όριο του εύρους θυρών στο οποίο θα ακούει ο εξυπηρετητής License Daemon
3. parameter3: το άνω όριο του εύρους θυρών

Ο διαχειριστής θέτει τις παραπάνω τρεις παραμέτρους και αποθηκεύει τις αλλαγές στο αρχείο. Κατόπιν εκτελεί το LicenseDaemon.bat και του εμφανίζεται η παρακάτω κονσόλα.



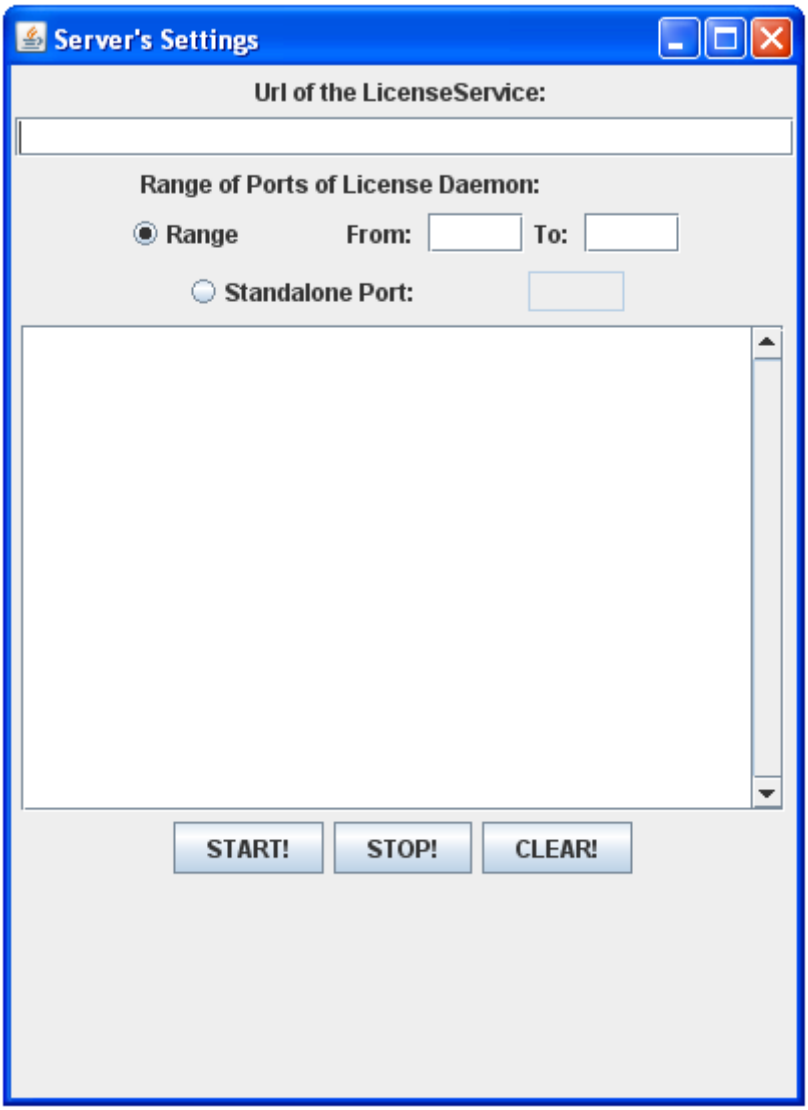
Εικόνα 24: Εκτέλεση License Daemon από τερματικό

Στην κονσόλα, ο διαχειριστής επιβεβαιώνει τις επιλογές που έκανε, εφόσον βλέπει να εκτελούνται License Daemons που ακούνε στις θύρες που προηγουμένως επέλεξε.

Η κονσόλα παραμένει ανοιχτή έως ότου ο διαχειριστής επιλέξει να την κλείσει οπότε και τερματίζεται η λειτουργία του License Daemon.

5.2.1.2 Μέσω παραθυριακής γραφικής διεπαφής

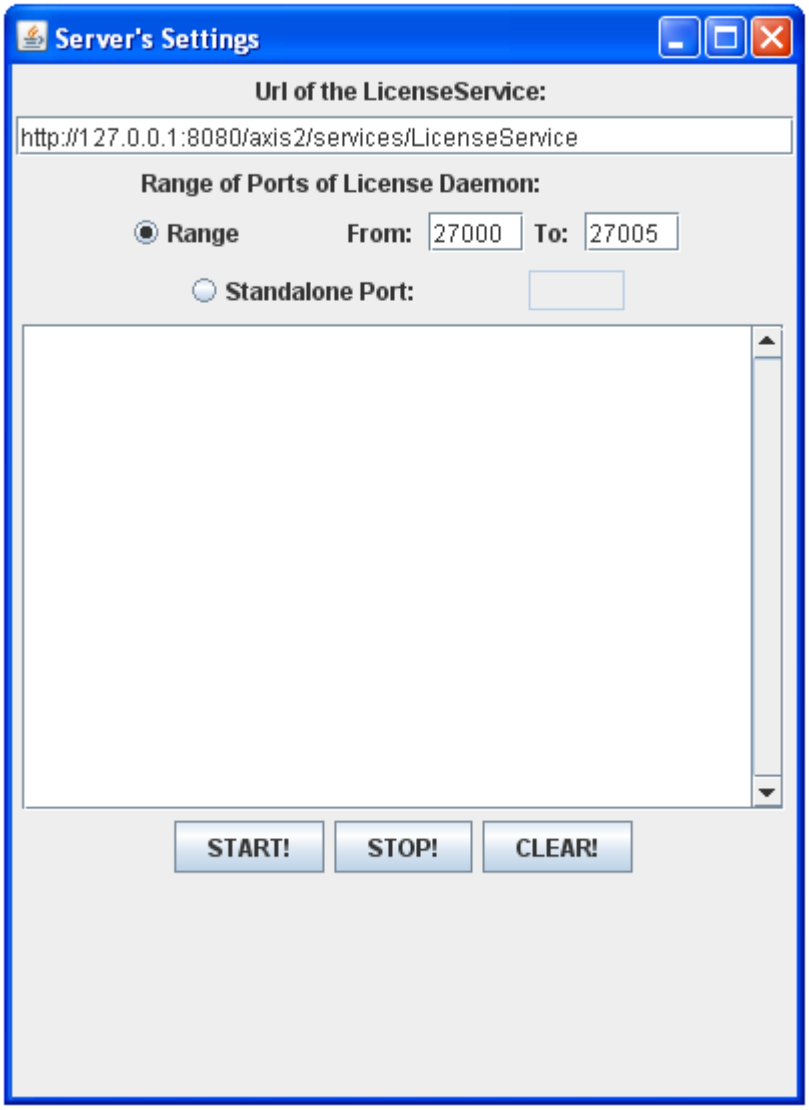
Ο διαχειριστής του License Daemon διαθέτει το LicenseDaemonGUI.bat. Εκτελώντας το, εμφανίζεται η παρακάτω γραφική διεπαφή:



Εικόνα 24: Εκτέλεση License Daemon από γραφική διεπαφή (1)

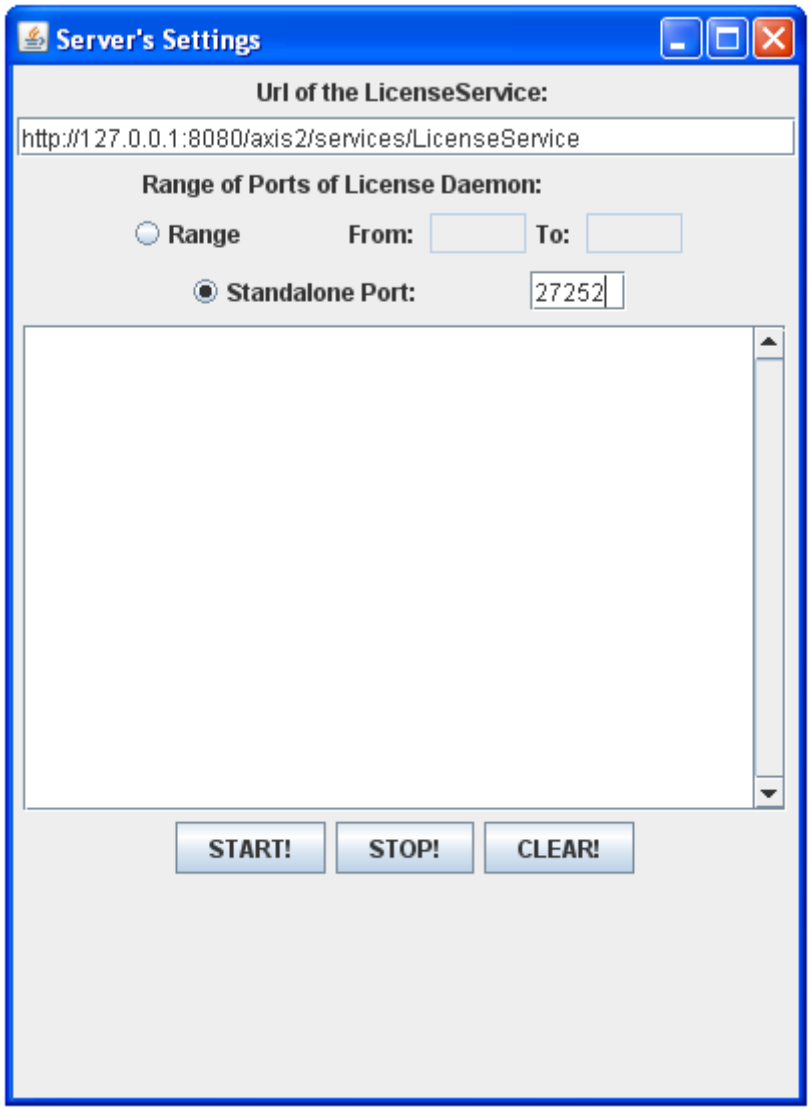
Στην γραφική διεπαφή, ο διαχειριστής υποχρεούται να συμπληρώσει την διαδικτυακή διεύθυνση του License Service στο πεδίο “Url of the License Service” καθώς και να επιλέξει τη/τις θύρες στις οποίες θα ακούει ο License Daemon. Ο διαχειριστής έχει τις εξής επιλογές:

Να επιλέξει ένα φάσμα θυρών επιλέγοντας την ρύθμιση “Range”.



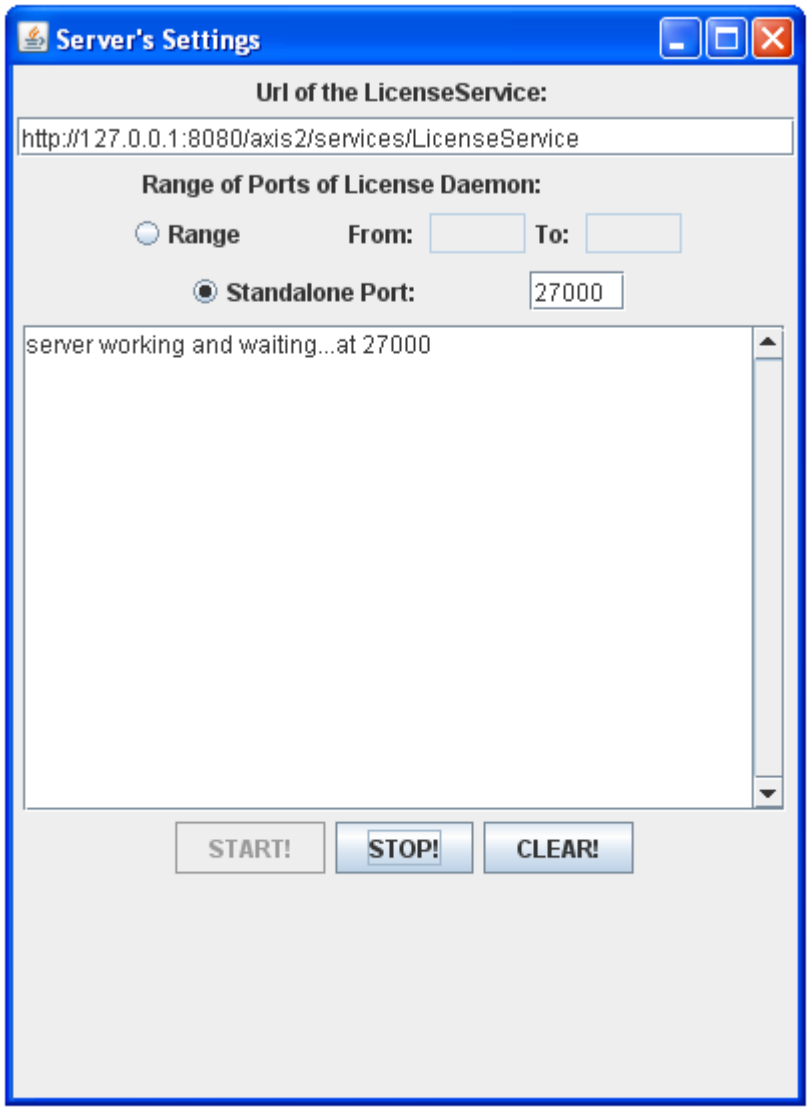
Εικόνα 25: Εκτέλεση License Daemon από γραφική διεπαφή (2)

Να επιλέξει μία μόνο θύρα, επιλέγοντας την ρύθμιση “Standalone Port” και συμπληρώνοντας στο αντίστοιχο πεδίο την επιθυμητή θύρα.



Εικόνα 26: Εκτέλεση License Daemon από γραφική διεπαφή (3)

Για να ολοκληρωθεί η εκκίνηση του License Daemon, ο διαχειριστής πατά το κουμπί "START!". Με το πάτημα του κουμπιού "START!", αρχίζουν την εκτέλεση τους οι License Daemons, κάθε ένας εκ των οποίων ακούει σε μια από τις θύρες που προηγουμένως επέλεξε. Η εκτέλεση αυτή, είναι ορατή από την γραφική διεπαφή μέσω της περιοχής κειμένου που εμφανίζεται στην διεπαφή αμέσως μετά το πάτημα του κουμπιού "START!".



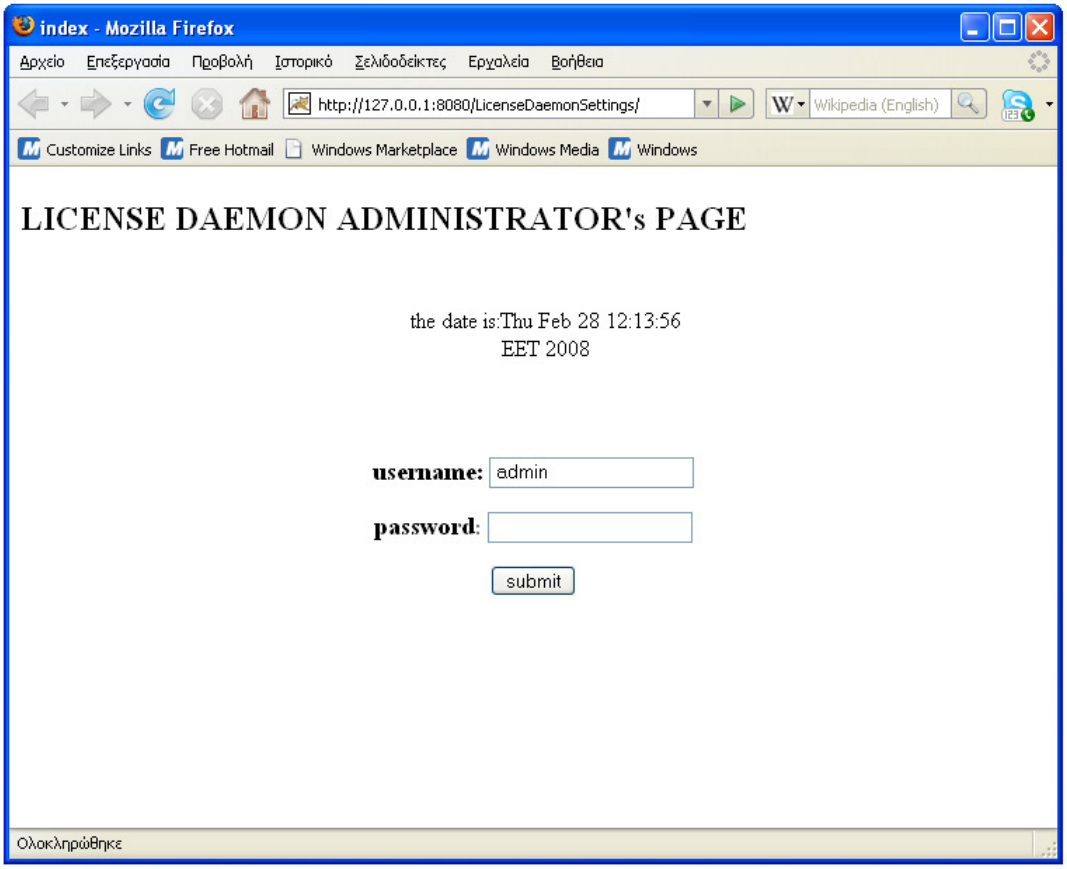
Εικόνα 27: Εκτέλεση License Daemon από γραφική διεπαφή (4)

Ο διαχειριστής μπορεί να τερματίσει την λειτουργία του License Daemon πατώντας το κουμπί "STOP!" ενώ μπορεί να εκκαθαρίσει την περιοχή κειμένου πατώντας το κουμπί "CLEAR!".

5.2.1.3 Μέσω JSP σελίδων

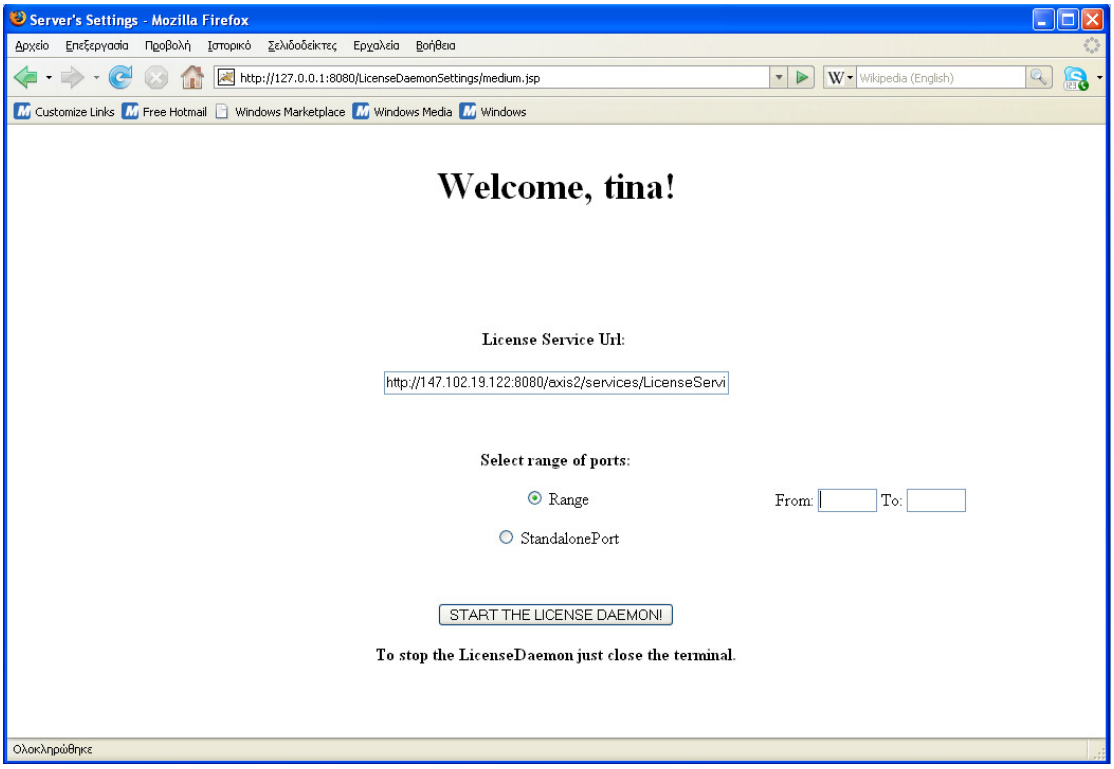
Ο διαχειριστής προκειμένου να εκκινήσει τον License Daemon επισκέπτεται την τοποθεσία http://127.0.0.1:8080/LicenseDaemonSettings/ .

Στην σελίδα που εμφανίζεται (index.jsp) ο διαχειριστής υποχρεούται να εισάγει το όνομα και τον κωδικό πρόσβασης του προκειμένου να αποκτήσει πρόσβαση στις ρυθμίσεις του License Daemon.



Εικόνα 28: Εκτέλεση License Daemon από JSP σελίδες(1)

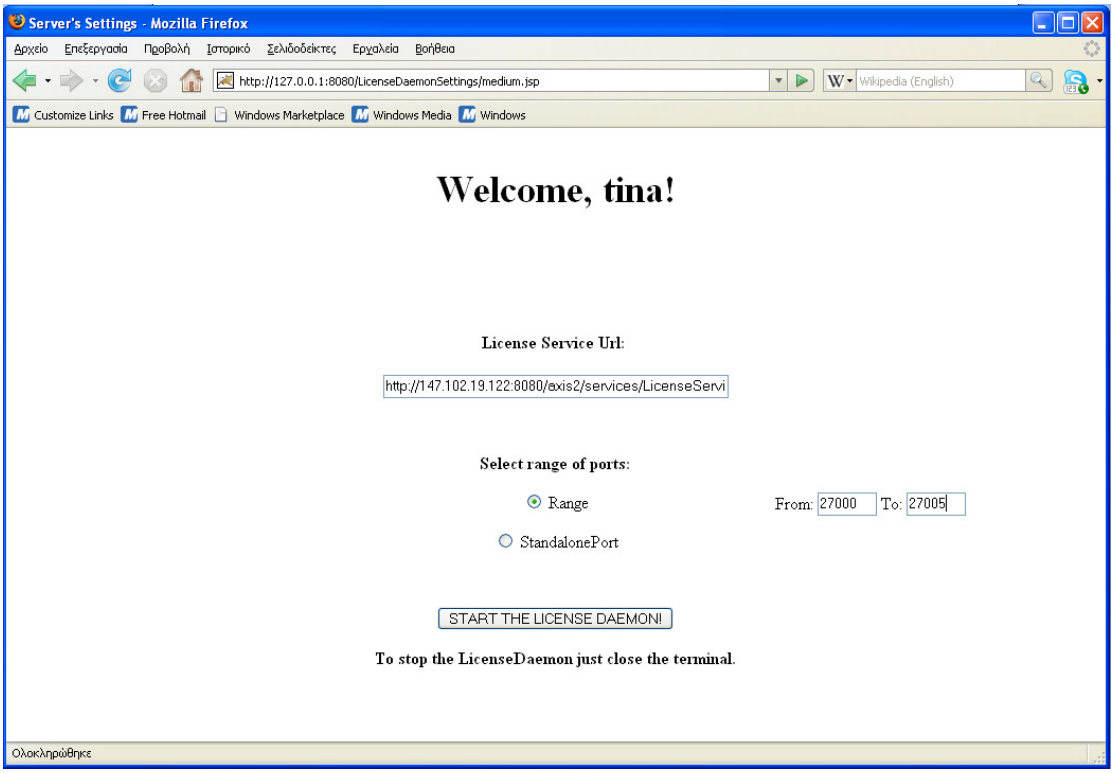
Με τη συμπλήρωση έγκυρων κωδικών, ο διαχειριστής μεταφέρεται στην σελίδα ρυθμίσεων του License Daemon.



Εικόνα 29: Εκτέλεση License Daemon από JSP σελίδες(2)

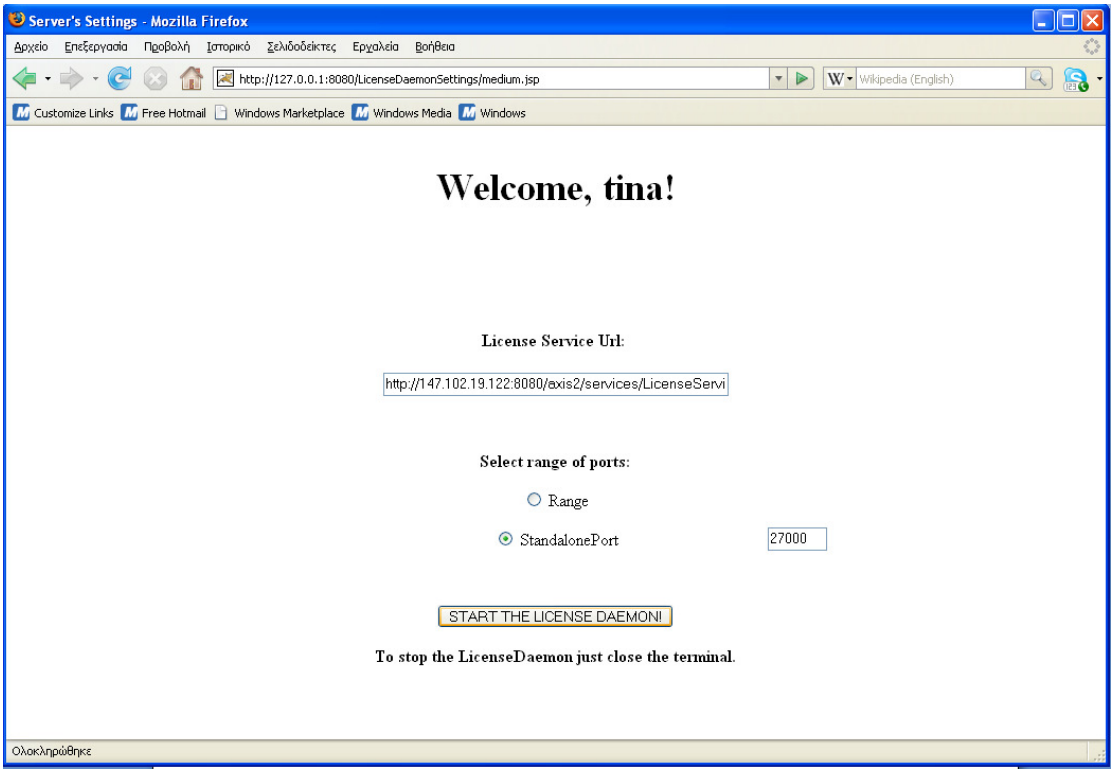
Εδώ, ο διαχειριστής υποχρεούται να συμπληρώσει την διαδικτυακή διεύθυνση του License Service στο πεδίο “License Service Url” καθώς και να επιλέξει τη/τις θύρες στις οποίες θα ακούει ο License Daemon. Ο διαχειριστής έχει τις εξής επιλογές:

Να επιλέξει ένα φάσμα θυρών επιλέγοντας την ρύθμιση “Range”.



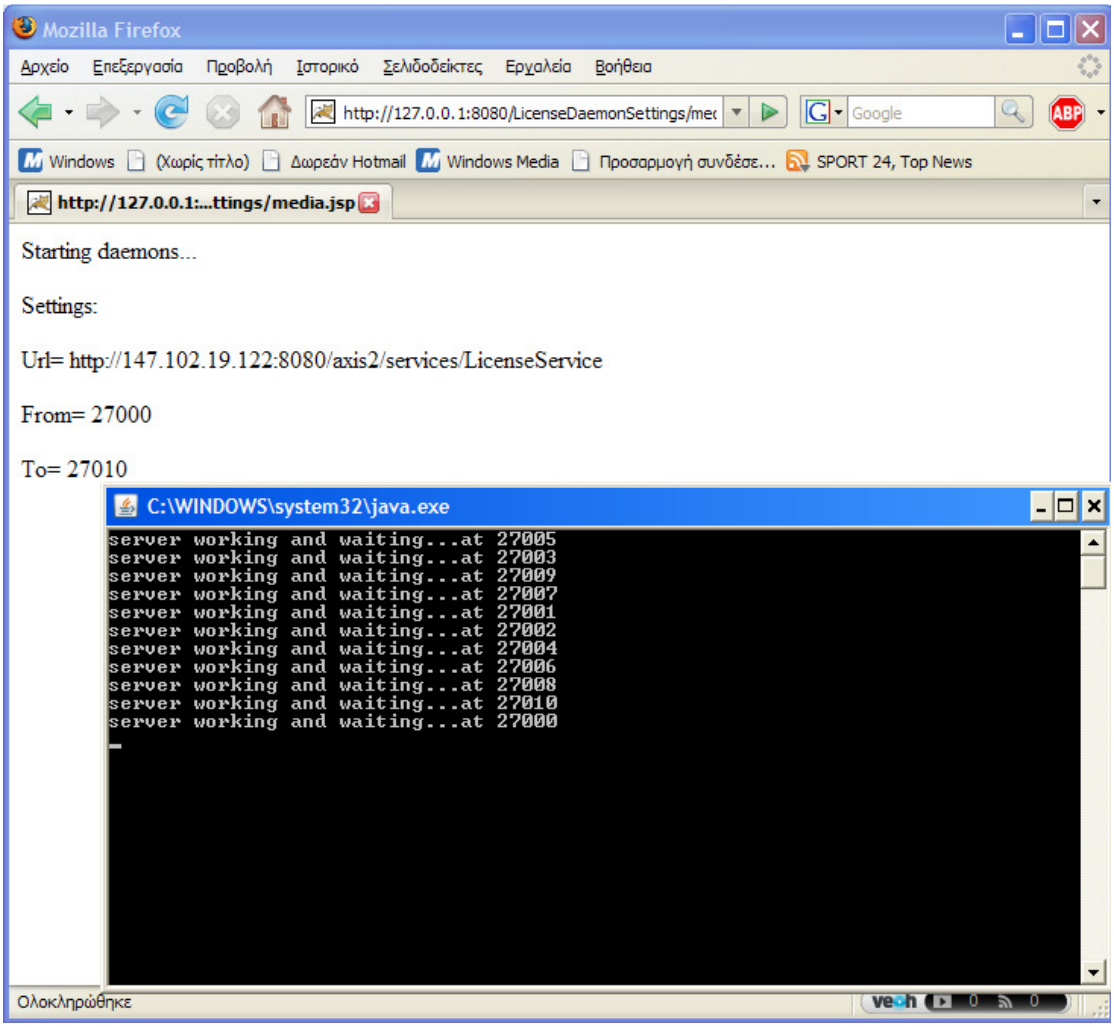
Εικόνα 30: Εκτέλεση License Daemon από JSP σελίδες(3)

Να επιλέξει μία μόνο θύρα, επιλέγοντας την ρύθμιση “Standalone Port” και συμπληρώνοντας στο αντίστοιχο πεδίο την επιθυμητή θύρα.



Εικόνα 31: Εκτέλεση License Daemon από JSP σελίδες(4)

Για να ολοκληρωθεί η εκκίνηση του License Daemon, ο διαχειριστής πατά το κουμπί “START THE LICENSE DAEMON!”. Με το πάτημα του κουμπιού αυτού, αρχίζουν την εκτέλεση τους οι δαίμονες, κάθε ένας εκ των οποίων ακούει σε μια από τις θύρες που προηγουμένως επέλεξε. Η εκτέλεση αυτή, είναι ορατή από την κονσόλα που εμφανίζεται.

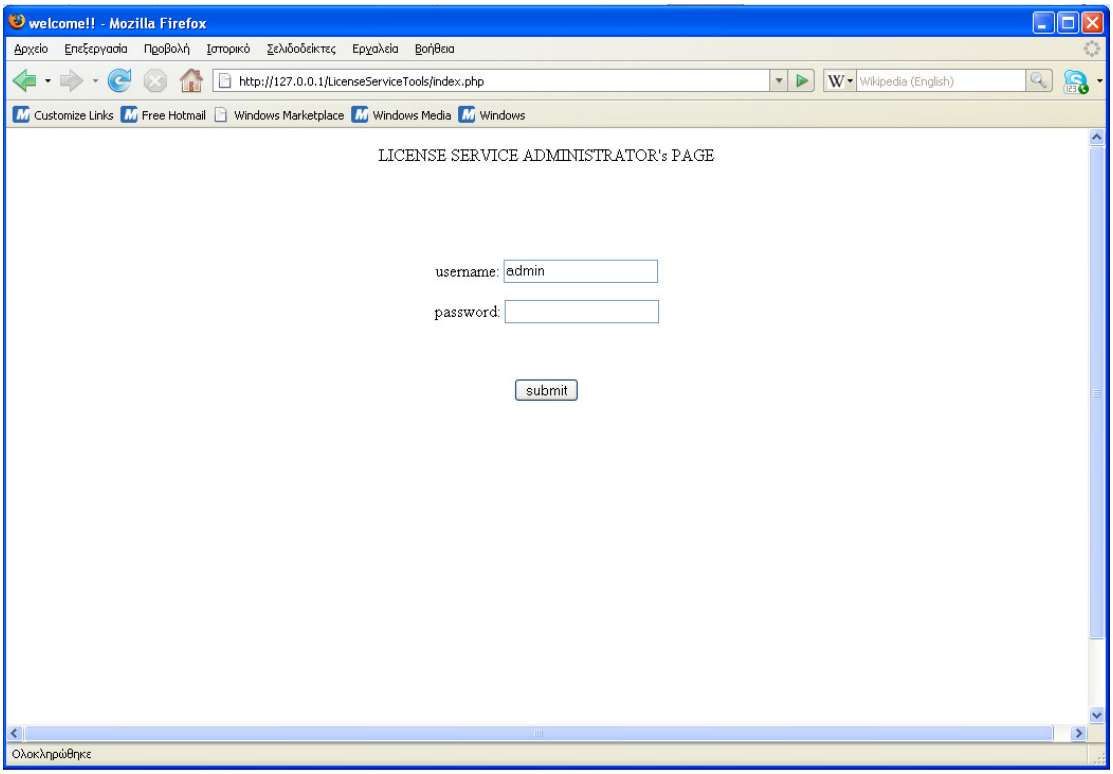


Εικόνα 32: Εκτέλεση License Daemon από JSP σελίδες(5)

Η κονσόλα παραμένει ανοιχτή έως ότου ο διαχειριστής επιλέξει να την κλείσει οπότε και τερματίζεται η λειτουργία του License Daemon.

5.2.2 Εκτέλεση License Service

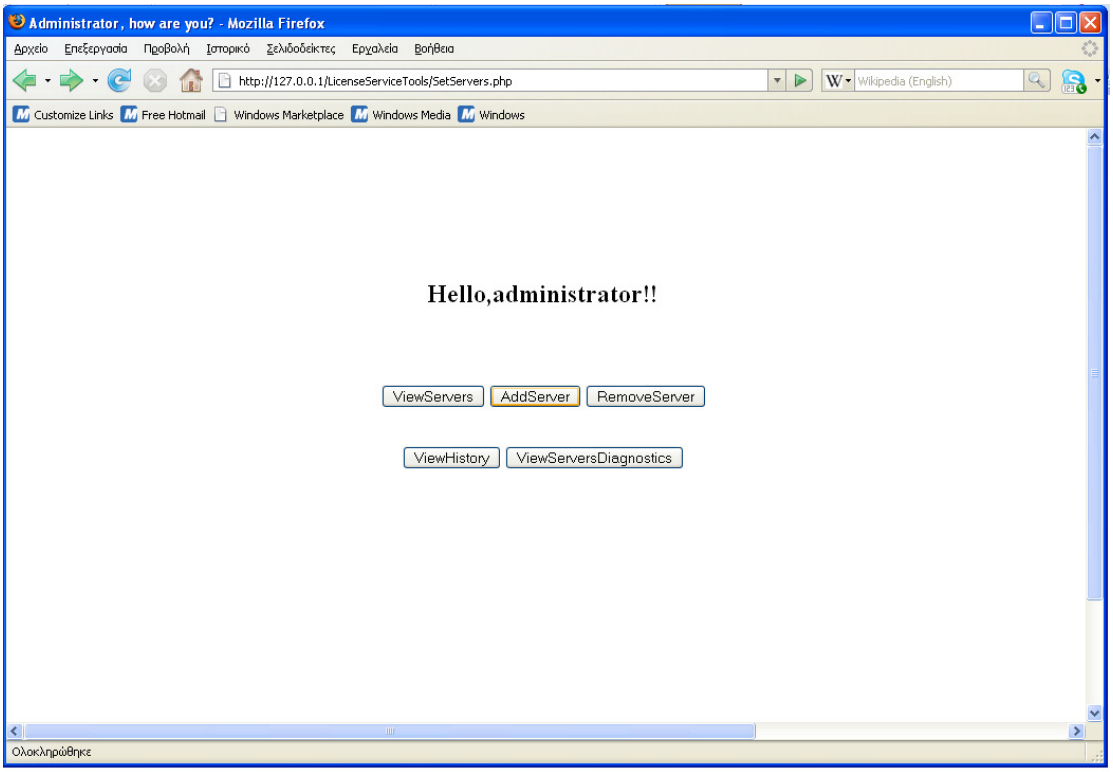
Ο διαχειριστής του License Service παρακολουθεί την χρήση του και ρυθμίζει τη λειτουργία του μέσω του δικτυακού τόπου http://127.0.0.1/LicenseServiceTools/.



Εικόνα 33: Ρύθμιση License Service από PHP σελίδες(1)

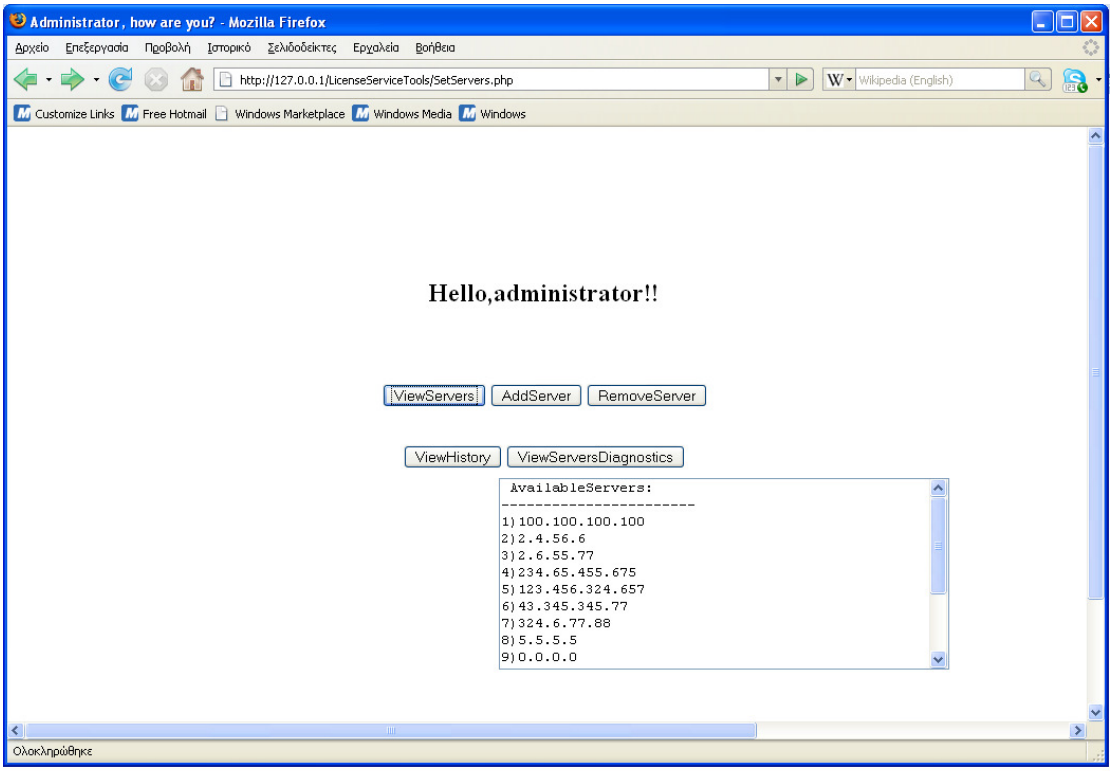
Στην σελίδα αυτή, ο διαχειριστής υποχρεούται να εισάγει το όνομα και τον κωδικό πρόσβασης προκειμένου να έχει πρόσβαση στα εργαλεία του License Service.

Στην επόμενη σελίδα (SetServers.php), ο διαχειριστής έχει την δυνατότητα να καθορίσει τους FLEXlm Servers με τους οποίους θα επικοινωνεί το License Service. Ο καθορισμός αυτός πραγματοποιείται από τα κουμπιά “AddServer”, “RemoveServer” και “ViewServers”.



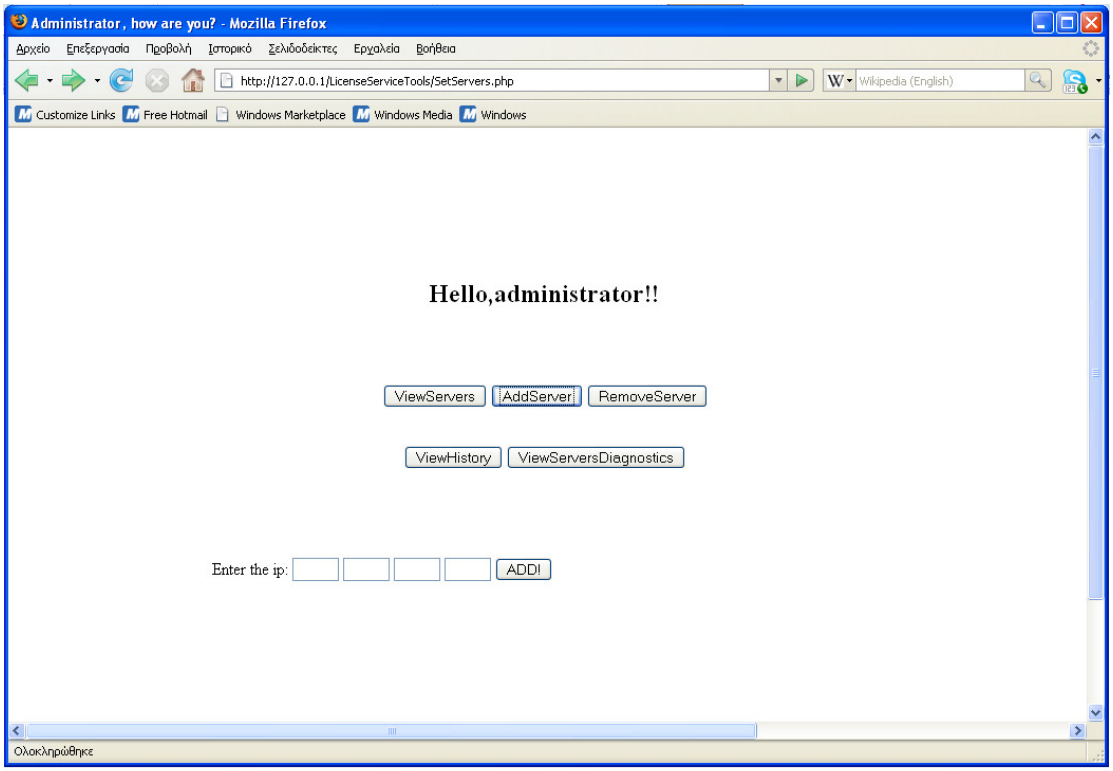
Εικόνα 34: Ρύθμιση License Service από PHP σελίδες(2)

Πατώντας το κουμπί "ViewServers", προβάλλεται το αρχείο AvailableServers.txt που περιέχει τους FLEXlm Servers προς σύνδεση. Η προβολή του αρχείου γίνεται σε μία περιοχή κειμένου που γίνεται ορατή με το πάτημα του κουμπιού.



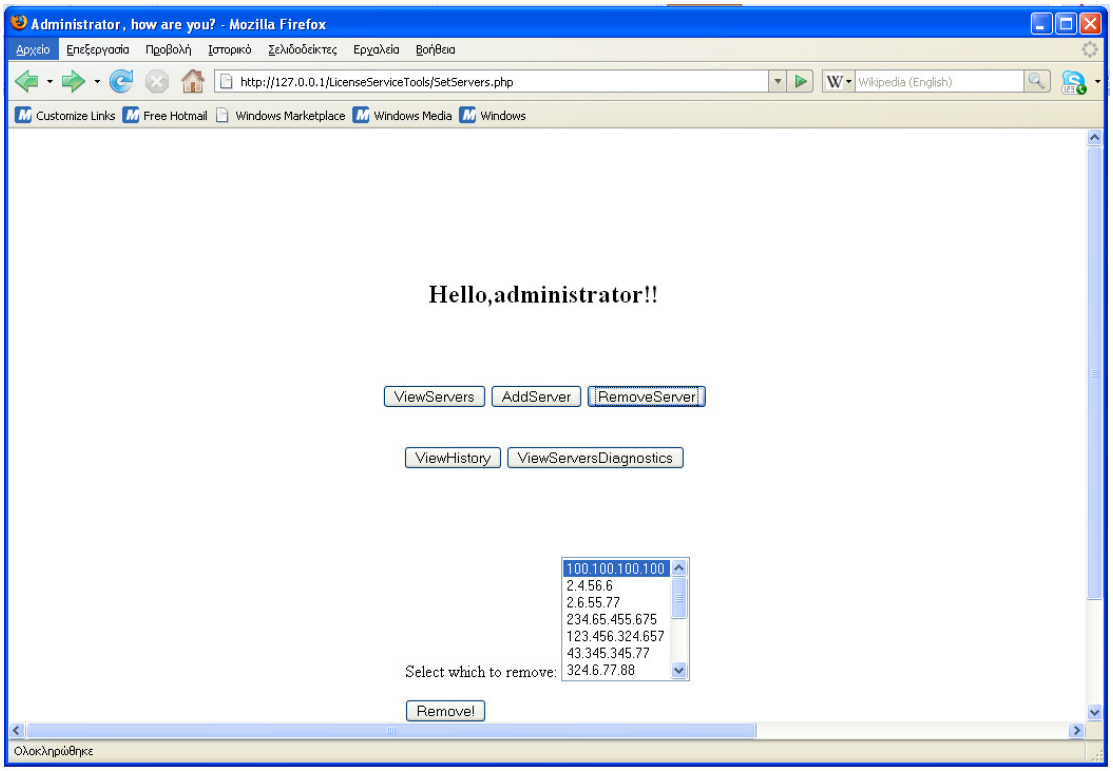
Εικόνα 35: Ρύθμιση License Service από PHP σελίδες(3)

Πατώντας το κουμπί “AddServer”, προβάλλεται ειδική φόρμα εισαγωγής μιας IP διεύθυνσης. Ο διαχειριστής συμπληρώνοντας τα πεδία, και πατώντας το κουμπί “ADD!”, προσθέτει στο αρχείο την IP διεύθυνση ενός νέου FLEXlm Server.



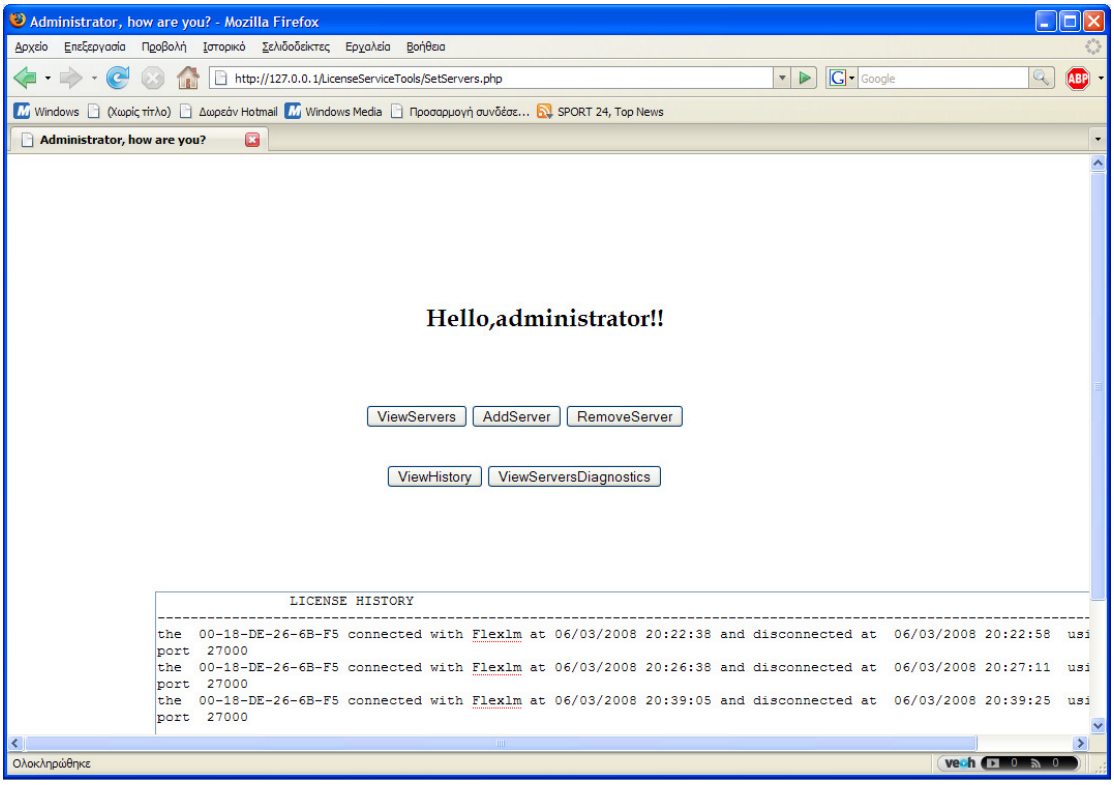
Εικόνα 36: Ρύθμιση License Service από PHP σελίδες(4)

Πατώντας το κουμπί “RemoveServer”, εμφανίζεται η λίστα των υπαρχόντων FLEXlm Servers, έτσι όπως διαβάζεται δυναμικά από το AvailableServers.txt. Ο διαχειριστής επιλέγει την IP του FLEXlm Server προς αφαίρεση, και πατώντας το κουμπί “REMOVE!” τον αφαιρεί.



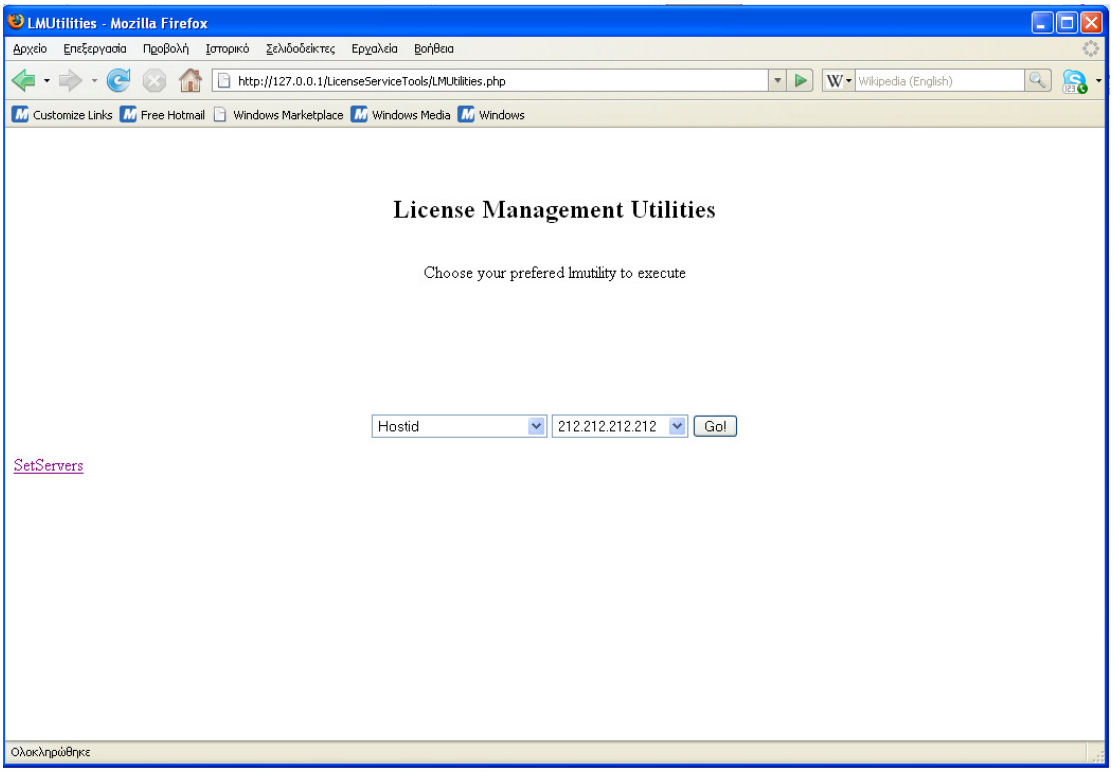
Εικόνα 37: Ρύθμιση License Service από PHP σελίδες(5)

Πατώντας το κουμπί "ViewHistory", απεικονίζεται το αρχείο καταγραφής του LicenseService όπως αυτό έχει διαμορφωθεί έως τώρα. Η απεικόνιση πραγματοποιείται πάνω σε μία περιοχή κειμένου η οποία γίνεται ορατή με το πάτημα του κουμπιού.

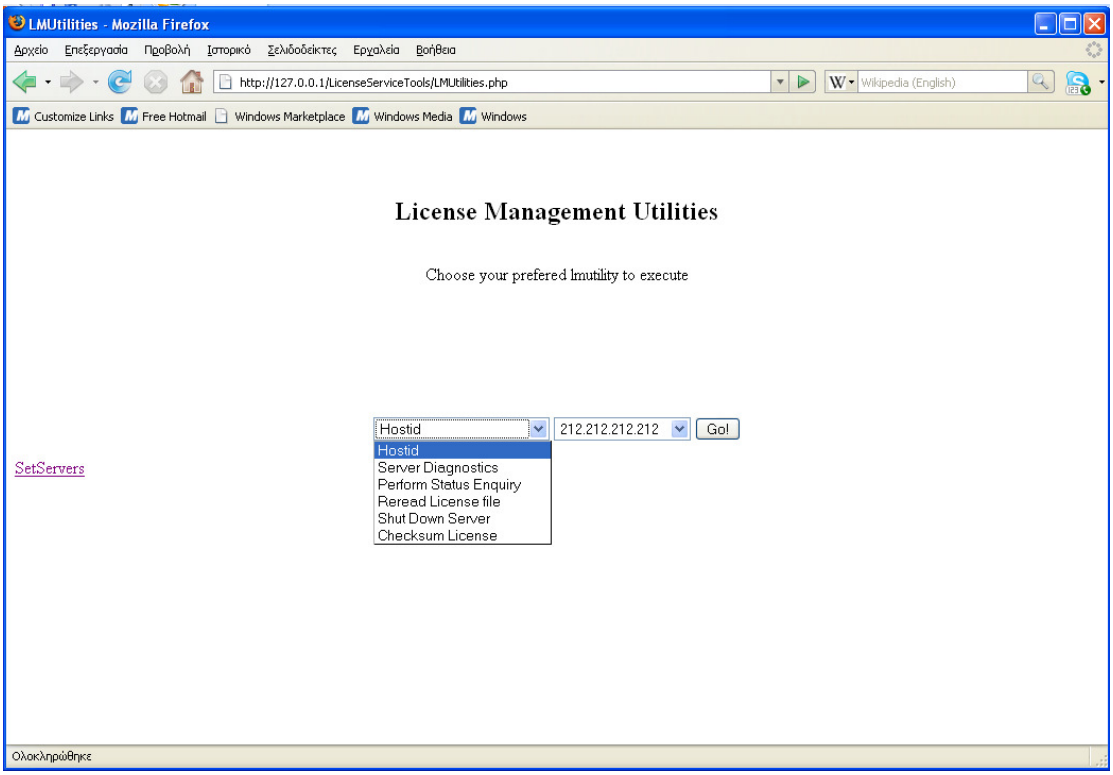


Εικόνα 38: Ιστορικό License Service

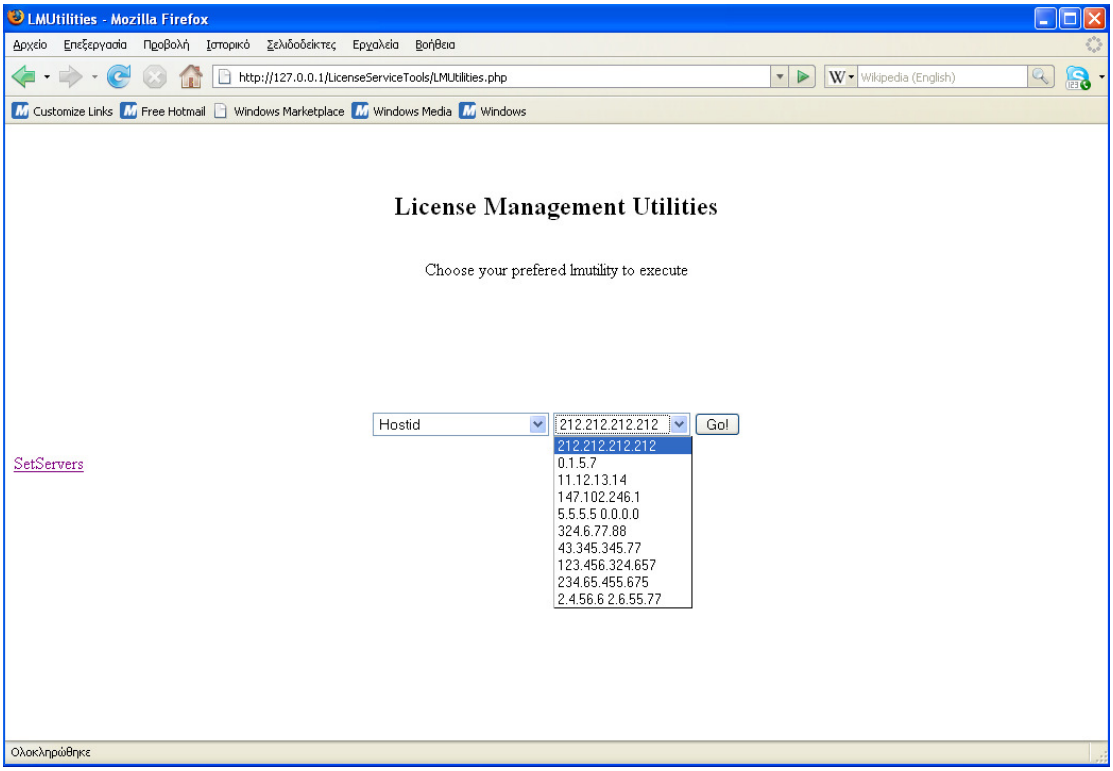
Πατώντας το κουμπί “ViewServerDiagnostics” ο διαχειριστής μεταβαίνει στην σελίδα LMUtilities.php στην όποια μπορεί να χρησιμοποιήσει ορισμένα από τα εργαλεία που παρέχει ένας FLEXlm Server, η λειτουργία των οποίων περιγράφεται στο κεφάλαιο 2. Σε περίπτωση που επιθυμεί να επιστρέψει στην SetServers.php, ο διαχειριστής αρκεί να πατήσει στον σύνδεσμο.



Εικόνα 39: Εργαλεία License Service (1)



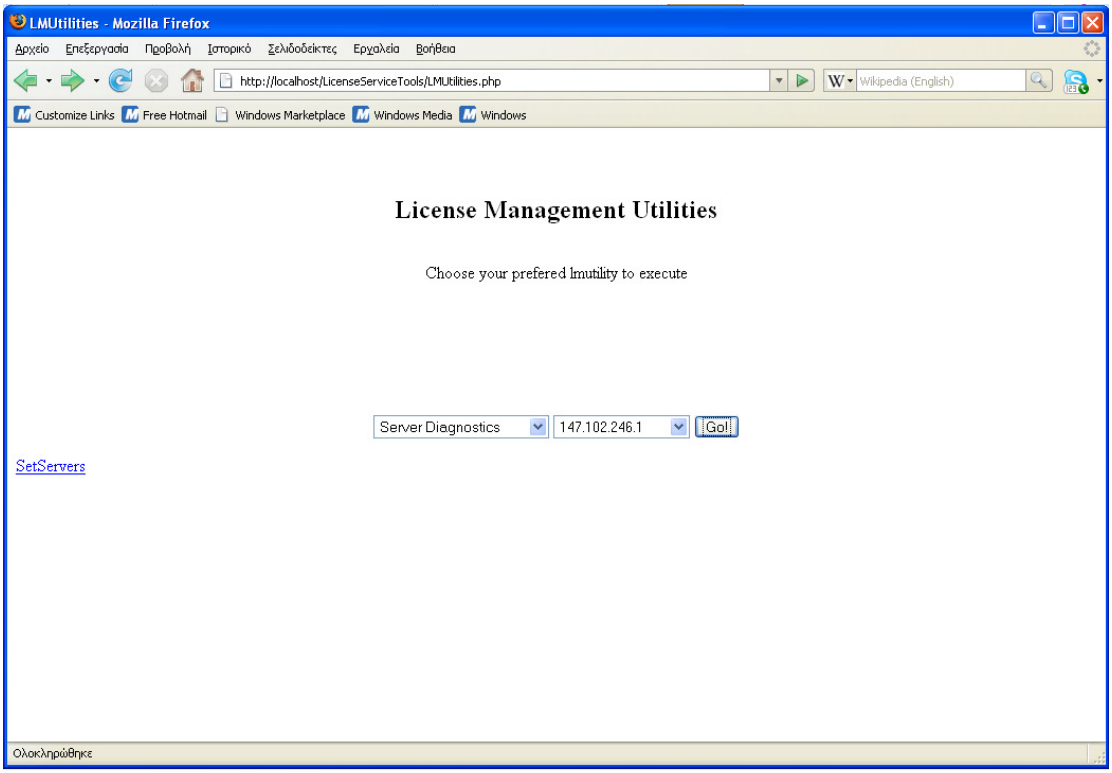
Εικόνα 40: Εργαλεία License Service (2)



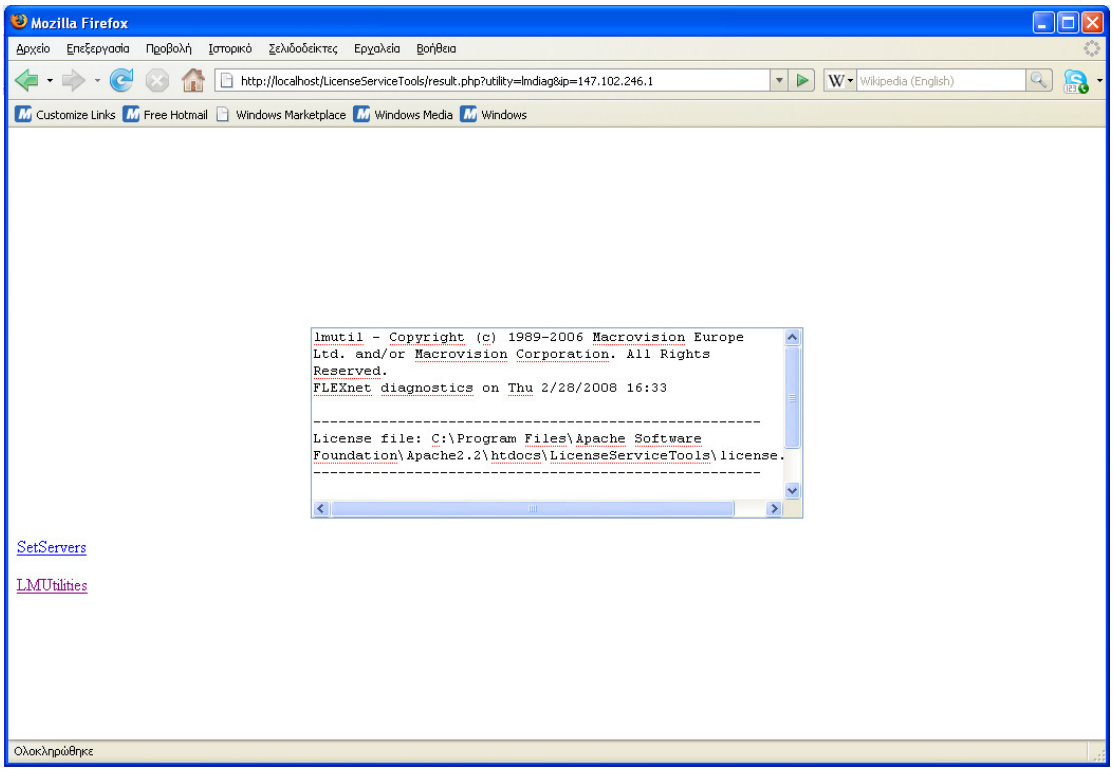
Εικόνα 41: Εργαλεία License Service (3)

Η πρώτη λίστα περιλαμβάνει τα προς χρήση εργαλεία όπως αναλύθηκαν στο κεφάλαιο 3, ενώ η δεύτερη τους FLEXlm Servers στους οποίους θα εφαρμοστούν. Η δεύτερη λίστα περιέχει τους servers που διαβάζει από το αρχείο AvailableServers.txt.

Ο διαχειριστής επιλέγει ένα εργαλείο από την πρώτη λίστα καθώς και τον server στο οποίο θα το εφαρμόσει και πατώντας το κουμπί “GO!” μεταβαίνει στην επόμενη σελίδα όπου παρουσιάζονται τα αποτελέσματα της χρήσης του εργαλείου σε μία περιοχή κειμένου. Στην σελίδα των αποτελεσμάτων ο διαχειριστής μπορεί να μεταβεί στην σελίδα SetServers.php ή στηνLMUtilities.php πατώντας στον αντίστοιχο σύνδεσμο.



Εικόνα 42: Εργαλεία License Service (4)



Εικόνα 43: Εργαλεία License Service (5)

5.2.3 Συνολική Εκτέλεση

Παρακάτω περιγράφεται αναλυτικά όλη η διαδικασία που λαμβάνει χώρα κατά την εκτέλεση μιας FLEXlm εφαρμογής. Όπως έχει αναφερθεί και σε προηγούμενο κεφάλαιο, οι οντότητες που λαμβάνουν μέρος στην διαδικασία είναι:

- Ο End User ο οποίος αποτελείται από την εφαρμογή – πελάτη και τον License Daemon. Η εφαρμογή, διαθέτει την στατική βιβλιοθήκη πελάτη και το License File
- Ο License Provider ο οποίος αποτελείται από τον FLEXlm Server (lmgrd και Vendor Daemon) και το License Service

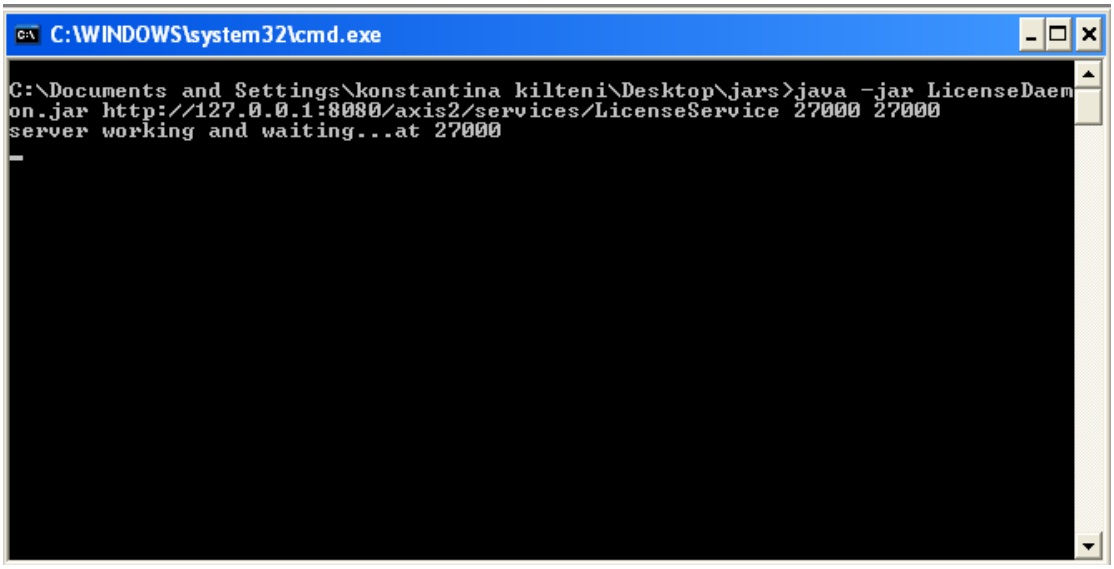
Η διαδικασία που ακολουθείται είναι η εξής:

i. Η στατική βιβλιοθήκη του πελάτη διαβάζει το License File. Συγκεκριμένα, διαβάζει το όνομα του FLEXlm Server με τον οποίο θα συνδεθεί και την θύρα στην οποία θα συνδεθεί. Η διαφορά είναι ότι πλέον στο License File δεν αναγράφεται το όνομα και η θύρα του FLEXlm Server, αλλά του License Daemon, ο οποίος θα παίξει το ρόλο ενός εικονικού FLEXlm Server.

ii. Η εφαρμογή λοιπόν δημιουργεί socket με ορίσματα το όνομα και τη θύρα του License Daemon.

iii. (Σημ: αν γνωρίζει την IP και όχι το όνομα του server, δημιουργεί socket με όρισμα την IP και την θύρα)

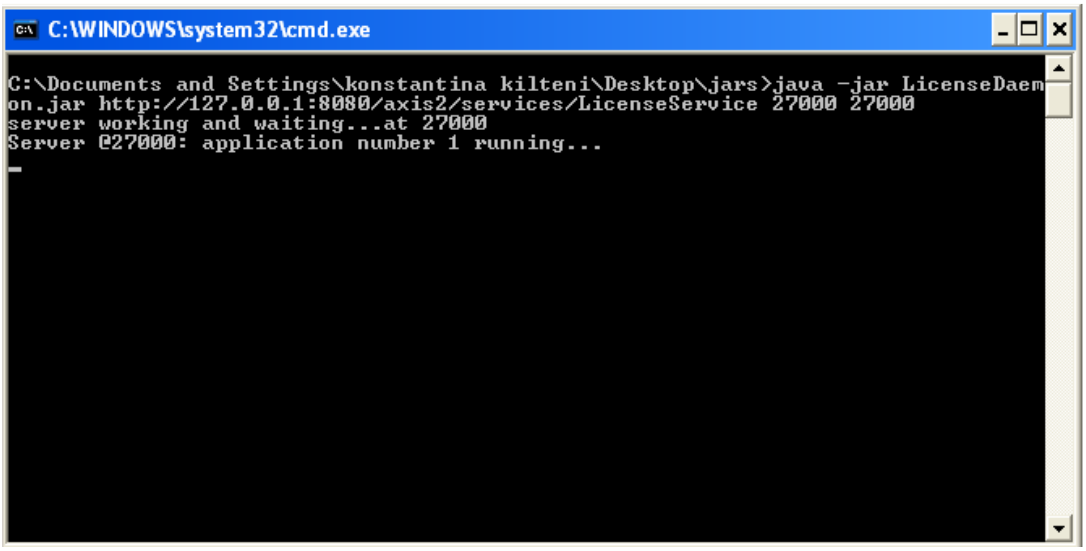
iv. Ο License Daemon έχει υλοποιηθεί προκειμένου να εκτελείται διαρκώς. Έχει ρυθμιστεί από τον διαχειριστή του να ακούει σε συγκεκριμένες θύρες. Στην κονσόλα τερματικού οι ρυθμίσεις αυτές φαίνονται:



Εικόνα 44: Κονσόλα τερματικού με τις ρυθμίσεις του License Daemon

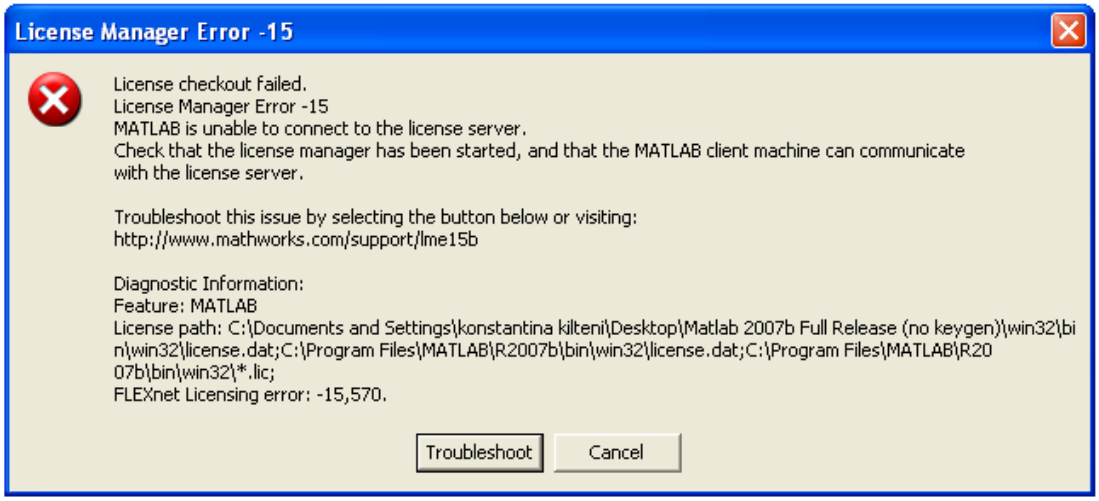
Ο License Daemon εφόσον είναι ρυθμισμένος να ακούει στην συγκεκριμένη θύρα, δέχεται την αίτηση της εφαρμογής και πραγματοποιούν τριμερή χειραψία.

Η κονσόλα τερματικού διαμορφώνεται αντίστοιχα.



Εικόνα 44: Κονσόλα τερματικού – εκκίνηση αδειοδότησης(1)

v. Αν ο License Daemon δεν ακούει στην συγκεκριμένη θύρα, τότε στην πλευρά της εφαρμογής εμφανίζεται μήνυμα λάθους που δηλώνει ότι δεν βρέθηκε ο License Daemon.



Εικόνα 45: Μήνυμα σφάλματος όταν δεν εκτελείται ο License Daemon

vi. Μετά την τριμερή χειραψία, η FLEXlm εφαρμογή αποστέλλει ένα TCP μήνυμα στον License Daemon. Το μήνυμα μεταφέρεται μέσα σε έναν πίνακα τύπου byte από το stream εισόδου.

vii. Με τη λήψη του μηνύματος αυτού (που είναι και το πρώτο) , ο License Daemon καλεί την διαδικτυακή υπηρεσία με την οποία συνεργάζεται, όπως έχει οριστεί από το διαχειριστή του. Στην κλήση περνά στην υπηρεσία ως παράμετρο, το μήνυμα που απεστάλη παραπάνω, το όνομα της μεθόδου της υπηρεσίας από την οποία θα εξυπηρετηθεί και την θύρα. Πρόκειται για την μέθοδο authorization.

viii. Κατά την κλήση της authorization, η διαδικτυακή υπηρεσία αρχικά διαβάζει το αρχείο AvailableServers.txt το οποίο περιέχει τις IP ενδεχόμενων FLEXlm Servers. Επιλέγει τον πρώτο και δημιουργεί socket με όρισμα την IP και την θύρα.

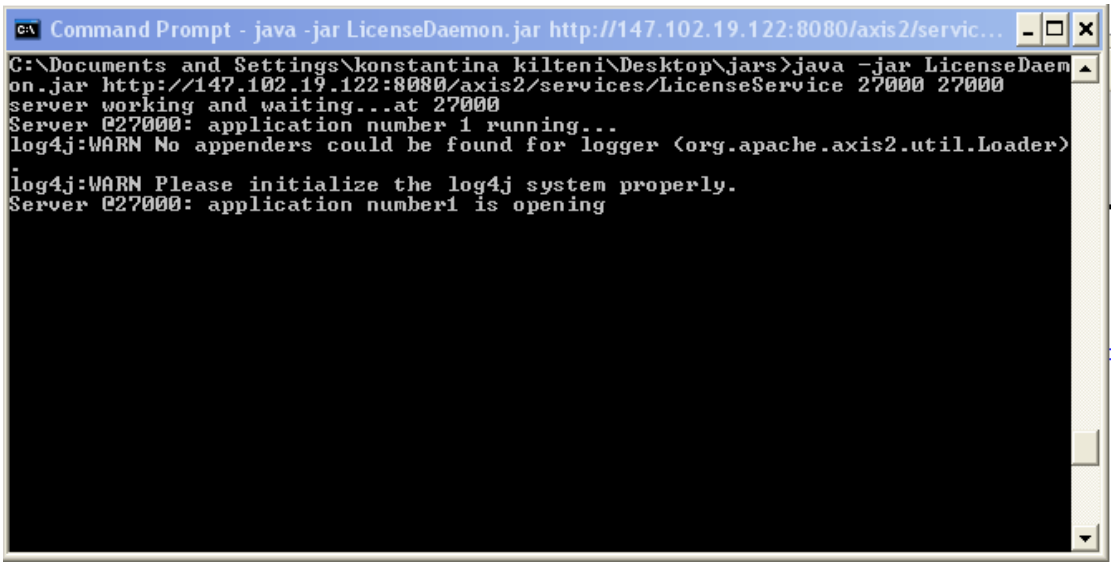
ix. Αν ο εξυπηρετητής που επέλεξε η διαδικτυακή υπηρεσία δεν δεχτεί την αίτηση της, είτε γιατί δεν είναι FLEXlm Server είτε γιατί δεν ακούει στην δεδομένη θύρα, τότε επιστρέφεται σφάλμα κατά την τριμερή χειραψία. Η Διαδικτυακή Υπηρεσία ενημερώνεται και λαμβάνει την επόμενη IP από το αρχείο και ακολουθείται η ίδια διαδικασία. Αν ο FLEXlm Server δεχτεί την αίτηση, ολοκληρώνεται η τριμερής χειραψία μεταξύ διαδικτυακής υπηρεσίας και FLEXlm Server.

x. Η διαδικτυακή υπηρεσία αποστέλλει τον πίνακα που περιέχει το πρώτο μήνυμα, στον FLEXlm Server εγγράφοντας το στο stream εξόδου του.

xi. Ο FLEXlm Server αποκρίνεται στην αίτηση με ένα μήνυμα. Το μήνυμα αυτό, το διαβάζει η διαδικτυακή υπηρεσία και το εγγράφει σε έναν πίνακα τύπου byte. Στην συνέχεια η σύνδεση μεταξύ διαδικτυακής υπηρεσίας και FLEXlm διακόπτεται.

xii. Ο καινούργιος πίνακας που φέρει την απόκριση, επιστρέφεται στον License Daemon ως αποτέλεσμα της μεθόδου authorization.

xiii. Ο License Daemon εγγράφει τον πίνακα απόκρισης στο stream εξόδου αφού πρώτα απομονώσει δύο στοιχεία του. Έτσι στο τέλος, η εφαρμογή λαμβάνει την απόκριση από τον FLEXlm Server που ανταποκρίνεται στην αρχική αίτηση, χωρίς να πραγματοποιηθεί καμία επικοινωνία μεταξύ τους (τουλάχιστον άμεσα).
Εδώ ολοκληρώνεται ο πρώτος κύκλος επικοινωνίας. Η κονσόλα διαμορφώνεται ανάλογα.



Εικόνα 46: Κονσόλα τερματικού – εκκίνηση αδειοδότησης(2)

xiv. Τα δύο στοιχεία που απομόνωσε ο License Daemon τα αποκωδικοποιεί κατάλληλα και λαμβάνει έναν αριθμό που αντιπροσωπεύει την δεύτερη θύρα. Αμέσως, ο License Daemon ρυθμίζεται να ακούει και στην δεύτερη θύρα.

xv. Η εφαρμογή δημιουργεί socket με ορίσματα το ίδιο όνομα (ή την ίδια IP) και την καινούργια θύρα του License Daemon. Ο License Daemon και η εφαρμογή πραγματοποιούν τριμερή χειραψία.

xvi. Ο License Daemon καλεί την μέθοδο makeConnection της διαδικτυακής υπηρεσίας. Κατά την κλήση της makeConnection, δημιουργείται socket με όρισμα την IP του FLEXlm Server και την δεύτερη θύρα.

xxvii. Η εφαρμογή αποστέλλει μήνυμα στον License Daemon. Ο License Daemon αποθηκεύει το μήνυμα σε ένα πίνακα τύπου byte.

xxviii. Ο License Daemon καλεί την μέθοδο της διαδικτυακής υπηρεσίας dataTranfer με όρισμα τον πίνακα.

xxix. Η dataTransfer εγγράφει στο stream εξόδου προς τον FlexLM Server, τον πίνακα.

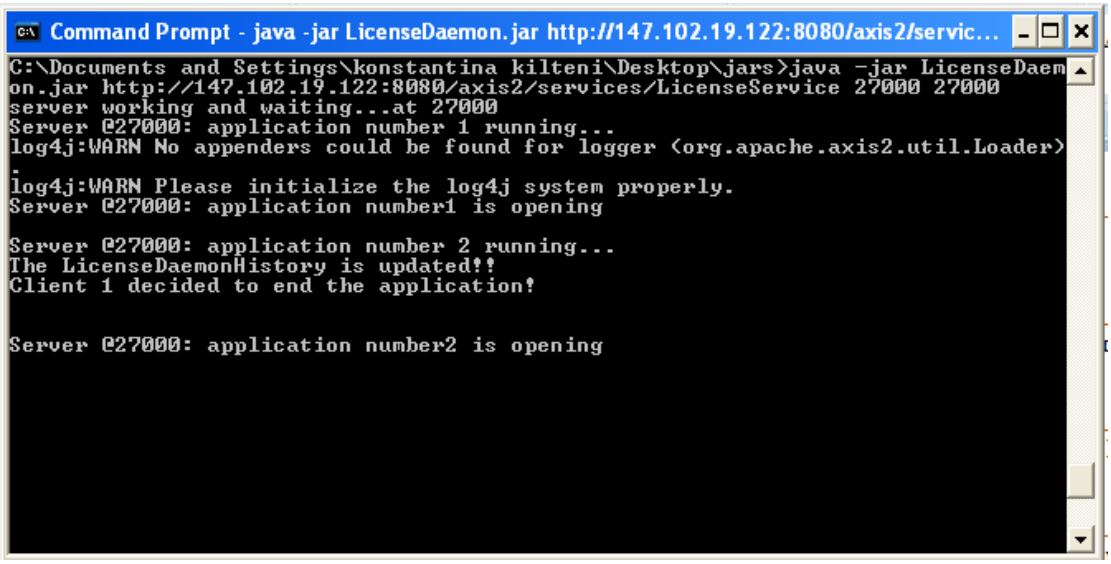
xxx. Ο FlexLM Server αποκρίνεται στο μήνυμα με νέο μήνυμα. Η διαδικτυακή υπηρεσία διαβάζει το μήνυμα από το stream εισόδου και το αποθηκεύει σε έναν νέο πίνακα.

xxxi. Κατόπιν ο πίνακας επιστρέφεται στον License Daemon.

xxxii. Ο License Daemon επιστρέφει τον πίνακα στην εφαρμογή, εγγράφοντας τον στο stream εξόδου.

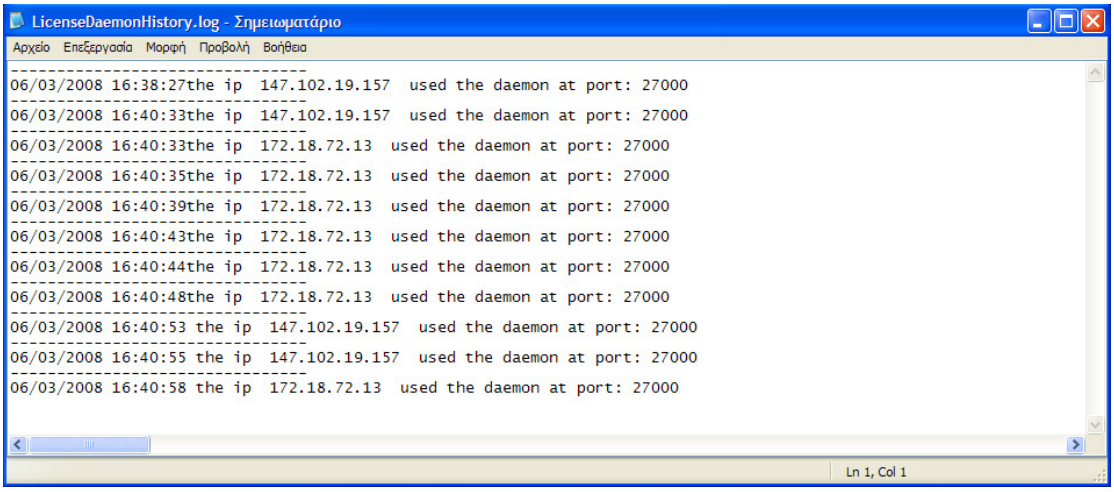
Τα βήματα xvii-xxxii επαναλαμβάνονται διαρκώς. Ακόμη κι όταν ανοίξει πλήρως η εφαρμογή, ανταλλάσσονται μηνύματα μεταξύ εφαρμογής , License Daemon, License Service και FLEXlm Server.

Σε περίπτωση τερματισμού της εφαρμογής, είτε εκούσια είτε ακούσια, η κονσόλα διαμορφώνεται ανάλογα, ενώ κλείνουν οι συνδέσεις μεταξύ εφαρμογής και License Daemon.



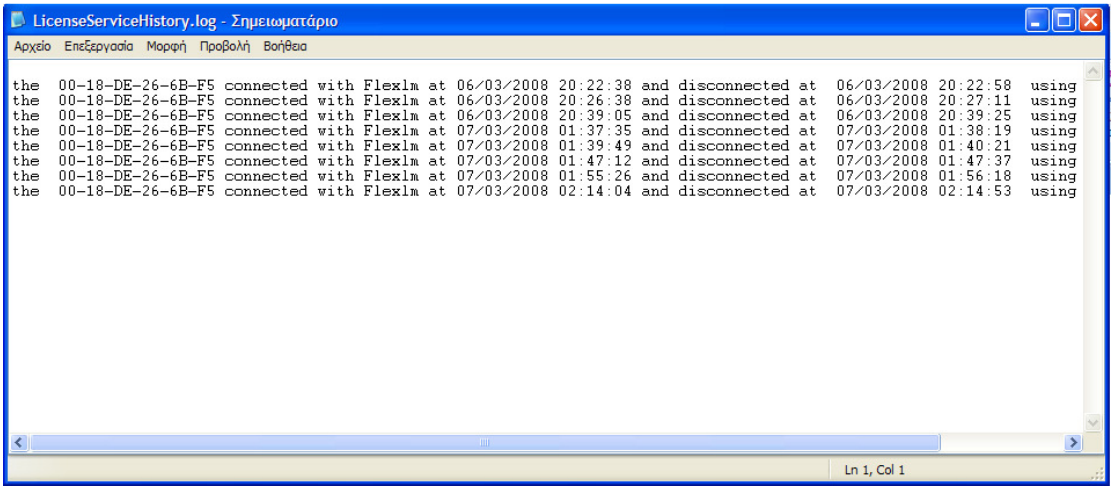
Εικόνα 47: Κονσόλα τερματικού – τερματισμός εφαρμογής

Όλες οι ενέργειες που πραγματοποιήθηκαν στην πλευρά του LicenseDaemon καταγράφονται στο LicenseDaemonHistory.log. Αναφέρεται ποιος συνδέθηκε με ποιόν License Daemon, πότε και ποια ήταν η πρώτη θύρα σύνδεσης.



Εικόνα 48: Ιστορικό License Daemon

Όλες οι ενέργειες που πραγματοποιήθηκαν στην πλευρά του License Service καταγράφονται στο LicenseServiceHistory.log. Αναφέρεται ποιός (ποιά MAC address) και πότε συνδέθηκε με τον FLEXlm Server, ποιά ήταν η πρώτη θύρα και πότε αποσυνδέθηκε.



Εικόνα 49: Ιστορικό License Service

5.2.4 Μετρήσεις, Γραφήματα και Συμπεράσματα

Οι βασικοί παράγοντες που επηρεάζουν την επίδοση του μηχανισμού που αναπτύχθηκε είναι δύο: ο χρόνος και το εύρος ζώνης. Αυτοί οι παράγοντες αφορούν την καθυστέρηση που προστίθεται στη διαδικασία παροχής άδειας καθώς και την αύξηση στο μέγεθος των πακέτων που ανταλλάσσονται εξαιτίας του πρωτοκόλλου SOAP με το οποίο πακετάρονται κατά την κλήση του License Service.

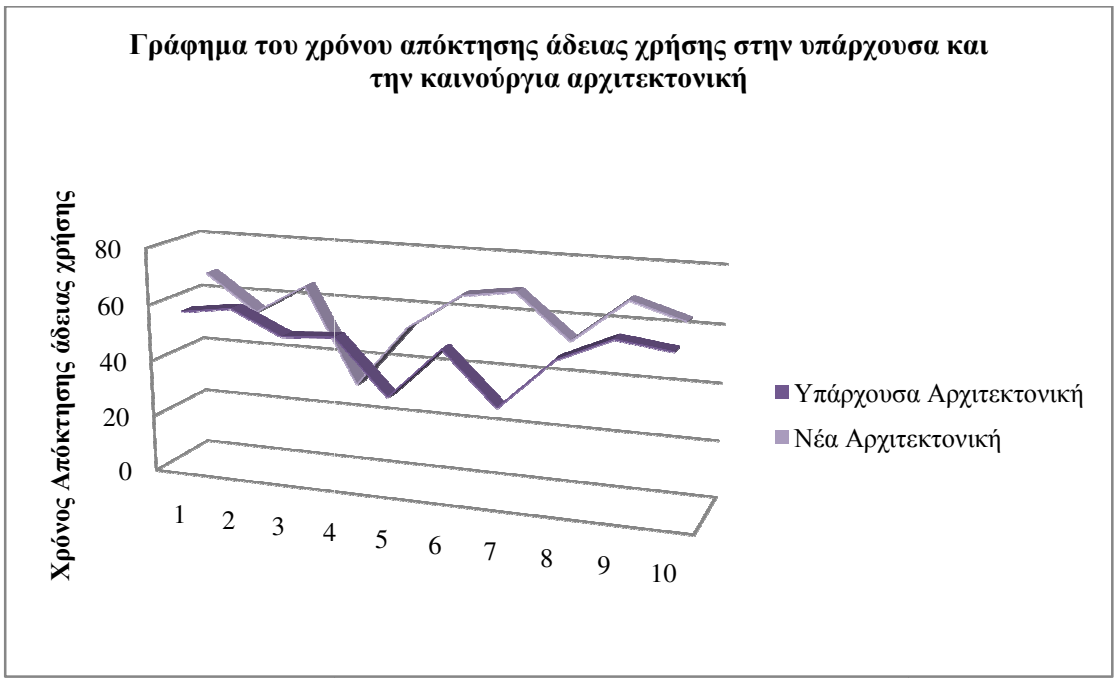
Η ανταλλαγή των TCP μηνυμάτων είναι συνεχής κατά τη διάρκεια εκτέλεσης της εφαρμογής, ενώ η διαδικασία παροχής άδειας διάρκεια μικρότερο χρονικό διάστημα. Μετά την απόκτηση της άδειας, αυτή η δοσοληψία μηνυμάτων συνεχίζεται, για να ελέγχει περιοδικά τη σύνδεση της εφαρμογής με τον εξυπηρετητή. Οι μετρήσεις που παρουσιάζονται στη συνέχεια, αφορούν το χρονικό διάστημα που διαρκεί η διαδικασία παροχής άδειας.

Οι μετρήσεις παρουσιάζονται στον επόμενο πίνακα:

TESTINGS	EXISTING ARCHITECTURE	NEW ARCHITECTURE
1	57.258	67.476
2	59.726	54.689
3	51.294	65.187
4	52.65	31.482
5	33.529	53.432
6	51.65	65.989
7	33.529	68.623

8	51.294	53.322
9	59.726	68.525
10	57.258	62.756

Πίνακας 3: Μετρήσεις χρόνου απόκτησης άδειας στην υπάρχουσα και την νέα αρχιτεκτονική



Εικόνα 50: Γράφημα χρόνου

Παρατηρώντας τον παραπάνω πίνακα, βλέπουμε μια πολύ μικρή καθυστέρηση, η οποία αναμενόταν, αφού η επικοινωνία μεταξύ του FLEXlm Server και της εφαρμογής δεν είναι άμεση.

Οι μετρήσεις του εύρους ζώνης το οποίο αντιπροσωπεύει το μέσο όρο του μεγέθους των πακέτων που ανταλλάσσονται κατά τη διάρκεια της διαδικασίας παροχής άδειας φαίνονται στο σχήμα.

Οι μετρήσεις παρουσιάζονται στον επόμενο πίνακα:

TESTINGS	EXISTING ARCHITECTURE (Bytes/sec)	NEW ARCHITECTURE (Bytes/sec)
1	947	8611
2	812	4034
3	1470	8575
4	447	3950

5	1179	4089
6	986,56	5377
7	1332	7041
8	762,28	1020.50
9	987,72	10106
10	1016	8966

Πίνακας 4: Μετρήσεις χρόνου απόκτησης άδειας στην υπάρχουσα και την νέα αρχιτεκτονική

Εικόνα 51: Γράφημα εύρους ζώνης

Παρατηρούμε ότι το εύρος ζώνης είναι μεγαλύτερο στη νέα αρχιτεκτονική, κάτι το οποίο περιμέναμε εφόσον πλέον το μήνυμα πακετάρεται σύμφωνα με το πρωτόκολλο SOAP κατά την επικοινωνία μεταξύ του License Daemon και του License Service.

6 Επίλογος

Στο παρόν κεφάλαιο γίνεται αρχικά μία ανακεφαλαίωση των αποτελεσμάτων της διπλωματικής εργασίας και παρουσιάζονται ορισμένα συμπεράσματα που εξήχθησαν κατά την πορεία εκπόνησή της και με την ολοκλήρωσή της. Στη συνέχεια γίνεται αναφορά σε μελλοντικές επεκτάσεις που η παρούσα εργασία και το σύστημα που αναπτύχθηκε στα πλαίσια αυτής μπορούν να έχουν.

6.1 Σύνοψη και συμπεράσματα

6.1.1 Σύνοψη

Το παρόν έγγραφο παρουσίασε εκτενώς και με σαφήνεια την υλοποίηση ενός μηχανισμού παροχής άδειας μέσω ενός FLEXlm Server, επιδιώκοντας τη μέγιστη δυνατή ασφάλεια και προστασία του Server, μετατρέποντας τον σε στοιχείο της οντότητας License Provider, η οποία επιτρέπει την πρόσβαση σε μία θύρα της μόνο(8080), ενώ η IP διεύθυνση του FLEXlm Server διατηρείται άγνωστη στις εφαρμογές που ζητούν άδεια από αυτόν. Επιπλέον ο μηχανισμός αυτός παρέχει ευελιξία, μεταφέροντας την επικοινωνία από το επίπεδο μεταφοράς, στο επίπεδο εφαρμογής και επομένως δίνοντας τη δυνατότητα σε μια εφαρμογή να ζητήσει διαδικτυακά την παροχή άδειας για την εκτέλεση της.

Για την αποτελεσματική λειτουργία και την επιτυχή διεκπεραίωση αυτών των στόχων χρησιμοποιήθηκαν τεχνολογίες αιχμής, με πιο βασική τη χρήση Web Service ως βασικό στοιχείο λειτουργίας του μηχανισμού. Κάθε τεχνολογία, πλατφόρμα ανάπτυξης, γλώσσα προγραμματισμού και περιβάλλον εκτέλεσης, επιλέχθηκε προσεχτικά και με σύνεση ώστε να πληροί τις προϋποθέσεις και τις ανάγκες του σχεδιασμού και των στόχων που έπρεπε να επιτελεί ο μηχανισμός παροχής άδειας καθώς και να παρέχει ευέλικτο γραφικό περιβάλλον που εξασφαλίζει ευκολία στη διαχείριση και στη χρήση του. Έτσι ως γλώσσα προγραμματισμού επιλέχθηκε η Java, ως πλατφόρμα ανάπτυξης του κώδικα το eclipse, οι δυναμικές σελίδες διαχείρισης υλοποιήθηκαν σε JSP και PHP, για την εγκατάσταση του web service και των JSP σελίδων επιλέχθηκε ο Tomcat, ενώ για την εγκατάσταση των PHP σελίδων χρησιμοποιήθηκε ο Apache HTTP Server.

Για τον έλεγχο της λειτουργίας της εφαρμογής και για την εξασφάλιση της λειτουργίας της σε διαφορετικά υπολογιστικά περιβάλλοντα, δοκιμές εκτέλεσης πραγματοποιήθηκαν στα λειτουργικά συστήματα Windows και Linux.

6.1.2 Συμπεράσματα

Η ανάγκη των εταιριών πώλησης λογισμικού για εξασφάλιση της γνησιότητας των αντιγράφων λογισμικού που διαθέτουν οι πελάτες τους, σε συνδυασμό με την ανάπτυξη της

τεχνολογίας του διαδικτύου καθιστούν την έρευνα στο αντίστοιχο επιστημονικό πεδίο εξαιρετικά ενδιαφέρουσα. Έπομένως, διαφαίνεται η σπουδαιότητα της υλοποίησης ένος συστήματος παροχής άδειας με το οποίο θα εξασφαλίζεται η μεγαλύτερη αξιοπιστία και ευελιξία της διαδικασίας παροχής άδειας, θα επιτρέπεται ο έλεγχος της επικοινωνίας μεταξύ πελάτη και εξυπηρετητή και θα παρέχεται η όσο το δυνατόν μεγαλύτερη ασφάλεια του παρόχου άδειας. Ο μηχανισμος ο οποίος περιγράφηκε στο παρόν έγγραφο, υλοποιήθηκε με σκοπό να πληροί όλες αυτές τις προϋποθέσεις. Επετεύχθη λοιπόν, με τη μεταφορά της επικοινωνίας μεταξύ πελάτη και εξυπηρετητή, από το επίπεδο μεταφοράς (TCP) στο επίπεδο εφαρμογής η απαραίτητη αξιοπιστία και ευελιξία στη διαχείριση παροχής άδειας. Η παροχή άδειας έχει τη δυνατότητα να πραγματοποιηθεί με την κλήση εξ' αποστάσεως μιας διαδικτυακής υπηρεσίας προσδίδοντας στην όλη διαδικασία εκτός από ευελιξία, και τη δυνατότητα να ελέγχεται επαρκώς. Επιπρόσθετα, η ύπαρξη των επιπλέον στοιχείων (components) τα οποία παρεμβάλλονται μεταξύ πελάτη και εξυπηρετητή εκτός από την μεταφορά της επικοινωνίας σε ανώτερο επίπεδο, διευκολύνει τη διαχείριση της αδειοδότησης σε κατανεμημένα συστήματα. Το επίπεδο της ασφάλειας του παρόχου της άδειας αναβαθμίζεται, αφού ο εξυπηρετητής επιτρέπει την πρόσβαση από τους πελάτες σε μία μόνο θύρα και όχι στο σύνολο των θυρών του, όπως προβλέπει το μοντέλο παροχής άδειας της υπάρχουσας αρχιτεκτονικής, ενώ η ταυτότητα του, παραμένει άγνωστη σε αυτούς.

Επιπλέον, οι προοπτικές και οι δυνατότητες της Grid τεχνολογίας και η ανάγκη για τη θεμελίωση και αξιοποίηση της, επισημαίνουν τη σημασία της ανάπτυξης του μηχανισμού που αναπτύχθηκε, ώστε να δημιουργηθεί η βάση για την υπηρεσία παροχής άδειας σε περιβάλλον Grid. Ο σχεδιασμός εξάλλου της αρχιτεκτονικής του μηχανισμού παροχής άδειας με τρόπο ώστε η χρήση διαδικτυακών υπηρεσιών να είναι βασικό στοιχείο της, είναι ενδεικτικός της δυνατότητας ένταξης σε Grid περιβάλλον.

Για την υλοποίηση του μηχανισμού παροχής άδειας, χρησιμοποιήθηκαν τεχνολογίες αιχμής, όπως είναι οι διαδικτυακές υπηρεσίες, οι οποίες προσέδωσαν την απαραίτητη λειτουργικότητα και συνέβαλλαν και αυτές στη δημιουργία ενός αξιόπιστου, ασφαλούς και ευέλικτου, σύμφωνα με τις προδιαγραφές, συστήματος. Η χρήση προγραμματιστικών λύσεων ελεύθερης χρήσης (open source solutions) όπως Java programming, Linux OS, Tomcat Server, δεν παρουσίασε κανένα απολύτως πρόβλημα κατά την ανάπτυξη του μηχανισμού, παρέχοντας όλες τις απαραίτητες και επιθυμητές λειτουργίες.

Τέλος, η επιτυχής διεκπεραίωση της υλοποίησης, η χρησιμότητα και η αποδοτικότητα της, επιβεβαιώθηκαν μετά από πολλαπλούς ελέγχους και έκδοση ικανοποιητικών αποτελεσμάτων.

6.2 Μελλοντικές επεκτάσεις

6.2.1 Δυνατές Επεκτάσεις-Εκκρεμότητες

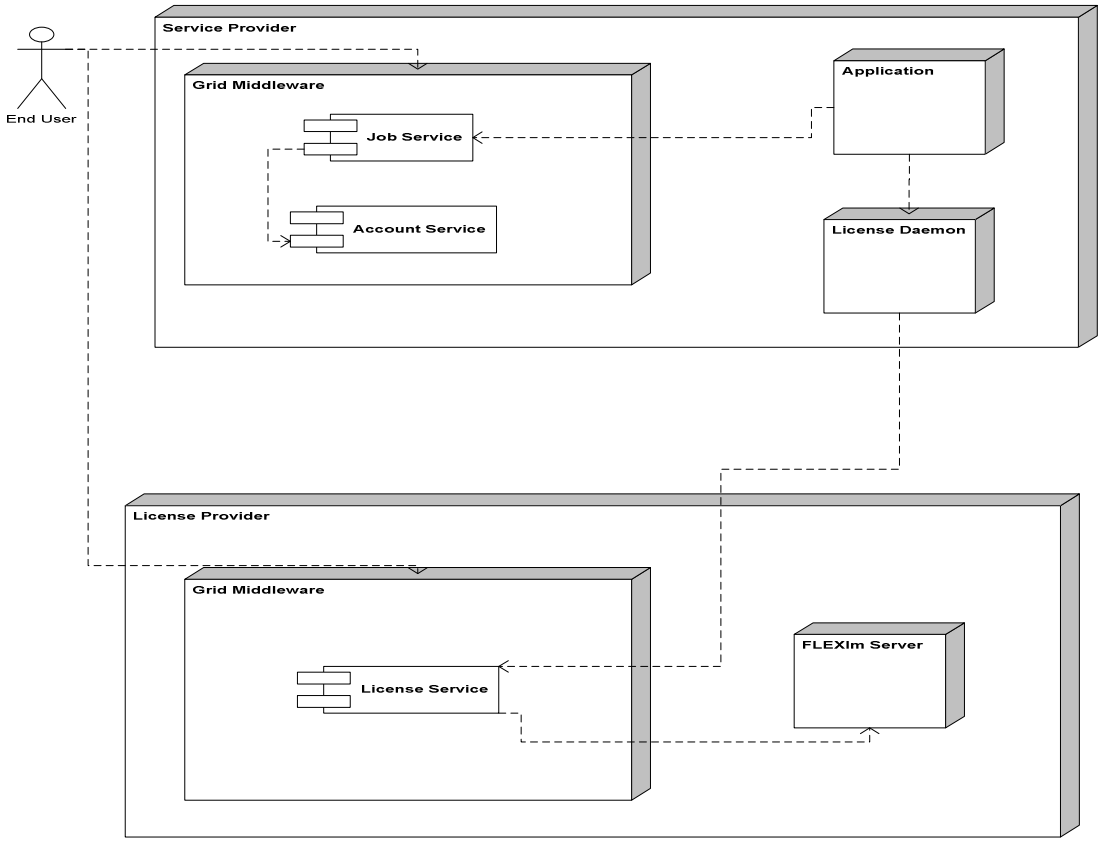
Ο μηχανισμός παροχής άδειας που υλοποιήθηκε αποτελεί μια λύση που λειτουργεί ολοκληρωμένα και ανεξάρτητα, ωστόσο επιδέχεται επεκτάσεις και συμπληρώσεις.

Μια επέκταση του μηχανισμού αποτελεί, η δημιουργία ενός μηχανισμού προστασίας και ελέγχου των υπολογιστών που συνδέονται στον License Provider ώστε να λάβουν άδεια με σκοπό να παρέχεται ακόμα μεγαλύτερη ασφάλεια και προστασία του. Ο License Provider θα

πρέπει επομένως να επιτρέπει μόνο σε trusted End Users να συνδέονται με αυτόν. Μια πιθανή υλοποίηση ενός τέτοιου μηχανισμού θα μπορούσε να γίνει με μηχανισμούς ελέγχου δημόσιου – ιδιωτικού κλειδιού (Public Key Infrastructure PKI) και χρήση πιστοποιητικών ασφαλείας (certificates). Με το σχεδιασμό που παρουσιάστηκε στις ενότητες αυτού του εγγράφου, παρέχεται μεγαλύτερη ασφάλεια από την ήδη υπάρχουσα, όπως έχει αναλυθεί, όμως δεν γίνεται έλεγχος των IP διευθύνσεων των πελατών που ζητούν άδεια ώστε να ελεγχθεί εκ των προτέρων αν διαθέτουν τη δυνατότητα χρήσης του λογισμικού.

Ένας επιπλέον μηχανισμός ασφάλειας μπορεί να εφαρμοστεί στη διαχείριση του License Daemon και του License Service. Στις δυναμικές σελίδες ο διαχειριστής υποχρεούται να εισάγει όνομα και κωδικό πρόσβασης προκειμένου να προβεί σε περαιτέρω ρυθμίσεις. Στην παρούσα υλοποίηση δε γίνεται έλεγχος των στοιχείων του διαχειριστή για να πιστοποιηθεί η ταυτότητά του. Ωστόσο μία ενδεχόμενη σύνδεση της φόρμας εισαγωγής δεδομένων με μία βάση δεδομένων θα μπορούσε να εξασφαλίσει την απαιτούμενη ασφάλεια.

Επιπλέον, μια επέκταση του μηχανισμού παροχής άδειας θα μπορούσε να είναι η ένταξη του σε περιβάλλον Grid. Αυτή η υλοποίηση θα περιλαμβάνει, δύο οντότητες, τον Service Provider και τον License Provider. Όταν ένας χρήστης (End User) του Grid ζητά την εκτέλεση μιας εργασίας (job), έπειτα από μια διαδικασία απαραίτητη για τη χρέωση του χρήστη, η προς εκτέλεση εφαρμογή, επικοινωνεί με το στοιχείο License Daemon του Service Provider, ο οποίος καλεί τη διαδικτυακή υπηρεσία License Service, η οποία αποτελεί στοιχείο του License Provider αλλά πλέον θα αποτελεί και Grid Υπηρεσία. Στη συνέχεια το Licenses Service θα επικοινωνεί με το στοιχείο FLEXlm Server του License Provider και έτσι θα παρέχεται η άδεια χρήσης του λογισμικού. Το σενάριο αυτό φαίνεται στο διάγραμμα της εικόνας 53.



Εικόνα 52: Μια μελλοντική υλοποίηση σε περιβάλλον Grid

7 Βιβλιογραφία

[ASK02] Alan Brown, Simon Johnston, Kevin Kelly, Service – Oriented Architecture and Component – Based Development to Build Web Service Applications, Rational Software Corporation, 2002

[AX08] Γενικές Πληροφορίες σχετικά με τον Apache Axis, διαθέσιμες στην επίσημη ιστοθέση http://ws.apache.org/axis2/

[BDG04] Miguel L. Bote-Lorenzo, Yannis A. Dimitriadis, and Eduardo G´omez-S´anchez , Grid Characteristics and Uses: a Grid Definition, Santiago de Compostela, Spain, Feb. 2004

[Dus05] Ivana Dusparic, Pervasive Application Rights Management Architecture, University of Dublin, July 2005

[EC] IPR – Helpdesk, European Commission DG Enterprise

[Erl04] Thomas Erl, Introduction to Web Services Technology: SOA, SOAP, WSDL and UDDI, September 2004

[GCR] Paul Gustavson, Tram Chase, Larry Root, Karl Crosson, Moving towards a Service – Oriented Architecture (SOA) for Distributed Computing Simulation Environments, SimVentions,Inc.

[JBF+05] Bart Jacob, Michael Brown, Kentaro Fukui, Nihar Trivedi, Introduction to Grid Computing, IBM, December 2005

[SC08] StackComparison, διαθέσιμο στην ιστοθέση http://wiki.apache.org/ws/StackComparison

[Mac04] FLEXlm End Users Guide Version 9.5, Macrovision Corporation, August 2004

[WAx08] Γενικές Πληροφορίες σχετικά με τον Apache Axis, διαθέσιμες στην ιστοθέση http://en.wikipedia.org/wiki/Apache_Axis, 2008

[Wax208] Γενικές Πληροφορίες σχετικά με τον Axis2, διαθέσιμες στην ιστοθέση http://en.wikipedia.org/wiki/Axis2, 2008

[WB08] Γενικές Πληροφορίες σχετικά με τις Διαδικτυακές Υπηρεσίες, διαθέσιμες στην ιστοθέση http://en.wikipedia.org/wiki/Web_service

[Zul08] Farhana Zulkernine , Service Oriented Architecture (SOA), 2008

Παράρτημα

License Daemon

CreateLog.java

```
package tcp;


import java.io.*;
import java.util.Date;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import javax.swing.JTextArea;


    public class CreateLog{
      protected Writer LogOutput ;
      File file;


    public void createFile()throws IOException{
            file = new File("LicenseDaemonHistory.log");
            LogOutput   =   new   BufferedWriter(new   FileWriter(file,
true));
           }


   public void updateFile(String str, int port) throws IOException{


            LogOutput.write('\n');
            LogOutput.write("--------------------------------");
            LogOutput.write('\n');
            DateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy
HH:mm:ss");
```

```
            Date date = new Date();

            LogOutput.write(dateFormat.format(date));

            LogOutput.write('\n');

            LogOutput.write(" the ip   "+str+ "   used the daemon at
port: "+port );

            LogOutput.write('\n');

            System.out.println("The      LicenseDaemonHistory      is
updated!!"); }

      public void updateFile(String str, int port, JTextArea area)
throws IOException{


            LogOutput.write('\n');

            LogOutput.write("--------------------------------");

            LogOutput.write('\n');

            DateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy
HH:mm:ss");

            Date date = new Date();

            LogOutput.write(dateFormat.format(date));

            LogOutput.write('\n');

            LogOutput.write("the ip   "+str+ "   used the daemon at
port: "+port );

            LogOutput.write('\n');

            area.append("The         LicenseDaemonHistory         is
updated!!"+'\n'); }


          public void closeLog(){

                  try{

                        LogOutput.close();}

                  catch(IOException ex){}

          }

}
```

LicenseDaemon.java

```
package tcp;


import java.net.*;

import java.util.concurrent.Callable;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import java.util.concurrent.Future;

import java.io.IOException;

import javax.swing.JTextArea;




public class LicenseDaemon extends Thread {

       ServerSocket s=null;

       ServerSocket s2 = null;

       int port;

       String WebServiceUrl;

       CreateLog log;

       JTextArea area=null;

       boolean SocketCreated = false;


public LicenseDaemon(int i, String Url, CreateLog logfile){

      port = i;

      WebServiceUrl = Url;

      log=logfile;

}


public LicenseDaemon(int i, String Url, CreateLog logfile, JTextArea
area1){
```

```
        port = i;

        WebServiceUrl = Url;

        log=logfile;

        area=area1;

    }

    public void run(){

        Socket incoming=null;

        Socket incoming2=null;

        Simple2 sim2 = null;


        try {

            s = new ServerSocket(port);

        } catch (IOException e1) {}

        try{

            int i = 1;

            if(area==null)

                System.out.println("server working and waiting...at
"+port);

            else

                area.append("server working and waiting...at "+port
+'\n');


        ExecutorService executor =
Executors.newFixedThreadPool(Integer.MAX_VALUE);

        for(;;){

             for(;;){

                incoming = s.accept( );

                Callable<Simple2> t = new
LicenseDaemonHandler(incoming, i,WebServiceUrl, port,area);

                Future<Simple2> submit = executor.submit(t);

                Future<Simple2> f = submit;

                sim2 = f.get();
```

```
                    if (sim2.valid==true)
                         continue;
                    else break;
        }
        if(!SocketCreated){
              s2=new ServerSocket(sim2.port2);
              SocketCreated = true;
        }
        for(;;){
              incoming2 = s2.accept();
              LicenseDaemonHandler2 t2 = new
LicenseDaemonHandler2(incoming2, i,WebServiceUrl,  sim2.port2, log,
port, area);
              t2.start();
              i++;
              break;
        }
}}  catch (Exception e) {
                try{
                      s.close();
                      s2.close();
                      incoming.close();
                      incoming2.close();}
                      catch(IOException e1){} e.printStackTrace();
                 }
    }
}
```

LicenseDaemonAux.java

```
package tcp;


import java.io.*;

import javax.swing.JTextArea;


public class LicenseDaemonAux {


       public int flag;

       public static void main(String[] args){

            try{

                   String Url = args[0];

            }catch(ArrayIndexOutOfBoundsException
a1){System.out.println("wrong. Balte URL!"); System.exit(0);}

            LicenseDaemonAux a=new LicenseDaemonAux(args[0], args[1],
args[2]);

}

      public LicenseDaemonAux(String url, String from, String to){


            CreateLog log=new CreateLog();

            if (from.equals(null)&&to.equals(null)) {

                  for (int m=27000; m<27011;m++){

                        LicenseDaemon  l=new  LicenseDaemon(m,  url,
log);

                        l.start();}

            }

            else{

                  int i=Integer.parseInt(from);

                  int j=Integer.parseInt(to);

                  if (i<=j){

                        for (int k=i; k<=j; k++){
```

```
                        LicenseDaemon  l=new  LicenseDaemon(k,  url,
log);
                        l.start();
                  }
         }
         if (i>j)
                  System.out.println("wrong range!");
      }
}
public LicenseDaemonAux(String url, String from, String to,JTextArea
area){
            CreateLog log=new CreateLog();
            if (from.equals(null)&&to.equals(null)) {
                  for (int m=27000; m<27011;m++){
                        LicenseDaemon  l=new  LicenseDaemon(m,  url,
log, area);
                        l.start();}
                  }
             else {
                  int i=Integer.parseInt(from);
                  int j=Integer.parseInt(to);
                  if (i<=j){
                        for (int k=i; k<=j; k++){
                              LicenseDaemon  l=new  LicenseDaemon(k,
url, log, area);
                              l.start();}
                        }

                  if (i>j){
                        System.out.println("wrong range!");
                        area.append("wrong range!");
}}}}
```

LicenseDaemonHandler.java

```
package tcp;


import java.io.*;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.concurrent.Callable;

import javax.swing.JTextArea;



 class LicenseDaemonHandler extends Thread  implements
Callable<Simple2> {


      Socket incoming;

      int counter;

      String url;

      String ip;

      int first;

      InputStream in;

      OutputStream out;

      private ServerSocket s2;

      private int secondport;

      private boolean valid = false;

      Simple sim = new Simple(null, counter);

      Simple2 sim2 ;

      JTextArea area;


       public   LicenseDaemonHandler(Socket   i,   int   c,   String
WebServiceUrl, int firstl)         {

            incoming = i;

            counter = c;
```

```
            url=WebServiceUrl;

            first=firstl;

}



       public   LicenseDaemonHandler(Socket   i,   int   c,   String
WebServiceUrl, int firstl, JTextArea area1)         {

             incoming = i;

             counter = c;

             url=WebServiceUrl;

             first=firstl;

             area=area1;

}

       public Simple2 call(){

            try {

                    int i=this.counter;

                    byte[] message=new byte[2048];


                   if(area==null)

                        System.out.println("Server        @"+first+":
application number "+i+" running...");

                   else

                        area.append("Server  @"+  first+":  application
number "+i+" running..."+'\n');


                   in = incoming.getInputStream();

                   out =incoming.getOutputStream();

                  int len= in.read(message);

                  LicenseServiceClient        lsc        =        new
LicenseServiceClient();


                 sim = lsc.deamon1(len, message, url, first);
```

```
                 out.write(sim.pinax, 0, sim.length);

                 int ib = sim.pinax[32] & 0xff;

                int ib2 = sim.pinax[33] & 0xff;

                String port2 =
Integer.toString(ib)+Integer.toString(ib2);

                  secondport=(sim.pinax[32]&0xff)*16*16
+(sim.pinax[33] & 0xff);

       }catch (Exception e){

                     e.printStackTrace();

                     valid = true;

                     if(area==null)

                              System.out.println("Server  @"+first+":
Authorization failed.");

                    else

                              area.append("Server         @"+first+":
Authorization failed."+'\n');


       try {

            in.close();

            out.close();

            incoming.close();

            counter++;

      } catch (IOException e1) {

                  e1.printStackTrace();

      }

}

sim2 = new Simple2(secondport,valid);

return sim2;

} }
```

LicenseHandler2.java

```
package tcp;

import java.io.InputStream;
import java.io.OutputStream;
import java.io.IOException;
import java.net.Socket;
import javax.swing.JTextArea;



public class LicenseDaemonHandler2 extends Thread{

       int counter;
       Socket incoming2;
       int port;
       String url;
       String ip;
       LicenseServiceClient lsc2;
       CreateLog log;
       int port1;
       JTextArea area=null;
       public LicenseDaemonHandler2(Socket i,int number, String urll,
int portl, CreateLog logo, int firstPort){
      counter=number;
      incoming2=i;
      port=portl;
      url=urll;
      log=logo;
      port1=firstPort;
}

       public LicenseDaemonHandler2(Socket i,int number, String urll,
int portl, CreateLog logo, int firstPort, JTextArea area1){
                  counter=number;
                  incoming2=i;
                  port=portl;
                  url=urll;
                  log=logo;
                  port1=firstPort;
                  area=area1;
}
       public void  run(){

                  InputStream in2 = null;
                  try {in2 = incoming2.getInputStream();} catch
(IOException e1) {e1.printStackTrace();}
                  OutputStream out2 = null;

                  try {out2 = incoming2.getOutputStream();
                  } catch (IOException e1) {e1.printStackTrace(); }

               int i=this.counter;
               byte[] data=new byte[2048];
               int len3=0;
               Simple sim = new Simple(null, counter);
               try{
                       lsc2 = new LicenseServiceClient();
                       lsc2.makeConnection(port, url);
```

```
                if(area==null)
                    System.out.println("Server @"+port1+":
application number"+counter+" is opening"+ '\n');
                else
                  area.append("Server @"+port1+": application
number"+counter+" is opening"+'\n');

                try{
                    do{ len3=transfer(len3, in2, out2, data,sim);}
                    while(true);
                }catch(IOException e){}

              try{
              this.log.createFile();
              if (area==null)

                  this.log.updateFile(incoming2.getInetAddress().toSt
                  ring(), port1 );
              else

                  this.log.updateFile(incoming2.getInetAddress().toSt
                  ring(), port1, area );
                  this.log.closeLog();
              }catch (IOException ex){}
             NetworkInfo nti= new NetworkInfo();

             lsc2.closeConnection(url, nti.getMacAddress());


             if(area==null){
                  System.out.println("Client "+i+ " decided to end
            the application!");
                  System.out.println('\n');
            }
            else{
                  area.append("Client  "+i+ " decided to end the
            application!"+'\n');
                  System.out.println('\n');}

                  incoming2.close();
                  in2.close();
                  out2.close();
                  i++;
            } catch (Exception e){}

          }

          public synchronized int transfer(int len3, InputStream in2,
OutputStream out2, byte[] data, Simple sim) throws IOException,
InterruptedException{


              len3 = in2.read(data);
              sim =  lsc2.deamon2(len3, data, url);
              out2.write(sim.pinax, 0, sim.length);
              return len3;

          }
        }
```

LicenseServiceClient.java

```
package tcp;


import java.io.IOException;

import javax.xml.namespace.QName;

import org.apache.axis2.AxisFault;

import org.apache.axis2.addressing.EndpointReference;

import org.apache.axis2.client.Options;

import org.apache.axis2.rpc.client.RPCServiceClient;

import sun.misc.BASE64Decoder;

import tcp.Simple;


public class LicenseServiceClient  {


      public synchronized Simple  deamon1( int len, byte[] message,
      String url, int firstPort)  throws IOException {

            BASE64Decoder en1 = new BASE64Decoder();

            RPCServiceClient serviceClient = new RPCServiceClient();

            Options options = serviceClient.getOptions();

            EndpointReference targetEPR = new EndpointReference(url);

            options.setTo(targetEPR);

            Integer len1 = new Integer(len);

            Integer len2= new Integer(firstPort);

            QName  opGetl  =  new  QName("http://ws.apache.org/axis2",
"authorisation");

            Object[] opLArgs = new Object[] { len1, message, len2 };

            Class[] returnTypes = new Class[] {String.class};

            Object[]  response  =  serviceClient.invokeBlocking(opGetl,
opLArgs, returnTypes );

            String result = (String)response[0];

            byte [] pinakas = new byte[result.length()];
```

```
            pinakas = en1.decodeBuffer(result);

            Simple result1 =  new Simple(pinakas,pinakas.length);

            return result1;

            }


      public synchronized Simple   deamon2( int len, byte[] data,
String url) throws AxisFault {


            BASE64Decoder en1 = new BASE64Decoder();

            RPCServiceClient serviceClient = new RPCServiceClient();

            Options options = serviceClient.getOptions();

            EndpointReference targetEPR = new EndpointReference(url);

            options.setTo(targetEPR);

            QName opGetl = new QName("http://ws.apache.org/axis2",
"dataTranfer");

            Integer len1=new Integer(len);

            Object[] opLArgs = new Object[] { len1, data };

            Class[] returnTypes = new Class[] {String.class };

            Object[] response = serviceClient.invokeBlocking(opGetl,
opLArgs, returnTypes);


           String result = (String) response[0];

           byte [] pinakas = new byte[result.length()];

           try {

                  pinakas = en1.decodeBuffer(result);

            } catch (IOException e) {

                        e.printStackTrace();

            }


           Simple result2 =  new Simple(pinakas,pinakas.length);

           return result2;

            }
```

```
      public synchronized  void makeConnection(int port, String url)
throws AxisFault{

            RPCServiceClient serviceClient = new RPCServiceClient();

            Options options = serviceClient.getOptions();

            EndpointReference targetEPR = new EndpointReference(url);

            options.setTo(targetEPR);

            QName  opGetl  =  new  QName("http://ws.apache.org/axis2",
"makeConnection");//

            Integer p=new Integer(port);

            Object[] opLArgs = new Object[] {p};

            Class[] returnTypes = new Class[] {String.class};

            serviceClient.invokeBlocking(opGetl, opLArgs, returnTypes
);

   }


 public   void   closeConnection(String   url,   String   MAC)   throws
AxisFault{

            RPCServiceClient serviceClient = new RPCServiceClient();

            Options options = serviceClient.getOptions();

            EndpointReference targetEPR = new EndpointReference(url);

            options.setTo(targetEPR);

            QName  opGetl  =  new  QName("http://ws.apache.org/axis2",
"closeConnection");

            Object[] opLArgs = new Object[] { MAC };

            Class[] returnTypes = new Class[] {String.class};

            serviceClient.invokeBlocking(opGetl, opLArgs, returnTypes
);

}}
```

NetworkInfo.java

```
package tcp;

import java.net.InetAddress;
import java.io.InputStream;
import java.io.BufferedInputStream;
import java.io.IOException;
import java.text.ParseException;
import java.util.StringTokenizer;


public final class NetworkInfo {

	static String getMacAddress() throws IOException {

		try {
			return parseMacAddress(runIpConfigCommand());
		} catch(ParseException ex) {
			ex.printStackTrace();
			throw new IOException(ex.getMessage());
		}
	}



	private final static String parseMacAddress(String
ipConfigResponse) throws ParseException {
		String localHost = null;
		try {
			localHost =
InetAddress.getLocalHost().getHostAddress();
		} catch(java.net.UnknownHostException ex) {
			ex.printStackTrace();
			throw new ParseException(ex.getMessage(), 0);
		}

		StringTokenizer tokenizer = new
StringTokenizer(ipConfigResponse, "\n");
		String lastMacAddress = null;

		while(tokenizer.hasMoreTokens()) {
			String line = tokenizer.nextToken().trim();

			// see if line contains IP address
			if(line.endsWith(localHost) && lastMacAddress !=
null) {
				return lastMacAddress;
			}

			// see if line contains MAC address
			int macAddressPosition = line.indexOf(":");
			if(macAddressPosition <= 0) continue;

			String macAddressCandidate =
line.substring(macAddressPosition + 1).trim();
			if(isMacAddress(macAddressCandidate)) {
				lastMacAddress = macAddressCandidate;
				continue;
			}
```

```
            }

            ParseException ex = new ParseException("cannot read MAC
address from [" + ipConfigResponse + "]", 0);
            ex.printStackTrace();
            throw ex;
      }


      private final static boolean isMacAddress(String
macAddressCandidate) {
            if(macAddressCandidate.length() != 17) return false;

            return true;
      }


      private final static String runIpConfigCommand() throws
IOException {
            Process p = Runtime.getRuntime().exec("ipconfig /all");
            InputStream stdoutStream = new
BufferedInputStream(p.getInputStream());

            StringBuffer buffer= new StringBuffer();
            for (;;) {
                  int c = stdoutStream.read();
                  if (c == -1) break;
                  buffer.append((char)c);
            }
            String outputText = buffer.toString();

            stdoutStream.close();

            return outputText;
      }



}
```

Simple.java

```
package tcp;


public class Simple {


      public byte [] pinax;
      public int length;

      public Simple(byte[] p, int l){
            pinax = p ;
            length = l ;
      }
}
```

Simple2.java

```
package tcp;

public class Simple2 {

      public int port2;
      public boolean valid;

      public Simple2(int p2, boolean flag){
            port2 = p2 ;
            valid = flag;
      }

}
```

UserInterface.java

```
package tcp;
import javax.swing.*;
import java.awt.event.*;



public class UserInterface extends JFrame implements ActionListener{

      private static final long serialVersionUID = 1L;
      boolean choices=false;
      String Ip;
      String Url=null;
      String From=null;
      String To=null;
      String Standalone=null;
      int i=0;

      //JTextFields
      JTextField url=new JTextField(35);

      JTextField standalone=new JTextField(4);
      JTextField from=new JTextField(4);
      JTextField to=new JTextField(4);

      //JButtons & JRadioButtons

      ButtonGroup choice =new ButtonGroup();
      JButton start=new JButton("START!");
      JButton stop=new JButton("STOP!");
      JButton clear=new JButton("CLEAR!");

      JRadioButton RangePorts= new JRadioButton ("Range
", true);
      JRadioButton StandPort =new JRadioButton("Standalone Port:
");


      JTextArea area=new JTextArea(15, 33);
```

```
      JScrollPane scroll;

      JLabel toLabel=new JLabel("To: ");
      JLabel fromLabel=new JLabel("From: ");

      JPanel pane;


      public UserInterface(){
            super("Server's Settings");
            setSize(400, 550);
            setDefaultCloseOperation(EXIT_ON_CLOSE);
            pane= new JPanel();

            JLabel urlLabel=new JLabel("Url of the LicenseService:
");
            JLabel range= new JLabel("                  Range of
Ports of License Daemon:          ");
            JLabel allign= new JLabel("
");

            standalone.setEditable(false);
            ButtonGroup ranges=new ButtonGroup();
            ranges.add(RangePorts);
            ranges.add(StandPort);

            RangePorts.addActionListener(this);
            StandPort.addActionListener(this);
            stop.addActionListener(this);

            area.setLineWrap(true);
            area.setEditable(false);

            scroll=new JScrollPane(area,
            ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS,
            ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);

            start.addActionListener(this);

            clear.addActionListener(this);


            pane.add(urlLabel);
            pane.add(url);
            pane.add(range);
            pane.add(allign);
            pane.add(allign);
            pane.add(allign);
            pane.add(RangePorts);
            pane.add(fromLabel);
            pane.add(from);
            pane.add(toLabel);
            pane.add(to);
            pane.add(StandPort);
            pane.add(standalone);
            pane.add(scroll);
            pane.add(start);
            pane.add(stop);
            pane.add(clear);
            setContentPane(pane);
```

```
            setVisible(true);


}
      public void actionPerformed(ActionEvent evt){

            if (evt.getSource()==RangePorts){
                   from.setEditable(true);
                   to.setEditable(true);
                   standalone.setEditable(false);

            }

            if (evt.getSource()==StandPort){
                  standalone.setEditable(true);
                  from.setEditable(false);
                  to.setEditable(false);
            }



            if (evt.getSource()==start) {

            if (RangePorts.isSelected()){
                  Url=url.getText();
                  From=from.getText();
                  To=to.getText();


      if((!Url.equals(""))&&(!From.equals(""))&&(!To.equals("")))
                  {LicenseDaemonAux a=new LicenseDaemonAux(Url, From,
To, area); start.setEnabled(false);}
            else ;
            }

            else //if (StandPort.isSelected())  {

                  Url=url.getText();
                  Standalone=standalone.getText();
                  if((!Url.equals(""))&&(!Standalone.equals("")){

                        LicenseDaemonAux a=new LicenseDaemonAux(Url,
Standalone, Standalone, area);
                        start.setEnabled(false);
                  }
                  else ;
                  }

        }


            if(evt.getSource()==clear){
                  area.setText("");
            }
            if (evt.getSource()==stop) {
                  System.exit(0);
         }
```

```
	}
	public static void main(String[] args){
		UserInterface u=new UserInterface();
	}

}
```

License Service

LicenseService.java

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.InetSocketAddress;
import java.net.Socket;
import java.net.SocketException;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.StringTokenizer;
import sun.misc.BASE64Encoder;



public class LicenseService {
	byte[] ans = new byte[2048];

	IOException exe=new IOException();
	OutputStream  outFlexlm;
	InputStream  inFlexlm;

	OutputStream  outFlexlmData;
	InputStream  inFlexlmData;

	Socket outgoingFlexlmForAuthorisation =null;
	Socket outgoingFlexlmForData;
	int len3,leng;
	//byte[] pin = new byte[2048];
	String ip;
	int totalServers=1;
	int len4=0;

	public String hostname;
	public String timeStart;
	public String timeStop;
	public int portf;

	String filename = "/Documents and
Settings/LicenseServiceHistory.log";
```

```
      File f;
      BufferedWriter output;


      public String[] viewServers() throws IOException{

            totalServers=1;
            String line="tina";
            BufferedReader in = null;
            String[] servers= new String[20];
            File f=new File("/Documents and
Settings/AvailableServers.txt");

            in = new BufferedReader(new   FileReader(f));
            line=in.readLine(); //read a line as a a String

            while (line!=null){
                  StringTokenizer strToken = new
StringTokenizer(line, "");

            while(strToken.hasMoreTokens()){
                  servers[totalServers]=strToken.nextToken();
                  totalServers++;
            }
            line=in.readLine();
                      }
            return servers;

            }

      public String authorisation (Integer len1, byte[] message,
Integer len2) throws IOException {

      String[] ips=viewServers();
      outFlexlm = null;
      inFlexlm = null;
      int port1=len2.intValue();
      int i=0;
      double counter=Math.random();
      for(;;){
              i++;
              ip=ips[i];
              System.out.println(ip + " "+ counter);
              try{
                  outgoingFlexlmForAuthorisation=new Socket(ip,
port1);
                  outgoingFlexlmForAuthorisation.setSoTimeout(1000);
                  outFlexlm =
outgoingFlexlmForAuthorisation.getOutputStream();

                  inFlexlm =
                  outgoingFlexlmForAuthorisation.getInputStream();

                  outFlexlm.write(message, 0,len1.intValue() );

                  len4 = inFlexlm.read(ans);
                  if ((len4==-1)||(len4==0))
                        throw exe;

                  else break;
```

```
                  }catch(IOException ex){System.out.println("io. " +
counter+ " stin " + ip);
        outgoingFlexlmForAuthorisation= new Socket(ips[1], port1);

        outgoingFlexlmForAuthorisation.close();
        outFlexlm.close();
        inFlexlm.close();

        continue;}

        }
       System.out.println("connected with FlexlmServer at " + port1+
"!");

 String resp= null;
 byte[] ans2 = new byte[len4];
 portf = len2.intValue();

for(int j=0;j<len4;j++){
      ans2[j]= ans[j];
                  }
BASE64Encoder en = new BASE64Encoder();
resp = en.encode(ans2);
return resp;
}



      public void makeConnection(int port) throws IOException{

            DateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy
HH:mm:ss");
            Date date = new Date();
            timeStart = dateFormat.format(date);

            f = new File(filename);
            output = new BufferedWriter(new FileWriter(f, true));


            outgoingFlexlmForData= new Socket();
            outgoingFlexlmForData.connect(new InetSocketAddress(ip,
port));
            System.out.println("connected with Flexlm at "+port+"
!");

      }

      public String dataTranfer(int len2, byte[] data) throws
IOException{

             System.out.println("data transfer started");
             outFlexlmData = outgoingFlexlmForData.getOutputStream();
             inFlexlmData = outgoingFlexlmForData.getInputStream();
             String resp2 = null;

             try {

                   outFlexlmData.write(data, 0,len2 );
                   len3 = inFlexlmData.read(ans);
                   byte[] ans8 = new byte[len3];
```

```
                  for(int j=0;j<len3;j++){
                        ans8[j]= ans[j];
                  }
                  BASE64Encoder en = new BASE64Encoder();
                  resp2 = en.encode(ans8);

             }catch(IOException exception) {
                  System.out.println("problhma edw");}

                  System.out.println("data transfer ended");

     return resp2;
}




      public void closeConnection(String host) throws
SocketException{


            try{
                  DateFormat dateFormat = new
SimpleDateFormat("dd/MM/yyyy HH:mm:ss");
                  Date date = new Date();
                  timeStop = dateFormat.format(date);

                  output.newLine();
                  output.write( "the  " + host +" connected with
Flexlm at " + timeStart + " and disconnected at  " + timeStop + "
using the port  " + portf);
                  output.close();
                  outFlexlm.close();
                  inFlexlm.close();
                  outgoingFlexlmForAuthorisation.close();
                  System.out.println("close");
                  }
            catch(Exception ex4){System.out.println("edw exoume
exception");}


      }
}
```

JSP

index.jsp

```
<%@ page contentType="text/html; charset=iso-8859-1" language="java"
import="java.sql.*" errorPage="" %>

<%@ page import="java.util.Date"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
1" />

<title>index</title>

<style type="text/css">

<!--

body {

      background-color: #FFFFFF;

      background-image: url();

}

.style1 {

      color: #000000;

      font-weight: bold;

      font-size: 24px;

}

.style2 {

      color: #FFFFFF;

      font-weight: bold;

}

#Layer1 {

      position:absolute;

      left:282px;

      top:100px;

      width:202px;

      height:43px;

      z-index:1;

}

.style3 {color: #000000}

#Layer2 {

      position:absolute;
```

```
      left:497px;
      top:139px;
      width:243px;
      height:123px;
      z-index:2;
}
.style5 {color: #000000; font-size: 18px; }
#Layer3 {
      position:absolute;
      left:444px;
      top:299px;
      width:135px;
      height:61px;
      z-index:3;
}
-->
</style>
<script type="text/JavaScript">
<!--
function MM_findObj(n, d) { //v4.01
  var          p,i,x;                     if(!d)          d=document;
if((p=n.indexOf("?"))>0&&parent.frames.length) {
    d=parent.frames[n.substring(p+1)].document; n=n.substring(0,p);}
  if(!(x=d[n])&&d.all) x=d.all[n]; for (i=0;!x&&i<d.forms.length;i++)
x=d.forms[i][n];
  for(i=0;!x&&d.layers&&i<d.layers.length;i++)
x=MM_findObj(n,d.layers[i].document);
  if(!x && d.getElementById) x=d.getElementById(n); return x;
}


function MM_validateForm() { //v4.0
```

```
  var
i,p,q,nm,test,num,min,max,errors='',args=MM_validateForm.arguments;

  for    (i=0;    i<(args.length-2);    i+=3)    {    test=args[i+2];
val=MM_findObj(args[i]);

    if (val) { nm=val.name; if ((val=val.value)!="") {

      if (test.indexOf('isEmail')!=-1) { p=val.indexOf('@');

        if (p<1 || p==(val.length-1)) errors+='- '+nm+' must contain
an e-mail address.\n';

      } else if (test!='R') { num = parseFloat(val);

        if (isNaN(val)) errors+='- '+nm+' must contain a number.\n';

        if (test.indexOf('inRange') != -1) { p=test.indexOf(':');

          min=test.substring(8,p); max=test.substring(p+1);

          if (num<min || max<num) errors+='- '+nm+' must contain a
number between '+min+' and '+max+'.\n';

    } } } else if (test.charAt(0) == 'R') errors += '- '+nm+' is
required.\n'; }

  } if (errors) alert('The following error(s) occurred:\n'+errors);

  document.MM_returnValue = (errors == '');

}

//-->

</script>

</head>


<body>

<div class="style3" id="Layer1">

   <div align="center">

     <%! Date date=new Date();%>

     <%  out.println("the date is:" +date); %>

   </div>

</div>

<p class="style1">LICENSE DAEMON ADMINISTRATOR's PAGE</p>

<p> </p>
```

```
<form id="form1" name="form1" method="post" action="medium.jsp">

  <label> </label>

  <div align="center">

    <p><span class="style2"><br />

          <br />

          <br />

          <br />

    </span></p>

    <p            align="center"><span            class="style2"><span
class="style5">username:</span></span> <span class="style3">

      <input name="username" type="text" value=" admin" />

    </span></p>

    <p                                          align="center"><span
class="style5"><strong>password</strong></span><span
class="style3"><strong>: </strong></span><span class="style2">

      <input name="password" type="password" />

    </span></p>

    <p align="center"><span class="style2">

      <input                name="submit"                type="submit"
onclick="MM_validateForm('username','','R','password','','R');return
document.MM_returnValue" value="submit" />

    </span></p>

    <p><br />

    </p>

    <label></label>

  </div>

  <p align="center"> </p>

  <label></label>

</form>

<p> </p>

<p align="center"> </p>

</body>
```

```
</html>
```

media.jsp

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>

<%
if (request.getMethod().equals("POST"))
{
String str5 = request.getParameter("choice");
String str2 = request.getParameter("url");
String str3 = request.getParameter("from");
String str4 = request.getParameter("to");
String str6 = request.getParameter("StandalonePort");
//out.println(str2+"&"+str3+"&"+str4+"&"+str5+"&"+str6);
//out.println(str5);
out.println("Starting daemons...");
out.println("<p/>");

if (str5.equals("Range"))
{
out.println("Settings: "+'\n');
out.println("<p/>");
out.println("Url= "+ str2+'\n');
out.println("<p/>");
out.println("From= "+ str3+'\n');
out.println("<p/>");
out.println("To= "+ str4+'\n');
```

```
Process p1 = Runtime.getRuntime().exec("cmd.exe /c start java -jar
LicenseDaemon.jar "+str2+" "+str3+" "+str4);


}

if (str5.equals("StandalonePort"))

{

out.println("Settings: ");

out.println("<p/>");

out.println("Url= "+ str2);

out.println("<p/>");

out.println("StandAlonePort= "+ str6);


Process p2 = Runtime.getRuntime().exec("cmd.exe /c start java -jar
LicenseDaemon.jar "+str2+" "+str6+" "+str6);


}




}

%>


<body>


</body>

</html>
```

medium.jsp

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<html>

<c:choose>
```

```
<c:when test="${param.username=='tina'}">
<jsp:include page="ServerSettings.jsp">
<jsp:param name="username" value="${param.username}"/>
<jsp:param name="password" value="${param.password}"/>
</jsp:include>
</c:when>

<c:when test="${param.username=='eleni'}">
<jsp:include page="ServerSettings.jsp">
<jsp:param name="username" value="${param.username}"/>
<jsp:param name="password" value="${param.password}"/>
</jsp:include>
</c:when>

<c:otherwise>
<jsp:include page="WrongPasswords.jsp">
<jsp:param name="username" value="${param.username}"/>
</jsp:include>
</c:otherwise>



</c:choose>
<body>

</body>
</html>
```

ServerSettings.jsp

```
<%@ page contentType="text/html; charset=iso-8859-1" language="java"
import="java.sql.*" errorPage="" %>


<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
1" />

<title>Server's Settings</title>

<style type="text/css">

<!--

#Layer1 {

      position:absolute;

      left:726px;

      top:364px;

      width:218px;

      height:36px;

      z-index:1;

      visibility: hidden;

}

#Layer2 {

      position:absolute;

      left:21px;

      top:181px;

      width:407px;

      height:152px;

      z-index:2;

}

.style1 {

      font-size: 18px;

      font-weight: bold;
```

```
}
.style2 {font-size: 36px; font-weight: bold; }
#Layer3 {
      position:absolute;
      left:748px;
      top:390px;
      width:127px;
      height:125px;
      z-index:1;
      visibility: hidden;
}
#Layer4 {
      position:absolute;
      left:726px;
      top:344px;
      width:220px;
      height:45px;
      z-index:1;
      visibility: visible;
}
#Layer5 {
      position:absolute;
      left:719px;
      top:379px;
      width:191px;
      height:38px;
      z-index:2;
      visibility: hidden;
}
-->
```

```
</style>

<script type="text/JavaScript">

<!--

function MM_findObj(n, d) { //v4.01

  var          p,i,x;                     if(!d)          d=document;
if((p=n.indexOf("?"))>0&&parent.frames.length) {

    d=parent.frames[n.substring(p+1)].document; n=n.substring(0,p);}

  if(!(x=d[n])&&d.all) x=d.all[n]; for (i=0;!x&&i<d.forms.length;i++)
x=d.forms[i][n];

  for(i=0;!x&&d.layers&&i<d.layers.length;i++)
x=MM_findObj(n,d.layers[i].document);

  if(!x && d.getElementById) x=d.getElementById(n); return x;

}


function MM_changeProp(objName,x,theProp,theValue) { //v6.0

  var obj = MM_findObj(objName);

  if (obj && (theProp.indexOf("style.")==-1 || obj.style)){

    if (theValue == true || theValue == false)

      eval("obj."+theProp+"="+theValue);

    else eval("obj."+theProp+"='"+theValue+"'");

  }

}


function MM_validateForm() { //v4.0

  var
i,p,q,nm,test,num,min,max,errors='',args=MM_validateForm.arguments;

  for    (i=0;    i<(args.length-2);    i+=3)    {    test=args[i+2];
val=MM_findObj(args[i]);

    if (val) { nm=val.name; if ((val=val.value)!="") {

      if (test.indexOf('isEmail')!=-1) { p=val.indexOf('@');

        if (p<1 || p==(val.length-1)) errors+='- '+nm+' must contain
an e-mail address.\n';

      } else if (test!='R') { num = parseFloat(val);
```

```
        if (isNaN(val)) errors+='- '+nm+' must contain a number.\n';

        if (test.indexOf('inRange') != -1) { p=test.indexOf(':');

          min=test.substring(8,p); max=test.substring(p+1);

          if (num<min || max<num) errors+='- '+nm+' must contain a
number between '+min+' and '+max+'.\n';

    } } } else if (test.charAt(0) == 'R') errors += '- '+nm+' is
required.\n'; }

  } if (errors) alert('The following error(s) occurred:\n'+errors);

  document.MM_returnValue = (errors == '');

}



function MM_showHideLayers() { //v6.0

  var i,p,v,obj,args=MM_showHideLayers.arguments;

  for         (i=0;         i<(args.length-2);         i+=3)         if
((obj=MM_findObj(args[i]))!=null) { v=args[i+2];

    if             (obj.style)             {             obj=obj.style;
v=(v=='show')?'visible':(v=='hide')?'hidden':v; }

    obj.visibility=v; }

}




//-->

</script>

</head>




<body>




    <form id="form1" name="form1" method="post" action="media.jsp">
```

```
      <p            align="center"            class="style2">Welcome,
${param.username}!</p>

<p class="style1"> </p>

<p class="style1"> </p>

<label>

  <div align="center"><strong>License Service Url:</strong>

    <br>

    <br>

    <input          name="url"          type="text"          id="url"
value="http://147.102.19.122:8080/axis2/services/LicenseService"
size="50" />

  </div>

  </label>

  <p> </p>

  <p align="center"><strong>Select range of ports: </strong></p>

  <p align="center">

    <label>

    <input                 name="choice"                 type="radio"
onClick="MM_showHideLayers('Layer4','','show','Layer5','','hide')"
value="Range" checked="checked" />

    Range

    </label>

  </p>

  <p align="center">

    <label>

    <input                 name="choice"                 type="radio"
onClick="MM_showHideLayers('Layer4','','hide','Layer5','','show')"
value="StandalonePort" />

      StandalonePort</label>

  </p>

  <p align="center">

    <label></label>

    <br />
```

```
    <label></label>
  </p>
  <p>
    <label></label>
  </p>
  <p>
    <label></label>
  </p>
  <p>
    <label></label>
  </p>
  <p>
    <label></label>
  </p>
  <p align="center"></p>
  <p align="center"></p>
  <p align="center"></p>
  <p align="center"></p>
  <p align="center">  </p>
  <p align="center"> </p>
  <p align="center">
   <div id="Layer4">
      <label>      </label>


        <label>From:
        <input name="from" type="text" id="from" size="5">
        </label>
To:
<label>
<input name="to" type="text" id="to" size="5">
```

```
</label>



      <label></label>

</div>

    <div id="Layer5">

      <label>

      <input  name="StandalonePort"  type="text"  id="StandalonePort"
size="5">

      </label>

    </div>

    <div align="center">

      <input       name="button"       type="submit"       id="button"
onClick="MM_validateForm('url','','R')"    value="START  THE  LICENSE
DAEMON!" />

      </p>

       </div>

    <p  align="center"><strong>To  stop  the  LicenseDaemon  just  close
the terminal. </strong></p>

</form>



</body>

</html>
```

WrongPasswords.jsp

```
<%@ page contentType="text/html; charset=iso-8859-1" language="java"
import="java.sql.*" errorPage="" %>

<!DOCTYPE  html  PUBLIC  "-//W3C//DTD  XHTML  1.0  Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
1" />

<title>WrongPasswords</title>

<script type="text/JavaScript">

<!--

function MM_goToURL() { //v3.0

  var i, args=MM_goToURL.arguments; document.MM_returnValue = false;

  for            (i=0;            i<(args.length-1);            i+=2)
eval(args[i]+".location='"+args[i+1]+"'");

}

//-->

</script>

<style type="text/css">

<!--

body {

      background-image: url(baby.jpg);

      background-repeat: repeat;

}

.style2 {

      color: #FFFFFF;

      font-weight: bold;

      font-size: 36px;

}

-->

</style></head>


<body>

<p class="style2">SORRY ${param.username}, your username or/and your
password are invalid!</p>

<p class="style2">Please try again!</p>

<p> </p>

<form id="form1" name="form1" method="post" action="">
```

```
  <label>

  <input                  name="Submit"                  type="submit"
onclick="MM_goToURL('parent','index.jsp');return
document.MM_returnValue" value="try again" />

  </label>

</form>

<label></label>

<p><strong></strong></p>

</body>

</html>
```

PHP

index.php

```
<?php

$login=$_POST['submit'];
$Uname = $_POST['username'];
$Pword = $_POST['password'];
$uname = $_POST["username"];
$pword = $_POST["password"];


$login=$_POST['submit'];
if($login){
$username=$_POST['username'];
$password=$_POST['password'];
$url='SetServers.php';

if  (($username=="a") && ($password=="a"))
{print("edwww!");
//header('location: SetServers.php');
header('location:$url');
}

else {
print("edw");
header('location:index.php');
}}

 ?>
<html>
<head>
<title>welcome!!</title>

<script type="text/JavaScript">
<!--

function MM_findObj(n, d) { //v4.01
```

```
  var p,i,x;  if(!d) d=document;
if((p=n.indexOf("?"))>0&&parent.frames.length) {
    d=parent.frames[n.substring(p+1)].document; n=n.substring(0,p);}
  if(!(x=d[n])&&d.all) x=d.all[n]; for (i=0;!x&&i<d.forms.length;i++)
x=d.forms[i][n];
  for(i=0;!x&&d.layers&&i<d.layers.length;i++)
x=MM_findObj(n,d.layers[i].document);
  if(!x && d.getElementById) x=d.getElementById(n); return x;
}

function MM_validateForm() { //v4.0
  var
i,p,q,nm,test,num,min,max,errors='',args=MM_validateForm.arguments;
  for (i=0; i<(args.length-2); i+=3) { test=args[i+2];
val=MM_findObj(args[i]);
    if (val) { nm=val.name; if ((val=val.value)!="") {
      if (test.indexOf('isEmail')!=-1) { p=val.indexOf('@');
        if (p<1 || p==(val.length-1)) errors+='- '+nm+' must contain
an e-mail address.\n';
      } else if (test!='R') { num = parseFloat(val);
        if (isNaN(val)) errors+='- '+nm+' must contain a number.\n';
        if (test.indexOf('inRange') != -1) { p=test.indexOf(':');
          min=test.substring(8,p); max=test.substring(p+1);
          if (num<min || max<num) errors+='- '+nm+' must contain a
number between '+min+' and '+max+'.\n';
    } } } else if (test.charAt(0) == 'R') errors += '- '+nm+' is
required.\n'; }
  } if (errors) alert('The following error(s) occurred:\n'+errors);
  document.MM_returnValue = (errors == '');
}
//-->
</script>
<style type="text/css">
<!--
#Layer1 {
      position:absolute;
      left:732px;
      top:445px;
      width:303px;
      height:190px;
      z-index:1;
      visibility: hidden;
}
#Layer2 {
      position:absolute;
      left:453px;
      top:390px;
      width:337px;
      height:51px;
      z-index:2;
}
-->
</style>
</head>

<body>
<form name="form1" method="post" action="SetServers.php">
  <p align="center" class="style1">LICENSE SERVICE ADMINISTRATOR's
PAGE</p>
  <p> </p>
  <label></label>
```

```
  <p>
    <label></label>
  </p>
  <p> </p>
  <p align="center">username:
    <input name="username" type="text" id="username">
  </p>
  <p align="center">password:
    <input name="password" type="password" id="password">
  </p>
  <p align="center"> </p>
  <p align="center">
    <label>
    <input name="submit" type="submit"
onClick="MM_validateForm('username','','R','password','','R');Go To
Url;return document.MM_returnValue" value="submit">
    </label>
  </p>
  <p></p>
</form>
</body>
</html>
```

LMUtilities.php

```
<html>
<head>
<title>LMUtilities</title>
<style type="text/css">
<!--
.style1 {
      font-size: 24px;
      font-weight: bold;
}
.style {
      font-size: 24px;
}

-->
</style>

</head>
<body>

<p align="center"> </p>
<p align="center"><br>
<span class="style1">License Management Utilities</span></p>
<p align="center"><br/>
  <span class="style2">Choose your prefered lmutility to
execute</span><br/></p>
<p> </p>
<p> </p>
<p> </p>
<?php
$file = fopen("AvailableServers.txt","r");
$i=0;
$k=1;
while(!feof($file))
  {
  $server[$i]=fgets($file);
```

```
      $i=$i+$k;
  }

fclose($file);
?>
   <form action=result.php method="to">
       <div align="center">
         <select name="utility">
            <option value="lmhostid">Hostid</option>
            <option value="lmdiag">Server Diagnostics</option>
            <option value="lmstat">Perform Status Enquiry</option>
            <option value="lmreread">Reread License file</option>
            <option value="lmdown">Shut Down Server</option>
        </select>
          <select name="ip">
         <?php
      $i=$i-1;
      do{
      echo "<option>$server[$i]</option>";
      $i=$i-$k;
      }while($i>0)
?>
        </select>
       <input type="submit" value="Go!" />
       </div>
   </form>

<p>
  <label><a href="SetServers.php">SetServers</a></label>
</p>
</body>
</html>
```

result.php

```
<td> </td>
<p> </p>
<p align="center"> </p>
<p align="center"> </p>
<p align="center"> </p>
<p align="center">
  <textarea name="textarea" cols=55 rows=10>
<?php
$lmhost=$_GET['utility'];
$ip=$_GET['ip'];
$str=SERVER." ".$ip." "."27000\n";
$str2="USE_SERVER";
$file2 = fopen("license.dat","w");
fwrite($file2,$str);
fwrite($file2,$str2);
fclose($file2);
$lpath = realpath("license.dat");
echo passthru("lmutil ".$lmhost." -c ".'"'.$lpath.'"');
?>



        </textarea>
</p>
```

```
<p><a href="SetServers.php">SetServers</a></p>
<p><a href="LMUtilities.php">LMUtilities</a></p>
```

SetServers.php

```
<?php
$str1= $_POST["insert1"];
$str2= $_POST["insert2"];
$str3= $_POST["insert3"];
$str4= $_POST["insert4"];
$str5=".";
$str=$str1.$str5.$str2.$str5.$str3.$str5.$str4;
$add2=$_REQUEST['add2'];
//$Uname = $_POST["username"];
$filename=$_POST['datafile'];
echo $filename;
//-----------------------------------kleboume edw
$filename='/Documents and Settings/LicenseServiceHistory.log';
$lines3 = file($filename);





//--------REMOVE
$_SESSION["removal"]=$_POST["removal"];
$removal=$_POST["removal"];

//---READ TEXT ARXIKA
$lines = file('/Documents and Settings/AvailableServers.txt');
$ips = array();
$i=1;
foreach ($lines as $line_num => $line)
{
$token = strtok($line,"\n" );
while ($token)
{
     if ($token!=" ")
       {
       $ips[$i]=trim($token);


       $i=$i+1;
       }
      $token = strtok("\n");

      //-----------???????
        }
}

//----------ADD
if($str!="...")
{
$fh = fopen("/Documents and Settings/AvailableServers.txt", 'a') or
die("can't open file");
fwrite($fh, "\n");
fwrite($fh, $str);
fclose($fh);
}
```

```
//-------REMOVE
if ($removal!="") //ax exei patithei to remove
{$fh = fopen("/Documents and Settings/AvailableServers.txt", 'w') or
die("can't open file"); //anoikse to arxeio kai grapse null gia na to
sbhseis
fwrite($fh, "");
fclose($fh);

for ($k=1; $k<14; $k++)
{
//echo "--".$ips[$k];
if ($ips[$k]!=$removal)//gia kathe stoixeio tou pinaka, an einia
diaforetiko, typwse to

{
$fh = fopen("/Documents and Settings/AvailableServers.txt", 'a') or
die("can't open file");
fwrite($fh, $ips[$k]);
fwrite($fh, "\n");
fclose($fh);}
continue;
}

}
//READ TEXT SECOND TIME

$lines = file('/Documents and Settings/AvailableServers.txt');
$ips = array();
$i=1;
foreach ($lines as $line_num => $line)
{$token = strtok($line,"\n" );
while ($token)
{
   if ($token!="")
 {
$ips[$i]=trim($token);
$i=$i+1;
}
$token = strtok(" ");


        }
}

?>
```

```
<html>
<head>
<title>Administrator, how are you?</title>
<style type="text/css">
<!--
.style1 {
      font-size: 24px;
      font-weight: bold;
}
#Layer1 {
      position:absolute;
```

```
        left:465px;
        top:331px;
        width:291px;
        height:161px;
        z-index:2;
        visibility: hidden;
}
#Layer2 {
        position:absolute;
        left:193px;
        top:355px;
        width:598px;
        height:94px;
        z-index:3;
        visibility: hidden;
}
#Layer3 {
        position:absolute;
        left:376px;
        top:354px;
        width:512px;
        height:108px;
        z-index:4;
        visibility: hidden;
}
#Layer4 {
        position:absolute;
        left:140px;
        top:183px;
        width:214px;
        height:34px;
        z-index:4;
}
#Layer5 {
        position:absolute;
        left:376px;
        top:246px;
        width:585px;
        height:27px;
        z-index:1;
}
#Layer6 {
        position:absolute;
        left:446px;
        top:300px;
        width:295px;
        height:35px;
        z-index:5;
}
#Layer7 {
        position:absolute;
        left:84px;
        top:305px;
        width:256px;
        height:44px;
        z-index:5;
        visibility: hidden;
}
#Layer8 {
        position:absolute;
        left:780px;
```

```
      top:579px;
      width:238px;
      height:32px;
      z-index:5;
}
#Layer9 {
      position:absolute;
      left:143px;
      top:325px;
      width:307px;
      height:268px;
      z-index:6;
      visibility: hidden;
}
-->
</style>
<script type="text/JavaScript">
<!--
function mmLoadMenus() {
  if (window.null) return;
  window.null = new
Menu("root",190,18,"",12,"#000000","#FFFFFF","#CCCCCC","#000084","lef
t","middle",3,0,1000,-5,7,true,false,true,0,true,true);

null.addMenuItem("The ip $textfield is inserted!"
);
   null.hideOnMouseOut=true;
   null.bgColor='#555555';
   null.menuBorder=1;
   null.menuLiteBgColor='#FFFFFF';
   null.menuBorderBgColor='#777777';

null.writeMenus();
} // mmLoadMenus()

function MM_findObj(n, d) { //v4.01
  var p,i,x;  if(!d) d=document;
if((p=n.indexOf("?"))>0&&parent.frames.length) {
    d=parent.frames[n.substring(p+1)].document; n=n.substring(0,p);}
  if(!(x=d[n])&&d.all) x=d.all[n]; for (i=0;!x&&i<d.forms.length;i++)
x=d.forms[i][n];
  for(i=0;!x&&d.layers&&i<d.layers.length;i++)
x=MM_findObj(n,d.layers[i].document);
  if(!x && d.getElementById) x=d.getElementById(n); return x;
}

function MM_showHideLayers() { //v6.0
  var i,p,v,obj,args=MM_showHideLayers.arguments;
  for (i=0; i<(args.length-2); i+=3) if
((obj=MM_findObj(args[i]))!=null) { v=args[i+2];
    if (obj.style) { obj=obj.style;
v=(v=='show')?'visible':(v=='hide')?'hidden':v; }
    obj.visibility=v; }
}

function MM_validateForm() { //v4.0
  var
i,p,q,nm,test,num,min,max,errors='',args=MM_validateForm.arguments;
  for (i=0; i<(args.length-2); i+=3) { test=args[i+2];
val=MM_findObj(args[i]);
    if (val) { nm=val.name; if ((val=val.value)!="") {
```

```
      if (test.indexOf('isEmail')!=-1) { p=val.indexOf('@');
        if (p<1 || p==(val.length-1)) errors+='- '+nm+' must contain
an e-mail address.\n';
      } else if (test!='R') { num = parseFloat(val);
        if (isNaN(val)) errors+='- '+nm+' must contain a number.\n';
        if (test.indexOf('inRange') != -1) { p=test.indexOf(':');
          min=test.substring(8,p); max=test.substring(p+1);
          if (num<min || max<num) errors+='- '+nm+' must contain a
number between '+min+' and '+max+'.\n';
    } } } else if (test.charAt(0) == 'R') errors += '- '+nm+' is
required.\n'; }
  } if (errors) alert('The following error(s) occurred:\n'+errors);
  document.MM_returnValue = (errors == '');
}
function MM_popupMsg(msg) { //v1.0
  alert(msg);
}



function isNothing() {
            if (arguments[0].value == "") arguments[0].selectedIndex
= 0;
            document.getElementById("uploadfile").focus();
      }

function MM_goToURL() { //v3.0
  var i, args=MM_goToURL.arguments; document.MM_returnValue = false;
  for (i=0; i<(args.length-1); i+=2)
eval(args[i]+".location='"+args[i+1]+"'");
}
//-->
</script>
<script language="JavaScript" src="mm_menu.js"></script>
</head>


<body>
<div id="Layer9">
  <label></label>
  <form name="form3" method="post" action="<? echo $PHP_SELF; ?>">
    <div align="center">
      <p align="left"> </p>
      <p>

        <label>
        <textarea name="textarea2" cols="120" rows="15">
            <?php echo "LICENSE HISTORY";
      echo "\n";
      echo "-------------------------------------------------------
-------------------------------------------------------------";
      ;?>
      <?php


       foreach ($lines3 as $line_num => $line)
{

echo $line ;
}?>
```

```
            </textarea>
        <input name="close" type="button" id="close"
onClick="MM_showHideLayers('Layer9','','hide')" value="close">
        </label>
</p>
    </div>
  </form>
</div>
<script language="JavaScript1.2">mmLoadMenus();</script>

<p align="center"> </p>
<p align="center"> </p>
<p align="center"> </p>
<p align="center"><br>
    <span class="style1"> Hello,administrator!! </span><br/>
</p>
<p> </p>
<p> </p>
<p align="center">

<div id="Layer1">
  <label>
  <div align="center">

<textarea name="textarea" cols="50" rows="10" id="textarea">
 <?php

echo "AvailableServers:\n";
echo "-----------------------";
echo "\n";
  for($k=1; $k<$i; $k++)
       {
     echo "$k)", trim($ips[$k]);
       echo "\n";

       }

//$LENGTH=$_POST["$i"];
//echo $LENGTH;
?>

 </textarea>
  </div>
  </label>
</div>

<div id="Layer5">
  <div align="center">
    <input name="ViewServers" type="submit" id="ViewServers"
onClick="MM_showHideLayers('Layer9','','hide','Layer1','','show','Lay
er2','','hide','Layer3','','hide')" value="ViewServers">
    <input name="AddServer" type="submit" id="AddServer"
onClick="MM_showHideLayers('Layer9','','hide','Layer1','','hide','Lay
er2','','show','Layer3','','hide')" value="AddServer">
    <input name="RemoveServer" type="submit" id="RemoveServer"
onClick="MM_showHideLayers('Layer9','','hide','Layer1','','hide','Lay
er2','','hide','Layer3','','show')" value="RemoveServer">
    </p>

    <label> <br>
```

```
    <input name="ViewHistory" type="submit" id="ViewHistory"
onClick="MM_showHideLayers('Layer9','','show','Layer1','','hide','Lay
er2','','hide','Layer3','','hide')" value="ViewHistory">
    </label>
    <label>
    <input name="ViewServersDiagnostics" type="submit"
id="ViewServersDiagnostics"
onClick="MM_goToURL('parent','LMUtilities.php');return
document.MM_returnValue" value="ViewServersDiagnostics">
    </label>
    <label></label>
  </div>
</div>
<p> </p>
<p>  </p>
<div id="Layer2">
  <label></label>
  <label></label>
  <p> </p>

 <form name="form1" method="post" action="SetServers.php">
    <p>Enter the ip:

      <input name="insert1" type="text" id="insert1" size="3"
maxlength="3">

<input name="insert2" type="text" id="insert2" size="3"
maxlength="3">

<input name="insert3" type="text" id="insert3" size="3"
maxlength="3">

<input name="insert4" type="text" id="insert4" size="3"
maxlength="3">
<input name="ADD2" type="submit" id="ADD2"
onClick="MM_validateForm('insert1','','RisNum','insert2','','RisNum',
'insert3','','RisNum','insert4','','RisNum');MM_showHideLayers('Layer
7','','show','Layer2','','hide');return document.MM_returnValue"
value="ADD!">
      </p>
    <p> </p>
    <p> </p>
  </form>
  <p>     </p>
</div>

 <div id="Layer3">
  <label></label><p> </p>

  <form name="form2" method="post" action="SetServers.php">
    <label>Select which to remove:
    <select name="removal" size="<?php echo $i/2;?>" multiple
id="removal" onChange="isNothing(this)" >
      <option selected><?php echo "$ips[1]";?></option>
      <?php for($k=2; $k<$i; $k++){?>
      <option><?php if ($ips[$k]!="") echo "$ips[$k]";?></option>
      <?php }?>

    </select>

    </label>
```

```
    <p><input name="Submit" type="submit" onClick="MM_popupMsg('Are
you sure you want to delete the
ip?');MM_showHideLayers('Layer3','','hide')" value="Remove!"  >
    </p>
   </form>

  <p> </p>
  <p align="center">
    <label></label>
  </p>
</div>

<p>
  <?php

if ($str!="...")
{
echo "<script> MM_popupMsg('The ip :$str has been inserted
succesfully! ');</script>";
}  ?>
<?php if ($removal!="")
{echo "<script> MM_popupMsg('The ip :$removal has been removed
succesfully! ');</script>";

}
?>
<label></label>
</p>
</body>
</html>
```