ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μελέτη υπηρεσιών για τον προσδιορισμό παραμέτρων Ποιότητας Υπηρεσίας (Quality of Service) σε περιβάλλον Πολυπλέγματος (Grid).

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΓΡΗΓΟΡΙΟΥ Χ.ΚΑΤΣΑΡΟΥ

Επιβλέπουσα : Θεοδώρα Βαρβαρίγου
Αναπληρώτρια Καθηγήτρια Ε.Μ.Π.

ΑΘΗΝΑ, ΙΟΥΛΙΟΣ 2006

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Μελέτη υπηρεσιών για τον προσδιορισμό παραμέτρων Ποιότητας Υπηρεσίας (Quality of Service) σε περιβάλλον Πολυπλέγματος (Grid).

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΓΡΗΓΌΡΙΟΥ Χ.ΚΑΤΣΑΡΟΥ

Επιβλέπων : Θεοδώρα Βαρβαρίγου
Αναπληρώτρια Καθηγήτρια Ε.Μ.Π

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 31η Μαρτίου 2006.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Θεοδώρα Βαρβαρίγου	Εμμανουήλ Πρωτονοτάριος	Γεώργιος Στασινόπουλος
Αναπλ. Καθηγήτρια Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2006

(Υπογραφή)

.....

ΓΡΗΓΟΡΙΟΣ Χ. ΚΑΤΣΑΡΟΣ

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

Περίληψη

Σκοπός της παρούσας εργασίας είναι η ανάπτυξη ενός μηχανισμού πρόβλεψης της ποιότητας υπηρεσιών (Quality of Service) που παρέχονται από συστήματα Πολυπλέγματος (Grid Systems). Η παραπάνω προσέγγιση υλοποιείται μέσω παραμέτρων που σχετίζονται με την ποιότητα και την αξιοπιστία της υπηρεσίας.

Η έννοια της ποιότητας υπηρεσίας (Quality of Service - QoS) είναι πολύ σημαντική στον αναπτυσσόμενο χώρο των τεχνολογιών Πολυπλέγματος, αφού η χρήση των συστημάτων αυτών στον επιχειρηματικό κόσμο απαιτεί την εξασφάλιση και επιβεβαίωση της ποιότητας υπηρεσίας μεταξύ των επιχειρήσεων που συμμετέχουν.

Ο μηχανισμός που περιγράφεται στο παρόν κείμενο υλοποιήθηκε με χρήση δικτυακών υπηρεσιών (Web Services) που εγκαθίστανται σε υπολογιστικούς πόρους Grid συστημάτων. Οι δικτυακές υπηρεσίες καλούνται είτε από τους πάροχους των υπηρεσιών (Service Providers) με στόχο τον εσωτερικό έλεγχο και τη βελτίωση των υπηρεσιών, είτε από εξωτερικούς πιστοποιημένους πελάτες με σκοπό την επιβεβαίωση της ποιότητας της υπηρεσίας.

Οι πλατφόρμες και τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση είναι οι πιο διαδεδομένες για αυτή τη χρήση. Συγκεκριμένα έγινε χρήση του Apache Tomcat Server και προγραμματισμού στη γλώσσα Java για τις υπηρεσίες ενώ έγινε ανάπτυξη JSP δυναμικών σελίδων για τις διεπαφές (interfaces) των χρηστών.

Λέξεις Κλειδιά: <<Ποιότητα Υπηρεσίας, Grid, Java, JSP, Tomcat server, SLA, Πάροχος Υπηρεσίας, Καταναλωτής Υπηρεσίας>>

Abstract

The aim of this thesis is the development of a QoS provision mechanism, in Grid systems. This approach will be accomplished by extracting specific parameters that are relevant with quality and reliability of the Grid service.

The role of QoS in Grid is fundamental, since it provides the necessary verification and the guarantee between the participants of a business Grid.

This module will be implemented with the use of Web Services which will be deployed in every resource of the Grid system. Those services will be called by the Service Providers, for internal control and improvement, and by authorized users of the Grid, for their confirmation of the service's quality.

The technologies that will be used during the implementation of the module will be at the cutting edge. In detail, tomcat server and Java programming will be used for the web service implementation while JSP will implement the dynamic user interface.

Keywords: <<Grid, QoS, SLA, Web Service, Java, Tomcat, JSP, Service Provider, Service Consumer>>

Ευχαριστίες

Θα ήθελα να ευχαριστήσω την Αναπληρώτρια Καθηγήτρια Θεοδώρα Βαρβαρίγου για την ανάθεση της εργασίας, την παρακολούθηση και παροχή κατευθύνσεων για την υλοποίηση της. Επίσης, δεν θα ήθελα να παραλείψω τη συμβολή των Τσερπέ Κωνσταντίνου, Μενύχτα Αντρέα, Κυριαζή Δημοσθένη και Δόλκα Κωνσταντίνου, υποψήφιων διδακτόρων του Εργαστηρίου Τηλεπικοινωνιών, που με αμέριστη υπομονή και συμπαράσταση, συνέβαλαν με τον τρόπο τους στην ολοκλήρωση αυτού του έργου.

Επιπρόσθετα, θα ήθελα να αναφέρω τον Τσούλο Κωνσταντίνο, απόφοιτο της Σχολής Ηλεκτ/ων Μηχανικών και Μηχανικών Η/Υ, και πολύ καλό φίλο, για την βοήθειά του και τις πολύτιμες συμβουλές του.

Τέλος, οφείλω κυρίως να ευχαριστήσω τους γονείς μου που με ανέχτηκαν όλα αυτά τα χρόνια και προσέφεραν και αυτοί ότι μπορούσαν στην επιτυχή σταδιοδρομία μου.

Πίνακας περιεχομένων

1 *Εισαγωγή στο κεφάλαιο*

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 αλματώδης εξέλιξη των δικτυακών τεχνολογιών και η απαίτηση υπολογιστικής ισχύος οπουδήποτε στον πλανήτη και οποτεδήποτε, οδήγησε στην ανάπτυξη της Grid τεχνολογίας. Με την διάδοση αυτής της τεχνολογίας και εκτός του ακαδημαϊκού χώρου, ένας συνεχώς αυξανόμενος αριθμός από εμπορικές εφαρμογές άρχισαν να υιοθετούν τεχνολογίες Grids, προκειμένου να βελτιώσουν την απόδοση, την αξιοπιστία και τη διαθεσιμότητά τους με ένα οικονομικό τρόπο. Η ευκολία στην δυναμική διαχείριση των πόρων και άλλες υπηρεσίες που προσφέρονται από τα Grid, έχουν φέρει την τεχνολογία αυτή στο προσκήνιο τα τελευταία χρόνια. Σε αυτό το πλαίσιο, και δεδομένου ότι η αγορά απαιτεί τώρα περισσότερες εγγυήσεις στο Επίπεδο της παρεχόμενης Ποιότητας (QoS), η διπλωματική αυτή εργασία, έχει ως σκοπό την υλοποίηση ενός μηχανισμού, ικανού να παρέχει πληροφορία σχετικά με την ποιότητα των παρεχόμενων υπηρεσιών από το Grid, σε αρχιτεκτονικές βασισμένες σε υπηρεσίες. Ο μηχανισμός αυτός θα αναπτυχθεί πάνω σε τεχνολογίες Web Services και σκοπός του θα είναι να καταγράφει, να υπολογίζει, να καταχωρεί και να επιστρέφει σε όποιον του ζητήσει δομημένες (μοντελοποιημένες) πληροφορίες σχετικά με την ποιότητα των Grid υπηρεσιών.

Ο μηχανισμός αυτός θα παρέχει δυνατότητες διαχείρισης του συστήματος ελέγχου της ποιότητας της υπηρεσίας, με λειτουργίες εισαγωγής-εξαγωγής πόρων από το σύστημα και

επιλογής παραμέτρων εκτέλεσης. Επίσης θα υπάρχουν καταστάσεις λειτουργίας του μηχανισμού, όπως προγραμματισμένη (αυτόματη εκτέλεση), είτε κατ' απαίτηση, real time εκτέλεση από κάποιον χρήστη. Τέλος όλες οι πληροφορίες θα συλλέγονται για μελλοντική επεξεργασία και εξαγωγή συμπερασμάτων με σκοπό την διόρθωση και βελτίωση των παρεχόμενων υπηρεσιών.

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 Interface)
- Εξαγωγή συμπερασμάτων και συλλογή αποτελεσμάτων

1.2 Οργάνωση του εγγράφου

Το παρόν έγγραφο αποτελείται από 7 κεφάλαια και Παράρτημα. Στις ενότητες των κεφαλαίων αυτών παρουσιάζεται ουσιαστικά και με αναλυτικό τρόπο η διαδικασία ανάπτυξης της εφαρμογής, που περιγράφηκε στην προηγούμενη ενότητα.

Το κεφάλαιο 2 έχει τίτλο «Περιγραφή Θέματος» και τοποθετεί το θέμα ως προς την επιστημονική περιοχή στην οποία ανήκει (Ενότητα «Σχετικές Τεχνολογίες») . Επιπλέον δίνει το στόχο της εργασίας και το πρόβλημα που αυτή καλείται να αντιμετωπίσει (Ενότητα «Στόχος»).

Το κεφάλαιο 3 έχει τίτλο «Ανάλυση και Σχεδίαση» και αφορά το τμήμα της σχεδίασης του συστήματος λογισμικού. Σε αυτό το κεφάλαιο γίνεται προσπάθεια να περιγραφεί η αρχιτεκτονική και η λειτουργία του συστήματος με την παρουσίαση διαγραμμάτων της δομής του συστήματος και περιπτώσεων χρήσης του.

Το κεφάλαιο 4 με τίτλο «Υλοποίηση», αναφέρεται στα εργαλεία που χρησιμοποιήθηκαν στην εφαρμογή και επεξηγηματικές πληροφορίες σχετικά με αυτή, οι οποίες λειτουργούν επικουρικά στα έγγραφα του κεφαλαίου 3.

Το κεφάλαιο 5 έχει τίτλο «Έλεγχος» και περιλαμβάνει τον Οδηγό Εγκατάστασης του συστήματος και τα αποτελέσματα χρήσης του.

Τέλος, στον «Επίλογος» (κεφάλαιο 6) περιλαμβάνεται η σύνοψη της διπλωματικής εργασίας, καθώς και τα συμπεράσματα που εξήχθησαν με την ολοκλήρωσή της. Το κεφάλαιο 7 περιλαμβάνει τη Βιβλιογραφία και το $8^{ο}$ (Παράρτημα) αναλυτικά τον κώδικα που χρησιμοποιήσαμε.

2 Περιγραφή Θέματος

Στο κεφάλαιο αυτό παρουσιάζονται οι σχετικές με το θέμα και το σύστημα τεχνολογίες, δίνεται η γενική περιγραφή του προς ανάπτυξη συστήματος και προσδιορίζονται οι στόχοι του.

2.1 Σχετικές τεχνολογίες

2.1.1 Grid

Με τον όρο Grid εννοούμε το σύνολο της υποδομής και των υπηρεσιών για τη δημιουργία ενός ενιαίου υπολογιστικά περιβάλλοντος, που αν και γεωγραφικά κατανεμημένο, εμφανίζεται ως μια συμπαγής υπολογιστική πλατφόρμα σε όλους τους χρήστες. Χαρακτηριστικό της Grid τεχνολογίας είναι η διασύνδεση ετερογενών υπολογιστικών και όχι μόνο συστημάτων, με αποτέλεσμα τη δημιουργία νέων συνόλων υπηρεσιών με αυξημένες υπολογιστικές δυνατότητες και νέους τρόπους αξιοποίησης των διαμοιραζόμενων πόρων. [FK98]

2.1.1.1 Γιατί Grid;

Τι θα σήμαινε για σας αν θα μπορούσατε :

- να αναλύσετε την αξία ενός επενδυτικού χαρτοφυλακίου σε λεπτά αντί για ώρες;
- να ενωθείτε με ερευνητικές ομάδες από όλο τον κόσμο και να επωφεληθείτε από την άμεση ενημέρωση όλων των εξελίξεων;

- να μειώσετε τον χρόνο σχεδίασης των προϊόντων σας στο μισό μειώνοντας εξίσου και τυχόν ατέλειες του;

Πανεπιστημιακά εργαστήρια και άλλοι ερευνητικοί οργανισμοί χρησιμοποιούν τεχνολογία Grid εδώ και αρκετά χρόνια, αντιμετωπίζοντας και λύνοντας πολλά σημαντικά προβλήματα που μέχρι πρότινος φαίνονταν άλυτα. Οι ανταγωνιστικοί ρυθμοί στον ερευνητικό και επαγγελματικό τομέα απαιτούν συνεχείς καινοτομίες, έτσι ώστε να παράγουν αξιόπιστα προϊόντα και υπηρεσίες. Υπάρχει μια έντονη αλλαγή στον τρόπο αξιοποίησης της υπολογιστικής ισχύος, στον τρόπο πρόσβασης αλλά και στον τρόπο αξιολόγησής της. Η χρήση λοιπόν τεχνολογίας Grid θα μπορούσε να βελτιώσει ραγδαία την αποδοτικότητα οργανισμών που αντιμετωπίζουν τις προκλήσεις μιας σύγχρονης απαιτητικής αγοράς.

2.1.1.2 Τα πλεονεκτήματα του Grid

Τα σημερινά λειτουργικά περιβάλλοντα πρέπει να είναι πιο ανθεκτικά, ευέλικτα και ολοκληρωμένα από ποτέ. Πολλοί οργανισμοί σε όλο τον κόσμο βρίσκουν σημαντικά πλεονεκτήματα χρησιμοποιώντας Grid σε ουσιαστικές επαγγελματικές διαδικασίες επιτυγχάνοντας επαγγελματικό και τεχνολογικό κέρδος.

Επαγγελματικό Κέρδος

Επιτάχυνση του χρόνου με αποτελέσματα:

- βελτίωση της παραγωγικότητας και της συνεργασίας
- επίλυση προβλημάτων που θεωρούνταν μέχρι πριν άλυτα

Ενεργοποίηση συνεργασίας και προώθηση της λειτουργικής ευελιξίας:

- ενοποίηση όχι μόνο των τεχνολογιών Internet (IT) αλλά και των ανθρώπινων πόρων
- δημιουργία ‘εικονικών οργανισμών’ (VOs) από απομακρυσμένα τμήματα και επιχειρήσεις, με σκοπό την κοινή χρήση πόρων και πληροφοριών

Αποτελεσματική προσαρμογή στις διάφορες ανάγκες της αγοράς:

- δημιουργία ευέλικτων και ανθεκτικών λειτουργικών υποδομών
- αντιμετώπιση γρήγορων διακυμάνσεων στις απαιτήσεις/ανάγκες του πελάτη

Αύξηση παραγωγικότητας:

- απεριόριστη πρόσβαση του χρήστη σε υπολογιστικούς, αποθηκευτικούς και πόρους πληροφοριών οποιαδήποτε στιγμή
- ευκολία κινήσεων του εργαζόμενου στη φάση σχεδίασης του προϊόντος, του ερευνητικού έργου κλπ.

Τεχνολογικό Κέρδος

Βελτιστοποίηση υποδομών:

- συγκέντρωση της διαχείρισης φόρτου εργασίας (workload)
- παροχή αποθηκευτικού χώρου για εφαρμογές με υψηλές απαιτήσεις
- μείωση των χρονικών επαναλήψεων

Αύξηση της πρόσβασης σε δεδομένα και της συνεργασίας:

- κοινόχρηστα δεδομένα και παγκόσμια διανομή τους
- δημιουργία συνεργασιών μεταξύ οργανισμών και επιχειρήσεων

2.1.1.3 Μια άλλη ερμηνεία του Grid

Η υπαρκτή πλέον τεχνολογική υποδομή και τα πολλά πλεονεκτήματα της τεχνολογίας Grid, όχι μόνο σε ερευνητικό αλλά και σε επαγγελματικό τομέα, έχουν φέρει στην πραγματικότητα πολλές εφαρμογές Grid. Ακούμε για Compute Grids, Data Grids, Bio Grids, Cluster Grids, Tera Grids, αλλά ποια είναι η διαφορά και ποια η ομοιότητα όλων αυτών; Συνολικά θα μπορούσαμε να πούμε ότι τα Grids θα έπρεπε να αξιολογηθούν σε επίπεδο εφαρμογών, επαγγελματικής αξίας, επιστημονικής αποτελεσματικότητας και όχι αρχιτεκτονικής. Ο ορισμός που δώσαμε στην αρχή, σε αυτό το σημείο δεν φαίνεται αρκετός και θα παρουσιάσουμε λίγο πιο αναλυτικά ποιο σύστημα μπορεί να θεωρηθεί Grid. Μπορούμε να συγκεντρώσουμε τα χαρακτηριστικά ενός τέτοιου συστήματος σε τρεις ιδιότητες που όταν και οι τρεις ικανοποιούνται, τότε μιλάμε για Grid. Αρχικά πρέπει το σύστημα αυτό να συντονίζει τους πόρους, οι οποίοι δεν πρέπει να υποβάλλονται σε κεντρικό έλεγχο. Η κεντρική ιδέα της τεχνολογίας Grid είναι η συνεργασία και η ανάπτυξη κατάλληλων προτύπων ασφαλείας και επικοινωνίας πόρων που ανήκουν σε διαφορετικά σύνολα διαχείρισης , πχ οι υπολογιστές ενός εργαστηρίου, τα εργαστήρια ενός πανεπιστημίου μεταξύ τους, τα πανεπιστήμια μιας πόλης κλπ. Η ύπαρξη κεντρικού ελέγχου σε ολόκληρο το πλέγμα υπολογιστών θα το μετέτρεπε αυτομάτως σε τοπικό σύστημα διαχείρισης. Η δεύτερη βασική ιδιότητα είναι η χρήση γνωστών πρωτοκόλλων για την αντιμετώπιση θεμελιωδών προβλημάτων όπως η πρόσβαση σε πόρους, η πιστοποίηση του χρήστη, η εξουσιοδότηση του χρήστη κλπ. Για να είναι το σύστημα αυτό όσο το δυνατόν προσβάσιμο από περισσότερους χρήστες, έχει μεγάλη σημασία τα βασικά πρωτόκολλα επικοινωνίας να είναι διαδεδομένα, πρότυπα γενικής χρήσης. Τέλος, το σύστημα αυτό πρέπει να αποδίδει σημαντική ποιότητα υπηρεσιών προς του χρήστες του. Αυτό σημαίνει ότι η ομαλή συνεργασία όλων των πόρων θα πρέπει να έχει αποτέλεσμα παροχή υπηρεσιών με συγκεκριμένο χρόνο απόκρισης,

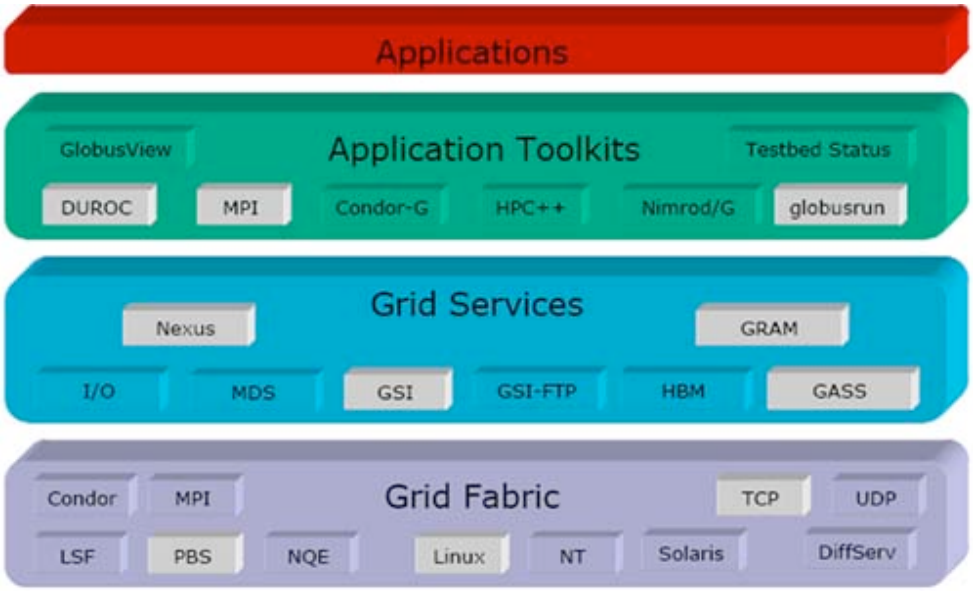
απόδοση, διαθεσιμότητα και ασφάλεια, έτσι ώστε η παραγωγικότητα και η χρησιμότητα των συνδυασμένων πόρων να είναι πολύ μεγαλύτερη από τις δυνατότητές τους ξεχωριστά. [F02]

2.1.1.4 Ιστορία του Grid

Το ιστορικό των τεχνολογιών διασύνδεσης των υπολογιστών αρχίζει στις αρχές της δεκαετίας του 70 στην Αμερική από την ερευνητική ομάδα ARPA (Advanced Research Projects Agency) που είχε δημιουργηθεί από την κυβέρνηση για στρατιωτικούς σκοπούς. Από την ομάδα αυτή αναπτύχθηκαν πολλά σημαντικά πρωτόκολλα και τεχνολογίες όπως το TCP/IP. Υπό την καθοδήγηση του Dr. J.C.R. Licklider και την συμβολή ερευνητών από διάφορα πανεπιστήμια της χώρας η έρευνα κατέληξε στη δημιουργία του πρώτου δικτύου υπολογιστών, προάγγελο του Internet, γνωστό ως ARPANET, στα 50 kbps. Η ερευνητική δραστηριότητα στον τομέα των δικτύων υπολογιστών συνεχίστηκε, με αποτέλεσμα την δημιουργία του NSFNET [1986], δικτύου στα 56Kbps που συνέδεε τα πέντε NSF κέντρα υπερ-υπολογιστών.

Ως συνέχεια και εξέλιξη αυτών των τεχνολογιών μπορούμε να θεωρήσουμε το πρόγραμμα Condor [1988] του πανεπιστημίου του Wisconsin. Το σύστημα αυτό είναι ένας 'διαχειριστής φόρτου εργασίας' (workload manager), με δυνατότητες παρακολούθησης και διαχείρισης πόρων, δρομολόγησης εργασιών και αποτελεί το πρώτο πρόγραμμα με κατεύθυνση προς την αξιοποίηση των Grid υπηρεσιών.

Η ανάπτυξη δικτύων υψηλών ταχυτήτων και η ανάγκη για μεγάλη επεξεργαστική ισχύ οδήγησε σε έντονη ερευνητική δραστηριότητα στον τομέα των Grid τεχνολογιών. Η έρευνα αυτή κατέληξε σε ενδιαφέροντα αποτελέσματα με πιο σημαντικά τα προγράμματα LEGION [1993], SRB [1997], GLOBUS [1998]. Το πρώτο βασίζεται στην ιδέα του 'εικονικού υπολογιστή (virtual computer) : όλοι οι πόροι συνδεδεμένοι μεταξύ τους, εμφανίζονται στον χρήστη ως μία εικονική μηχανή, με αρκετά μειονεκτήματα όμως, όπως η πολύπλοκη υλοποίηση και η μικρή αποδοτικότητα. Το SRB (Storage Resource Broker) ήταν μια πλατφόρμα διαχείρισης αποθηκευτικών πόρων που βοήθησε πολύ στην ανάπτυξη των Grid τεχνολογιών, αφού αντιμετώπισε τα προβλήματα μεταφοράς δεδομένων σε Grid περιβάλλον. Τέλος, το πιο διαδεδομένο σύστημα διαχείρισης Grid υπηρεσιών είναι το GLOBUS, που αναπτύχθηκε στο Argonne National Lab στο πανεπιστήμιο του Berkeley. Το GLOBUS προτυποποίησε πρωτοκόλλα για τη ασφάλεια, μεταφορά δεδομένων, ανακάλυψη πόρων και εκτέλεση εργασιών. Λειτουργεί σε χαμηλό επίπεδο και είναι ανεπτυγμένο σε επίπεδα υπηρεσιών:



Εικόνα 1 : Βασική Grid Αρχιτεκτονική

Η συνέχεια γίνεται με την ανάπτυξη των Web Services [2001] , υπηρεσιών που είναι προσβάσιμες μέσω του διαδικτύου και χρησιμοποιούν συγκεκριμένα πρωτόκολλα περιγραφής (XML, SOAP, WSDL) και τέλος με την εγκαθίδρυση του OGSA (Open Grid Service Architecture) [2002] ως κύριας αρχιτεκτονικής της Grid τεχνολογίας. Το πρότυπο αυτό είναι το πιο σημαντικό για τις τεχνολογίες του Grid, αφού περιγράφει τις δυνατότητες του συστήματος αυτού να αναλύσει την λειτουργία του σε όλα τα επίπεδά του.

2.1.2 OGSA

Οι συντονισμένες μελέτες του GGF (Global Grid Forum) - ενός οργανισμού αποτελούμενου από χρήστες, προγραμματιστές, ερευνητές με σκοπό την ανάπτυξη προτύπων για την υποστήριξη των Grid τεχνολογιών - κατέληξε πρόσφατα (2002) στην συγγραφή του OGSA. Το τελευταίο είναι ένα κείμενο που περιγράφει την δομή ενός Grid συστήματος και συγκεντρώνει όλα τα πρότυπα που χρησιμοποιεί η αρχιτεκτονική αυτή σε κάθε επίπεδο της.

2.1.2.1 Ποιοι είναι οι στόχοι του OGSA ;

Οι κύριοι στόχοι του OGSA είναι :

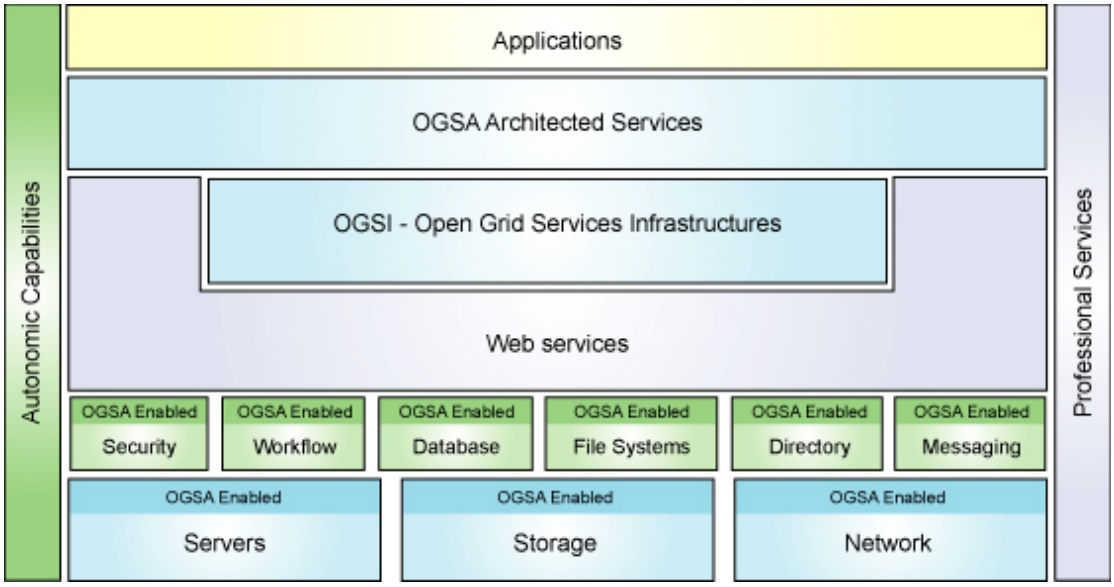
- να διαχειρίζεται πόρους απομακρυσμένων ετερογενών συστημάτων
- να αποδίδει σημαντική ποιότητα στις υπηρεσίες που παρέχει (QoS)

- να παρέχει τις βάσεις για αυτόνομες λύσεις διαχείρισης. Η διαφορετικότητα των πόρων που συμμετέχουν σε ένα Grid και ο δυναμικός τρόπος συνεργασίας τους απαιτεί ένα σύστημα διαχείρισης που θα προσαρμόζεται σύμφωνα με τις εκάστοτε ανάγκες.

- Να καθορίζει γνωστά πρότυπα και πρωτόκολλα λειτουργίας. Η δυναμική εισαγωγή και συνεργασία ετερογενών συστημάτων στο δίκτυο του Grid, βασίζεται στην υιοθέτηση γνωστών, ευρείας χρήσης προτύπων.

- Να εκμεταλλεύεται υπάρχουσες τεχνολογίες και να τις προσαρμόζει στο Grid, αν είναι εφικτό. Το OGSA βασίζεται στη τεχνολογία των Web Services (υπηρεσίες διαδικτύου), ένα τομέα πολύ διαδεδομένο αυτή την εποχή.

2.1.2.2 Η αρχιτεκτονική του OGSA



Εικόνα 2 : OGSA Αρχιτεκτονική

Το OGSA αποτελείται από τέσσερα βασικά επίπεδα :

- τους πόρους (φυσικούς και λογικούς)
- τις Web Services συμπεριλαμβανομένου και της OGSI υποδομής που έχει άμεση σχέση με τις υπηρεσίες
- τις υπηρεσίες σχεδιασμένες από το OGSA
- τις Grid εφαρμογές

Επίπεδο φυσικών και λογικών πόρων

Αυτό είναι το πιο χαμηλό επίπεδο από τα τέσσερα και έχει να κάνει με τους πόρους που συμμετέχουν στο Grid. Η έννοια του ‘πόρου’ στο OGSA είναι πολύ σημαντική και δεν είναι συγκεκριμένη. Ως πόρος (resource) σε ένα Grid μπορεί να θεωρηθεί από τον επεξεργαστή ενός υπολογιστή, μέχρι το τμήμα υπολογιστών μιας μεγάλης εταιρίας. Επίσης εκτός από υπολογιστικές μονάδες, συμμετέχουν και μονάδες αποθήκευσης, βάσεις δεδομένων, δικτυακές υποδομές κλπ. Αυτά που αναφέρθηκαν παραπάνω είναι κυρίως οι φυσικοί πόροι. Εκτός από αυτούς έχουμε και τους λογικούς, κάποια ενδιάμεσα συστήματα (middleware) που χρησιμοποιώντας τους φυσικούς πόρους, προσφέρουν στοιχειώδεις υπηρεσίες, όπως διαχείριση αρχείων ή βάσεων δεδομένων, στο παραπάνω επίπεδο.

Επίπεδο υπηρεσιών διαδικτύου (Web Services)

Μία πολύ βασική θεώρηση του OGSA είναι ότι όλοι οι πόροι του συστήματος αντιστοιχίζονται με υπηρεσίες. Έτσι η OGSI υποδομή (Open Grid Services Infrastructure) χρησιμοποιεί συγκεκριμένες, γνωστές υπηρεσίες δικτύου και πρωτόκολλα όπως XML και WSDL για να δημιουργήσει επαφές και διασυνδέσεις κάθε gird υπηρεσίας με τους πόρους του συστήματος. Η OGSI επεκτείνει τις δυνατότητες των δικτυακών υπηρεσιών έτσι ώστε να παρέχει δυναμικές και αξιόπιστες υπηρεσίες όπως απαιτείται για τη μοντελοποίηση των πόρων.

Επίπεδο Grid υπηρεσιών

Οι υπηρεσίες διαδικτύου και οι δυνατότητες του OGSI παρέχουν τη βασική υποδομή για το επόμενο επίπεδο, των Grid υπηρεσιών. Αυτή την εποχή υπάρχει μεγάλη δραστηριότητα προς την κατεύθυνση προσδιορισμού και προτυποποίησης τέτοιων υπηρεσιών, όπως υπηρεσίες υπολογισμού, πληροφοριών, πυρήνα, κ.α. Καθώς οι υλοποιήσεις αυτών των υπηρεσιών θα αρχίσουν να εμφανίζονται, η OGSA αρχιτεκτονική θα γίνεται όλο και πιο χρήσιμη βασιζόμενη πάντα στις υπηρεσίες (**S**ervice **O**riented **A**rchitecture)

Επίπεδο εφαρμογών Grid

Με το πέρασμα του χρόνου και όσο οι Grid υπηρεσίες θα αναπτύσσονται, καινούργιες εφαρμογές βασισμένες στην Grid τεχνολογία θα κάνουν την εμφάνισή τους. Αυτές οι εφαρμογές θα χρησιμοποιούν μια ή και περισσότερες Grid υπηρεσίες του προηγούμενου επιπέδου.

2.1.3 Γενικά για τις υπηρεσίες

Τον τελευταίο καιρό έχει γίνει αρκετά μεγάλη συζήτηση γύρω από το θέμα των υπηρεσιών των εφαρμογών. Οι υπηρεσίες τείνουν να γίνουν τμήματα της εφαρμογής που αθροιστικά σχηματίζουν το περιβάλλον αυτής. Φυσικά δεν αποτελούν απλά ένα κομμάτι της εφαρμογής, αλλά έχουν χαρακτηριστικά που τις μετατρέπουν σε μέρος μιας αρχιτεκτονικής προσανατολισμένης στις υπηρεσίες (**S**ervice **O**riented **A**rchitecture).

Ένα από τα χαρακτηριστικά αυτά είναι η αυτονομία από άλλες υπηρεσίες. Αυτό σημαίνει ότι κάθε υπηρεσία είναι υπεύθυνη για το δικό της εύρος λειτουργίας, περιορίζοντας έτσι και εξειδικεύοντάς την σε συγκεκριμένες επαγγελματικές χρήσεις.

Αυτός ο σχεδιασμός έχει ως αποτέλεσμα τη δημιουργία ανεξάρτητων μονάδων, ελαστικά συνδεδεμένων μεταξύ τους με κάποιο πρότυπο πλαίσιο επικοινωνίας. Εξαιτίας αυτής της ανεξαρτησίας που απολαμβάνουν οι υπηρεσίες στο πλαίσιο αυτό, η προγραμματιστική λογική που κάθε υπηρεσία χρησιμοποιεί, δεν χρειάζεται να προσαρμόζεται σε συγκεκριμένη πλατφόρμα ή τεχνολογία. Είναι χαρακτηριστικό των υπηρεσιών ενός πλαισίου, ο πολυμορφισμός των μερών του με ταυτόχρονη ομαλή συνεργασία.

2.1.3.1 XML Web services (Διαδικτυακές Υπηρεσίες)

Ο πιο ευρέως διαδεδομένος και επιτυχημένος τύπος υπηρεσιών είναι οι XML Services γνωστές ως Web Services.

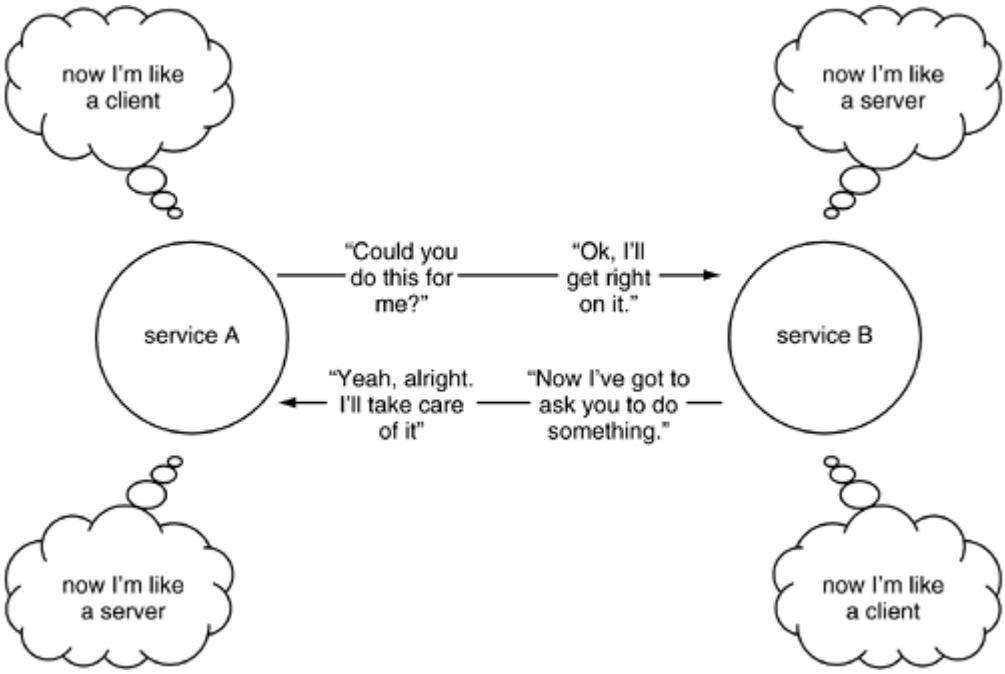
Αυτός ο τύπος υπηρεσίας έχει δύο βασικές προαπαιτήσεις:

- επικοινωνεί μέσω πρωτοκόλλου Internet (κυρίως HTTP)
- στέλνει και δέχεται δεδομένα μέσα από XML αρχεία

Η ευρεία αποδοχή του μοντέλου των Web Services είχε ως αποτέλεσμα την ανάγκη πρόσθετων τεχνολογιών που βασίζονται σε αυτές και τη δημιουργία καινούργιων προτύπων. Έτσι η "παραγωγή" τέτοιων υπηρεσιών απαιτεί:

- τη περιγραφή της υπηρεσίας αναλυτικά, τουλάχιστον με ένα WSDL έγγραφο
- τη δυνατότητα μεταφοράς ενός XML εγγράφου χρησιμοποιώντας SOAP μέσω HTTP.

Επιπρόσθετα είναι συνηθισμένο μια υπηρεσία να λειτουργεί και ως πελάτης (client/requestor) και ως πάροχος (provider) υπηρεσίας. Αναλόγως λοιπόν με τη δραστηριότητα της υπηρεσίας κάθε στιγμή, μετατρέπεται από το ένα στο άλλο. Στο παρακάτω σχήμα φαίνεται ένα παράδειγμα αυτής της συμπεριφοράς:

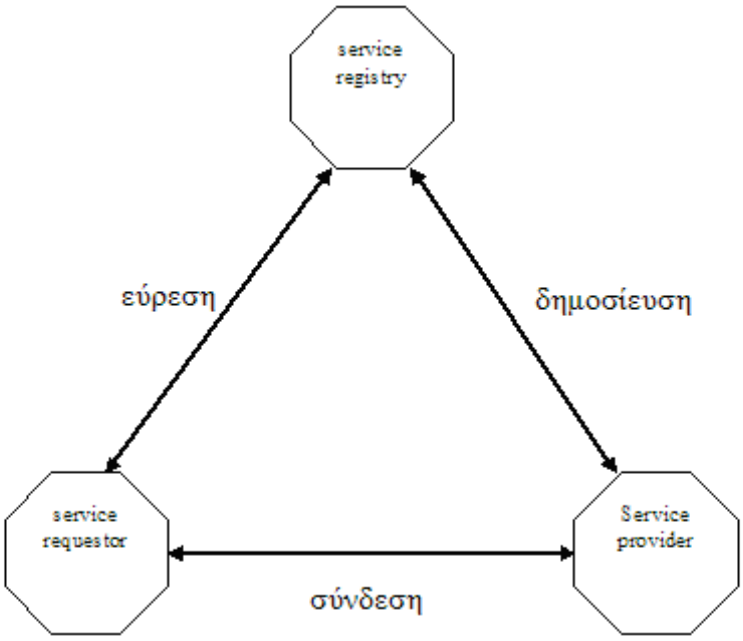


Εικόνα 3 : Ανταλλαγή ρόλων μεταξύ Web services κατά τη διάρκεια μιας επικοινωνίας

2.1.4 Service Oriented Architecture (Υπηρεστρεφής Αρχιτεκτονική)

Το να προσαρμόσεις σε μια εφαρμογή κάποιες υπηρεσίες (Web Services) δεν είναι μια δύσκολη διαδικασία και επιπρόσθετα θα αποδώσει στην εφαρμογή πρόσθετα λειτουργικά χαρακτηριστικά. Τα χαρακτηριστικά αυτά όμως δεν δημιουργούν μια υπηρεσιοστρεφή αρχιτεκτονική (Service Oriented Architecture), αφού η διαφορά τους είναι πιο μεγάλη.

Το SOA πρότυπο είναι ένα σχεδιαστικό μοντέλο με κύριο χαρακτηριστικό την ενσωμάτωση λογικής εφαρμογών μέσα σε υπηρεσίες που θα αλληλεπιδρούν μέσω επικοινωνιακών πρωτοκόλλων. Έτσι η υιοθέτηση σε μια εφαρμογή, SOA δομής σημαίνει αυτόματα την αποδοχή κάποιων σχεδιαστικών αρχών και πρόσθετων τεχνολογιών ως βασικού τμήματος του τεχνικού περιβάλλοντος της.



Εικόνα 4 : Λειτουργία SOA

Σε επίπεδο σχεδιασμού συστημάτων, η χρήση της τεχνολογίας των web services οδηγεί στην υιοθέτηση της λεγόμενης service-oriented αρχιτεκτονικής. Οι βασικοί ρόλοι και λειτουργίες στην αρχιτεκτονική αυτή παρουσιάζονται στην Εικόνα 4[ACK+04]. Η αρχιτεκτονική αυτή υποδεικνύει μια σχέση εξυπηρετητή-πελάτη (server-client) ανάμεσα στον προμηθευτή υπηρεσιών (service provider, που παίζει το ρόλο του server) και τον ζητώντα την υπηρεσία (service-requestor, που παίζει το ρόλο του client). Ο service-provider είναι αυτός που παρέχει την υπηρεσία δεχόμενος μηνύματα κλήσεις από τους requestors. Είναι επίσης υπεύθυνος για τη δημιουργία της περιγραφής της υπηρεσίας (service description) και τη δημοσίευσή της σε κάποιο κατάλογο-οδηγό υπηρεσιών (Universal Description, Discovery and Integration-UDDI). Ο service requestor αναζητά την υπηρεσία και την περιγραφή της σε κάποιο κατάλογο υπηρεσιών (service registry) και στη συνέχεια καλεί κατάλληλα την επιθυμητή υπηρεσία. Ο κατάλογος υπηρεσιών φέρνει ουσιαστικά τις δύο πλευρές, client και server σε επαφή. Η συνέχεια αφορά μόνο τις δύο άλλες συμμετέχουσες μονάδες (service requestor και service provider).

2.1.5 Web Services Description Language (WSDL)

Οι Web Services χρειάζεται να ορίζονται με συγκεκριμένο τρόπο έτσι ώστε να μπορούν να εντοπιστούν και χρησιμοποιηθούν από άλλες υπηρεσίες και εφαρμογές. Για αυτό το σκοπό

δημιουργήθηκε από τον οργανισμό W3C, μια γλώσσα περιγραφής γνωστή ως WSDL (Web Services Description Language).

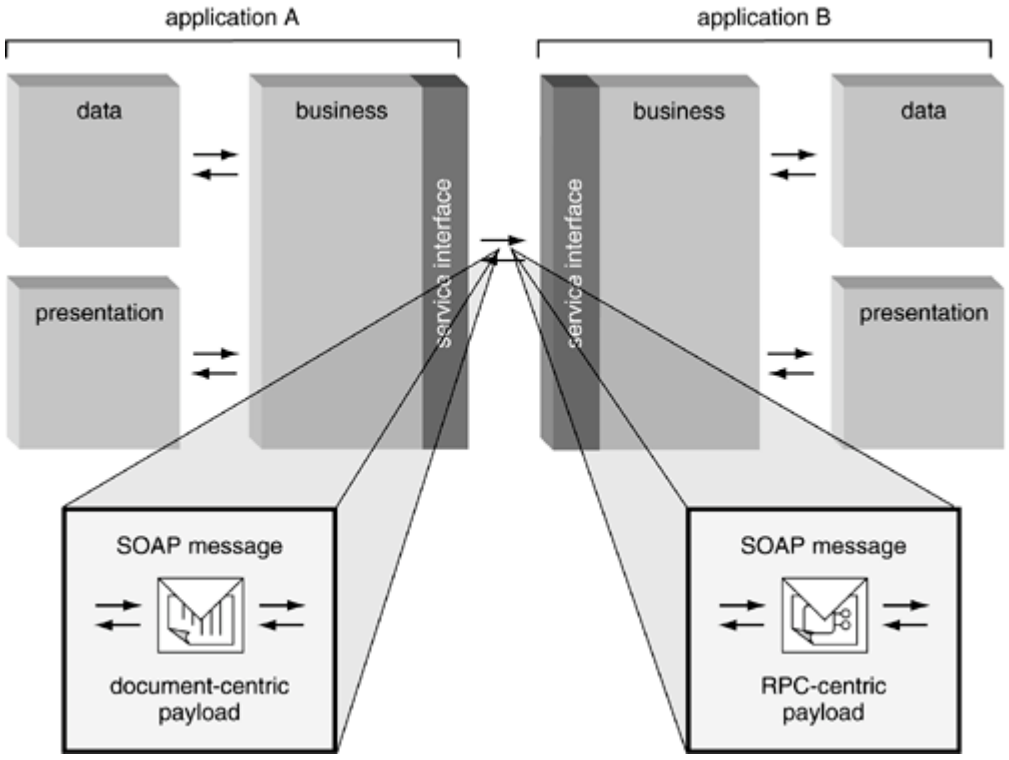
Η γλώσσα αυτή είναι μια XML δομή εγγράφου, που περιέχει όλες τις πληροφορίες σχετικά με τα δεδομένα εισόδου και εξόδου, τις μεθόδους και ότι άλλο χρειάζεται να γνωρίζει κάποιος για την ακριβή χρήση μιας Web υπηρεσίας.

2.1.6 Simple Object Access Protocol (SOAP)

Αν και αρχικά είχε θεωρηθεί ως η τεχνολογία που θα γεφυρώσει το κενό μεταξύ ανόμοιων πλατφορμών βασισμένων σε RPC επικοινωνία, το SOAP έχει εξελιχθεί στο ευρέως χρησιμοποιούμενο πρότυπο επικοινωνίας για τη χρήση XML Υπηρεσιών Διαδικτύου (Web Services). Μετά από αυτή την κατάσταση γίνεται πολλές φορές η παράφραση του ακρωνύμου από Simple Object Access Protocol σε Service-Oriented Architecture (or Application) Protocol.

Το πρωτόκολλο SOAP διαμορφώνει ένα πρότυπο μήνυμα που αποτελείται από ένα XML έγγραφο ικανό να περιγράψει δεδομένα όπως RPC κλήσεις κ.α. Το μήνυμα αυτό μεταφέρεται μεταξύ των υπηρεσιών και των εφαρμογών, χρησιμοποιώντας κυρίως το HTTP πρωτόκολλο δικτύου. Με τον τρόπο αυτό ολοκληρώνεται το πλαίσιο λειτουργίας και επικοινωνίας στην SOA δομή, αφού με την βοήθεια της περιγραφής WSDL είναι εφικτή η επικοινωνία και συνεργασία οποιωνδήποτε υπηρεσιών στο δίκτυο.

Στην Εικόνα 5 βλέπουμε τη χρήση του πρωτοκόλλου σε εφαρμογές είτε για προτυποποιημένη RPC επικοινωνία είτε για γενική χρήση μεταφοράς μηνύματος.



Εικόνα 5 : RPC και Document Centric επικοινωνία

Το πρωτόκολλο SOAP καθορίζει ένα XML έγγραφο με μια συγκεκριμένη δομή που μέσα του θα εμπεριέχονται οι πληροφορίες. Η βασική δομή ενός τέτοιου εγγράφου είναι η παρακάτω:

```
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>

   </soap:Body>
</soap:Envelope>
```

Εσωτερικά του "σώματος" (Body) του εγγράφου, ενσωματώνεται η πληροφορία που θέλουμε να μεταφέρουμε από τη μία υπηρεσία στην άλλη. Η τεχνολογία που χρησιμοποιήθηκε για απομακρυσμένη κλήση υπηρεσιών είναι η SOAP-RPC, (επέκταση της τεχνολογίας Remote Procedure Call) η οποία εισάγει στο SOAP έγγραφο τις απαιτούμενες πληροφορίες, σύμφωνα και με το WSDL της υπηρεσίας που θα κληθεί και στέλνει το μήνυμα.

Όπως είδαμε και στο σχήμα προηγουμένως, εκτός από την χρήση που περιγράφηκε, το SOAP μπορεί να μεταφέρει οποιαδήποτε άλλη πληροφορία ενσωματωμένη σε XML δομή, μέσα στο "φάκελο" του SOAP μηνύματος, αρκεί βέβαια ο παραλήπτης να γνωρίζει πώς να αποκωδικοποιήσει το έγγραφο αυτό.

2.1.7 Ποιότητα Υπηρεσίας (Quality of Service)

Ο όρος QoS (Quality of Service) χρησιμοποιήθηκε αρχικά ως ορολογία στα δίκτυα υπολογιστών για να αποδώσει την ικανότητα ενός δικτύου να παρέχει καλύτερες υπηρεσίες για συγκεκριμένη κίνηση (network traffic) χρησιμοποιώντας τα διάφορα πρωτόκολλα δικτύου. Η ανάγκη για αξιόπιστα δίκτυα και η βελτίωση των υπηρεσιών που προσέφεραν αυτά δημιούργησε αυτή την παράμετρο, που τώρα θεωρείται ένας πολύ βασικός παράγοντας αξιολόγησης.

2.1.7.1 Η έννοια της Ποιότητας Υπηρεσίας στο Grid

Η ανάπτυξη των Grid στον επιχειρηματικό κόσμο (business Grids) οδήγησε στην υιοθέτηση της ορολογίας αυτής και στον χώρο των Grid υπηρεσιών. Η σημασία της ποιότητας των παρερχομένων υπηρεσιών σε ένα Grid σύστημα είναι πολύ μεγάλη, αφού αυτό που κάνει μια υπηρεσία προσιτή στον πελάτη, εκτός από την ίδια την υπηρεσία, είναι η ποιότητά της. Επίσης η ιδιαιτερότητα των Grid συστημάτων πρώτον ως προς τη συνεργασία ετερογενών συστημάτων και δεύτερον ως προς την αρχιτεκτονική τους (Service Oriented Architecture) έκανε πιο εμφανή την ανάγκη προσδιορισμού της ποιότητας των υπηρεσιών. Όσον αφορά το πρώτο, η συμμετοχή πολλών διαφορετικών συστημάτων στο ίδιο Grid απαιτεί μια κοινή παράμετρο αξιολόγησης αυτών αλλά και των υπηρεσιών που αυτά προσφέρουν. Από την άλλη, η αρχιτεκτονική που χρησιμοποιείται σε τέτοια περιβάλλοντα είναι βασισμένη στις υπηρεσίες, έτσι η εισαγωγή του παράγοντα QoS είναι εφικτή σε όλα τα επίπεδα της δομής του Grid.

Τα χαρακτηριστικά του QoS μπορούν να διαχωριστούν σε δύο κατηγορίες : αυτά που βασίζονται στα ποσοτικά και αυτά που βασίζονται στα ποιοτικά χαρακτηριστικά μιας Grid υποδομής. Τα πρώτα αναφέρονται σε ιδιότητες όπως η απόδοση του επεξεργαστή, η καθυστέρηση του δικτύου, ο διαθέσιμος αποθηκευτικός χώρος κλπ., ενώ τα δεύτερα έχουν σχέση με την αξιοπιστία των υπηρεσιών και την ικανοποίηση του χρήστη.

Ο προσδιορισμός και η συλλογή των πιο συγκεκριμένων παραμέτρων για τον καθορισμό του QoS δεν είναι πάντα μια εύκολη διαδικασία. Κάποιες βασικές πληροφορίες μπορούν να εξαχθούν από προγράμματα παρακολούθησης (monitoring) και άλλα συστήματα μοντέλα προτυποποίησης των πόρων [CIM].

Από την άλλη υπάρχουν αρκετές παράμετροι που εξαιτίας της ιδιομορφίας του Grid συστήματος (δυναμική εισαγωγή ετερογενών υπολογιστικών και μη συστημάτων) δεν είναι εύκολο να προσδιοριστούν. Προσεγγίζοντάς το από τη πλευρά του πελάτη ενός business Grid, και παρατηρώντας τι πληροφορίες το Grid θα μπορούσε να δώσει, καταλήγουμε στις παρακάτω κατηγορίες QoS πληροφορίας:

- Διαθεσιμότητα : η πληροφορία αυτή μπορεί να δοθεί με μορφή ένδειξης, για το αν είναι διαθέσιμο το στιγμιότυπο της υπηρεσίας (service up-and-running), και άλλα απαραίτητα αρχεία και βιβλιοθήκες.
- Απόδοση : ο προσδιορισμός αυτής της παραμέτρου είναι αρκετά δύσκολος εξαιτίας της αδυναμίας μοντελοποίησης ενός γενικού τρόπου εξαγωγής της πληροφορίας. Παρόλα αυτά μπορούμε να θεωρήσουμε ότι υπολογίζοντας το χρόνο απόκρισης, τη ταχύτητα του επεξεργαστή και κάποιες άλλες τιμές ενδεικτικές για την αποδοτικότητα και ικανότητα του συστήματος, έχουμε μια αφηρημένη εικόνα για την παράμετρο αυτή.
- Αξιοπιστία : η παράμετρος αυτή πηγάζει από την επεξεργασία των αποτελεσμάτων της διαθεσιμότητας. Η διαθεσιμότητα μιας υπηρεσίας για μεγάλο διάστημα χρόνου, της αποδίδει αξιοπιστία.
- Κόστος : η παράμετρος αυτή έχει να κάνει αποκλειστικά με την τιμολόγηση των υπηρεσιών σε σχέση με το χρόνο κλπ.
- Στατιστικά στοιχεία για την ποιότητα του αποτελέσματος (Quality of the Result) : τα στοιχεία αυτά επιτυγχάνονται από την αξιολόγηση των υπηρεσιών από τους πελάτες, αποδίδοντας ένα βαθμό ικανοποίησης του χρήστη.
- Στατιστικά στοιχεία για την παραβίαση των SLA συμφωνιών : μια ένδειξη για την κατάσταση μιας υπηρεσίας σε σχέση με το συμφωνηθέν SLA θα αποτελούσε χρήσιμη πληροφορία

Οι δύο πρώτες κατηγορίες ανήκουν στα ποσοτικά χαρακτηριστικά, σύμφωνα με τον αρχικό διαχωρισμό, ενώ τα υπόλοιπα αφορούν την ποιότητα της παρεχομένης υπηρεσίας και ανήκουν στα ποιοτικά.. Παρόλο που οι ιδιότητες οι σχετικές με την ποιότητα της υπηρεσίας (ποιοτικά χαρακτηριστικά) είναι πολύ σημαντικές στην αξιολόγηση του QoS, είναι δύσκολο να υπολογιστούν αντικειμενικά, αφού βασίζονται κατά πολύ στην ανάδραση πληροφορίας από τον χρήστη. Έτσι θα επικεντρώσουμε την προσοχή μας στις πληροφορίες για την διαθεσιμότητα (availability) και αποδοτικότητα (performance).

2.1.7.2 Η εφαρμογή της Ποιότητας Υπηρεσίας στο Grid

Είναι απαραίτητο οι εφαρμογές που θα εκτελεστούν στο Grid να προσδιορίσουν τις απαιτήσεις τους για QoS ως τα χαρακτηριστικά των πόρων που απαιτούνται για την εκτέλεση αυτή (αποθηκευτικό χώρο, εύρος δικτύου, επεξεργαστική ισχύ κλπ.). Επίσης θα πρέπει να εξασφαλίζεται ο εντοπισμός του κατάλληλου πόρου σύμφωνα με τις απαιτήσεις αυτές. Η έρευνα σε αυτό το θέμα έχει καταλήξει σε ένα γενικό μοντέλο διαχείρισης QoS. Σύμφωνα με

αυτό η κάθε εφαρμογή υποβάλει τις απαιτήσεις της στον διαχειριστή Grid πόρων (resource manager), ο οποίος υπεύθυνος για την ανάθεση εργασιών στους πόρους. Έτσι ο κάθε πάροχος πόρων (resource provider) πρέπει να υποστηρίζει διαχειριστή ή δρομολογητή για να δέχεται τις αιτήσεις των εφαρμογών. Εκτός όμως από την ανάγκη ύπαρξης QoS πληροφορίας κατά την εκτέλεση μιας εφαρμογής (on demand), είναι απαραίτητη η ύπαρξη αρχείου πληροφοριών με το ιστορικό των παραμέτρων του QoS για λόγους επεξεργασίας και εξαγωγής χρήσιμων αποτελεσμάτων. Συνολικά λοιπόν, μπορούμε να παρατηρήσουμε ότι αφού έχουμε να κάνουμε με QoS σε Grid, δηλ. με πολλές εφαρμογές σε πολλούς πόρους ταυτόχρονα, το SLA απαιτεί να διευκρινίζει το επίπεδο της υπηρεσίας που ο χρήστης επιθυμεί και ο πάροχος πρέπει να προσφέρει.

2.1.7.3 Η Ιστορία της Ποιότητας της Υπηρεσίας στα Grids

Οι παλαιότερες προσπάθειες για την εισαγωγή της έννοιας του QoS στη διαχείριση των πόρων είχαν μεγάλο ενδιαφέρον και σημαντικά αποτελέσματα. Ο Sahai [SGM+03] πρότεινε την υποστήριξη της QoS πληροφορίας σε SLA οντότητες στα ευρείας χρήσης (εμπορικά) Grid. Αν και πολύ ενδιαφέρουσα προσέγγιση, δεν αναπτύχθηκε σημαντικά και είναι ακόμα σε ερευνητικό στάδιο. Ένα γενικό μοντέλο με την ονομασία SNAP (Service Negotiation and Acquisition Protocol) προτάθηκε από τον Czajkowski [CFK+02]. Το μοντέλο αυτό διακρίνει τρία είδη SLA για την ανάθεση των εργασιών στο Grid, Task SLA (TSLA), Resource SLA (RSLA), Bind SLA (BSLA). Το πρώτο από αυτά εμπεριέχει τις απαιτήσεις για την ποιότητα της υπηρεσίας που ζητήθηκε. Ο Keahey πρότεινε την αρχιτεκτονική VAS (Virtual Application Service)[9] για την διαχείριση QoS στα Grid. Η αρχιτεκτονική αυτή εισήγαγε καινούριους τρόπους διαπραγμάτευσης του επιπέδου της ποιότητα των υπηρεσιών και των απαιτήσεων. Τέλος το επικρατέστερο και πληρέστερο μοντέλο για την υποστήριξη QoS στα Grid είναι το General-purpose Architecture for Reservation and Allocation (GARA) [FKL+99]. Το GARA εξασφαλίζει ότι ο χρήστης ή η εφαρμογή θα λάβει συγκεκριμένη ποιότητα υπηρεσίας από τον διαχειριστή των πόρων. Αν και οι δυνατότητες του είναι αξιόλογες, έχει κάποιους περιορισμούς αφού δεν είναι συμβατό με το OGSA και δεν υποστηρίζει SLA.

2.2 Στόχος

Στην ενότητα αυτή περιγράφεται η ανάπτυξη των τεχνολογιών που αναλύθηκαν προηγουμένως και η χρήση τους πλέον εμπορικά. Αρχικά θα γίνει μια προσέγγιση της

τεχνολογίας του Grid στον επιχειρηματικό τομέα και στη συνέχεια θα εξετάσουμε την τεχνολογία αυτή ως κατεξοχήν αντικείμενο προς εμπορική εκμετάλλευση.

2.2.1 Από το ακαδημαϊκό στο επιχειρηματικό Grid (Business Grid)

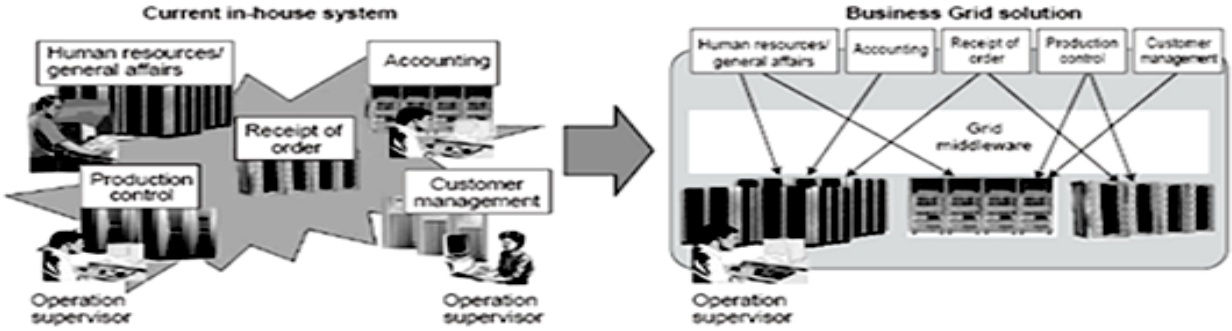
Όπως μιλήσαμε και νωρίτερα, ο όρος Grid αναφέρεται σε μεγάλης κλίμακας διαχείριση και κοινή χρήση πόρων, με σκοπό την επίλυση δύσκολων προβλημάτων. Αυτή η διαχείριση και ταυτόχρονα και η συμμετοχή στο Grid γίνεται από οργανωμένες ομάδες που ονομάζονται "εικονικοί οργανισμοί" (Virtual Organizations) γνωστοί ως VOs [F01]. Τέτοιοι οργανισμοί μπορούν να θεωρηθούν πάροχοι υπηρεσιών εφαρμογών (application service providers), αποθηκευτικού χώρου (storage service providers), βάσεις δεδομένων, συστήματα προσομοίωσης κ.α.

Η συνεργασία των VOs είναι απαραίτητο να καθορίζεται από συγκεκριμένους κανόνες, έτσι ώστε να είναι ξεκάθαρη η σχέση μεταξύ παροχών υπηρεσιών και καταναλωτών, ποιος επιτρέπεται να χρησιμοποιήσει τους πόρους, ποιους πόρους, κάτω από ποιες συνθήκες κλπ. Οι κανονισμοί αυτοί προτυποποιήθηκαν και έτσι δημιουργήθηκαν τα SLA (Service Level Agreement), συμβόλαια δηλαδή με όλες τις πληροφορίες και απαιτήσεις μεταξύ των παροχών των υπηρεσιών και των αποδεκτών τους.

Η χρήση SLA στις Grid υπηρεσίες έδωσε τις προδιαγραφές για την ανάπτυξη της Grid τεχνολογίας και εκτός του αυστηρά ερευνητικού τομέα. Η αλματώδης εξέλιξη του Internet και γενικά της δικτύωσης των υπολογιστών και ο αμείλικτος ανταγωνισμός των επιχειρήσεων, έδωσαν άλλο νόημα στον όρο e-business. Η ροή των εργασιών στις επιχειρήσεις, από την σχεδιασμό μέχρι την υλοποίηση είναι πιο γρήγορη από ποτέ. Στο χώρο όμως του e-business οι απροσδόκητες αλλαγές στο φόρτο εργασιών, στον όγκο πληροφοριών ή στις απαιτήσεις επεξεργασίας δημιούργησαν την απαίτηση για ευέλικτα, μικρού κόστους και μεγάλης αξιοπιστίας συστήματα. Πάνω σε αυτήν την απαίτηση στηρίχθηκε η Grid τεχνολογία για την δημιουργία του business Grid [LG]. Την υιοθέτηση δηλαδή Grid υποδομών στον επιχειρηματικό κόσμο με σκοπό την εμπορική εκμετάλλευση των Grid υπηρεσιών. Πιο συγκεκριμένα οι στόχοι του business Grid είναι :

- την κοινόχρηστη χρήση IT (Information Technology) πόρων, μέσω δικτύωσης υπολογιστών των υπολογιστών απομακρυσμένων κέντρων πληροφοριών (data centers).
- Τη μείωση του κόστους στις επιχειρήσεις με τη χρήση ευέλικτων υποδομών και αυτόνομων συστημάτων ελέγχου
- Την δυναμική εισαγωγή και χρήση πόρων με την ταυτόχρονη διοχέτευση φόρτου εργασίας σε διαφορετικές απομακρυσμένες περιοχές.

- Την απλοποίηση, συνολικά των υποδομών και τη δημιουργία αξιόπιστων συστημάτων που θα υποστηρίζουν διασπορά εργασίας, ανάκτηση κατεστραμμένων πληροφοριών και άλλα λειτουργικά συστήματα και συστήματα ασφαλείας.



Εικόνα 6 : Business Grid

Όπως βλέπουμε και στην Εικόνα 6, η υποδομή του Business Grid βελτιστοποιεί τη εκμετάλλευση των πόρων μιας επιχείρησης με αποτέλεσμα την μείωση του κόστους και την αύξηση της παραγωγικότητας.

2.2.2 Προσέγγιση λειτουργίας του επιχειρηματικού μοντέλου του Grid

Αναφερθήκαμε νωρίτερα στην διεύρυνση χρήσης του Grid όχι μόνο στον ερευνητικό τομέα αλλά και στον επαγγελματικό. Η χρήση αυτή όμως δεν περιορίζεται στην εκμετάλλευση των συστημάτων αυτών εντός μιας επιχείρησης με σκοπό την μείωση κόστους, την αύξηση της παραγωγικότητας και μείωση του κύκλου παραγωγής, αλλά και στην εμπορική εκμετάλλευση αυτών των τεχνολογιών με και νωρίτερα, στα πλαίσια του business Grid, την δυνατότητα εμπορικής εκμετάλλευσης των Grid υπηρεσιών. Η απαίτηση της αγοράς για εξειδίκευση στην παροχή υπηρεσιών και η συγκεκριμένη αρχιτεκτονική του Grid (Service Oriented Architecture) συστήματος, αν δεν μας δίνουν ένα συγκεκριμένο μοντέλο λειτουργίας, μας δίνουν τουλάχιστον τις προδιαγραφές αυτού.

Για να περιγράψουμε τη λειτουργία ενός τέτοιου συστήματος θα αναφέρουμε τους ρόλους και τις οντότητες που συμμετέχουν σε αυτό. Έτσι μπορούμε να ξεχωρίσουμε τον πάροχο της υπηρεσίας (Service Provider). Αυτός ο όρος είναι πολύ γενικός και μπορεί να διασπαστεί και σε άλλες υπό-οντότητες, αφού οι παρεχόμενες υπηρεσίες που σχετίζονται είναι υπηρεσίες δικτύου, πόρων (αποθηκευτικών, υπολογιστικών κλπ.), εφαρμογών (application services) κ.α. Κάποιες από τις υπηρεσίες αυτές μπορούν να παρέχονται από το ίδιο άτομο ή να είναι διαμοιρασμένες σε περισσότερους. Σε κάθε περίπτωση, για την ομαλή και αποδοτική

συνεργασία μεταξύ τους υπάρχουν τα συμβόλαια (Service Level Agreements) που διασφαλίζουν τους όρους συνεργασίας. Όπως αναφέρθηκε και σε προηγούμενα κεφάλαια, τα SLA θα πρέπει να διασφαλίσουν και την ποιότητα των παρεχομένων υπηρεσιών (QoS) και αυτό μπορεί να επιτευχθεί με την εισαγωγή στο σύστημα και ενός άλλου ρόλου, του QoS Provider. Μιας οντότητας δηλαδή, η οποία θα επικοινωνεί με τους άλλους πάροχους υπηρεσιών, θα συλλέγει πληροφορίες και θα προσδιορίζει την ποιότητα των προσφερομένων υπηρεσιών από και μεταξύ κάθε πλευράς. Ο ρόλος αυτός είναι πολύ σημαντικός αφού διασφαλίζει συνολικά την ποιότητα της Grid υπηρεσίας που παρέχεται και που μας ενδιαφέρει να εκμεταλλευτούμε. Φυσικά η τελευταία οντότητα στο σύστημα αυτό, που δε πρέπει να ξεχνούμε είναι ο πελάτης που θέλει την χρήση των υπηρεσιών του Grid. Έτσι και αλλιώς γύρω από τις απαιτήσεις του πελάτη γίνονται όλες οι προσπάθειες για τη βελτίωση της ποιότητας των υπηρεσιών.

2.2.2.1 QoS Provider

Αναλύοντας λοιπόν περισσότερο τον μηχανισμό, μιας και σκοπός μας είναι η μοντελοποίηση του και τελικά η υλοποίηση του, θα πρέπει να περιγράψουμε τις προδιαγραφές λειτουργίας του. Η υπηρεσία παροχής QoS πληροφορίας θα πρέπει :

- Να έχει πρόσβαση σε πληροφορίες των πόρων ενός Grid συστήματος που άλλοι χρήστες μπορεί να μην έχουν
- Να έχει υλοποίηση γενικής χρήσης έτσι ώστε να μπορεί να εφαρμοστεί σε διαφόρων τύπων πόρους
- Να είναι συμβατό με τις τεχνολογίες που εφαρμόζονται στα Grid συστήματα
- Η υλοποίησή της να είναι διαθέσιμη σε δημόσια χρήση, έτσι ώστε οποιοσδήποτε επιθυμεί on-demand πληροφορία να μπορεί να την αποκτήσει
- Να γίνεται καταγραφή και επεξεργασία των πληροφοριών αυτών όχι μόνο για τεκμηρίωση της ποιότητα των προσφερομένων υπηρεσιών αλλά και για βελτίωση της λειτουργίας όλου του Grid συστήματος.

Μιλώντας για την ποιότητα υπηρεσιών σε Grid περιβάλλον, ανατρέχουμε κατευθείαν στους όρους της διαθεσιμότητας, αποδοτικότητας, αξιοπιστίας, κόστους κλπ. Η παραμετροποίηση αυτών των εννοιών και η εξαγωγή της αντίστοιχης πληροφορίας θα μας δώσει τα δεδομένα που χρειαζόμαστε για τον προσδιορισμό του QoS. Έχουμε αναλύσει τις παραμέτρους της ποιότητας των Grid υπηρεσιών σε προηγούμενη ενότητα και σε αυτό το σημείο μας ενδιαφέρουν οι όροι διαθεσιμότητα και απόδοση. Αυτά τα δύο χαρακτηριστικά εμπεριέχουν

όλη την πληροφορία μέσω της οποίας μπορεί να γίνει αποτίμηση της αξιοπιστίας, του κόστους και της τήρησης των συμβολαίων (SLA).

2.2.2.2 Διαθεσιμότητα (Availability)

Με τον όρο διαθεσιμότητα μπορούμε να συμπεριλάβουμε παραμέτρους όπως τη διαθεσιμότητα των εφαρμογών, διαθεσιμότητα των πόρων, διαθεσιμότητα του δικτύου κ.α. Το σημαντικότερο στοιχείο στην ποιότητα μιας υπηρεσίας Grid, αλλά και γενικά μιας οποιαδήποτε υπηρεσίας, είναι το κατά πόσο αυτή και ότι απαιτήσεις έχει, είναι διαθέσιμα. Ποιο αναλυτικά τα χαρακτηριστικά που θεωρούμε έκφραση της διαθεσιμότητας είναι :

- Διαθέσιμος ελεύθερος χώρος
- Εύρος δικτύου
- Απαραίτητες βιβλιοθήκες-αρχεία
- Υποστηριζόμενα πρωτόκολλα υπηρεσιών
- Κατάσταση υπηρεσιών εφαρμογών

Τα παραπάνω όπως παρατηρούμε, είναι οι στοιχειώδεις προϋποθέσεις για την εκτέλεση μιας Grid εφαρμογής. Προσπαθώντας να τηρήσουμε τις αρχικές απαιτήσεις λειτουργίας ενός συστήματος γενικής χρήσης, δε θα αναζητήσουμε παραμέτρους συγκεκριμένους για κάθε είδος συστήματος ή πόρου. Η ισορροπία στην επιλογή αυτή είναι λίγο λεπτή αφού πρέπει να αποκτήσουμε όσον το δυνατόν περισσότερη πληροφορία αλλά παράλληλα να κρατήσουμε την μορφή της QoS υπηρεσίας εφαρμόσιμη σε κάθε σύστημα.

2.2.2.3 Αποδοτικότητα (Performance)

Ένα άλλο χαρακτηριστικό που είναι πολύ σημαντικό και καθοριστικό για την ποιότητα της υπηρεσίας που παρέχεται είναι η απόδοση των συστημάτων που συμμετέχουν στο Grid. Ο καθορισμός τέτοιων παραμέτρων που θα προσδιορίζουν την απόδοση, είναι αρκετά δύσκολος και διαφέρει από σύστημα σε σύστημα. Επίσης η ταχύτητα με την οποία μπορεί να αλλάξει η κατάσταση ενός συστήματος από αποδεκτής απόδοσης σε μη αποδεκτής δυσκολεύει περισσότερο την μοντελοποίηση τέτοιων μηχανισμών. Λαμβάνοντας υπόψη ότι το μεγαλύτερο μέρος των συστημάτων που συμμετέχουν θα είναι υπολογιστικές μονάδες, κάποιες μελέτες έχουν καταλήξει σε παραμέτρους όπως :

- Συχνότητα ρολογιού επεξεργαστή
- Υπολογισμός MFLOPs (Millions Floating Operations per second)
- Μέσο Εύρος ζώνης μνήμης

- Μέσο εύρος ζώνης σκληρού δίσκου
- Διαθέσιμη μνήμη

Ο προσδιορισμός πιο αξιόπιστων παραμέτρων αποτελεί αντικείμενο επιστημονικού ενδιαφέροντος και πεδίου συζήτησης.

2.2.2.4 Αξιοπιστία (Reliability)

Ένα τρίτο χαρακτηριστικό που είναι εξίσου σημαντικό με τα προαναφερθέντα είναι η αξιοπιστία της παρεχόμενης υπηρεσίας. Η αξιοπιστία της υπηρεσίας έχει άμεση σχέση με την διαθεσιμότητα σε σύγκριση όμως με την χρονική διάρκεια της. Εκτός από αυτό όμως, η αξιοπιστία εξαρτάται και από παραμέτρους όπως η συχνότητα εμφάνισης σφαλμάτων ή την πιθανότητα εμφάνισής τους.

Συνολικά όμως η αξιοπιστία της υπηρεσίας διαπιστώνεται από μακροχρόνια παρακολούθηση των παραμέτρων της διαθεσιμότητας και απόδοσης και συνολικής επεξεργασίας τους.

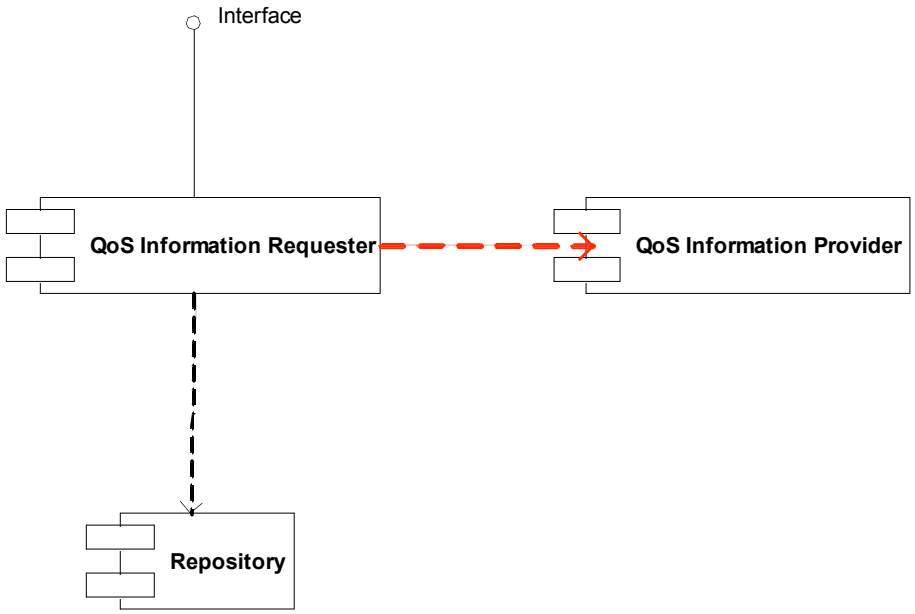
3 Ανάλυση και σχεδίαση

Στο κεφάλαιο αυτό θα γίνει μια παρουσίαση και ανάλυση της αρχιτεκτονικής και των λειτουργιών του συστήματος που θα αναπτύξουμε. Μια πρώτη προσέγγιση της αρχιτεκτονικής του μηχανισμού που θα ονομάσουμε “QoS Provisioning Module” φαίνεται παρακάτω.

3.1 Περιγραφή Αρχιτεκτονικής

3.1.1 Μηχανισμός Πρόβλεψης Ποιότητας Υπηρεσίας (QoS Provisioning Module)

Ο μηχανισμός παροχής της QoS πληροφορίας, όπως είπαμε, θα είναι μια υπηρεσία που θα είναι προσβάσιμη από οποιονδήποτε και εφαρμόσιμη σε διάφορες πλατφόρμες. Βασισμένοι στην αρχιτεκτονική του Grid (S.O.A.) και λαμβάνοντας υπόψη της προδιαγραφές του μηχανισμού αυτού, καταλήγουμε σε ένα μοντέλο με δύο μέρη : το πρώτο μέρος αποτελείται από την υπηρεσία εγκατεστημένη σε κάθε υπολογιστικό πόρο του Grid περιβάλλοντος και το δεύτερο μέρος οπουδήποτε στο σύστημα μας επιθυμούμε να συλλέξουμε και καταγράψουμε αυτή τη πληροφορία. Στο παρακάτω σχήμα φαίνεται ο μηχανισμός αυτός με τα δύο διακριτά μέρη QoS Information Requester και QoS Information Provider.



Εικόνα 7 : QoS Provisioning Module

Ο QoS Provider, αποτελεί το κομμάτι της υπηρεσίας που επικοινωνεί με τους πόρους και τις άλλες υπηρεσίες του Grid σε χαμηλό επίπεδο για να συλλέξει όλες τις πληροφορίες που χρειάζονται για την QoS εκτίμηση. Η υπηρεσία του Provider είναι εγκατεστημένη σε κάθε μέλος του Grid συστήματος έτσι ώστε τα δεδομένα από οποιαδήποτε πηγή να είναι προσβάσιμα.

Από την άλλη πλευρά ο Requester :

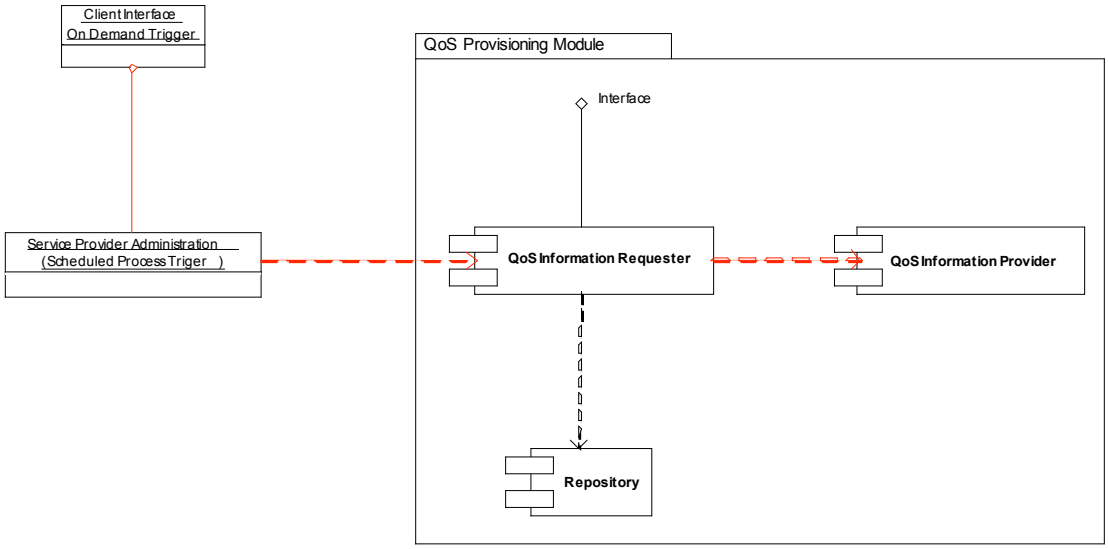
- Καλεί την υπηρεσία του Provider έτσι ώστε να εκκινήσει η διαδικασία συλλογής πληροφοριών
- Συγκεντρώνει, επεξεργάζεται και αποθηκεύει τα δεδομένα που επιστρέφονται από την υπηρεσία του Provider. Τα δεδομένα αυτά αποθηκεύονται σε κάποια βάση δεδομένων (βλ. σχήμα *Repository*) για μελλοντική χρήση.
- Δέχεται ερωτήματα (queries) από τρίτη πηγή που θέλει να έχει πρόσβαση στα αποθηκευμένα δεδομένα. Σύμφωνα με τις προδιαγραφές του μηχανισμού αυτού, η QoS υπηρεσία θα πρέπει να είναι ανοιχτή σε εξωτερικές κλήσεις, σε όλα τα άτομα με τα κατάλληλα φυσικά δικαιώματα πρόσβασης.

Ο Information Requester μπορεί να είναι εγκατεστημένος οπουδήποτε στο Grid σύστημα, όμως το πιο κατάλληλο σημείο λειτουργίας του είναι ως μέρος της υποδομής του παρόχου

υπηρεσιών (Service Provider). Αυτό γίνεται για καλύτερη ασφάλεια στα δεδομένα που ο πάροχος θέλει να κρατάει εσωτερικά του VO.

3.1.2 Ολοκλήρωση σχεδιασμού του μηχανισμού

Για να επιτύχουμε όσο το δυνατόν όλες τις λειτουργικές προδιαγραφές του μηχανισμού, πρέπει να σχεδιάσουμε ένα πιο ολοκληρωμένο μοντέλο από αυτό που παρουσιάσαμε προηγουμένως που θα παρουσιάζει και τις υπόλοιπες οντότητες που συμμετέχουν στην διαδικασία.



Εικόνα 8 : QoS Provisiong Module (Ολοκληρωμένη Έκδοση)

Όπως βλέπουμε παραπάνω, οι προστιθέμενες οντότητες είναι η Service Provider Administration και Client. Αυτοί οι δύο καινούργιοι ρόλοι υλοποιούν τις προδιαγραφές του συστήματος για απόκτηση πληροφοριών από τον Service Provider, με σκοπό τον εσωτερικό έλεγχο του συστήματός του, και ελεύθερη διάθεση, όσων πληροφοριών ο πάροχος της υπηρεσίας επιθυμεί, στους χρήστες του Grid.

Ο μηχανισμός που εμείς θα κατασκευάσουμε, είναι η υπηρεσία του QoS Provisioning και αποτελείται από τα συστατικά (components) που αναλύθηκαν και προηγουμένως. Το διαδραστικό περιβάλλον διαχείρισης και εκτέλεσης είναι αυτό που φαίνεται εκτός του μηχανισμού (QoS Provisioning Module)

Στο προηγούμενο κεφάλαιο αναλύθηκαν όλες οι παράμετροι που σχετίζονται με την ποιότητα της υπηρεσίας των Grid εφαρμογών και υπηρεσιών. Στην υλοποίηση μας θα ασχοληθούμε

μόνο με την Διαθεσιμότητα (Availability) των εφαρμογών, πόρων και υπηρεσιών στο Grid, συλλέγοντας και αποθηκεύοντας κατάλληλες παραμέτρους.

3.1.3 Ταξινόμηση υπηρεσιών (Core and Application Services)

Για την ομαλή λειτουργία ενός GRID συστήματος είναι απαραίτητη η ύπαρξη κάποιων μηχανισμών που θα ελέγχουν και θα καθοδηγούν τις βασικές δραστηριότητες του περιβάλλοντος αυτού, όπως η δυναμική εισαγωγή πόρων, η τήρηση των συμβολαίων κ.α. Αυτούς τους μηχανισμούς θα μπορούσαμε να τους ονομάσουμε υπηρεσίες πυρήνα (Core Services) αφού χωρίς αυτές δεν είναι δυνατή η λειτουργία του Grid. Από την άλλη οι υπηρεσίες αυτές που απλά εξυπηρετούνται από το σύστημα αυτό όπως εφαρμογές που εκτελούνται σε Grid περιβάλλον μπορούν να θεωρηθούν ως υπηρεσίες εφαρμογών (Application Services). Ο μηχανισμός για την ποιότητα των υπηρεσιών, που εμείς εξετάζουμε βρίσκεται στο μέσο αυτής της διαβάθμισης των Grid υπηρεσιών, αφού ούτε υπηρεσία πυρήνα είναι αλλά ούτε Grid εφαρμογή. Το QoS module είναι μια υπηρεσία θεμελίωσης και προτυποποίησης της προσφοράς υπηρεσιών του Grid. Δεν είναι μια λειτουργική υπηρεσία, κρίσιμη για την λειτουργία του συστήματος αλλά δεν είναι και μια απλή εφαρμογή που εκτελείται στο στρώμα εφαρμογών του. Αν θα έπρεπε να την ταξινομήσουμε σε μία από τις δύο κατηγορίες που αναφέρθηκαν, από τη μια η σημαντικότητα της υπηρεσίας στην αξιόπιστη λειτουργία του business Grid την καθιστά Core Service, από την άλλη όμως η διαφάνειά της ως προς τους χρήστες του GIRD και η αρχιτεκτονική της ως Grid Service της αποδίδει Application Service χαρακτηριστικά. Η μελέτη και έρευνα στο μέλλον θα αξιολογήσουν πιο αποτελεσματικά την έννοια του QoS στην ιεραρχία των υπηρεσιών του Grid.

3.1.4 Εγκατάσταση συστήματος

Όπως παρουσιάσαμε το σύστημα αποτελείται από διάφορα μέρη. Θεωρούμε σκόπιμο σε αυτό το σημείο να διευκρινίσουμε την εμβέλεια και αρχιτεκτονική της εγκατάστασης όλων των συνιστωσών του μηχανισμού.

Ο Information Provider λοιπόν αποτελεί το κέντρο της υπηρεσίας του QoS Provisioning και είναι απαραίτητη η ύπαρξη του σε κάθε πόρο που θα συμμετέχει στην υπηρεσία αυτή. Δεν θα είναι προσβάσιμος εκτός των ορίων του Service Provider και οι αρμοδιότητες του είναι όταν κληθεί να επιστρέφει τις πληροφορίες σε αυτόν που τις ζήτησε. Δεν χρειάζεται να επεξεργάζεται και να αποθηκεύει δεδομένα. Αυτή τη δουλειά την αναλαμβάνει ο Information Requester, που βρίσκεται και αυτός μέσα στα όρια του Service Provider. Πρόσβαση σε αυτόν

μπορούν να έχουν υπό όρους και εξωτερικοί χρήστες. Ο Information Requester λειτουργεί σαν μεσάζοντας ανάμεσα στην υπηρεσία παροχής πληροφοριών (Information Provider) και τους υπόλοιπους εξωτερικά του Service Provider. Αρμοδιότητα του είναι να καλεί την υπηρεσία του Information Provider όποτε χρειαστεί, να επεξεργάζεται τα δεδομένα που επιστρέφει αυτή και να τα καταγράφει στο Repository. Κατ' επέκταση το Repository είναι εγκατεστημένο εσωτερικά του συστήματος του Service Provider και μόνο ο Information Requester επιτρέπεται να έχει πρόσβαση. Τέλος τα δύο συστατικά (components) που απομένουν έχουν να κάνουν με την εκτέλεση της υπηρεσίας του QoS Provisioning. Το ένα αφορά την εκκίνηση εσωτερικά του Service Provider με χρήση χρονοπρογραμματιστή ενώ το δεύτερο αφορά την κλήση κατ' απαίτηση κάποιου εξωτερικού χρήστη.

3.2 Περιγραφή Λειτουργιών

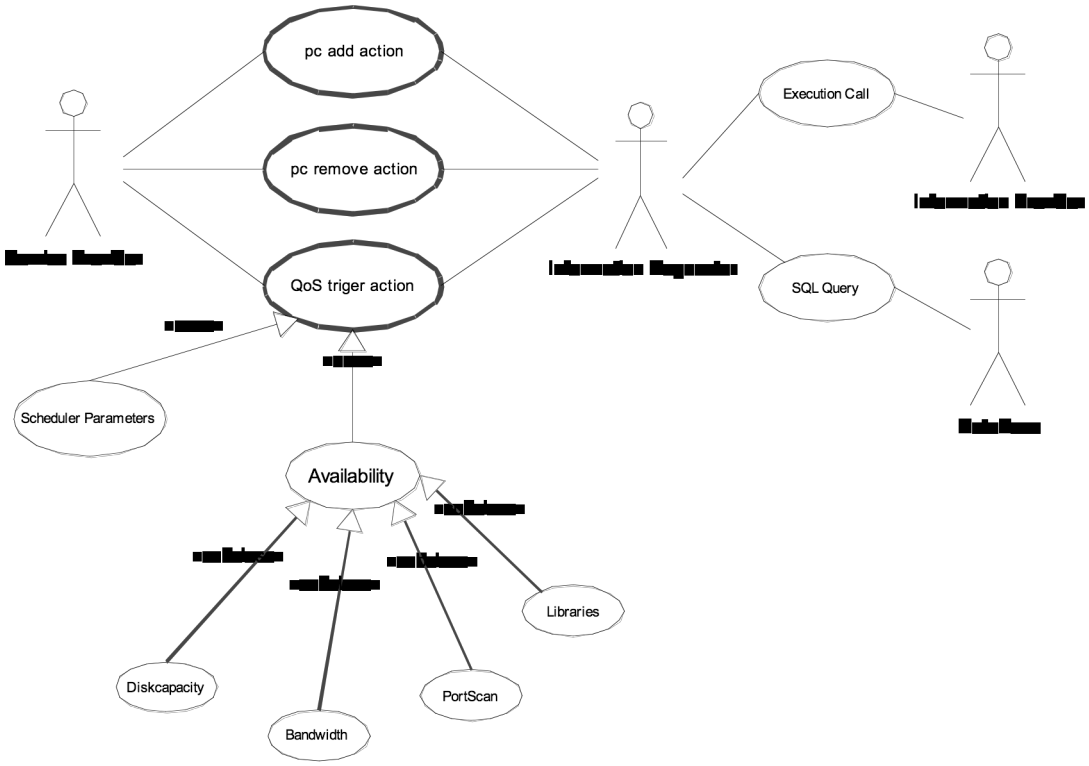
3.2.1 Καταστάσεις Λειτουργίας

Η υπηρεσία που θα κατασκευάσουμε, όπως έχουμε αναφέρει, θα έχει δύο καταστάσεις λειτουργίας, την προγραμματισμένη (scheduled) και την κατά απαίτηση (on demand) κατάσταση. Και οι δύο καταστάσεις θα είναι διαθέσιμες ταυτόχρονα, αφού η καθεμία έχει τον δικό της σκοπό λειτουργίας. Στην πρώτη περίπτωση γίνεται συλλογή πληροφοριών που μπορούν να αξιολογηθούν συνολικά σε βάθος χρόνου και να εξαχθούν συγκεκριμένα συμπεράσματα για την ποιότητα της υπηρεσίας, ενώ η δεύτερη περίπτωση είναι έκφραση της Service Oriented δομής που ακολουθείται, αφού ο μηχανισμός θα είναι διαθέσιμος για σύγχρονες κλήσεις (real time calls) από τους χρήστες του Grid.

3.2.2 Προγραμματισμένη Εκτέλεση

Κατά την προγραμματισμένη εκτέλεση σκοπός της υπηρεσίας QoS Provisioning, είναι η συλλογή πληροφοριών των πόρων εκ μέρους του διαχειριστή του Service Provider. Η έννοια αυτή αναφέρθηκε και νωρίτερα και έχει να κάνει με την οντότητα του πάροχου οποιασδήποτε υπηρεσίας βασισμένη σε Grid τεχνολογία. Η σημασία του QoS Module για αυτό το σκοπό είναι μεγάλη, αφού καθένας που προσφέρει υπηρεσίες πόρων, εφαρμογών, αποθήκευσης σε Grid περιβάλλον επιθυμεί να γνωρίζει την ποιότητα της υπηρεσίας που ο ίδιος προσφέρει με στόχο την βελτίωση, την σύγκριση ή ακόμα και την κοστολόγηση.

Για καλύτερη κατανόηση και παρουσίαση της λειτουργίας αυτής θα παρουσιάσουμε όλες τις δυνατές περιπτώσεις χρήσης για προγραμματισμένη εκτέλεση.



Εικόνα 9 : Use Case 1

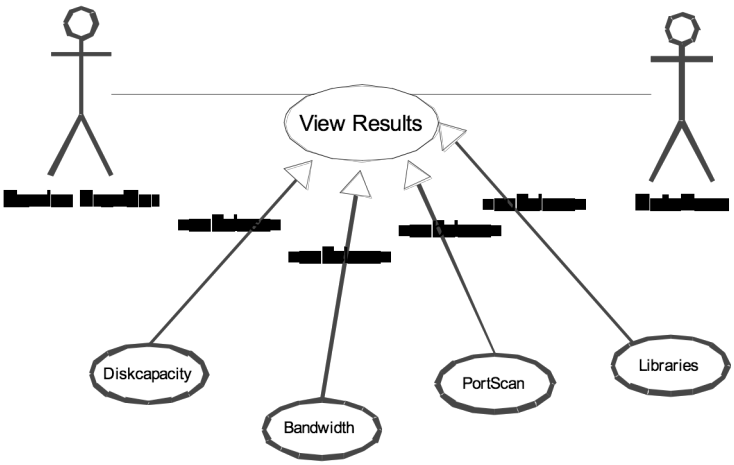
Στο διάγραμμα περίπτωσης χρήσης που παρουσιάστηκε βλέπουμε τις δυνατές δραστηριότητες του διαχειριστή του πάροχου υπηρεσίας (Service Provider Administrator), και πως αυτές επηρεάζουν άλλες οντότητες.

Αρχικά ο SP (Service Provider) μπορεί να προσθέσει ή να αφαιρέσει πόρους στους διαθέσιμους προς έλεγχο από το QoS Provisioning Module, μέσω της οντότητας του Requester. Χρησιμοποιώντας συγκεκριμένη λειτουργία και συμπληρώνοντας τις απαραίτητες πληροφορίες προσθέτει στην βάση δεδομένων του πόρους που θέλει να εντάξει στον QoS έλεγχο.

Σημαντικότερη λειτουργία είναι η εκκίνηση του QoS module, καλώντας την υπηρεσία QoS Provisioning από την Information Requester. Αρχικά πρέπει να επιλέξει τις παραμέτρους του προγραμματιστή (scheduler) και στη συνέχεια να επιλέξει τις παραμέτρους της διαθεσιμότητας που επιθυμεί να εκτελέσει στο σύνολο των πόρων του Grid.

Κατά αναλογία και με την αρχιτεκτονική που παρουσιάστηκε νωρίτερα, ο Information Requester επικοινωνεί με την βάση δεδομένων και με την υπηρεσία παροχής των QoS πληροφοριών για την ολοκλήρωση της δραστηριότητας αυτής.

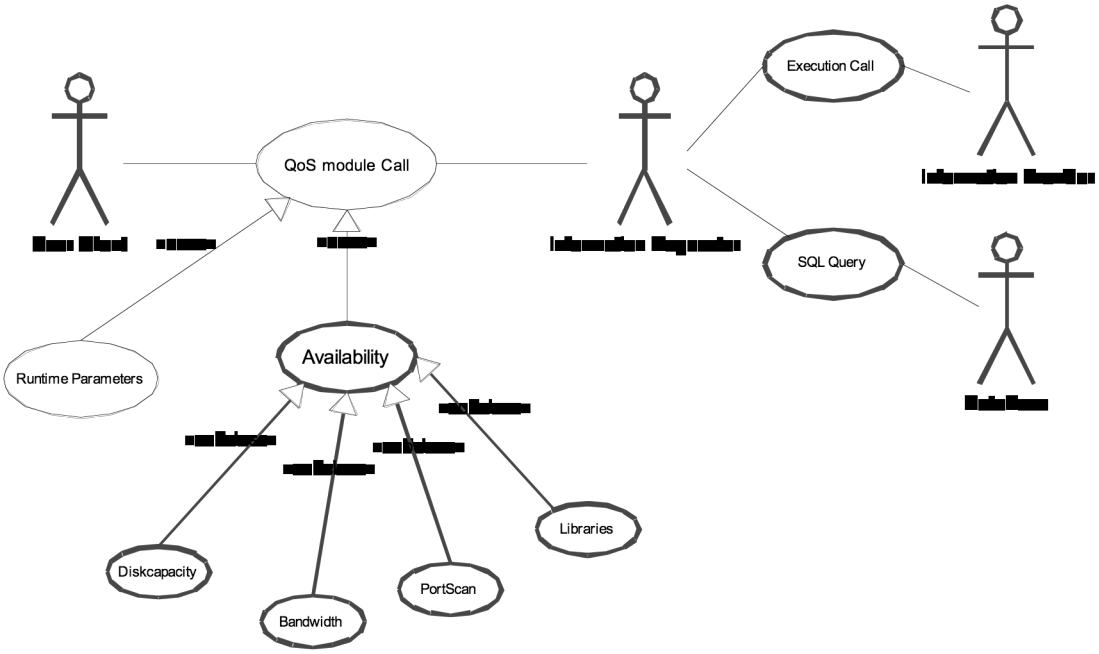
Τέλος ο SP μπορεί να δει πίνακες και διαγράμματα των αποτελεσμάτων προηγούμενων εκτελέσεων του QoS module:



Εικόνα 10 : Use Case 2

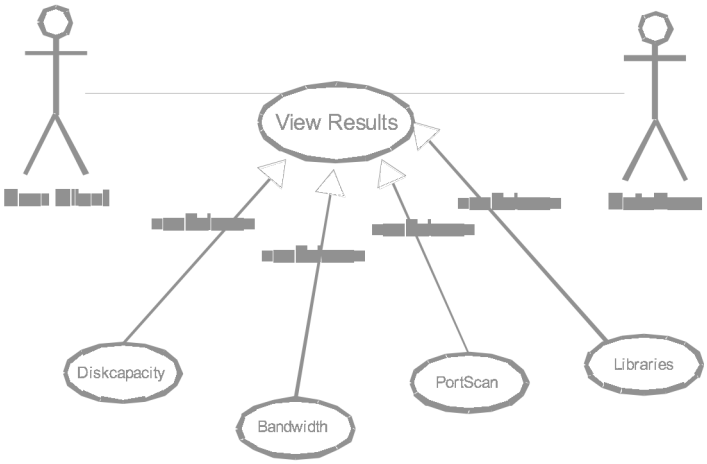
3.2.3 Εκτέλεση κατά απαίτηση

Αντίστοιχα με την προγραμματισμένη εκτέλεση, ένας χρήστης που επιθυμεί να εκτελέσει την QoS Provisioning υπηρεσία αρκεί να επιλέξει τις κατάλληλες παραμέτρους εκτέλεσης (τους πόρους που επιθυμεί, παραμέτρους διαθεσιμότητας). Αφού εκτελεστεί η κλήση του μέσω του Information Requester, τα αποτελέσματα του επιστρέφονται και καταγράφονται και αυτόματα στην βάση δεδομένων:



Εικόνα 11 : Use Case 3

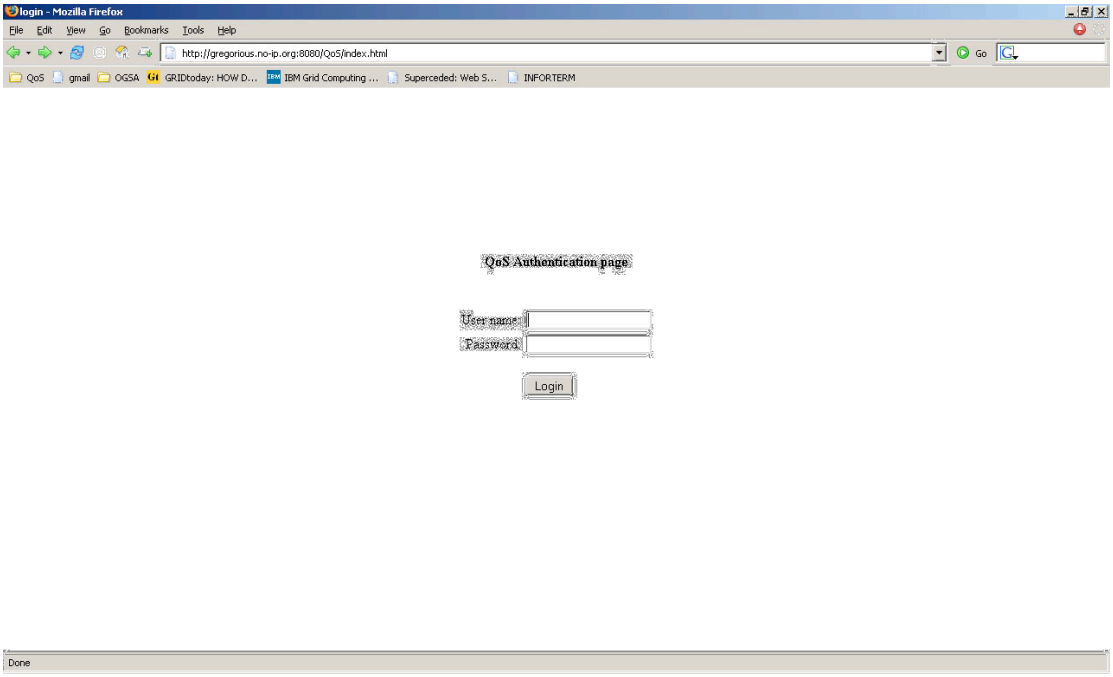
Τέλος , ο χρήστης μπορεί να δει τα αποτελέσματα και διαγράμματα παλαιότερων εκτελέσεων χρησιμοποιώντας συγκεκριμένη εφαρμογή:



Εικόνα 12 :Use Case 4

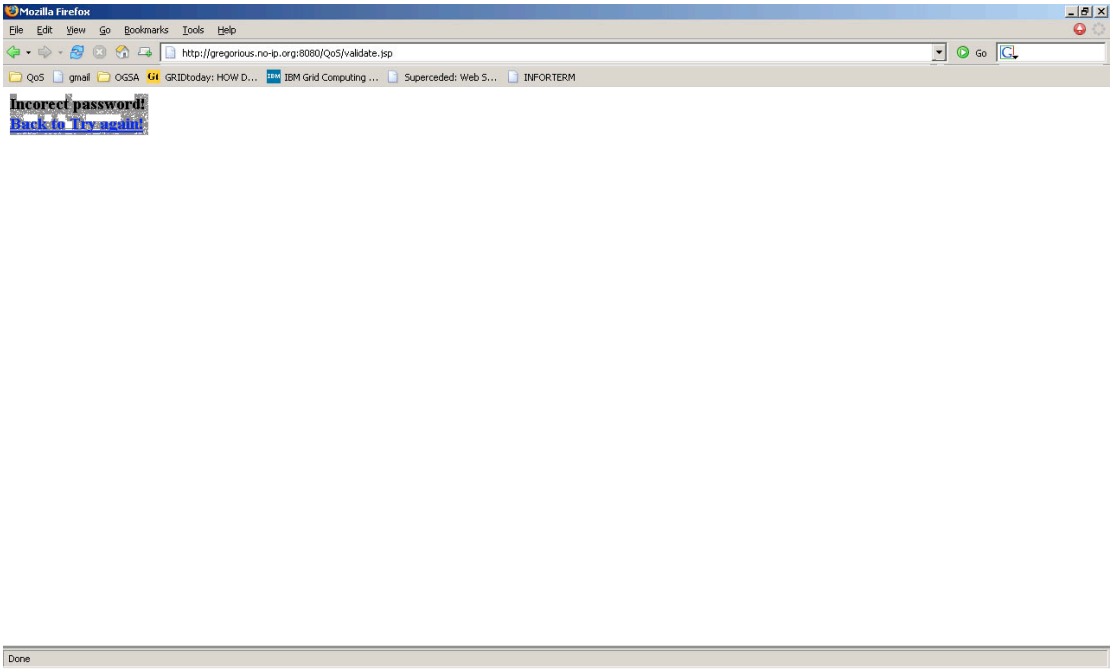
3.3 Διασυνδέσεις Χρήστη (User Interfaces)

Για τις λειτουργίες διαχείρισης και εκτέλεσης της υπηρεσίας που υλοποιήσαμε απαιτείται ένα περιβάλλον διασύνδεσης. Έτσι παρακάτω παρουσιάζουμε τις δυναμικές σελίδες που κατασκευάστηκαν για αυτό το σκοπό:

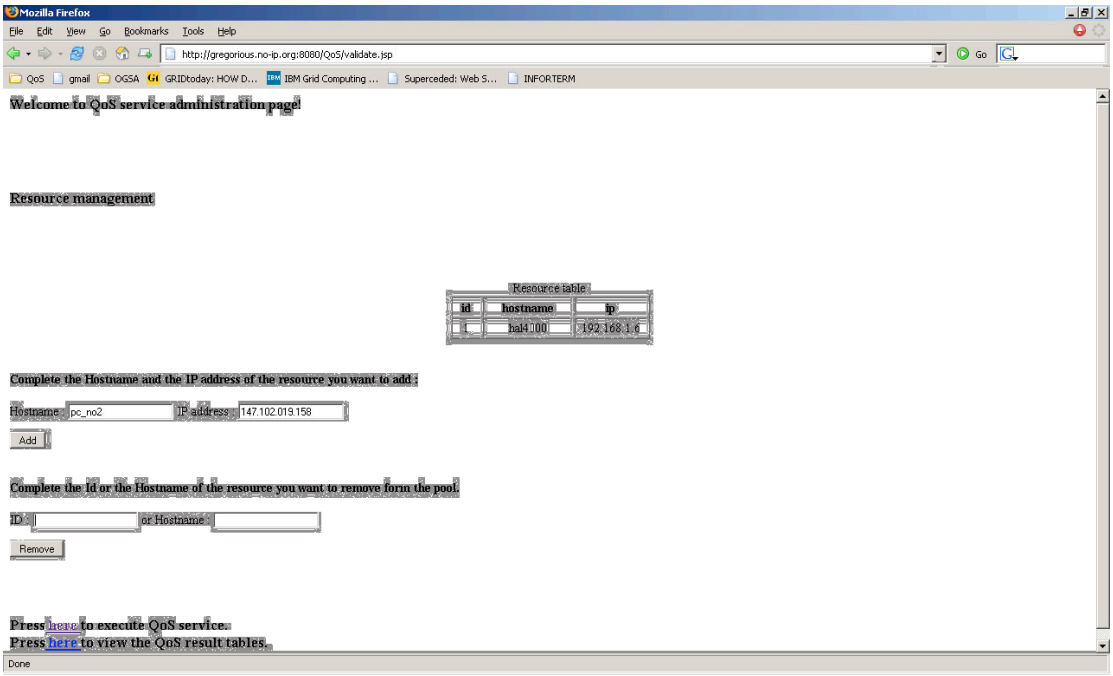


Εικόνα 13 : Login Page

Για την πρόσβαση στη σελίδα διαχείρισης απαιτείται πιστοποίηση χρήστη. Το όνομα χρήστη και κωδικός πρόσβασης είναι καταχωρημένα σε βάση δεδομένων που ελέγχεται από τον διαχειριστή της υπηρεσίας. Σε περίπτωση λάθους συμπλήρωσης στοιχείων κατάλληλα μηνύματα μας ενημερώνουν και μας καθοδηγούν.

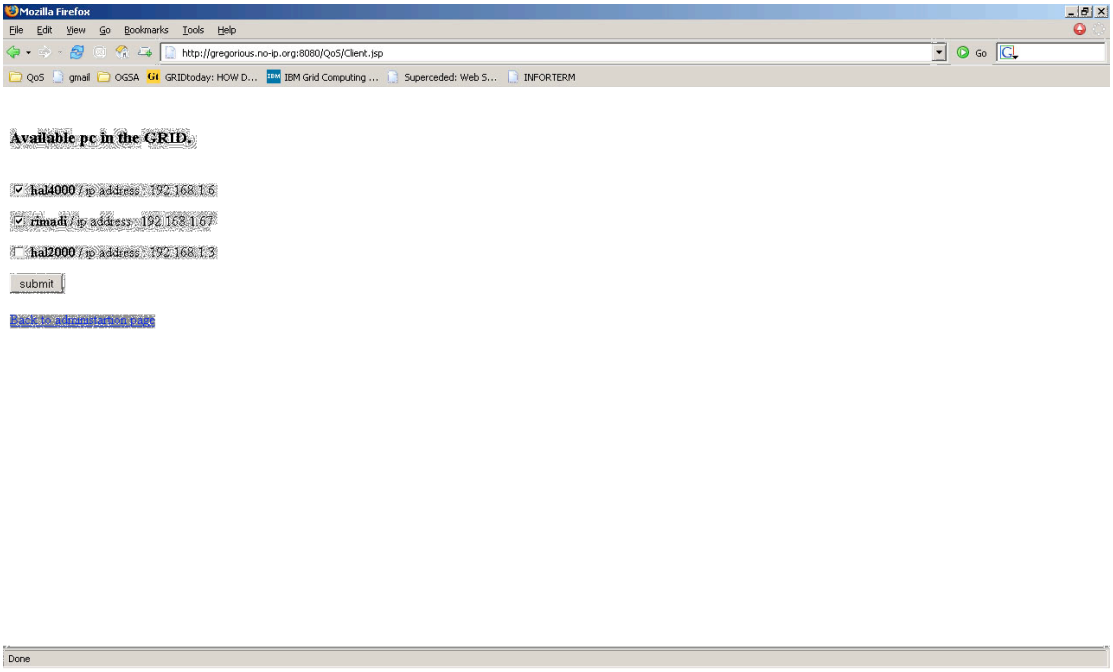


Εικόνα 14 : Login Error Page



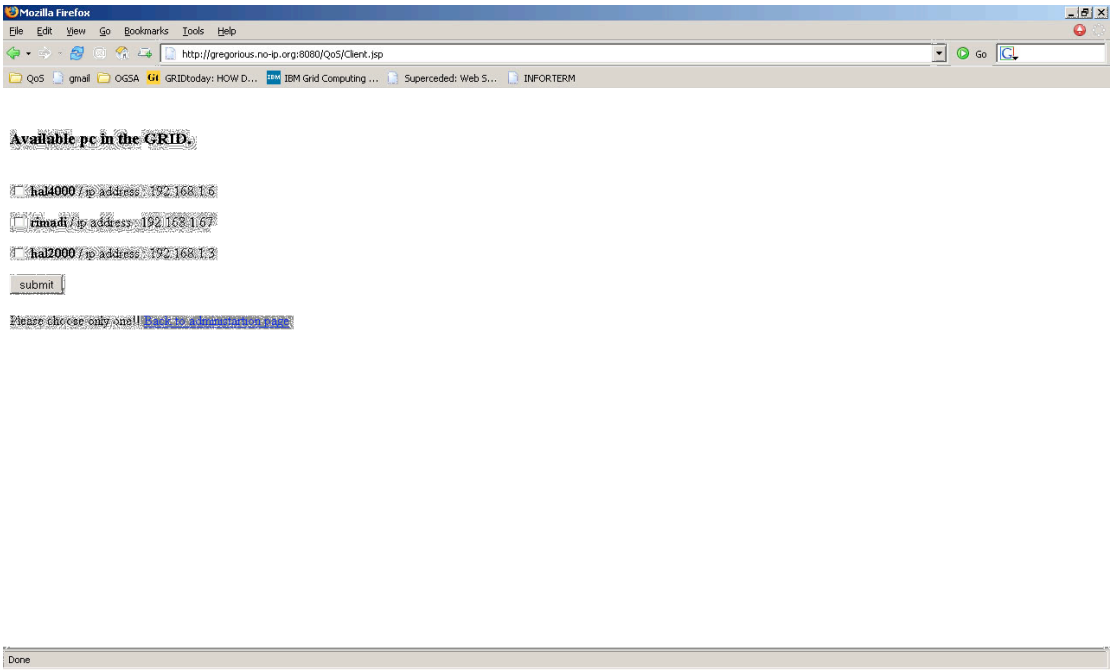
Εικόνα 15 : Administration Page

Αφού ο χρήστης συμπληρώσει σωστά το όνομα και κωδικό πρόσβασης, περνά στην σελίδα διαχείρισης. Σε αυτή τη σελίδα φαίνονται οι δυνατότητες προσθήκης/αφαίρεσης πόρων, προβολή αποτελεσμάτων και εκτέλεσης της QoS υπηρεσίας. Για να προσθέσει πόρους στο σύστημα ελέγχου ποιότητας υπηρεσίας χρειάζεται να συμπληρώσει hostname και ip-address, ενώ για αφαίρεση πόρων μπορούμε να προσδιορίσουμε βάση του κωδικού ή του ονόματος.



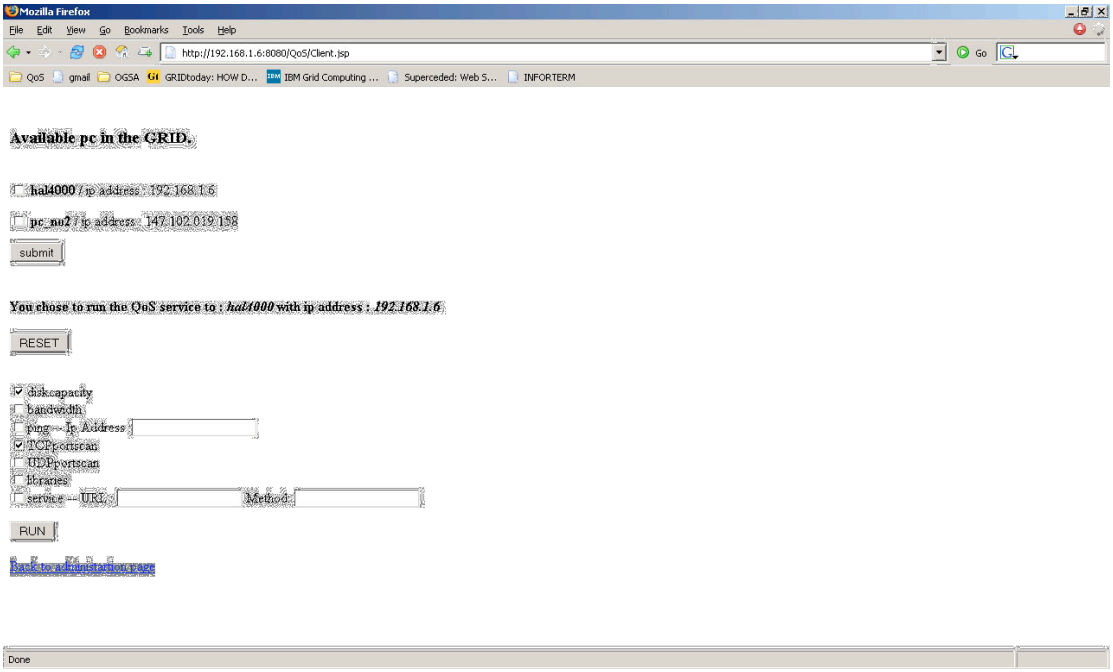
Εικόνα 16 : Execution Page

Η σελίδα εκτέλεσης της υπηρεσίας κατ' απαίτηση έχει αυτή τη μορφή. Διακρίνουμε τις δυνατότητες επιλογών πόρων και μεθόδων για την εκτέλεση.



Εικόνα 17 : Execution Error Message

Δίνεται η δυνατότητα επιλογής εκτέλεσης της υπηρεσίας μόνο σε ένα πόρο την φορά.



Εικόνα 18 : Execution Page

Αφού επιλέξουμε τον υπολογιστή που επιθυμούμε, επιλέγουμε τις παραμέτρους διαθεσιμότητας της υπηρεσίας, που θέλουμε να εκτελέσουμε.

4 *Υλοποίηση*

4.1 Πλατφόρμες και προγραμματιστικά εργαλεία

Η υλοποίηση του μηχανισμού παροχής πληροφορίας για την ποιότητα της υπηρεσίας (QoS Provision Module), όπως αναφέραμε θα αποτελείται από διάφορα μέρη χρησιμοποιώντας διαφορετικές τεχνολογίες με σκοπό την προσέγγιση όλων των προδιαγραφών που περιγράφηκαν. Έτσι σε μια γενική ανάλυση του μηχανισμού έχουμε :

1. availability_service : web service υλοποιημένο σε Java κώδικα και εγκατεστημένο σε κάθε υπολογιστικό πόρο του Grid δικτύου. Η υπηρεσία αυτή είναι η πλευρά του Provider, στο μοντέλο που παρουσιάστηκε νωρίτερα, και ο ρόλος της είναι να επιστρέφει σε όποιον την καλέσει πληροφορίες σχετικές με το σύστημα που είναι εγκατεστημένη.
2. service_client : πρόγραμμα υλοποιημένο σε Java κώδικα, εγκατεστημένο σε ένα μηχάνημα του πλέγματος υπολογιστών του Service Provider. Σύμφωνα με το μοντέλο που αναφέρθηκε, το πρόγραμμα αυτό αποτελεί τον Requester και ρόλος του είναι να καλεί την προηγούμενη υπηρεσία στο πλήθος των πόρων και να συγκεντρώνει τα δεδομένα που επιστρέφονται.
3. database : βάση δεδομένων υλοποιημένη σε Mysql. Σκοπός της είναι η αποθήκευση τόσο πληροφοριών του πλέγματος του Service Provider αλλά και των αποτελεσμάτων εκτέλεσης της υπηρεσίας αυτής. Αποτελεί το Repository σύμφωνα με το μοντέλο που παρουσιάστηκε.
4. admin_service : web service υλοποιημένο σε Java κώδικα και εγκατεστημένο στο ίδιο μηχάνημα με το service_client. Είναι βοηθητική υπηρεσία διαχείρισης των δεδομένων του πλέγματος του Service Provider. Παρέχει λειτουργίες προσθήκης και

αφαίρεσης πόρων στο Grid, προσθήκης και αφαίρεσης υπηρεσιών στο Grid κλπ. Κύριος στόχος της είναι η ελεγχόμενη πρόσβαση στη βάση δεδομένων από την οποία ενημερώνεται η υπηρεσία availability_service για την κατάσταση του πλέγματος του SP.

5. front_end : δυναμικές σελίδες διαδικτύου, υλοποιημένες είτε σε JSP είτε σε PHP, για την διαχείριση της βάσης, την προβολή των αποτελεσμάτων , την on_demand εκτέλεση της υπηρεσίας και την εφαρμογή ερωτημάτων (queries) στο πλήθος των αποτελεσμάτων.

4.1.1 JAVA

Η επιλογή της JAVA ως κώδικα ανάπτυξης του μηχανισμού έγινε πολύ προσεκτικά βάσει των μεγάλων δυνατοτήτων που αυτή προσφέρει. Σύμφωνα με την κατασκευάστρια της εταιρία την Sun, «Η Java είναι μια απλή, αντικειμενοστραφής, κατανεμημένη, ερμηνευόμενη, εύρωστη, αρχιτεκτονικά ουδέτερη, μεταφερόμενη, υψηλής απόδοσης, πολυνηματική, και ασφαλής γλώσσα».

Η JAVA, από την αρχή σχεδιάστηκε για εφαρμογές που θα λειτουργούσαν πάνω από το Internet. Έχει έτσι πολλά πλεονεκτήματα σε σχέση με την γρήγορη και ασφαλή διαδικτυακή επικοινωνία κάτι αναγκαίο στις υπηρεσίες του Grid. Εξαιτίας αυτού έχει καταστεί ως μία από τις βασικότερες γλώσσες προγραμματισμού για την ανάπτυξη εφαρμογών και υπηρεσιών διαδικτύου (web services).

Το κυριότερο βέβαια χαρακτηριστικό της JAVA είναι ότι είναι Platform Independent, μπορεί δηλαδή να λειτουργήσει σε κάθε υπολογιστικό σύστημα, είτε αυτό είναι προσωπικός υπολογιστής με Linux ή Windows, είτε ένα οποιοδήποτε ενσωματωμένο σύστημα (embedded system) το οποίο ενσωματώνει αυτή τη τεχνολογία. Αυτή η ιδιαιτερότητα της δίνει και το μεγάλο προβάδισμα αφού στη τεχνολογία Grid είναι χαρακτηριστική η συνεργασία πολλών διαφορετικών συστημάτων. Η έκδοση που θα χρησιμοποιήσουμε για την υλοποιήση μας είναι η Java 2 Platform, Standard Edition 5.0 Update 6 (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 MySql

Η επιλογή της MySql ως βάση δεδομένων οφείλεται κυρίως στην πολύ καλή της συνεργασίας με τη JAVA. Έτσι ο συνδυασμός αυτός μας δίνει μια ευέλικτη και χρηστική δομή με όλες τις ιδιότητες που μας ενδιαφέρουν. Τα βασικά χαρακτηριστικά που κάνουν αυτή τη βάση δεδομένων κατάλληλη είναι :

- ταχύτητα ανταπόκρισης
- ευκολία εκτέλεσης ενσωματωμένων σε γλώσσα προγραμματισμού ερωτημάτων
- λειτουργία της βάσης σε διάφορες πλατφόρμες (Linux, Windows)

Εναλλακτική λύση για την επιλογή του συστήματος βάσης δεδομένων, θα μπορούσε να είναι η επιλογή MS SQL Server ή ακόμα και κάποιας πιο πολύπλοκης βάσης όπως Oracle. Για την χρήση όμως που επιθυμούμε, τα χαρακτηριστικά της MySql μας υπερκαλύπτουν δίνοντας περισσότερο σημασία στην συνεργασία της βάσης με την επιλεχθείσα γλώσσα προγραμματισμού.

4.1.3 Γραφικό Περιβάλλον (User Interaface)

Είναι φυσικό για τη λειτουργία και τη διαχείριση της υπηρεσίας αυτής να απαιτείται η ύπαρξη ενός περιβάλλοντος ή μιας διεπαφής (interface) μέσω της οποίας θα γίνεται και η προβολή των αποτελεσμάτων. Η δικτυακή δομή όλου του Grid συστήματος και η ανάγκη για απομακρυσμένη πρόσβαση μας οδήγησε σε χρήση δυναμικών σελίδων διαδικτύου βασισμένες στις τεχνολογίες JSP και PHP . περιγραφή αυτών των σελίδων, της χρήσης τους και της ανάπτυξη τους θα γίνει αργότερα σε αντίστοιχο κεφάλαιο.

Αντί της χρήσης δυναμικών ιστοσελίδων θα μπορούσαμε να διαχειριζόμαστε την υπηρεσία από μία παραθυρική εφαρμογή. Αυτό αν και θα είχε κάποια λειτουργικά πλεονεκτήματα όπως ταχύτητα εκτέλεσης, μεταφερσιμότητα κλπ έχει το βασικό μειονέκτημα ότι δεν είναι προσβάσιμο από οποιονδήποτε στο δίκτυο. Μία από τις προδιαγραφές της υπηρεσίας ήταν δυνατή η πρόσβαση στα αποτελέσματα από κάθε ένας στο δίκτυο με την κατάλληλη πάντα ταυτοποίηση (authentication).

4.1.4 Εξυπηρετητής-Web Server

Ένας web server είναι ένα πρόγραμμα που δέχεται αιτήσεις σε HTTP (HTTP Requests) και απαντά παρέχοντας αρχεία HTML.

4.1.4.1 Tomcat

Ο Tomcat είναι ένα sevlet container ή servlet engine.Είναι ένα υπερσύνολο ενός web-server, που υποστηρίζει servlets και JSPs καθώς και στατικές σελίδες δεχόμενος requests στην πόρτα 8080. Είναι ένα εγχείρημα της Apache μέρος του jakarta project και διατίθεται δωρεάν στο διαδίκτυο (http://jacarta.apache.org/tomcat.). Θεωρείται από τους πλέον αξιόπιστους web servers και χρησιμοποιείται ευρέως στην ανάπτυξη δικτυακών υπηρεσιών. Στην παρούσα εφαρμογή χρησιμοποιήθηκε η έκδοση 5.1.

Τα Servlets είναι προγράμματα JAVA τα οποία επεκτείνουν τη λειτουργικότητα ενός web server, παράγοντας δυναμικό περιεχόμενο, αλληλεπιδρώντας με clients με ένα μοντέλο αίτησης-απάντησης. Είναι η τεχνολογία της JAVA για τη δημιουργία δυναμικών σελίδων στο διαδίκτυο. Τα Servlets έχουν τη δυνατότητα να εμπεριέχουν HTML tags στον JAVA κώδικα.

Οι JSPs είναι ουσιαστικά HTML σελίδες οι οποίες περιέχουν και JAVA κώδικα. Είναι δηλαδή το συμμετρικό μοντέλο ως προς τα Servlets και εξυπηρετούν τους ίδιους σκοπούς.

4.1.4.2 AXIS

Το Apache Axis είναι ένα πλαίσιο εργασίας για την ανάπτυξη δικτυακών υπηρεσιών (web service development framework), βασισμένο σε Java και XML και χρησιμοποιώντας SOAP κωδικοποίηση επικοινωνίας. Το Axis σε συνδυασμό με τον Tomcat Web Server, μας παρέχουν όλες τις απαραίτητες λειτουργίες για την ανάπτυξη και διαχείριση των υπηρεσιών. Το framework αυτό διατίθεται δωρεάν από την ιστοσελίδα του apache.org (http://ws.apache.org/axis/). Η έκδοση που θα χρησιμοποιήσουμε είναι η 1.4.

4.1.5 Bash Commands

Για την συλλογή των πληροφοριών που χρειαζόμαστε από τους πόρους του συστήματος, θα χρησιμοποιήσουμε διαφορές εντολές της κονσόλας εντολών σε λειτουργικό σύστημα Linux.

4.1.5.1 Η εντολή df

Εντολή της κονσόλας του Linux, που μας δίνει πληροφορίες για τα αποθηκευτικά μέσα του συστήματος. Μας ενημερώνει για το πόσο ελεύθερο χώρο έχουμε σε κάθε τοποθετημένο (mounted) δίσκο έχουμε στο σύστημά μας. Η εντολή αυτή είναι διαθέσιμη εξαρχής σε όλα τα Linux συστήματα.

4.1.5.2 Η εντολή nmap

Πρόγραμμα που εντοπίζει της ενεργοποιημένες θύρες του συστήματος και της εφαρμογές που τις χρησιμοποιούν. Λειτουργεί για TCP και UDP τύπους θυρών. Η έκδοση που χρησιμοποιήσαμε εμείς είναι η nmap 3.81

4.1.5.3 Η εντολή iperf

Εφαρμογή Client-Server για την μέτρηση του εύρους ζώνης μεταξύ δύο συστημάτων. Ο πελάτης (client) στέλνει πακέτα στον εξυπηρετητή (server) , που εκτελείται σε άλλο σύστημα και υπολογίζει τον χρόνο που χρειάζεται για την αποστολή. Λειτουργεί για μέτρηση την μέτρηση UDP και TCP πακέτων. Η έκδοση που χρησιμοποιήσαμε εμείς ήταν η 2.0.2 που προσφέρεται δωρεάν στο διαδίκτυο (http://dast.nlanr.net/Projects/Iperf).

4.2 Λεπτομέρειες υλοποίησης

Το σύστημα που σχεδιάζουμε μπορούμε να το χωρίσουμε σε επίπεδο υλοποίησης σε δύο μέρη:

- Server Side : η υλοποίηση της υπηρεσίας του QoS Provisioning, που θα είναι εγκατεστημένη σε καθέναν από τους πόρους που επιθυμούμε να εντάξουμε στο σύστημα του Provisioning.
- Client Side : η υλοποίηση της εφαρμογής που ο χρήστης θα χειρίζεται για να εκτελέσει κατ' απαίτηση την υπηρεσία, οι δυναμικές σελίδες προβολής των αποτελεσμάτων και η σελίδα διαχείρισης της υπηρεσίας.

4.2.1 Παραμετροποίηση Διαθεσιμότητας

Ο εντοπισμός των παραμέτρων που αποδεικνύουν την διαθεσιμότητα είναι μια δύσκολη και όχι αντικειμενική διαδικασία. Ειδικά όταν επιδιώκουμε οι παράμετροι αυτοί να μας δώσουν

ένδειξη της ποιότητας μιας υπηρεσίας. Για αυτό το λόγο μετά από δοκιμές και μελέτες καταλήξαμε στην επιλογή τον παρακάτω μεγεθών ως ένδειξη της διαθεσιμότητας των πόρων:

- Αποθηκευτικός χώρος του πόρου
- Εύρος ζώνης μεταξύ των πόρων και του χρήστη
- Διαθέσιμα πρωτόκολλα επικοινωνίας και ανοιχτές θύρες στους πόρους
- Διαθέσιμες βιβλιοθήκες ανά υπηρεσία εγκατεστημένη
- Κατάσταση υπηρεσίας (up/down)

4.2.2 Server Side

Τα απαραίτητα αρχεία/κλάσεις αυτής της υλοποίησης έχουν να κάνουν κυρίως με την υπηρεσία για την εξαγωγή των πληροφοριών από τους πόρους και την υπηρεσία διαχείρισης του μηχανισμού (προσθήκη-αφαίρεση πόρων). Έτσι τα βασικά αρχεία είναι :

- Availability.java : το αρχείο αποτελεί την υλοποίηση της υπηρεσίας για την εξαγωγή την παραμέτρων για τη διαθεσιμότητα. Αποτελεί ουσιαστικά τον πυρήνα του QoS Provisioning Module και χρησιμοποιείται ως υπηρεσία (availability_service) μέσω του tomcat
- Admin.java : είναι η υλοποίηση της υπηρεσίας για την διαχείριση του συστήματος που σχεδιάζουμε. Παρέχει προς το παρόν δυνατότητες προσθήκης και αφαίρεσης πόρων από τον μηχανισμό του QoS Provisioning.

Ολοκληρωμένα ο κώδικας όλων των αρχείων δίνεται στο Παράρτημα, στο τέλος του εγγράφου.

4.2.2.1 Αρχείο Availability.java

Παρακάτω παρουσιάζουμε τις μεθόδους που εμπεριέχονται στην υπηρεσία availability_service που δημιουργείται από την εγκατάσταση του κώδικα του αρχείου που εξετάζουμε στον tomcat server.

DiskCapacity : η μέθοδος αυτή μας επιστρέφει στοιχεία για τον ελεύθερο αποθηκευτικό χώρου που διαθέτει το παρόν σύστημα. Χρησιμοποιώντας την εντολή df με ορίσματα –klx tmpfs συγκεντρώνουμε πληροφορίες για τους σκληρούς δίσκους και την διαθεσιμότητά τους.

TCPportmap : με αυτή τη μέθοδο εκτελούμε έναν έλεγχο όλων των ελεύθερων θυρών του συστήματος μας και των εφαρμογών που χρησιμοποιούν αυτές τις θύρες. Με αυτό τον τρόπο μπορούμε να εντοπίσουμε τα πρωτόκολλα επικοινωνίας και άλλων δραστηριοτήτων που είναι

ενεργά. Την διαδικασία αυτήν την επιτυγχάνουμε χρησιμοποιώντας την εντολή *nmap* με ορίσματα *–sT localhost.* Σε αυτή τη μέθοδο μας επιστρέφονται δεδομένα σχετικά με τις TCP θύρες.

UDPportmap : Αντίστοιχη διαδικασία με την TCPportmap αλλά για UDP θύρες. Η σύνταξη της εντολής είναι *nmap –sU localhost*

Ping : Η μέθοδος αυτή εκτελεί ping request στην IP διεύθυνση του χρήστη και υπολογίζει το εύρος ζώνης μεταξύ τους. Το μέγεθος του πακέτου και το πλήθος ρυθμίζονται δυναμικά. Το αποτέλεσμα υπολογίζεται βάση του χρόνου της εκτέλεσης και του όγκου των πακέτων που μεταφέρονται.

BandWidth : Με τη μέθοδο αυτή υπολογίζουμε ξανά το εύρος ζώνης, μεταξύ όμως το συστήματος και του gateway του Grid συστήματος μας. Ο στόχος αυτής της μέτρησης είναι να πάρουμε μια ένδειξη εύρους ζώνης σε περίπτωση που η προηγούμενη μέτρηση (ping request) αποτύχει. Για την εκτέλεση αυτή χρησιμοποιούμε το πρόγραμμα *iperf*..

servive_test : Με αυτή τη μέθοδο ελέγχουμε την κατάσταση μιας εγκατεστημένης, στο πόρο που εξετάζουμε, υπηρεσίας. Ως δεδομένα εισόδου έχει τον URI της υπηρεσίας και το όνομα της μεθόδου που επιθυμεί να καλέσει.

listpath : Η μέθοδος αυτή μας επιστρέφει τα αρχεία που εντοπίζει μέσα σε ένα κατάλογο του συστήματος. Χρησιμοποιείται για τον εντοπισμό των βιβλιοθηκών και άλλων απαραίτητων αρχείων των υπηρεσιών που είναι εγκατεστημένες στο σύστημα.

4.2.2.2 Αρχείο Admin.java

Η δυνατότητες προσθήκης και αφαίρεσης πόρων στο σύστημα του QoS Provision γίνεται με την χρήση της υπηρεσίας availability_admin που υλοποιείται με το αρχείο που εξετάζουμε. Οι μέθοδοι *add_pc* και *delete_pc* που περιέχονται στο αρχείο καταγράφουν ή διαγράφουν στη βάση δεδομένων τις πληροφορίες για του πόρους.

4.2.3 Client Side

4.2.3.1 Αρχείο Client.java

Η κλάση αυτή είναι η κυριότερη για την πλευρά του χρήστη. Αναφέρεται στην προγραμματισμένη λειτουργία εκτέλεσης, αφού εμπεριέχει χρονοπρογραμματιστή (time scheduler). Όταν η κλάση αυτή εκτελεστεί, για κάθε πόρο που συμμετέχει στην υπηρεσία QoS Provisioning, θα δημιουργήσει ένα νήμα (thread) μέσω της μεθόδου *timer*. Το κάθε νήμα

από αυτά θα εκτελεί την κλάση *ExecTask* που εμπεριέχεται στο αρχείο που αναφερόμαστε, κάθε ένα κβάντο χρόνου (στην περίπτωση μας 10sec). Στο σημείο αυτό ανάλογα με τις σημαίες (flags) στις παραμέτρους της διαθεσιμότητας θα κληθούν και οι αντίστοιχες μέθοδοι για την εκτέλεση της QoS Provisioning υπηρεσίας στην κάθε πόρο του συστήματος. Τέλος η μέθοδος scheduler που καλείται στο τέλος κάθε κύκλου ρολογιού του χρονοπρογραμματιστή, επαναπροσδιορίζει τις τιμές στις σημαίες για την επόμενη εκτέλεση, σύμφωνα με την λογική που έχουμε επιλέξει (smart scheduling).

4.2.3.2 Αρχείο Methods.java

Η κλάση αυτή περιέχει μεθόδους για τις κλήσεις των παραμέτρων της διαθεσιμότητας από την υπηρεσία που είναι εγκατεστημένη στον server (availability_service). Η κάθε μέθοδος στο αρχείο αυτό καλεί την αντίστοιχή της στον server, με τα κατάλληλα ορίσματα που δέχθηκε ως είσοδο. Τα αποτελέσματα που επιστρέφονται σε αυτήν αποθηκεύονται στους κατάλληλους πίνακες της βάσης μετά από την απαραίτητη επεξεργασία.

4.2.3.3 Σελίδα Validate.jsp

Αποτελεί τη δυναμική σελίδα που εμφανίζεται μετά την επαλήθευση του κωδικού χρήσης. Απευθύνεται στον διαχειριστή της υπηρεσίας που υλοποιούμε, αφού προσφέρει λειτουργίες προσθήκης-αφαίρεσης πόρων στο σύστημα του QoS Provisioning. Επιπλέον παρέχει συνδέσμους για κατ' απαίτηση εκτέλεση και προβολή παλαιότερων αποτελεσμάτων.

4.2.3.4 Σελίδα Client.jsp

Αυτή η δυναμική σελίδα εκφράζει την υπηρεσία εκτέλεσης κατ' απαίτηση. Ο χρήστης επιλέγει έναν από τους διαθέσιμους πόρους που είναι εγκατεστημένοι και αφού διαλέξει και τις παραμέτρους της διαθεσιμότητας που επιθυμεί, εκτελεί την υπηρεσία στον πόρο αυτό. Τα αποτελέσματα της εκτέλεσης παρουσιάζονται στην δυναμική σελίδα *results.jsp* που εμφανίζεται αμέσως μετά.

4.2.4 Βάση Δεδομένων

4.2.4.1 Αρχείο Database.java

Σε αυτό το αρχείο υπάρχουν όλες οι βασικές μέθοδοι για την σύνδεση, αποσύνδεση και εφαρμογή ερωτημάτων στην βάση δεδομένων.

4.2.4.2 Πίνακες Βάσης Δεδομένων

Παρακάτω παρουσιάζονται οι πίνακες που ορίστηκαν στη βάση δεδομένων για την αποθήκευση των πληροφοριών και για την λειτουργικότητα του συστήματος.

Diskcapacity

Εμπεριέχει όλες τις πληροφορίες από τα αποτελέσματα της εκτέλεσης της μεθόδου για τον διαθέσιμο αποθηκευτικό χώρο. Το πεδίο kbfee περιέχει τον ελεύθερο χώρο σε Kbytes και το mounted_on τη συσκευή που είναι εγκατεστημένος ο χώρος αυτός.

Field	Type	Null	Key	Default	Extra
id	smallint (5) unsigned		PRI	NULL	auto_increment
timestamp	timestamp (14)	YES		NULL	
hostname	char (60)	YES		NULL	
ip	char (20)	YES		NULL	
kbfree	char (30)	YES		NULL	
mounted_on	char (25)	YES		NULL	

Bandwidth

Καταγράφει τα δεδομένα από την εκτέλεση της αντίστοιχης μεθόδου. Το pc_id είναι ο κωδικός του πόρου από τον οποίο θα γίνει η μέτρηση, target_ip ο πόρος με τον οποίο θα επικοινωνήσει έτσι ώστε να μετρήσει το εύρος ζώνης, και bandwidth η μέτρηση που έκανε.

Field	Type	Null	Key	Default	Extra
id	smallint (5) unsigned		PRI	NULL	auto_increment
timestamp	timestamp (14)	YES		NULL	
pc_id	smallint (5) unsigned			0	
target_ip	char (20)	YES		NULL	
bandwidth	char (15)	YES		NULL	

pc_pool

Περιέχει τα στοιχεία των πόρων που εντάσσονται στο σύστημα του QoS Provisioning

Field	Type	Null	Key	Default	Extra
id	smallint (5) unsigned		PRI	NULL	auto_increment
hostname	char (25)	YES		NULL	
ip_address	char (20)	YES		NULL	

ping_connection

Κκαταγράφει τα δεδομένα από την εκτέλεση της μεθόδου ping της υπηρεσίας availability_service. Τα πεδία pc_id και target_ip λειτουργούν όπως και στο bandwidth απλά εδώ η μέτρηση γίνεται βάση της χρήσης της ping εντολής.

Field	Type	Null	Key	Default	Extra
id	smallint (5) unsigned		PRI	NULL	auto_increment
timestamp	timestamp (14)	YES		NULL	
pc_id	smallint (5) unsigned			0	
target_ip	char (20)	YES		NULL	
packets_sent	smallint (6)	YES		NULL	
packets_received	smallint (6)	YES		NULL	
avg_time_ms	decimal (6,5)	YES		NULL	
bandwidth_used_Mbps	decimal (6,5)	YES		NULL	

port_map

Περιέχει τα δεδομένα από την διερεύνηση των ενεργών θυρών του συστήματος. Το πεδίο ip περιέχει την διεύθυνση του πόρου, port την θύρα που εξετάζουμε, protocol το πρωτόκολλο επικοινωνίας (TCP-UDP) και service την υπηρεσία που χρησιμοποιεί την θύρα αυτή.

Field	Type	Null	Key	Default	Extra

id	smallint (5) unsigned		PRI	NULL	auto_increment
timestamp	timestamp (14)	YES		NULL	
hostname	char (60)	YES		NULL	
ip	char (20)	YES		NULL	
port	char (10)	YES		NULL	
protocol	char (5)	YES		NULL	
service	char (30)	YES		NULL	

Scheduler

Βοηθητικός πίνακα για την προγραμματισμένη εκτέλεση. Τα πεδία diskcapacity,port_map,list_path,bandwidth περιέχουν μεταβλητές σημαίες (flags) για την εκτέλεση του χρονο-προγραμματισμού.

Field	Type	Null	Key	Default	Extra
pc_id	smallint (6)		PRI	0	
diskcapacity	decimal (1,0)	YES		NULL	
d_timestamp	timestamp (14)	YES		NULL	
port_map	decimal (1,0)	YES		NULL	
p_timestamp	timestamp (14)	YES		0	
list_path	decimal (1,0)	YES		NULL	
l_timestamp	timestamp (14)	YES		0	
bandwidth	decimal (1,0)	YES		NULL	
b_timestamp	timestamp (14)	YES		0	

services_pool

Καταγραφή των διαθέσιμων υπηρεσιών ανά εγκατεστημένο πόρο, μαζί με τους καταλόγους των απαραίτητων βιβλιοθηκών. Στο πεδίο lib_paths εγγράφουμε την διεύθυνση του καταλόγου που η υπηρεσία με servicename και wsdl, που δίνεται, χρησιμοποιεί ως βιβλιοθήκη.

Field	Type	Null	Key	Default	Extra

id	smallint (5) unsigned		PRI	NULL	auto_increment
servicename	char (25)	YES		NULL	
installed_pc_id	smallint (5) unsigned			0	
wsdl	char (50)	YES		NULL	
lib_paths	char (50)	YES		NULL	

service_libraries Καταγραφή των αρχείων που εντοπίστηκαν στους καταλόγους των αντίστοιχων υπηρεσιών

Field	Type	Null	Key	Default	Extra
id	smallint (5) unsigned		PRI	NULL	auto_increment
timestamp	timestamp (14)	YES		NULL	
service_id	smallint (5) unsigned			0	
pc_id	smallint (5) unsigned			0	
filename	char (50)	YES		NULL	
path	char (50)	YES		NULL	

Οι πίνακες pc_pool, services_pool και scheduler πρέπει να πάρουν τιμές από τον διαχειριστή του συστήματος πριν την εκτέλεση της υπηρεσίας. Εκτέλεση κατ' απαίτηση μπορεί να γίνει χωρίς χρήση του τελευταίου πίνακα. Οι υπόλοιποι πίνακες χρησιμοποιούνται για αποθήκευση αποτελεσμάτων των εκτελέσεων.

5 Έλεγχος

5.1 Οδηγός Εγκατάστασης

5.1.1 Εγκατάσταση του Tomcat και AXIS

Το αρχείο για την εγκατάσταση του Tomcat διατίθεται δωρεάν στο διαδίκτυο στη διεύθυνση (http://jacarta.apache.org/tomcat). Στην παρούσα εφαρμογή χρησιμοποιήθηκε η έκδοση 5.5.1.5.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. Σε Windows περιβάλλον, απλά εκτελούμε το αρχείο που κατεβάσαμε από την ιστοσελίδα και ακολουθούμε τις οδηγίες.

Το AXIS προσφέρεται στο διαδίκτυο δωρεάν (http://ws.apache.org/axis/). Για την εγκατάσταση του αρκεί να αντιγράψουμε τον κατάλογο webapps/axis, 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/axis/, και να επιβεβαιώσουμε την σωστή εγκατάσταση του προγράμματος αυτού. Η έκδοση που χρησιμοποιήσαμε εμείς ήταν η 1.4. Για την ορθή λειτουργία του AXIS απαιτείται η ρύθμιση κάποιων μεταβλητών του συστήματος (system variables). Παρακάτω εξηγούμε ακριβώς την διαδικασία, μαζί με όλες τις υπόλοιπες απαιτήσεις.

5.1.2 Ρύθμιση μεταβλητών συστήματος και βιβλιοθηκών

Εκτός από τα προηγούμενα που αναφέρθηκαν είναι απαραίτητη και η ύπαρξη Java Runtime Environment στο μηχάνημα που αναφερόμαστε. Η προτεινομένη έκδοση είναι η 1.5.0_06 που προσφέρεται δωρεάν από την ιστοσελίδα της SUN (http://java.sun.com/javase/downloads/index.jsp). Επίσης για την ολοκληρωμένη λειτουργία της Java στην ανάπτυξη Web Services απαιτείται η εγκατάσταση των Xerces API, Javamail API και Jaf API. Αυτό σημαίνει την αντιγραφή των αντίστοιχων αρχείων (xerces.jar, mail.jar, activation.jar) στην βιβλιοθήκη αρχείων του tomcat. Στην εφαρμογή μας χρησιμοποιήσαμε τις εκδόσεις Xerces 1.4.4, Javamail 1.4 και Jaf 1.1.

Για την ρύθμιση των μεταβλητών του συστήματος χρειάζεται να δημιουργήσουμε καινούριες μεταβλητές . Θεωρώντας ότι ο κατάλογος που είναι εγκατεστημένο το AXIS είναι ο usr/axis τοποθετούμε τις εντολές αυτές στο αρχείο ets/bash.bashrc :

```
set AXIS_HOME=/usr/axis
set AXIS_LIB=$AXIS_HOME/lib
set AXISCLASSPATH=$AXIS_LIB/axis.jar:$AXIS_LIB/commons-discovery.jar:
  $AXIS_LIB/commons-
logging.jar:$AXIS_LIB/jaxrpc.jar:$AXIS_LIB/saaj.jar:
  $AXIS_LIB/log4j-1.2.8.jar:$AXIS_LIB/xml-
apis.jar:$AXIS_LIB/xercesImpl.jar:$AXIS_LIB /mail.jar:$AXIS_LIB
/activation.jar
export AXIS_HOME; export AXIS_LIB;
export AXISCLASSPATH
```

Εκτός από τις παραπάνω μεταβλητές συστήματος πρέπει να ορίσουμε και τις μεταβλητές για την Java, JAVA_HOME και JRE_HOME που θα περιέχουν το κατάλογο που βρίσκεται εγκατεστημένη η Java και Jre αντίστοιχα.

5.1.3 Εγκατάσταση Βάσης Δεδομένων

Ακολουθώντας τις οδηγίες που υπάρχουν στην ιστοσελίδα της MySQL κατεβάζουμε την διανομή που διατίθεται (http://dev.mysql.com) ανάλογα με το λειτουργικό που χρησιμοποιούμε στο μηχάνημα που θα εγκαταστήσουμε τη βάση. Η προτεινομένη έκδοση είναι η 5.0 και η εγκατάσταση της εξαρτάται από το λειτουργικό που χρησιμοποιούμε. Για λεπτομέρειες ακολουθήστε οδηγίες που υπάρχουν στην ιστοσελίδα.

Για την δημιουργία των πινάκων που χρειάζονται στην λειτουργία της υπηρεσίας μας, θα χρησιμοποιήσουμε τον κώδικα που υπάρχει στο αρχείο sqlscripts.txt που συνοδεύεται σε cd με το έγγραφο αυτό. Εκτελώντας διαδοχικά τα scripts που υπάρχουν στο αρχείο, δημιουργούμε την βάση δεδομένων και όλους τους απαραίτητους πίνακες. Απαραίτητοι για λειτουργία του κώδικα του module είναι οι οδηγοί της Java για την σύνδεση με την βάση

δεδομένων. Στο συνοδευτικό cd, μέσα στον κατάλογο mysql drivers υπάρχει ένα jar αρχείο το οποίο πρέπει να τοποθετήσουμε στην βιβλιοθήκη της java (java lib directory) και στον WEB-INF/lib κατάλογο της εφαρμογής που θα εγκαταστήσουμε στον tomcat.

5.1.4 Εγκατάσταση QoS Provisioning Module

Για την εγκατάσταση της υπηρεσίας μας χρειάζονται δύο κινήσεις. Αρχικά πρέπει να κάνουμε deploy τις δικτυακές υπηρεσίες (web services) στον tomcat server και στη συνέχεια να εγκαταστήσουμε την jsp εφαρμογή για τις δυναμικές σελίδες. Αρχικά, ενώ ο tomcat server "τρέχει", χρειάζεται να εκτελέσουμε την παρακάτω εντολή ενώ βρισκόμαστε στους καταλόγους availability_service και admin_service, για να εγκαταστήσουμε το availability_service και την admin_service αντίστοιχα:

```
% java org.apache.axis.client.AdminClient deploy.wsdd
```

Για την επιτυχή εγκατάσταση των δύο υπηρεσιών πρέπει η εκτέλεση της εντολής να έχει την παρακάτω έξοδο:

```
<Admin>Done processing</Admin>
```

Για το δεύτερο μέρος της εγκατάστασης αρκεί να αντιγράψουμε τον κατάλογο QoS από το συνοδευτικό cd και να τον τοποθετήσουμε στον κατάλογο εφαρμογών του tomcat (webapps directory) και να κάνουμε restart τον server. Μετά από τις παραπάνω κινήσεις είμαστε έτοιμοι να εκτελέσουμε το QoS Provisioning Module.

5.1.5 Εκτέλεση QoS υπηρεσίας

Έχοντας ολοκληρώσει τις παραπάνω προετοιμασίες, είμαστε έτοιμοι να δοκιμάσουμε την υπηρεσία. Για την κατ' απαίτηση εκτέλεση, θα πρέπει να επικοινωνήσουμε με τον διαχειριστή της υπηρεσίας για να μας προμηθεύσει με κωδικό πρόσβασης και στη συνέχεια να κάνουμε login στην σελίδα *http://web_server:8080/QoS*. Όπου web_server είναι η ip address ή το domain name του μηχανήματος στο οποίο έχει εγκατασταθεί ο tomcat server για την διαχείριση της υπηρεσίας. Στη συνέχεια επιλέγουμε τον σύνδεσμο execute QoS service και μεταφερόμαστε στην σελίδα εκτέλεσης. Εδώ επιλέγουμε τον πόρο που θα εκτελέσουμε την QoS Provisioning Service και πατάμε submit. Μπορούμε να επιλέξουμε μόνο έναν πόρο από την λίστα κάθε φορά. Εν συνεχεία επιλέγουμε τις παραμέτρους της διαθεσιμότητας που θέλουμε να εκτελέσουμε. Μπορούμε να τις επιλέξουμε όλες είτε να τις εκτελέσουμε μια-μια. Οι παράμετροι είναι :

- Diskcapacity : έλεγχος ελεύθερου αποθηκευτικού χώρου

- Bandwidth : μέτρηση του εύρους ζώνης από το σημείο που βρίσκεται η υπηρεσία μέχρι το gateway του υποδικτύου που βρισκόμαστε.
- Ping : μέτρηση του εύρους ζώνης από το σημείο που βρίσκεται η υπηρεσία, μέχρι τον χρήστη που την κάλεσε.
- TCPportscan : έλεγχος θυρών επικοινωνίας και προγραμμάτων-πρωτοκόλλων επικοινωνίας TCP πακέτων.
- UDPportscan έλεγχος θυρών επικοινωνίας και προγραμμάτων-πρωτοκόλλων επικοινωνίας UDP πακέτων.
- Libraries : έλεγχος βιβλιοθηκών και άλλων αρχείων που σχετίζονται με εγκατεστημένες υπηρεσίες.
- Service : έλεγχος κατάστασης υπηρεσίας.

Αφού συμπληρώσουμε και επιλέξουμε ότι επιθυμούμε πατάμε RUN και συνδεόμαστε στην σελίδα αποτελεσμάτων. Εδώ αφού μας παρουσιαστούν οι επιλογές που κάναμε προηγουμένως ως επιβεβαίωση, θα δούμε και τα αποτελέσματα των εκτελέσεων.

Μπορούμε να δοκιμάσουμε ξανά κάποια άλλη παράμετρο είτε να μεταφερθούμε στην αρχική σελίδα χρησιμοποιώντας του αντίστοιχους συνδέσμους.

Για την προγραμματισμένη εκτέλεση χρησιμοποιούμε την κονσόλα του τερματικού διαχείρισης, όπου είναι εγκατεστημένος και ο tomcat server. Ενώ βρισκόμαστε στον κατάλογο που υπήρχε στο συνοδευτικό cd, με το όνομα scheduled_run , εκτελούμε την εντολή :

Java Client arg1 arg2 arg3 arg4 arg5 arg6

Όπου argx είναι η χρονική συχνότητα που θέλουμε να εκτελεστούν οι παράμετροι για το diskcapacity, bandwidth, ping, TCPportscan, UDPportscan και libraries αντίστοιχα , σε mseconds. Αφού πατήσουμε ENTER η εκτέλεση ξεκινά, και τα αποτελέσματα τυπώνονται στην οθόνη. Η σύγχρονη εκτέλεση των διάφορων παραμέτρων στο σύνολο των υπολογιστών μπορεί να προκαλέσει μια σύγχυση με την εμφάνιση των αποτελεσμάτων αλλά κατάλληλα μηνύματα βοηθούν σε αυτό. Η εκτέλεση δεν θα σταματήσει μέχρι να την διακόψουμε εμείς. Πατάμε λοιπόν Ctrl+C όποτε επιθυμούμε να διακόψουμε. Τα αποτελέσματα έχουν ήδη καταγραφεί στη βάση, έτσι από τη σελίδα διαχείρισης μπορούμε να συνδεθούμε με τις σελίδες προβολής αποτελεσμάτων όπου παρουσιάζονται συνολικά όλα τα αποτελέσματα.

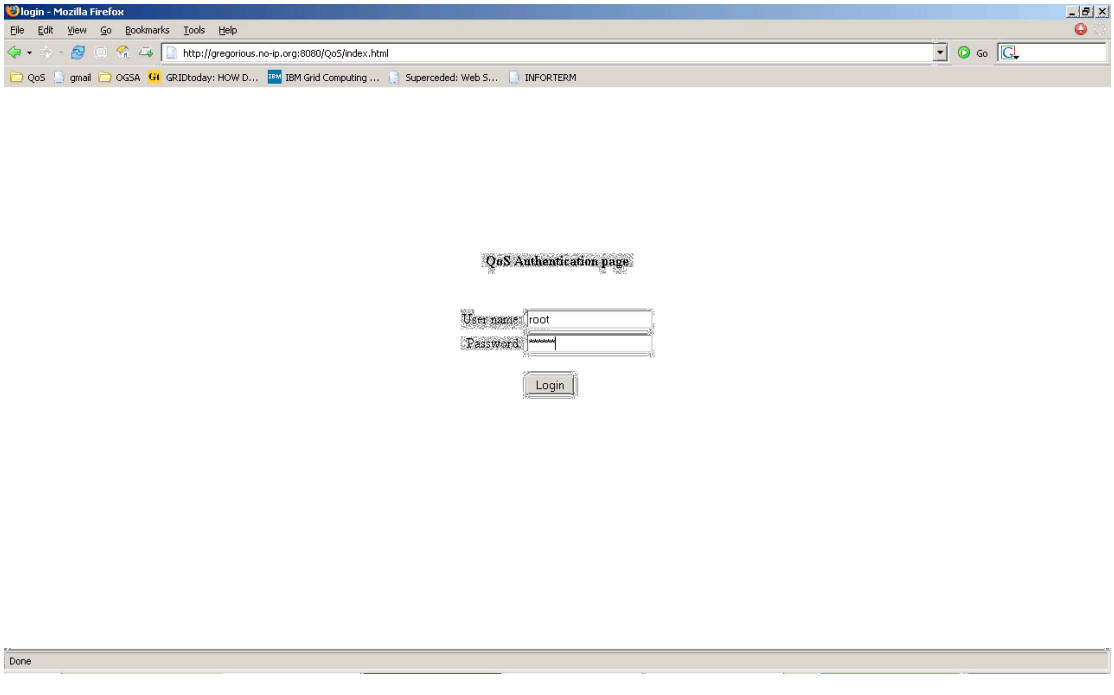
5.2 Αναλυτική παρουσίαση ελέγχου

5.2.1 Τα αποτελέσματα της χρήσης του συστήματος

Στην παράγραφο αυτή θα παρουσιάσουμε την εκτέλεση ενός σεναρίου χρήσης του συστήματος, από κάποιον που έχει πρόσβαση και στη σελίδα διαχείρισης για να καλύψουμε όσο το δυνατόν περισσότερες λειτουργίες. Στη συνέχεια παρουσιάζονται και σχολιάζονται τα αποτελέσματα.

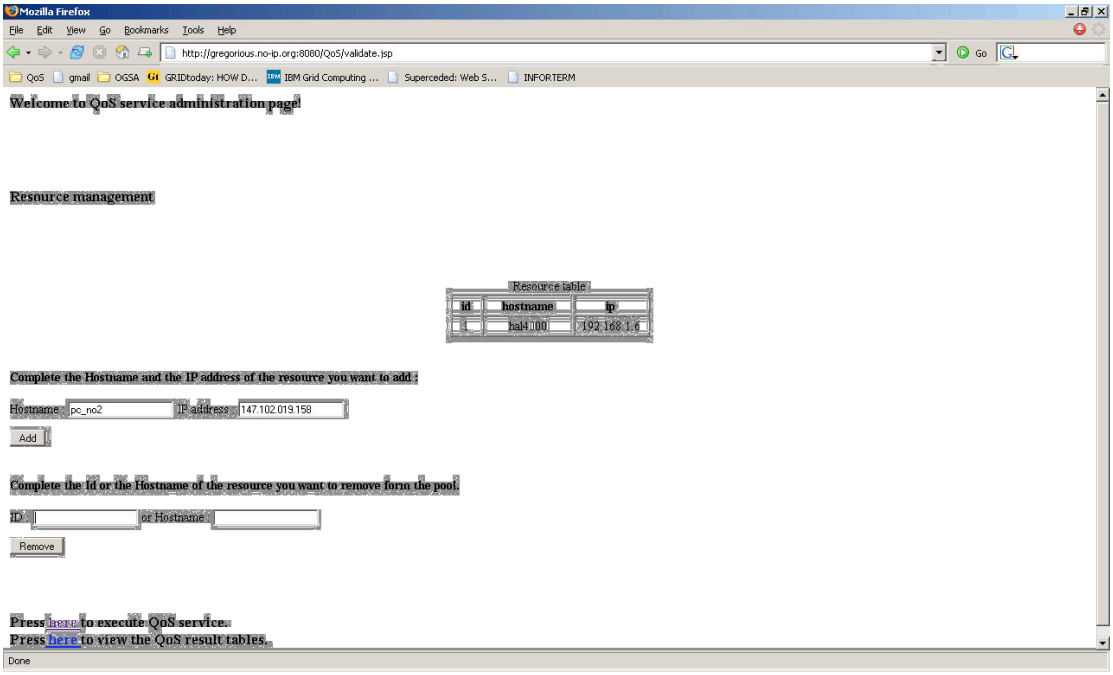
5.2.1.1 Εκτέλεση Σεναρίου Χρήσης

Αρχίζουμε την διαδικασία με την παρακάτω εισαγωγική οθόνη:

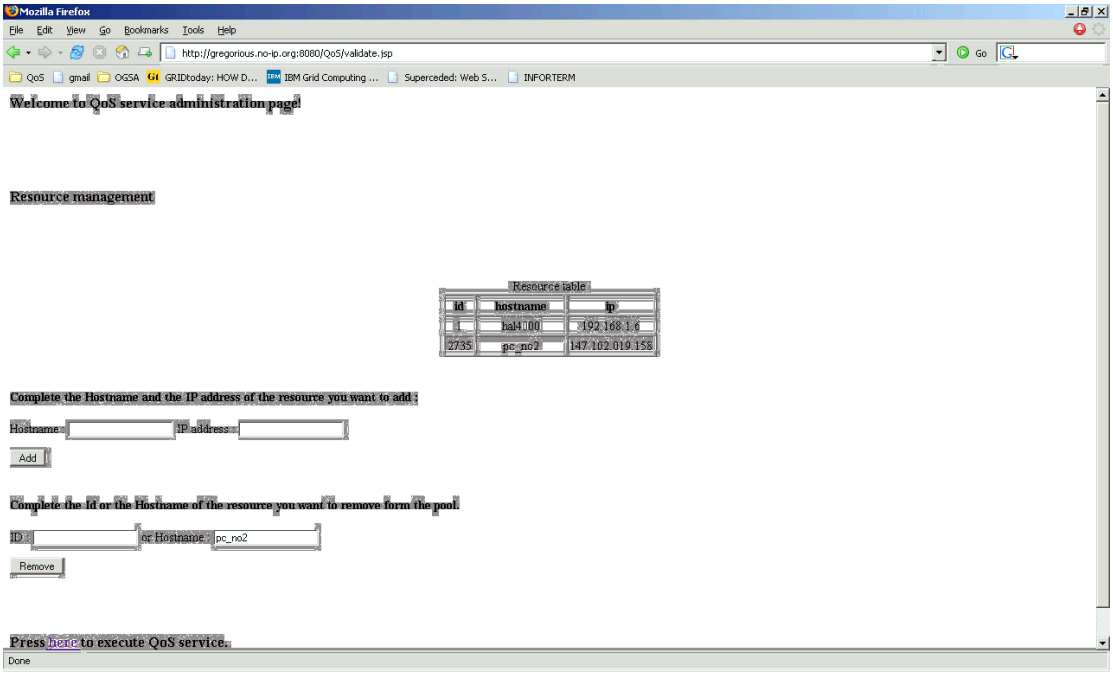


Εικόνα 19 : Login Page

Μετά την επιτυχημένη εισαγωγή προχωράμε στην επόμενη σελίδα που μας παρέχει δυνατότητες προσθήκης πόρων και αφαίρεσης. Στις δύο παρακάτω οθόνες παρουσιάζεται το περιβάλλον διαχείρισης πόρων και δοκιμάζονται οι λειτουργίες. Για την πρόσθεση πόρου, αρκεί να συμπληρώσουμε το όνομά του (hostname) και την διεύθυνσή του (ip address). Πατώντας το ADD προστίθεται στην λίστα που παρουσιάζεται από πάνω. Αντίστοιχα για την αφαίρεση, συμπληρώνουμε στα πεδία είτε τον ID κωδικό του κάθε πόρου είτε το hostname του και πατάμε REMOVE.

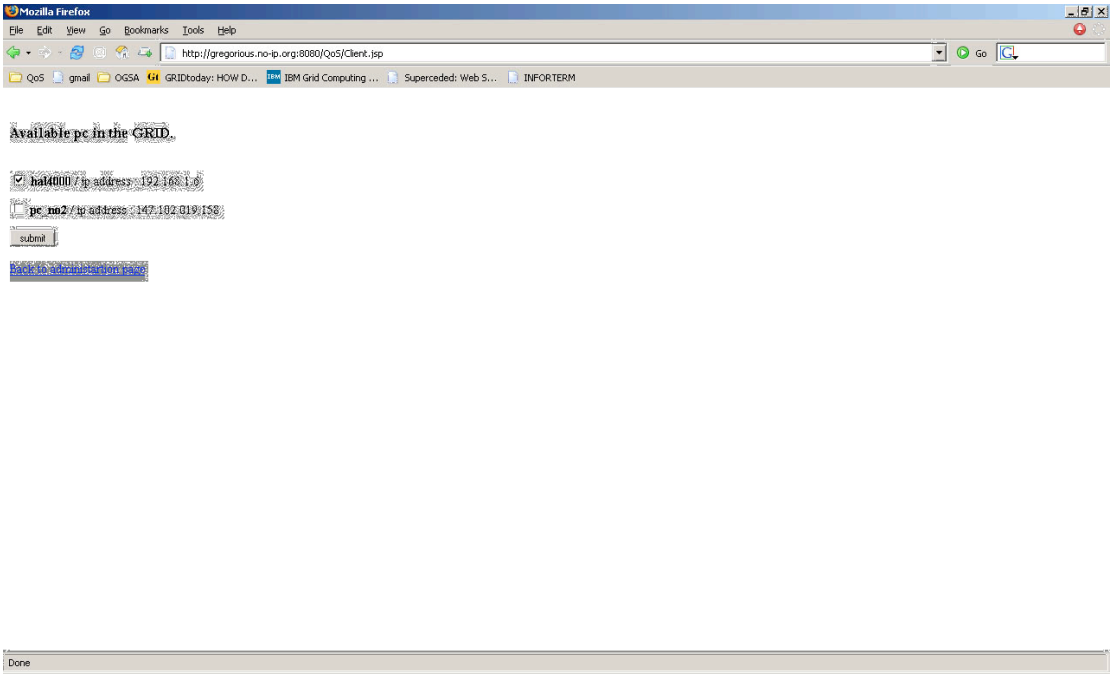


Εικόνα 20 : Adding Resource



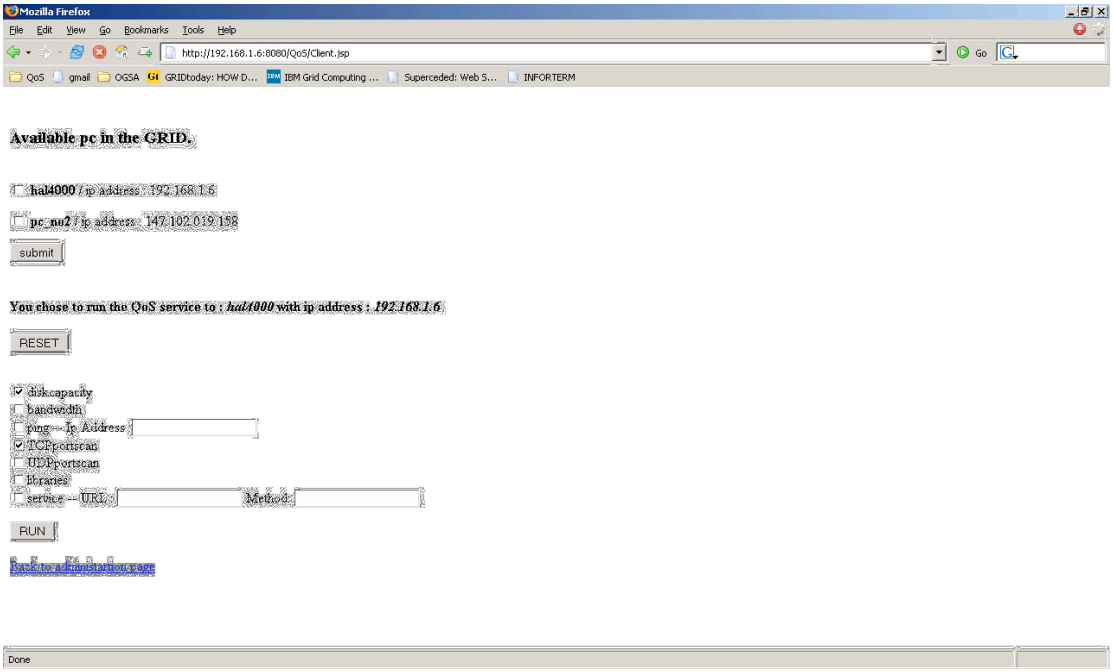
Εικόνα 21 : Removing Resource

Από την σελίδα διαχείρισης μπορούμε να συνδεθούμε με την σελίδα εκτέλεσης της υπηρεσίας πιέζοντας τον αντίστοιχο σύνδεσμο (execute QoS Service). Η σελίδα που θα ανοίξει θα μοιάζει με αυτή:



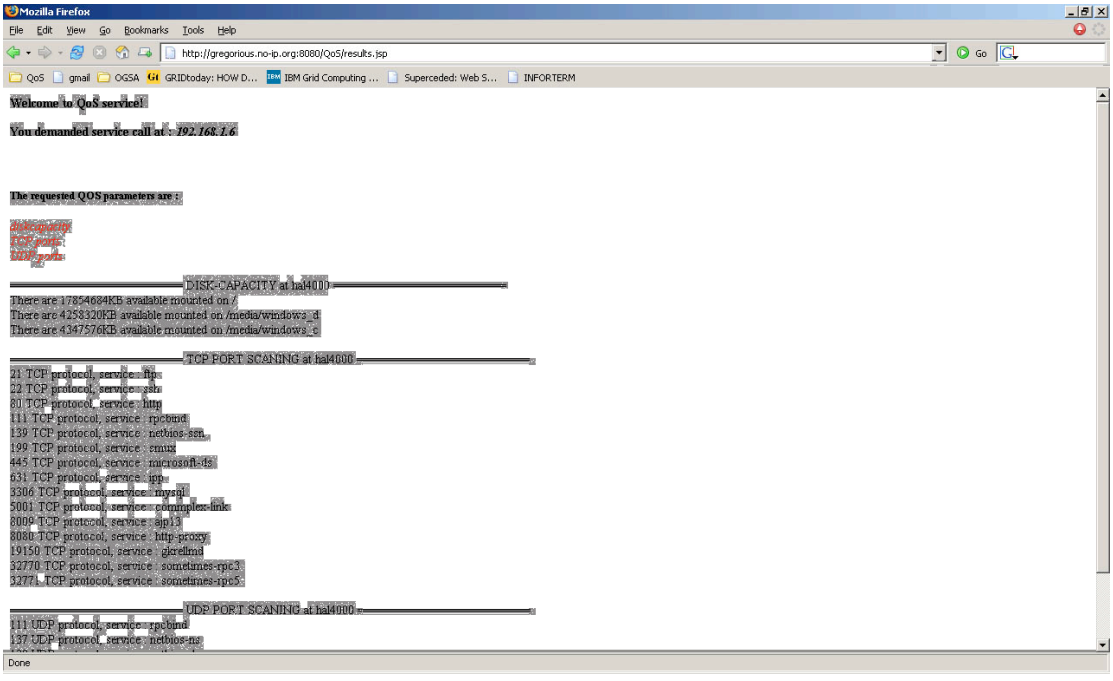
Εικόνα 22 : Executing On-Demand Service

Μας παρουσιάζονται οι διαθέσιμοι πόροι, πάνω στους οποίους μπορούμε να εκτελέσουμε την QoS υπηρεσία. Επιλέγουμε έναν και πατάμε SUBMIT. Στην συνέχεια εμφανίζονται οι παράμετροι της διαθεσιμότητας που υλοποιήθηκαν σε αυτήν την υπηρεσία. Μπορούμε να επιλέξουμε ανάμεσα στις diskcapacity, bandwidth, ping, TCPportscan, UDPportscan, libraries και service. Η λειτουργία κάθε μία από αυτές παρουσιάστηκε νωρίτερα. Κατά την δοκιμή χρήσης που κάνουμε επιλέξαμε diskcapacity, TCPportscan και UDPportscan.



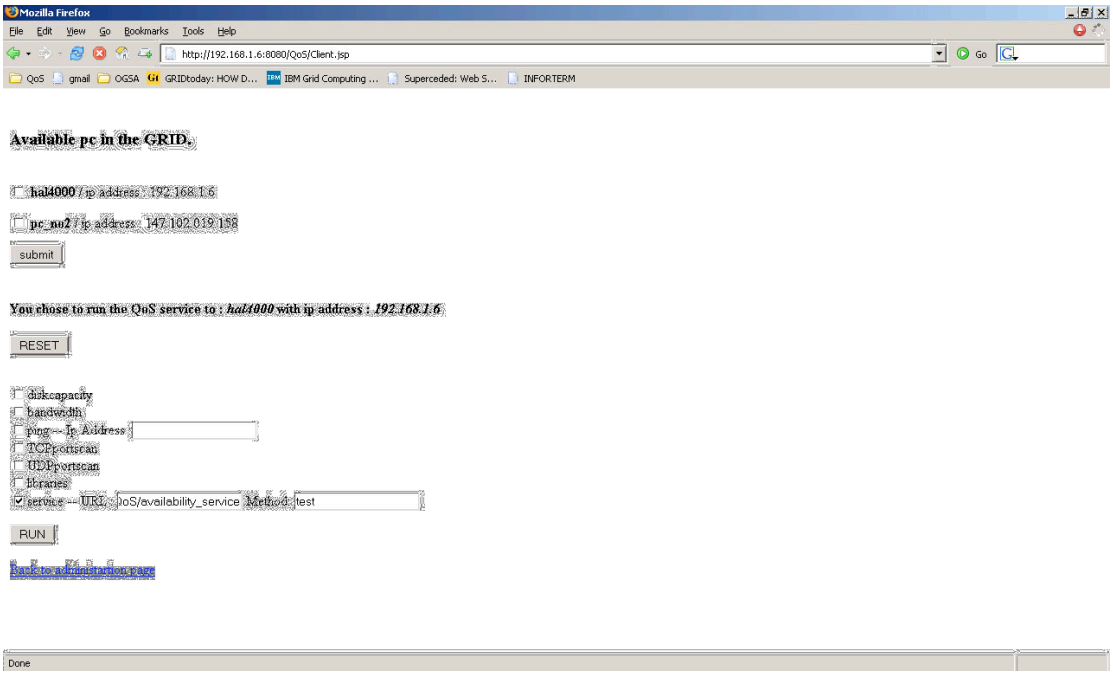
Εικόνα 23 : Completing Parameters

Τα αποτελέσματα της εκτέλεσης εμφανίζονται στην επόμενη οθόνη:

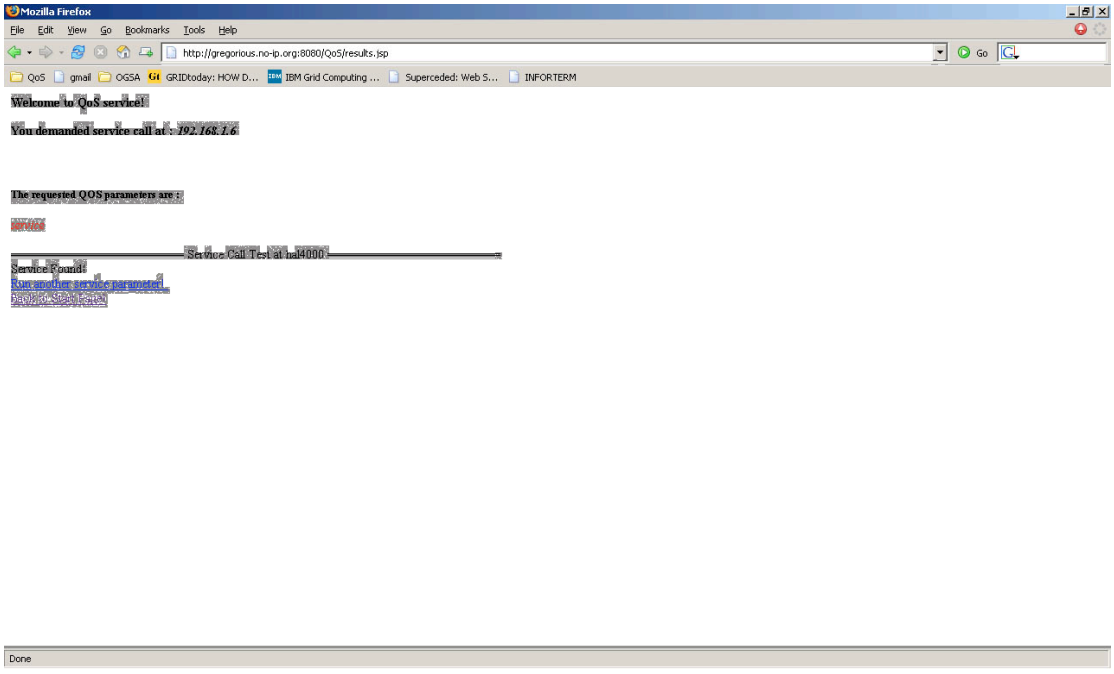


Εικόνα 24 : Result Page

Μπορούμε να επιστρέψουμε στην προηγούμενη οθόνη με τον κατάλληλο σύνδεσμο και να εκτελέσουμε και πάλι την υπηρεσία με άλλες παραμέτρους εκτέλεσης. Εδώ παρουσιάζουμε την εκτέλεση της παραμέτρου service, συμπληρώνοντας δοκιμαστικά την υπηρεσία που κατασκευάσαμε availability_service με την μέθοδο test.

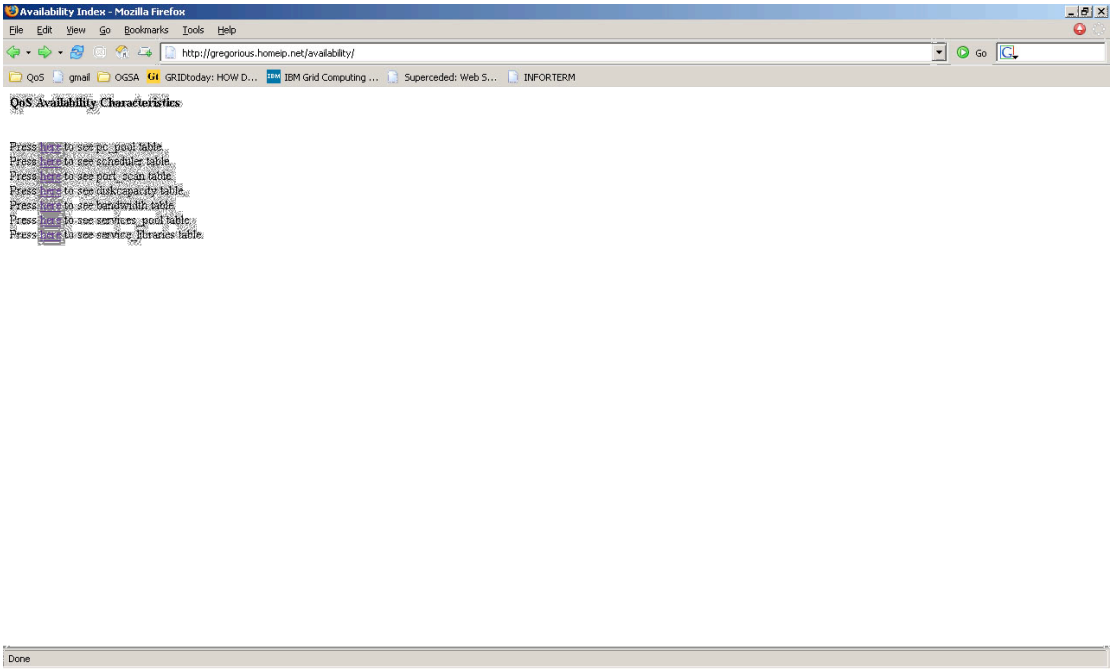


Εικόνα 25 : Execution Page



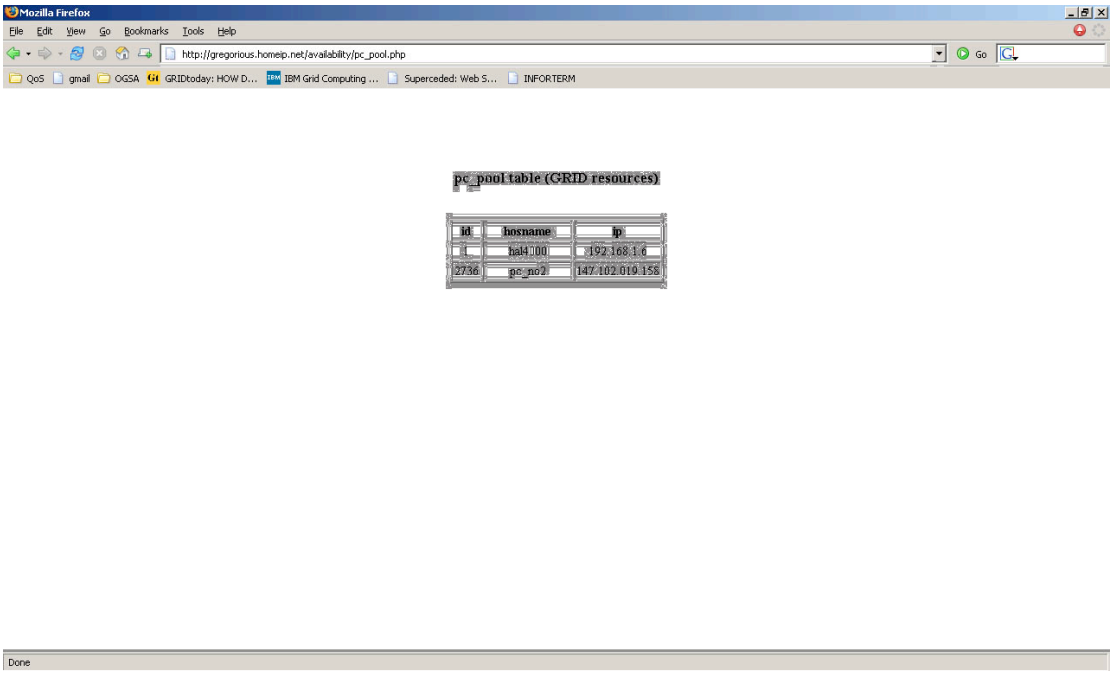
Εικόνα 26 : Result Page

Στην οθόνη των αποτελεσμάτων ενημερωνόμαστε για τον επιτυχή έλεγχο της κατάστασης της υπηρεσίας. Αφού εκτελέσουμε όποια παράμετρο θέλουμε, όσες φορές θέλουμε και σε όποιον πόρο επιθυμούμε μπορούμε να περάσουμε στην προβολή των αποτελεσμάτων. Σύνδεσμος προς τη σελίδα των αποτελεσμάτων παρέχεται από στην σελίδα διαχείρισης. Η αρχική σελίδα των αποτελεσμάτων θα φαίνεται κάπως έτσι:

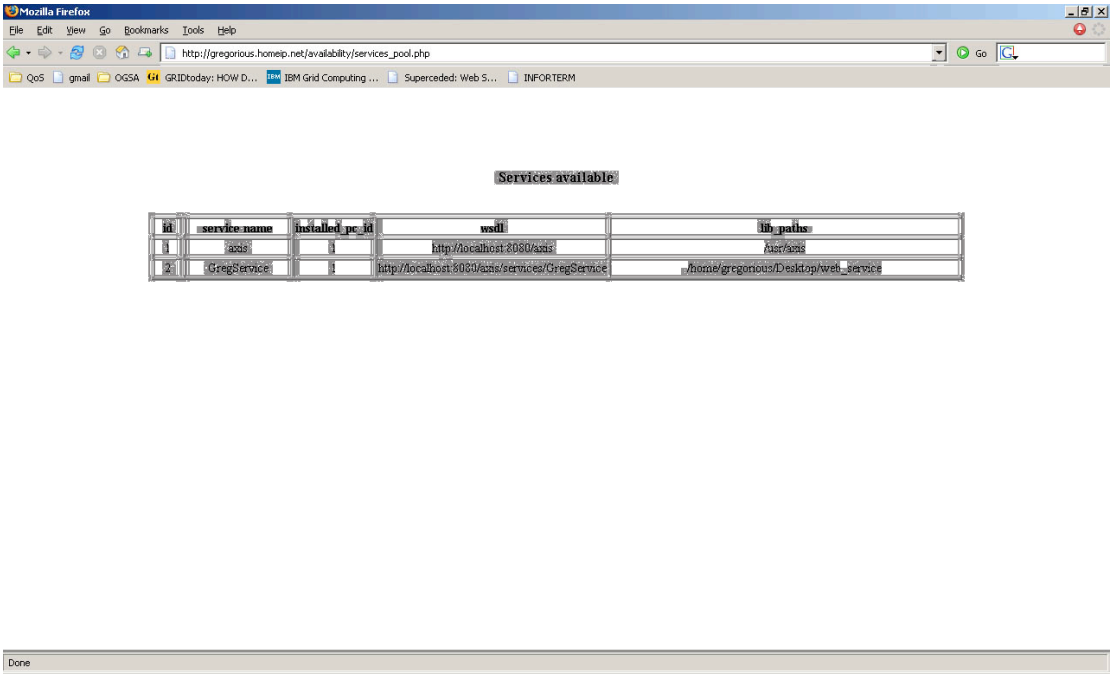


Εικόνα 27 : Παρουσίαση Πινάκων αποτελεσμάτων

Στο μενού που παρουσιάστηκε υπάρχει δυνατότητα παρουσίασης όλων των πινάκων της βάσης. Αρχικά θα εμφανίσουμε τους πίνακες που περιέχουν δεδομένα για την εκτέλεση, τον pc_pool πίνακα και τον service_pool:

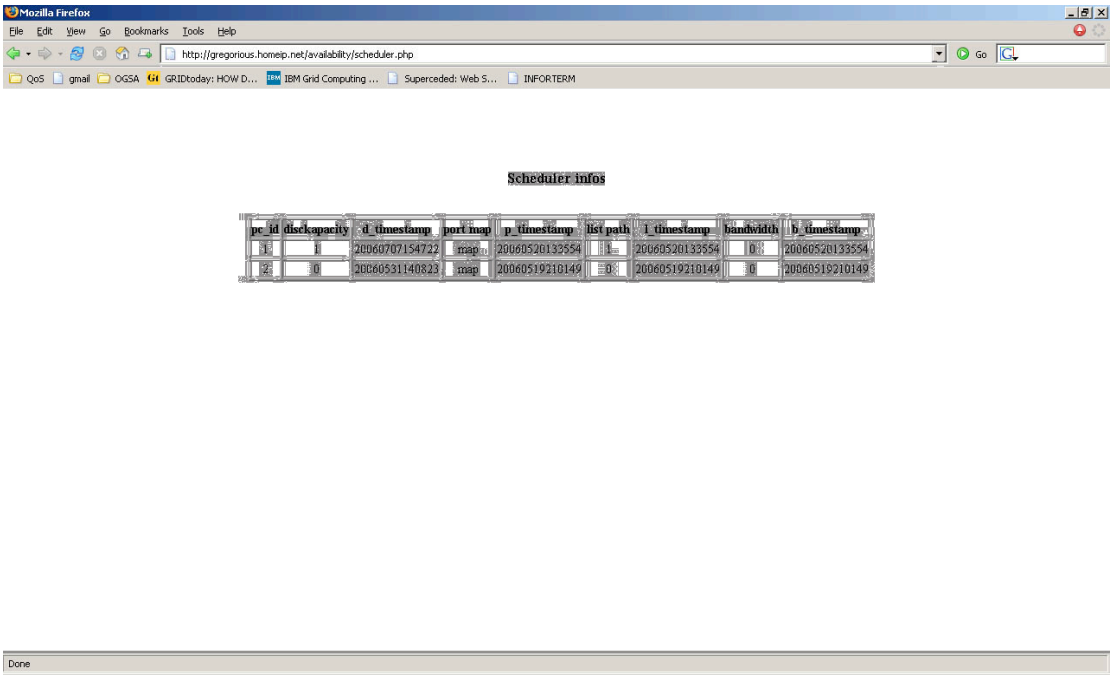


Εικόνα 28 : pc_pool table



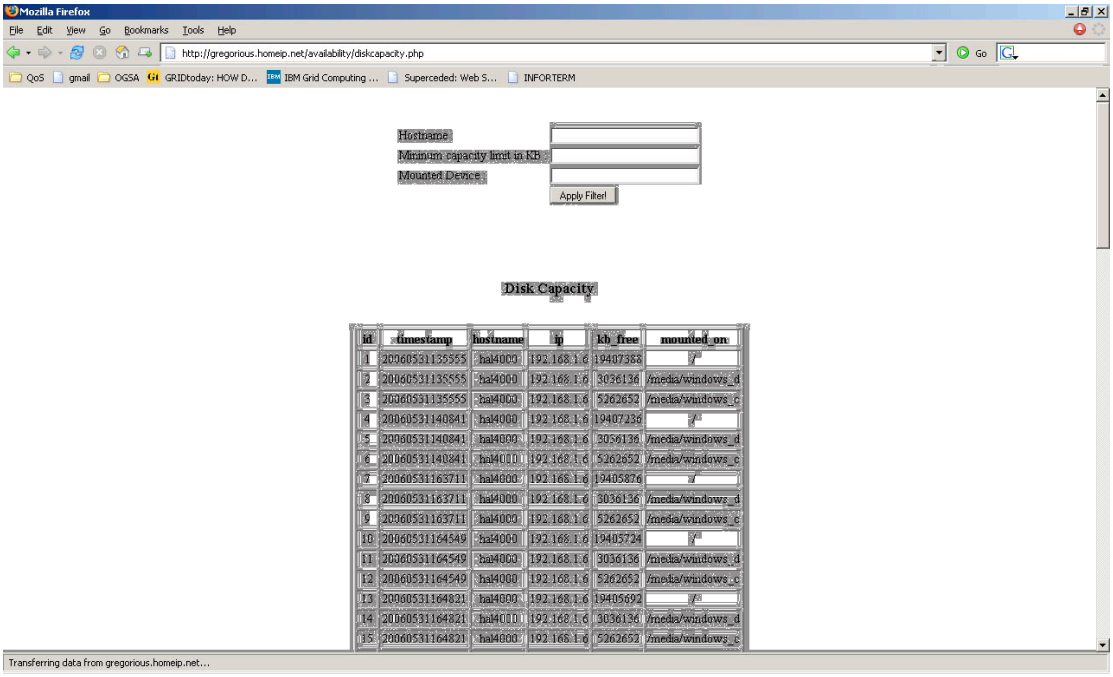
Εικόνα 29 : Service_pool table

Εκτός από αυτούς του δύο πίνακες και ο πίνακας scheduler περιέχει πληροφορίες για την εκτέλεση σχετικά με την χρονο-προγραμματισμό:

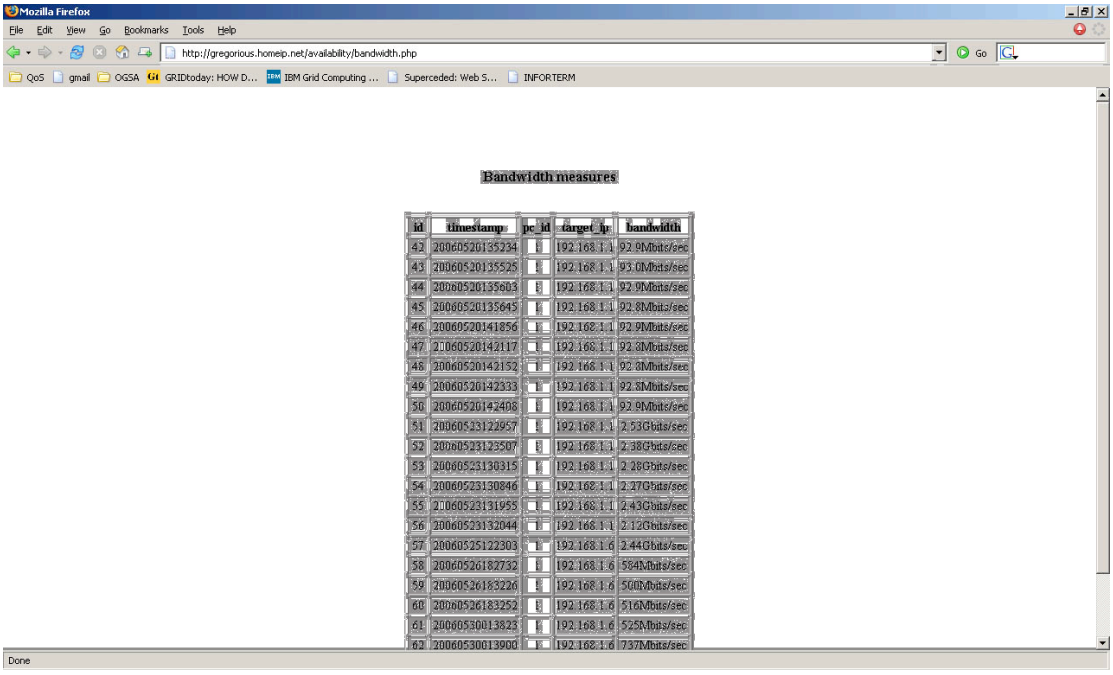


Εικόνα 30 : Scheduler table

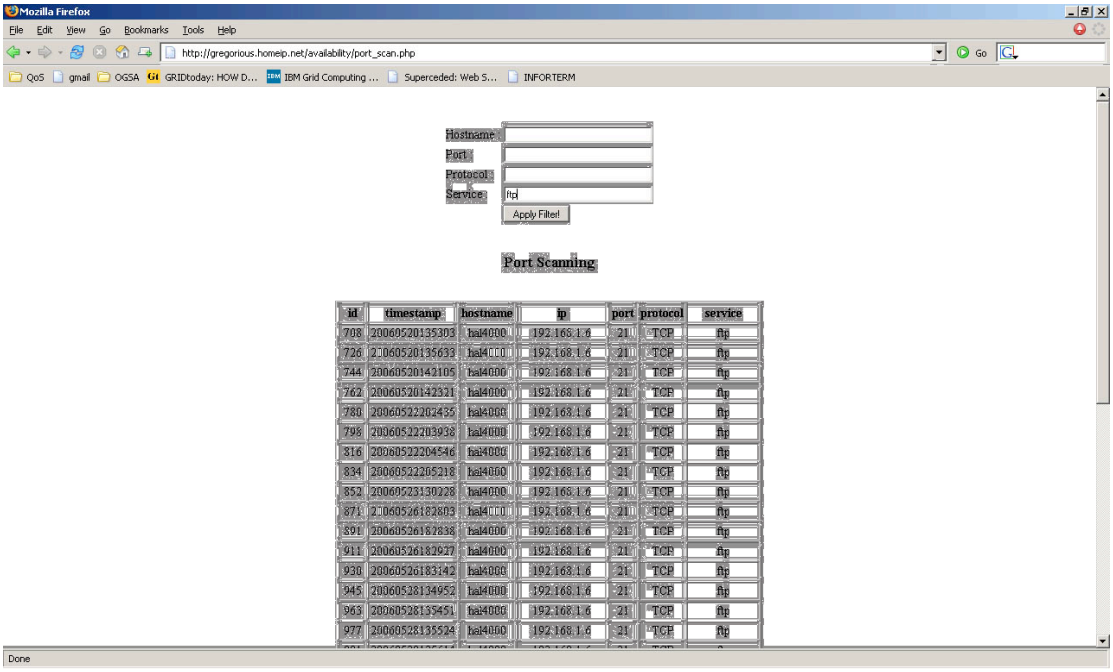
Παρακάτω παρουσιάζονται οι πίνακες αποτελεσμάτων κατά σειρά, για το diskcapacity, bandwidth, portscan, και libraries:



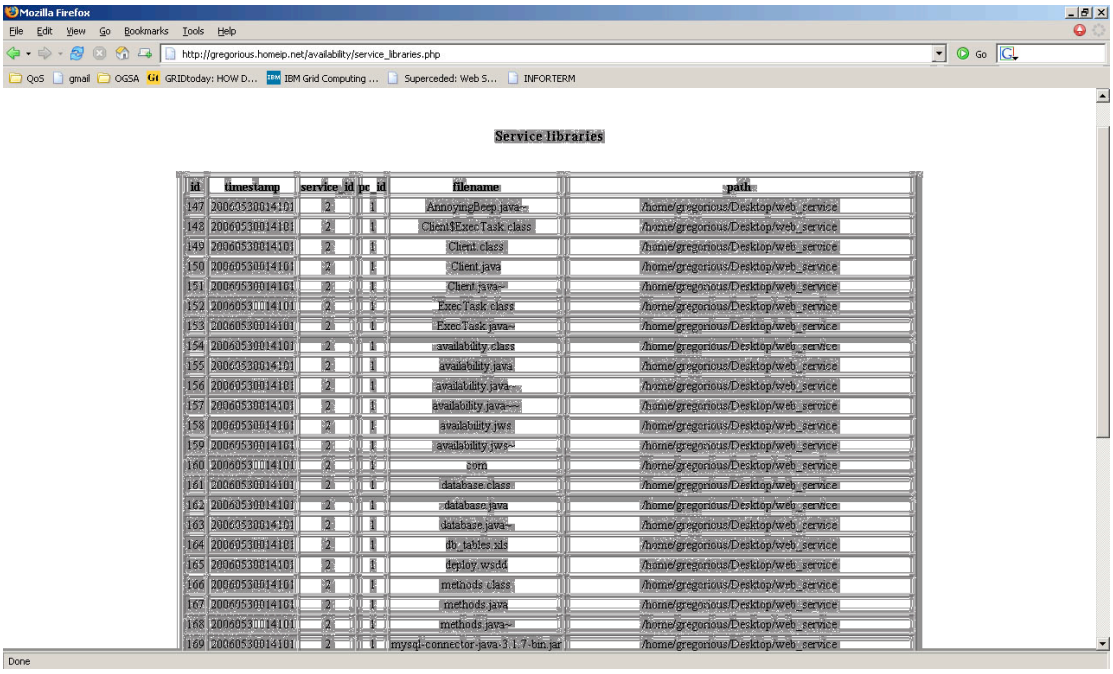
Εικόνα 31 :Diskcapacity table



Εικόνα 32 : Bandwidth table

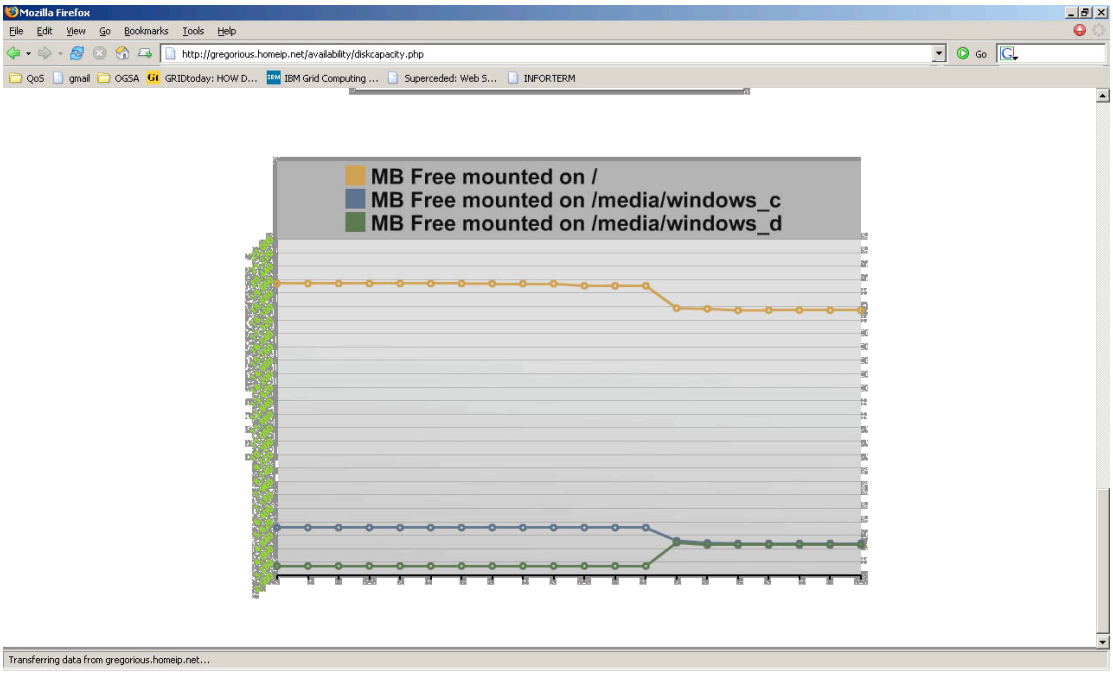


Εικόνα 33 : Portscan table



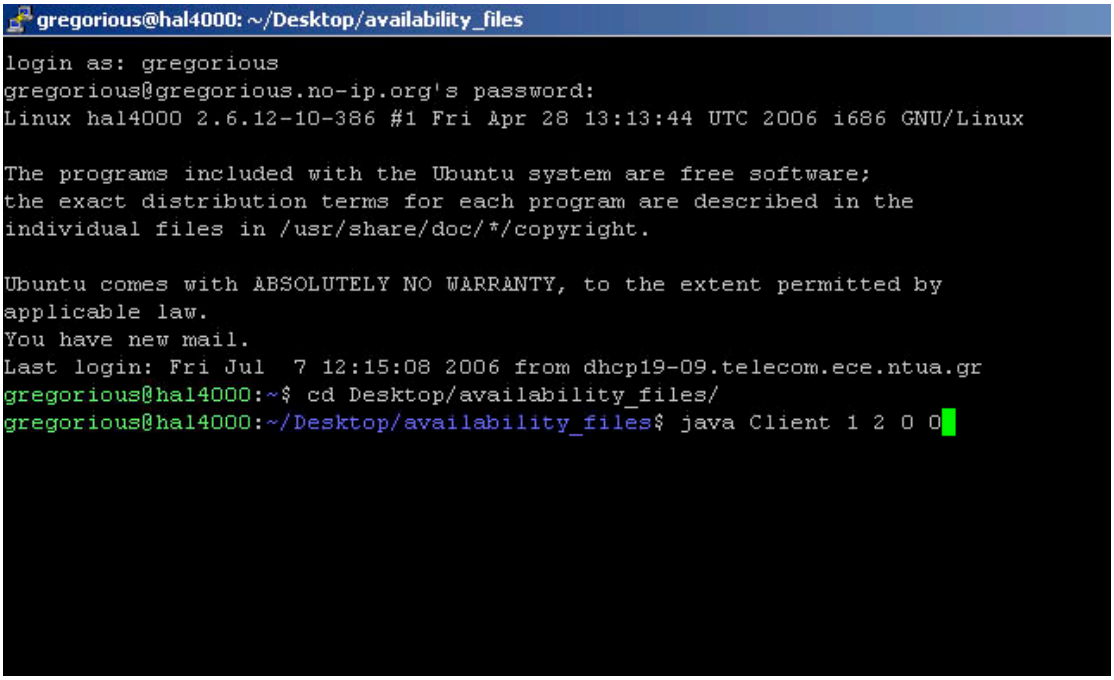
Εικόνα 34 : Services table

Όπως φαίνεται και στις εικόνες, σε μερικές παραμέτρους (diskcapacity, portscan) μπορούμε να εφαρμόσουμε φίλτρο αναζήτησης στα αποτελέσματα και να εντοπίσουμε αυτά που επιθυμούμε. Επίσης υπάρχει και προβολή δυναμικών γραφημάτων (charts) με την αναπαράσταση των αποτελεσμάτων σε βάθος εκτελέσεων. Από αυτά μπορούμε να βγάλουμε κάποια συμπεράσματα για την αξιοπιστία και την ποιότητα της υπηρεσίας αυτής.



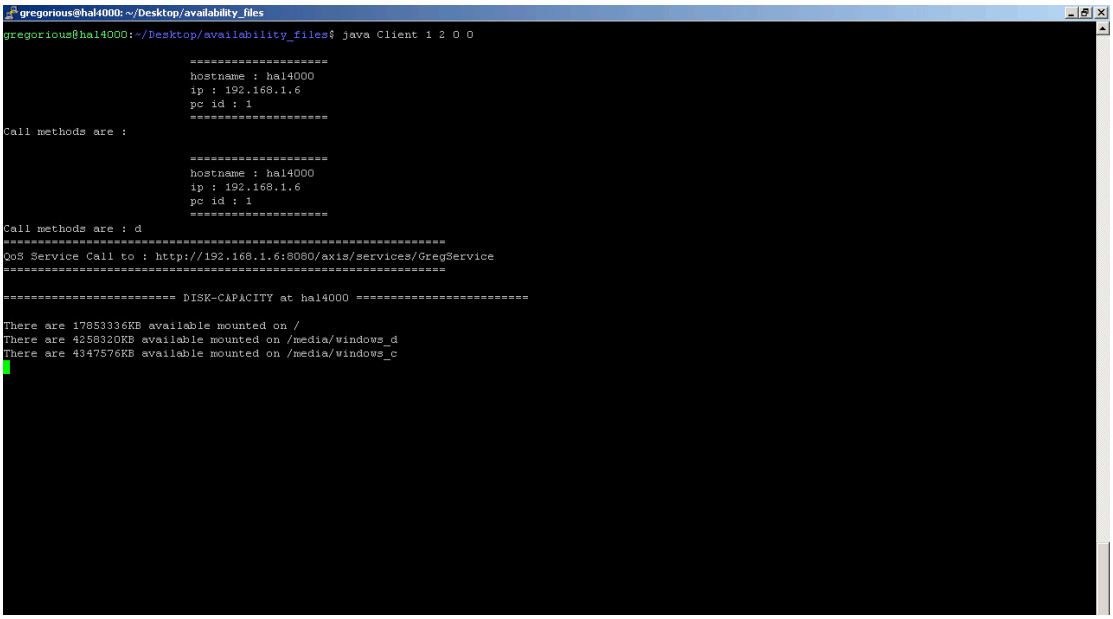
Εικόνα 35 : Διάγραμμα Diskcapacity αποτελεσμάτων

Τέλος για την εκκίνηση του χρονο-προγραμματιστή, εκτελούμε το αρχείο client.class με την εντολή java Clinet arg1 arg2 arg3 arg4 , ενώ βρισκόμαστε στον κατάλογο availability_files. Όπου arg1,2,3,4 είναι οι παράμετροι για τον χρόνο εκτέλεσης την μεθόδων diskcapacity, portscaning, bandwidth και libraries αντίστοιχα. Με παράμετρο 1 η εκτέλεση θα επαναλαμβάνεται ανά 10 sec, με 2 για 20 κτλ.



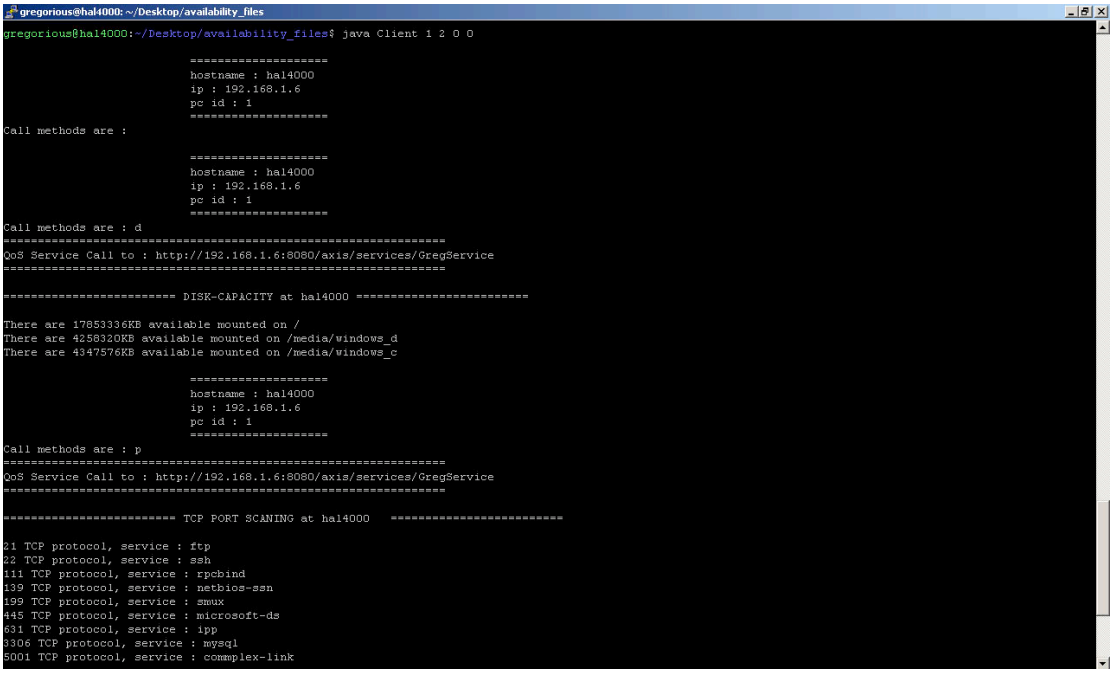
Εικόνα 36 : Εκτέλεση Client στην κονσόλα

Κατά την εκτέλεση αυτή έχουμε ρυθμίσει 1 κβάντο για το diskcapacity, 2 για το portscan και μηδέν στα άλλα. Έτσι παρακάτω βλέπουμε ότι περνάει ένας κύκλος εκτέλεσης χωρίς παράμετρο μεθόδου (Call methods are : EMPTY) και μετά από ένα κβάντο εκτέλεσης, εκτελείται η diskcapacity.



Εικόνα 37 : Αποτελέσματα Client Εκτέλεσης

Στη συνέχεια, στον επόμενο κύκλο εκτέλεσης, θα εφαρμοστεί η μέθοδος portscan αφού έχουν περάσει δύο κβάντα από την αρχική εκτέλεση και βλέπουμε τα αποτελέσματα για τις θύρες και τις υπηρεσίες που τις χρησιμοποιούν:



Εικόνα 38 : Αποτελέσματα Client Εκτέλεσης

Πατώντας Ctrl+C σταματάμε την εκτέλεση όποτε επιθυμούμε. Το SOAP μήνυμα που μεταφέρεται από τον Client προς την υπηρεσία μας μέσο του δικτύου για την κλήση αυτή είναι το παρακάτω:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:Body><DiskCapacity
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
</soapenv:Body>
</soapenv:Envelope>

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:Body><TCPportmap
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
</soapenv:Body>
</soapenv:Envelope>

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:Body><UDPportmap
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
</soapenv:Body>
</soapenv:Envelope>
```

Στο πρώτο XML μήνυμα αποστέλλεται η κλήση για την diskcapacity, στο δεύτερο για την TCPportmap και στο τρίτο για την UDPportmap.

Η απάντηση της πρώτης κλήσης από την υπηρεσία προς τον client είναι ένα άλλο SOAP μήνυμα που μεταφέρεται μέσο του δικτύου και περιέχει την απάντηση της μεθόδου μαζί με τα αποτελέσματα που υπολογίστηκαν:

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:Body>
<DiskCapacityResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<DiskCapacityReturn href="#id0"/></DiskCapacityResponse>

<multiRef id="id0" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="ns1:Map"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://xml.apache.org/xml-soap">

<item>
<key xsi:type="soapenc:string">mount</key>
```

```
<value soapenc:arrayType="xsd:string[50]" xsi:type="soapenc:Array">
<value xsi:type="xsd:string">Mounted</value>
<value xsi:type="xsd:string">/</value>
<value xsi:type="xsd:string">/media/windows_d</value>
<value xsi:type="xsd:string">/media/windows_c</value>

</value></item>

<item><key xsi:type="soapenc:string">count</key>
<value xsi:type="soapenc:int">4</value></item>
<item><key xsi:type="soapenc:string">available</key>
<value soapenc:arrayType="xsd:string[50]" xsi:type="soapenc:Array">
<value xsi:type="xsd:string">Available</value>
<value xsi:type="xsd:string">17853392</value>
<value xsi:type="xsd:string">4258320</value>
<value xsi:type="xsd:string">4347576</value>

</value></item></multiRef></soapenv:Body>
</soapenv:Envelope>
```

Μπορούμε να ξαναδοκιμάσουμε με ότι παράμετρο θέλουμε. Τα αποτελέσματα καταγράφονται κάθε φορά στη βάση δεδομένων, όπου πηγαίνοντας την αντίστοιχη σελίδα μπορούμε να τα δούμε.

5.2.1.2 Παρουσίαση Αποτελεσμάτων

Μετά από ένα μεγάλο διάστημα δοκιμών και εκτελέσεων, έχουμε συγκεντρώσει τα παρακάτω αποτελέσματα για τις παραμέτρους του diskcapacity, bandwidth, libraries :

Diskcapacity:

id	timestamp	hostname	ip	kb_free	mounted_on
1	2006/05/31-13:55:55	hal4000	192.168.1.6	19407388	/
2	2006/05/31-13:55:55	hal4000	192.168.1.6	3036136	/media/windows_d
3	2006/05/31-13:55:55	hal4000	192.168.1.6	5262652	/media/windows_c
4	2006/05/31-14:08:41	hal4000	192.168.1.6	19407236	/
5	2006/05/31-14:08:41	hal4000	192.168.1.6	3036136	/media/windows_d
6	2006/05/31-14:08:41	hal4000	192.168.1.6	5262652	/media/windows_c
7	2006/05/31-16:37:11	hal4000	192.168.1.6	19405876	/
8	2006/05/31-16:37:11	hal4000	192.168.1.6	3036136	/media/windows_d
9	2006/05/31-16:37:11	hal4000	192.168.1.6	5262652	/media/windows_c
10	2006/05/31-16:45:49	hal4000	192.168.1.6	19405724	/
11	2006/05/31-16:45:49	hal4000	192.168.1.6	3036136	/media/windows_d
12	2006/05/31-16:45:49	hal4000	192.168.1.6	5262652	/media/windows_c
13	2006/05/31-16:48:21	hal4000	192.168.1.6	19405692	/
14	2006/05/31-16:48:21	hal4000	192.168.1.6	3036136	/media/windows_d
15	2006/05/31-16:48:21	hal4000	192.168.1.6	5262652	/media/windows_c
16	2006/05/31-17:03:53	hal4000	192.168.1.6	19405616	/
17	2006/05/31-17:03:53	hal4000	192.168.1.6	3036136	/media/windows_d
18	2006/05/31-17:03:53	hal4000	192.168.1.6	5262652	/media/windows_c
19	2006/05/31-17:03:57	hal4000	192.168.1.6	19405616	/

20	2006/05/31-17:03:57	hal4000	192.168.1.6	3036136	/media/windows_d
21	2006/05/31-17:03:57	hal4000	192.168.1.6	5262652	/media/windows_c
22	2006/05/31-17:49:11	hal4000	192.168.1.6	19399684	/
23	2006/05/31-17:49:11	hal4000	192.168.1.6	3036136	/media/windows_d
24	2006/05/31-17:49:11	hal4000	192.168.1.6	5262652	/media/windows_c
25	2006/05/31-17:49:15	hal4000	192.168.1.6	19399680	/
26	2006/05/31-17:49:15	hal4000	192.168.1.6	3036136	/media/windows_d
27	2006/05/31-17:49:15	hal4000	192.168.1.6	5262652	/media/windows_c
28	2006/05/31-17:49:22	hal4000	192.168.1.6	19399672	/
29	2006/05/31-17:49:22	hal4000	192.168.1.6	3036136	/media/windows_d
30	2006/05/31-17:49:22	hal4000	192.168.1.6	5262652	/media/windows_c
31	2006/06/01-16:56:26	hal4000	192.168.1.6	19278164	/
32	2006/06/01-16:56:26	hal4000	192.168.1.6	3036136	/media/windows_d
33	2006/06/01-16:56:26	hal4000	192.168.1.6	5262652	/media/windows_c
34	2006/06/01-16:57:03	hal4000	192.168.1.6	19278052	/
35	2006/06/01-16:57:03	hal4000	192.168.1.6	3036136	/media/windows_d
36	2006/06/01-16:57:03	hal4000	192.168.1.6	5262652	/media/windows_c
37	2006/06/01-17:34:52	hal4000	192.168.1.6	19275708	/
38	2006/06/01-17:34:52	hal4000	192.168.1.6	3036136	/media/windows_d
39	2006/06/01-17:34:52	hal4000	192.168.1.6	5262652	/media/windows_c
40	2006/06/09-17:49:06	hal4000	192.168.1.6	17983008	/
41	2006/06/09-17:49:06	hal4000	192.168.1.6	4383200	/media/windows_d
42	2006/06/09-17:49:06	hal4000	192.168.1.6	4503376	/media/windows_c
43	2006/07/02-20:34:00	hal4000	192.168.1.6	17941168	/
44	2006/07/02-20:34:01	hal4000	192.168.1.6	4258320	/media/windows_d
45	2006/07/02-20:34:01	hal4000	192.168.1.6	4376376	/media/windows_c
46	2006/07/06-19:27:38	hal4000	192.168.1.6	17849748	/
47	2006/07/06-19:27:38	hal4000	192.168.1.6	4258320	/media/windows_d
48	2006/07/06-19:27:38	hal4000	192.168.1.6	4347576	/media/windows_c
49	2006/07/06-19:38:35	hal4000	192.168.1.6	17866968	/
50	2006/07/06-19:38:35	hal4000	192.168.1.6	4258320	/media/windows_d
51	2006/07/06-19:38:35	hal4000	192.168.1.6	4347576	/media/windows_c
52	2006/07/06-19:39:21	hal4000	192.168.1.6	17866952	/
53	2006/07/06-19:39:21	hal4000	192.168.1.6	4258320	/media/windows_d
54	2006/07/06-19:39:21	hal4000	192.168.1.6	4347576	/media/windows_c
55	2006/07/06-19:39:59	hal4000	192.168.1.6	17866940	/
56	2006/07/06-19:39:59	hal4000	192.168.1.6	4258320	/media/windows_d
57	2006/07/06-19:39:59	hal4000	192.168.1.6	4347576	/media/windows_c
58	2006/07/07-15:37:18	hal4000	192.168.1.6	17854684	/
59	2006/07/07-15:37:18	hal4000	192.168.1.6	4258320	/media/windows_d
60	2006/07/07-15:37:18	hal4000	192.168.1.6	4347576	/media/windows_c
61	2006/07/07-15:46:52	hal4000	192.168.1.6	17854664	/
62	2006/07/07-15:46:52	hal4000	192.168.1.6	4258320	/media/windows_d
63	2006/07/07-15:46:52	hal4000	192.168.1.6	4347576	/media/windows_c
64	2006/07/09-17:02:01	hal4000	192.168.1.6	17853380	/
65	2006/07/09-17:02:01	hal4000	192.168.1.6	4258320	/media/windows_d
66	2006/07/09-17:02:01	hal4000	192.168.1.6	4347576	/media/windows_c
67	2006/07/09-17:02:38	hal4000	192.168.1.6	17853360	/
68	2006/07/09-17:02:38	hal4000	192.168.1.6	4258320	/media/windows_d
69	2006/07/09-17:02:38	hal4000	192.168.1.6	4347576	/media/windows_c
70	2006/07/09-17:03:18	hal4000	192.168.1.6	17853352	/
71	2006/07/09-17:03:18	hal4000	192.168.1.6	4258320	/media/windows_d
72	2006/07/09-17:03:18	hal4000	192.168.1.6	4347576	/media/windows_c

73	2006/07/09-17:03:47	hal4000	192.168.1.6	17853340	/
74	2006/07/09-17:03:47	hal4000	192.168.1.6	4258320	/media/windows_d
75	2006/07/09-17:03:47	hal4000	192.168.1.6	4347576	/media/windows_c
76	2006/07/09-17:04:23	hal4000	192.168.1.6	17853336	/
77	2006/07/09-17:04:23	hal4000	192.168.1.6	4258320	/media/windows_d
78	2006/07/09-17:04:23	hal4000	192.168.1.6	4347576	/media/windows_c
79	2006/07/09-17:42:06	hal4000	192.168.1.6	17853392	/
80	2006/07/09-17:42:06	hal4000	192.168.1.6	4258320	/media/windows_d

Bandwidth :

id	timestamp	pc_id	target_ip	bandwidth
42	2006/05/20-13:52:34	1	192.168.1.1	9,2900E-02
43	2006/05/20-13:55:25	1	192.168.1.1	9,3000E-02
44	2006/05/20-13:56:03	1	192.168.1.1	9,2900E-02
45	2006/05/20-13:56:45	1	192.168.1.1	9,2800E-02
46	2006/05/20-14:18:56	1	192.168.1.1	9,2900E-02
47	2006/05/20-14:21:17	1	192.168.1.1	9,2800E-02
48	2006/05/20-14:21:52	1	192.168.1.1	9,2800E-02
49	2006/05/20-14:23:33	1	192.168.1.1	9,2800E-02
50	2006/05/20-14:24:08	1	192.168.1.1	9,2900E-02
51	2006/05/23-12:29:57	1	192.168.1.1	2,5300E+00
52	2006/05/23-12:35:07	1	192.168.1.1	2,3800E+00
53	2006/05/23-13:03:15	1	192.168.1.1	2,2800E+00
54	2006/05/23-13:08:46	1	192.168.1.1	2,2700E+00
55	2006/05/23-13:19:55	1	192.168.1.1	2,4300E+00
56	2006/05/23-13:20:44	1	192.168.1.1	2,1200E+00

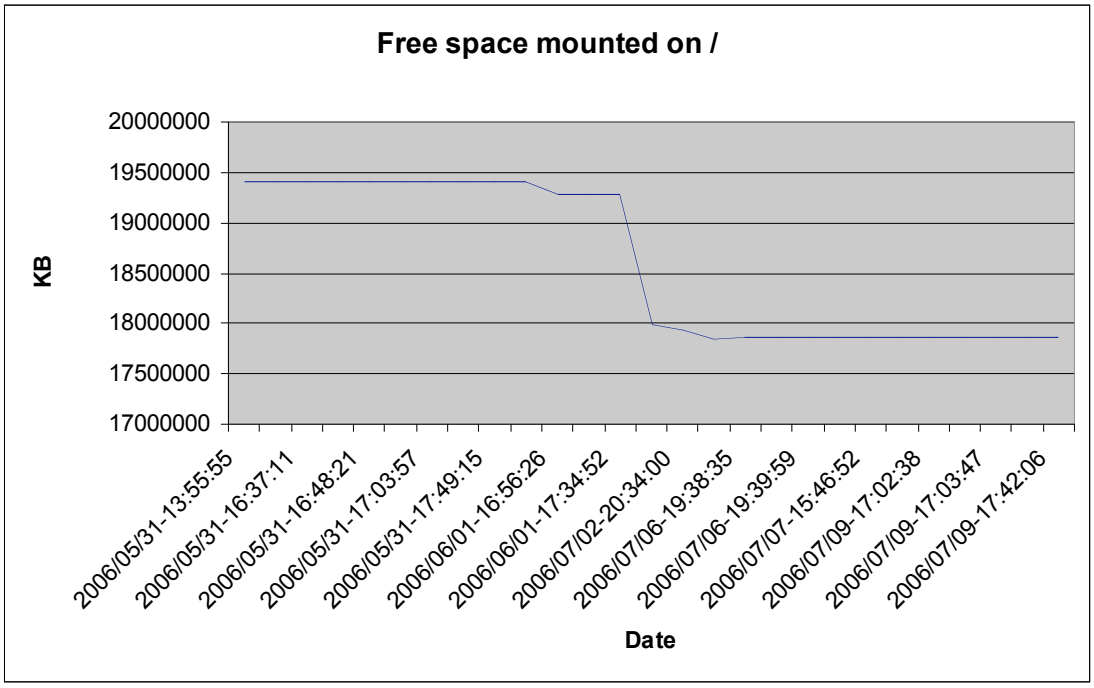
Libraries :

id	timestamp	service_id	pc_id	filename	path
147	2006/05/30-01:41:01	2	1	AnnoyingBeep.java~	/home/gregorious/Desktop/web_service
148	2006/05/30-01:41:01	2	1	Client$ExecTask.class	/home/gregorious/Desktop/web_service
149	2006/05/30-01:41:01	2	1	Client.class	/home/gregorious/Desktop/web_service
150	2006/05/30-01:41:01	2	1	Client.java	/home/gregorious/Desktop/web_service
151	2006/05/30-01:41:01	2	1	Client.java~	/home/gregorious/Desktop/web_service
152	2006/05/30-01:41:01	2	1	ExecTask.class	/home/gregorious/Desktop/web_service
153	2006/05/30-01:41:01	2	1	ExecTask.java~	/home/gregorious/Desktop/web_service
154	2006/05/30-01:41:01	2	1	availability.class	/home/gregorious/Desktop/web_service
155	2006/05/30-01:41:01	2	1	availability.java	/home/gregorious/Desktop/web_service

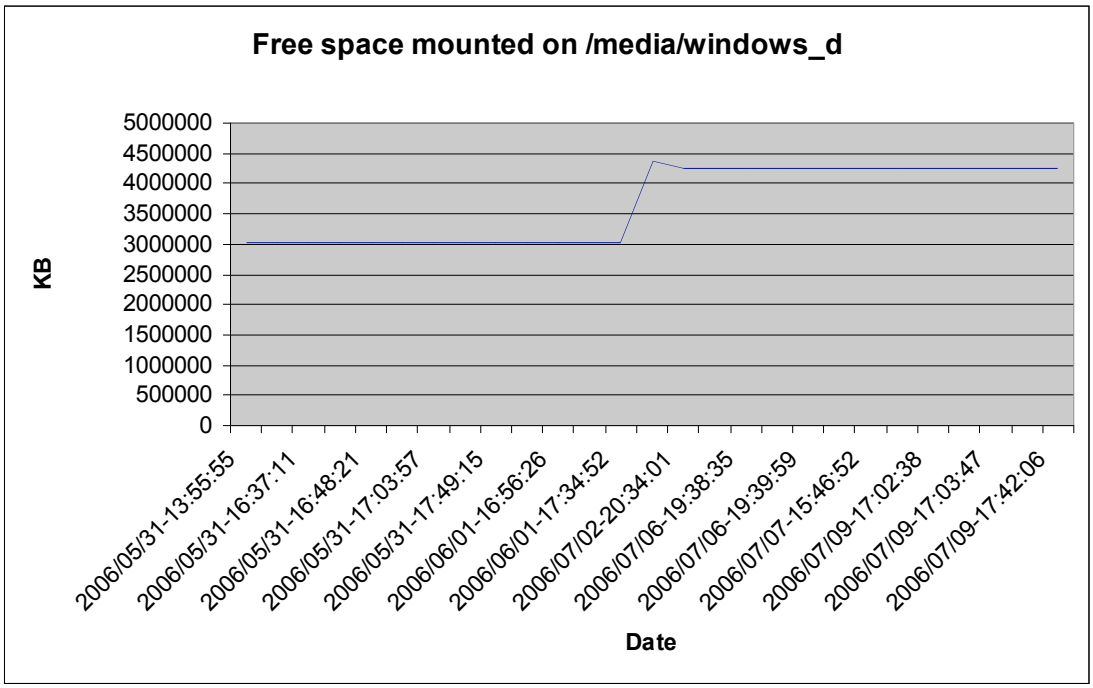
156	2006/05/30-01:41:01	2	1	availability.java~	/home/gregorious/Desktop/web_service
157	2006/05/30-01:41:01	2	1	availability.java~~	/home/gregorious/Desktop/web_service
158	2006/05/30-01:41:01	2	1	availability.jws	/home/gregorious/Desktop/web_service
159	2006/05/30-01:41:01	2	1	availability.jws~	/home/gregorious/Desktop/web_service
160	2006/05/30-01:41:01	2	1	com	/home/gregorious/Desktop/web_service
161	2006/05/30-01:41:01	2	1	database.class	/home/gregorious/Desktop/web_service
162	2006/05/30-01:41:01	2	1	database.java	/home/gregorious/Desktop/web_service
163	2006/05/30-01:41:01	2	1	database.java~	/home/gregorious/Desktop/web_service
164	2006/05/30-01:41:01	2	1	db_tables.xls	/home/gregorious/Desktop/web_service
165	2006/05/30-01:41:01	2	1	deploy.wsdd	/home/gregorious/Desktop/web_service
166	2006/05/30-01:41:01	2	1	methods.class	/home/gregorious/Desktop/web_service
167	2006/05/30-01:41:01	2	1	methods.java	/home/gregorious/Desktop/web_service
168	2006/05/30-01:41:01	2	1	methods.java~	/home/gregorious/Desktop/web_service
169	2006/05/30-01:41:01	2	1	mysql-connector-java-3.1.7-bin.jar	/home/gregorious/Desktop/web_service
170	2006/05/30-01:41:01	2	1	new file~	/home/gregorious/Desktop/web_service
171	2006/05/30-01:41:01	2	1	org	/home/gregorious/Desktop/web_service
172	2006/05/30-01:41:01	2	1	portal	/home/gregorious/Desktop/web_service
173	2006/05/30-01:41:01	2	1	sql-tables	/home/gregorious/Desktop/web_service
174	2006/05/30-01:41:01	2	1	sql-tables~	/home/gregorious/Desktop/web_service
175	2006/05/30-01:41:01	2	1	test.html~	/home/gregorious/Desktop/web_service
176	2006/05/30-01:41:01	2	1	test_wsrf	/home/gregorious/Desktop/web_service
177	2006/05/30-01:41:01	2	1	undeploy.wsdd	/home/gregorious/Desktop/web_service
178	2006/05/30-01:41:01	2	1	various	/home/gregorious/Desktop/web_service
179	2006/07/07-15:47:22	1	1	LICENSE	/usr/axis
180	2006/07/07-15:47:22	1	1	README	/usr/axis
181	2006/07/07-15:47:22	1	1	docs	/usr/axis
182	2006/07/07-15:47:22	1	1	lib	/usr/axis
183	2006/07/07-15:47:22	1	1	release-notes.html	/usr/axis
184	2006/07/07-15:47:22	1	1	samples	/usr/axis

Από τα παραπάνω αποτελέσματα μπορούμε να εξάγουμε κάποια διαγράμματα και από αυτά να συμπεράνουμε στοιχεία για την ποιότητα της υπηρεσίας συνολικά.

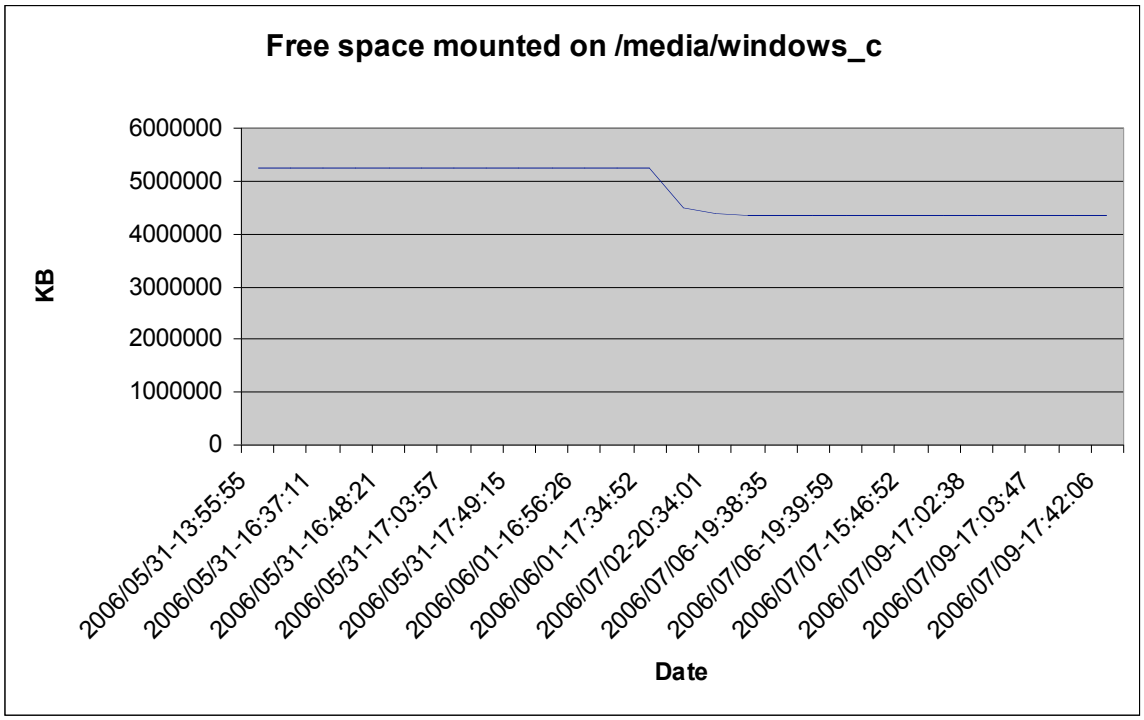
Έτσι από τον πρώτο πίνακα παρατηρούμε ότι οι εγκατεστημένοι δίσκοι στο συγκεκριμένο μηχάνημα είναι τρεις. Η διακύμανση του ελεύθερου χώρου όλο αυτό το διάστημα φαίνεται παρακάτω:



Εικόνα 39 : Διάγραμμα διακύμανσης διαθέσιμου χώρου στο /



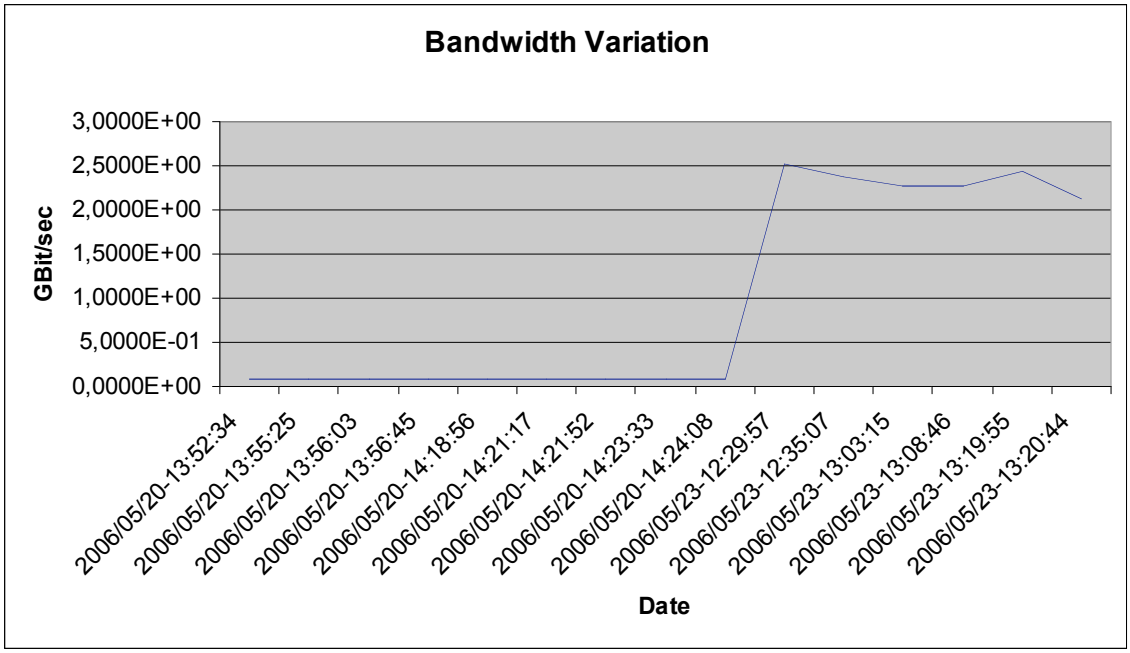
Εικόνα 40 : Διάγραμμα διακύμανσης διαθέσιμου χώρου στο /media/windows_d



Εικόνα 41 : Διάγραμμα διακύμανσης διαθέσιμου χώρου στο /media/windows_c

Από τα διαγράμματα, παρατηρούμε μια μεγάλη αλλαγή στον διαθέσιμο χώρο, και για τους τρεις δίσκους, την περίοδο από 01/06/2006 μέχρι 02/07/2006. Αυτό μπορεί να σημαίνει μεταφορά όγκου πληροφορίας από τον ένα δίσκο στον άλλο, είτε εγκατάσταση προγραμμάτων κλπ. Συμπερασματικά σε σχέση με την διαθεσιμότητα και την ποιότητα της υπηρεσίας μπορούμε να πούμε ότι ο πρώτος δίσκος έχει μεγάλη κινητικότητα και χρήση, με διαθεσιμότητα ελεύθερου σε κάποια φάση φθίνουσα, ο δεύτερος είχε μια διακύμανση αύξησης στην ίδια περίοδο και ο τρίτος μικρή πτώση κατά την ίδια περίοδο. Το τελευταίο όμως διάστημα και οι τρεις δίσκοι είναι σταθεροί στις τιμές διαθέσιμου χώρου.

Από τον πίνακα του εύρους ζώνης σχηματίζουμε το διάγραμμα διακύμανσης των μετρήσεων για το εύρος ζώνης μεταξύ του μηχανήματος που εγκαταστήσαμε την υπηρεσία και το μηχάνημα που χρησιμοποιήσαμε ως πελάτη (client) της υπηρεσίας.



Εικόνα 42 : Διάγραμμα διακύμανσης εύρους ζώνης

Από το διάγραμμα αυτό εξάγουμε σημαντικά συμπεράσματα για την ποιότητα της υπηρεσίας στην πορεία του χρόνου. Το εύρος ζώνης είναι πολύ σημαντικό για την εκτέλεση και χρήση υπηρεσιών σε Grid περιβάλλοντα αφού ο ρυθμός μεταφοράς των δεδομένων παίζει καθοριστικό ρόλο στον χρόνο που θα χρειαστεί και φυσικά στο κόστος. Έτσι παρατηρούμε ότι αρχικά το εύρος ζώνης ήταν σταθερό κοντά στα 100Mbit/sec και στη συνέχεια την περίοδο 20/05/2006-23/05/2006 έχουμε μια μεγάλη αύξηση με τιμές εύρους ζώνης κοντά στο 2.5 Gbit/sec. Αυτή η μεγάλη αύξηση μπορεί να σημαίνει αναβάθμιση του δικτύου που ήταν εγκατεστημένο ή γενικά αναβάθμιση του hardware εξοπλισμού (κάρτες δικτύου, routers κλπ). Η αύξηση του παρεχόμενου εύρους ζώνης αναβαθμίζει την ποιότητα των υπηρεσιών που το συγκεκριμένο Grid προσφέρει και φυσικά των περιθωρίων χρήσης του.

Η επεξεργασία και η παρατήρηση και άλλων παραμέτρων για διαθεσιμότητα και απόδοση μπορούν να μας προσφέρουν σημαντικά αποτελέσματα για την αξιολόγηση της ποιότητας και αξιοπιστίας ενός Grid συστήματος και των υπηρεσιών που αυτό προσφέρει.

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

Κατά τη διάρκεια αυτής της διπλωματικής υλοποιήσαμε έναν μηχανισμό για την πρόβλεψη της ποιότητας της υπηρεσίας (QoS Provisioning Module), Grid εφαρμογών. Χρησιμοποιώντας τεχνολογίες αιχμής όπως SOAP-XML, Web Services υλοποιήσαμε μία υπηρεσία που μας παρέχει με πληροφορίες σχετικές με την παρεχόμενη ποιότητα. Η χρήση βάσης δεδομένων μας βοήθησε στην αποθήκευση των πληροφοριών για μεταγενέστερη ανάκληση και επεξεργασία. Επίσης αναπτύχθηκε γραφικό περιβάλλον επικοινωνίας με το χρήστη είτε για την ρύθμιση των παραμέτρων των εκτελέσεων αλλά και για την προβολή των αποτελεσμάτων.

Οι τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση αυτή ήταν :

- Java ως γλώσσα προγραμματισμού για την υλοποίηση της υπηρεσίας
- Eclipse ως η πλατφόρμα ανάπτυξης του κώδικα
- MySQL βάση δεδομένων για την αποθήκευση των αποτελεσμάτων

- JSP δυναμικές σελίδες για την κατασκευή των διασυνδέσεων χρήστη
- PHP δυναμικές σελίδες για την προβολή των αποτελεσμάτων
- Tomcat Server για την εγκατάσταση της υπηρεσίας
- Linux Ubuntu 5.10 για την δοκιμαστική εκτέλεση της υπηρεσίας

Η διαδικασία ανάπτυξης της εφαρμογής περιλάμβανε τα ακόλουθα βήματα:

- Εξοικείωση με το πρόβλημα και τις σχετικές τεχνολογίες
- Βασικός Σχεδιασμός

  Το τμήμα αυτό περιλαμβάνει σχεδιασμό του της υπηρεσίας, τον προσδιορισμό των παραμέτρων εκτέλεσης και των ζητούμενων αποτελεσμάτων.
- Λεπτομερής σχεδιασμός

  Το τμήμα αυτό περιλαμβάνει το αναλυτικό σχεδιασμό των υπηρεσιών που θα χρησιμοποιηθούν, τον σχεδιασμό της βάσης δεδομένων και την επιλογή των απαιτούμενων μονάδων κώδικα για την ανάπτυξη της εφαρμογής.
- Συγγραφή κώδικα και ανάπτυξη της εφαρμογής (deployment)
- Εξαγωγή συμπερασμάτων- Δυνατές Επεκτάσεις

Μετά την ολοκλήρωση καθενός από τα βήματα σχεδίασης και υλοποίησης ακολούθησε έλεγχος και τεκμηρίωση.

6.1.2 Συμπεράσματα

Η ενασχόληση μας με την Grid τεχνολογία, κατά την ανάπτυξη αυτής της διπλωματικής εργασίας μας παρουσίασε τις μεγάλες δυνατότητες και τις προοπτικές αυτής της τεχνολογίας. Βρισκόμαστε ήδη στην ολοκλήρωση του πρώτου κύκλου ανάπτυξης των Grid τεχνολογιών, δηλαδή της λειτουργικής θεμελίωσης του Grid, και βαδίζουμε αυτή τη στιγμή στο επόμενο στάδιο, αυτό της αξιοποίησης και χρήσης του. Η υπηρεσιοστραφής μορφή αυτής της τεχνολογίας (service oriented), μας οδηγεί στην υιοθέτηση του όρου QoS (Ποιότητα Υπηρεσίας) και στη προσπάθεια ορισμού του με συγκεκριμένες παραμέτρους.

Η σημασία παρακολούθησης και επεξεργασίας αυτών των παραμέτρων είναι μεγάλη αφού μπορούν να μας δώσουν μια ένδειξη για την κατάσταση του πάροχου μιας Grid υπηρεσίας, τις δυνατότητές του, τους περιορισμούς του και συνολικά από όλα τα παραπάνω μια πρώτη εικόνα της ποιότητας που προσφέρει. Όταν υπεισέρχονται στην χρήση υπηρεσιών Grid, επιχειρήσεις και συμφωνίες μεταξύ αυτών, η έννοια της ποιότητας της υπηρεσίας παίρνει άλλες διαστάσεις.

Επιπρόσθετα με τα παραπάνω καταλήξαμε ότι υπηρεσίες διαδικτύου (web services) είναι ένα ισχυρό εργαλείο για περιπτώσεις χρήσης σχετικές με Grid, και γενικά απομακρυσμένης εκτέλεσης και παροχής πληροφοριών. Η προσεκτική υλοποίηση αυτών των υπηρεσιών μπορούν να μας προσφέρουν ένα ασφαλές και ευέλικτο μοντέλο επικοινωνίας.

Τέλος η χρήση προγραμματιστικών λύσεων ελεύθερης χρήσης (open source solutions) όπως Java programming, Linux OS, MySQL Databases, Tomcat Server, δεν παρουσίασε κανένα απολύτως πρόβλημα κατά την ανάπτυξη του μηχανισμού, παρέχοντας μας όλες τις απαραίτητες λειτουργίες που επιθυμούσαμε.

6.2 Μελλοντικές επεκτάσεις

6.2.1 Δυνατές Επεκτάσεις-Εκκρεμότητες της Υλοποίησης

Ο μηχανισμός που υλοποιήσαμε στην παρούσα διπλωματική παρουσιάζει και εφαρμόζει τον έλεγχο της διαθεσιμότητας των υπηρεσιών σε Grid συστήματα. Το κομμάτι αυτό λειτουργεί μεν ολοκληρωμένα και ανεξάρτητα αλλά επιδέχεται επεκτάσεων και συμπληρώσεων.

Ως πρώτη επέκταση μπορεί να θεωρηθεί η κατασκευή ενός έξυπνου χρονο-προγραμματιστή (smart scheduler). Η υπάρχουσα υπηρεσία QoS Provisioning χρησιμοποιεί μια απλή έκδοση scheduler, με προκαθορισμένο χρόνο εκτέλεσης της κάθε μεθόδου. Μια δυνατή επέκταση είναι η κατασκευή ενός δυναμικού χρονο-προγραμματιστή ο οποίος θα προσαρμόζει τους χρόνους συχνότητας των εκτελέσεων ανάλογα με την συχνότητα και εύρος αλλαγών των αποτελεσμάτων. Η ιδέα αυτή πηγάζει από την τεχνολογία των νευρωνικών δικτύων, όπου μπορεί να φτάσει μέχρι και την πρόβλεψη της επόμενης τιμής. Η υλοποίηση αυτού του μηχανισμού είναι αρκετά πολύπλοκη αφού πρέπει να χρησιμοποιηθούν αλγόριθμοι πρόβλεψης τιμών με επεξεργασία προηγούμενων αποτελεσμάτων εκτέλεσης. Παρόλα αυτά ο σχεδιασμός της υπάρχουσας υπηρεσίας βοηθάει στην μελλοντική επέκταση, αφού για την τροποποίηση αυτή αρκεί η υλοποίηση του κώδικα του έξυπνου χρονο-προγραμματιστή και η τοποθέτηση στην υπάρχουσα μέθοδο scheduler της Client.java κλάσης.

Επόμενη τροποποίηση αναβάθμισης της υπάρχουσας υλοποίησης είναι η προσθήκη παραμέτρων για την κάλυψη της απόδοσης των υπηρεσιών. Για την αναβάθμιση αυτή απαιτείται μελέτη για τον προσδιορισμό των παραμέτρων και κατασκευή μιας υπηρεσίας που θα εγκαθίσταται στους πόρους του συστήματος για την πρόβλεψη της ποιότητας της υπηρεσίας (QoS Provisioning).

Φυσικά ως συνέχεια των παραπάνω δε μπορούμε να ξεχάσουμε την επέκταση του μηχανισμού και κάλυψη των παραμέτρων της αξιοπιστίας. (Reliability). Αυτή η αναβάθμιση απαιτεί αναλυτική έρευνα αφού η έννοια της αξιοπιστίας παραμένει λίγο αόριστη.

7 *Βιβλιογραφία*

[FK98] The Grid ,Blueprint for a New Computing Infrastructure Edited by Ian Foster and Carl Kesselman,1998.

[F02] What is the Grid? A three point Checklist by Ian Foster, 2002.

[F01] The Anatomy of the Grid: Enabling Scalable Virtual Organizations, Ian Foster

Mathematics and Computer Science Division, Argonne National Laboratory

Department of Computer Science, The University of Chicago, USA, 2001.

[FKL+99] I. Foster, C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt, and A. Roy. A distributed resource management architecture that supports advance reservation and coallocation.In Proceedings of the International Workshop on Quality of Service, pages 27–36, 1999.

[CFF+05] From Open Grid Services Infrastructure to WSResource

Framework: Refactoring & Evolution,

Karl Czajkowski (Globus Alliance / USC Information Sciences Institute)

Don Ferguson (IBM), Ian Foster (Globus Alliance / Argonne National Laboratory), Jeff Frey (IBM), Steve Graham (IBM), Tom Maguire (IBM), David Snelling (Fujitsu Laboratories of Europe), Steve Tuecke (Globus Alliance / Argonne National Laboratory), May 2005.

[LG] The Business Grid: Providing Transactional Business Processes via Grid Services Frank Leymannand, Kai Güntzel.

[SSK04] Business Grid Computing Project Activities

Andreas Savva, Toshiyuki, Suzuki, Hiro Kishimoto.

(Manuscript received August 27, 2004)

[M03] Web Services Technologies Report for the JISC Technology Watch Service Diane McDonald, University of Strathclyde, May 2003.

[TSD+05] NEXTGrid P4.8.2 Modelling Resources, Kostas Tserpes, Fabrizio Silvestri, Konstantinos Dolkas, Dimos Kyriazis, Andreas Menychtas, September 2005.

[TKL+05] An Open Architecture for Providing QoS Information in Business Grids Konstantinos Tserpes, Dimosthenis Kyriazis, Antonios Litke, Andreas Menychtas, and Theodora Varvarigou, 2005.

[F02] How does one really characterize Grid computing? By Richard F. Freund, CTO, GridIQ (http://www.Gridtoday.com/02/0819/100249.html

[CIM] *Common Infromation Model (CIM), Infrastructure Specification ,DMTF* (http://www.dmtf.org/standards/cim/cim_schema_v23/)

A Complete History of the Grid, Dr Rob Baxter Software Development Group Manager ,University of Edinburgh.

[SGM+03] A. Sahai and S. Graupner and V. Machiraju and A. Moorsel. (Specifying and Monitoring) Guarantees in Commercial Grids through SLA. Proceedings of the 3rd IEEE/ACM CCGrid2003, 2003.

[CFK+02] K. Czajkowski and I. Foster and C. Kesselman and V. Sander and S. Tuecke SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems. Proceedings of the 8th Workshop on Job Scheduling Strategies for Parallel Processing, 2002.

[KM03] K. Keahey and K. Motawi The Taming of the Grid: Virtual Application Service.Argonne National Laboratory Technical Memorandum No. 262, May 2003.

[RAK+04] Analysis and Provision of QoS for Distributed Grid Applications Rashid J. Al-Ali, Kaizar Amin, Gregor von Laszewski, Omer F. Rana, David W. Walker, Mihael Hategan, and Nestor Zaluzec, April 2004.

[T05] Ολοκλήρωση Συστημάτων και Εφαρμογών με χρήση Web Services «Διαλειτουργικότητα με PDA και Γεωγραφική Πληροφορία» , Διπλωματική Εργασία Κωνσταντίνου Τσούλου , 2005.

[M04] Προγραμματισμός σε Περιβάλλον Πολυπλέγματος, Διπλωματική Εργασία Μενύχτα Αντρέα, 2004.

[ACK+04] G. Alonso, F. Casati, H. Kuno, V. Machiraju.
"Web Services: Concepts, Architectures and Applications".

Springer-Verlag Berlin Heidelberg 2004

The Next Generation Grid (NEXTGrid), www.nextGrid.org

IBM About Grid Computing

http://www-1.ibm.com/Grid

JAVA Online Help
http://www.jguru.com

Apache Tomcat
http://jakarta.apache.org/tomcat/

W3C –XML Techonologies
http://www.w3.org

JDBC Drivers
http://www.cis.upenn.edu/~cis550/TRASH/JDBC_doc/jdbcoci4.htm
http://www.oracle.com/technology/tech/java/sqlj_jdbc/htdocs/jdbc_faq.htm

Global Grid Forum (GGF)

http://www.Gridforum.org/

Παράρτημα

Server Side

Availability.java

```
import java.io.BufferedReader;
import java.io.File;
import java.io.InputStreamReader;
import java.util.Arrays;
import java.util.Hashtable;
import java.util.StringTokenizer;
import java.lang.Exception;
import org.apache.axis.AxisFault;
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;

public class availability {

   public String service_test (String endpoint, String
method) throws Exception{


try{

           Service service = new Service ();
           Call call = (Call)
```

```
            service.createCall ();
            try{
            call.setTargetEndpointAddress (new
java.net.URL (endpoint));
            }catch (Exception e){ return ("Call Exception:
"+e.getMessage ());}

            call.setOperationName (method);
            call.invoke (new Object[] {});
            return "Service Found!";
          } catch (AxisFault ex) { return ("Call
Exception: " +  ex.getFaultString () ); }

}


   public Hashtable listPath (String pathString) {


         File files[];
         File path=new File (pathString);
         String[] filelist =new String[100];

         files = path.listFiles ();

         Arrays.sort (files);

         for (int i=0; i<files.length;i++)
               filelist[i]=files[i].toString ();



   Hashtable table = new Hashtable ();
   table.put ("files",filelist);
```

```
   table.put ("count",new Integer (files.length));

   return table;

   }

   public Hashtable DiskCapacity () throws
java.io.IOException {
         String[] cmd = { "df", "-klx", "tmpfs" };
         String line = "";
         String[] available = new String[50];
         String[] mount = new String[50];
         Process prcs;
         int i = 0;

   try
         {
         prcs = Runtime.getRuntime ().exec (cmd);
         InputStreamReader inp2 = new InputStreamReader
(prcs.getInputStream ());
         InputStreamReader inp3 = new InputStreamReader
(prcs.getErrorStream ());
         BufferedReader in3 = new BufferedReader (inp3);
         prcs.waitFor ();
         if (in3.readLine () != null){
               System.out.println ("Disc Capacity
service:");
               System.out.println ("Syntax Error :
"+in3.readLine ().toString ());
               return null;

         }else{
         in3 = new BufferedReader (inp2);
         while ( (line = in3.readLine ()) != null) {
```

```
                StringTokenizer data = new StringTokenizer
(line);
                for (int counter = 0; counter < 3;
counter++) {
                      data.nextToken ();
                }
                available[i] = data.nextToken ();
                data.nextToken ();
                mount[i] = data.nextToken ();
                i++;
          }

    Hashtable table = new Hashtable ();
    table.put ("available",available);
    table.put ("mount",mount);
    table.put ("count",new Integer (i));
    return table;
    }//end else
            }
          catch (Throwable t)
           {
        System.out.println ("Disc Capacity service:");
             System.out.println ("Runtime Error :
"+t.getMessage ());
        return null;

      }
    }

    public Hashtable TCPportmap () throws
java.io.IOException {

          String line = "";
          String[] cmd = { "nmap", "-sT","localhost" };
```

```
        Process prcs;
        int i = 0;
        boolean flag = false;
        String[] ports = new String[200];
        String[] proto_names = new String[200];

   try
        {
        prcs = Runtime.getRuntime ().exec (cmd);
        InputStreamReader inp2 = new InputStreamReader
(prcs.getInputStream ());
        InputStreamReader inp3 = new InputStreamReader
(prcs.getErrorStream ());
        BufferedReader in3 = new BufferedReader (inp3);
        prcs.waitFor ();

        if (in3.readLine () != null){
              System.out.println ("TCP portmap service:");
              System.out.println ("Syntax Error :
"+in3.readLine ().toString ());
              return null;

        }else{
        i = 0;
        in3 = new BufferedReader (inp2);
        while ( (line = in3.readLine ()) != null) {

              if (line.contains ("PORT") && line.contains
("STATE")
                        && line.contains ("SERVICE"))
                    flag = true;
              if (line.contains ("finished"))
                    flag = false;
```

```
              if (flag == true) {
                    StringTokenizer data2 = new
StringTokenizer (line);

                    if (data2.hasMoreTokens ()) {
                          ports[i] = data2.nextToken ();
                          data2.nextToken ();
                          proto_names[i] = data2.nextToken
();
                          i++;
                    }
              }
         }//end while

              Hashtable table = new Hashtable ();
              table.put ("ports",ports);
              table.put ("protocols",proto_names);
              table.put ("count",new Integer (i));

              return table;

         }//end else
           }
         catch (Throwable t)
          {
        System.out.println ("TCP portmap service:");
            System.out.println ("Runtime Error :
"+t.getMessage ());
        return null;

     }

   }
```

```
public Hashtable UDPportmap () throws java.io.IOException
{

        String line = "";
        String[] cmd = { "nmap", "-sU","localhost" };
        Process prcs;
        int i = 0;
        boolean flag = false;
        String[] ports = new String[200];
        String[] proto_names = new String[200];


   try
        {
        prcs = Runtime.getRuntime ().exec (cmd);
        InputStreamReader inp2 = new InputStreamReader
(prcs.getInputStream ());
        InputStreamReader inp3 = new InputStreamReader
(prcs.getErrorStream ());
        BufferedReader in3 = new BufferedReader (inp3);
        prcs.waitFor ();

        if (in3.readLine () != null){

              System.out.println ("UDP portpam service:");
              System.out.println ("Syntax Error :
"+in3.readLine ().toString ());
              return null;

        }else{
        i = 0;
        in3 = new BufferedReader (inp2);
        while ( (line = in3.readLine ()) != null) {
```

```
                if (line.contains ("PORT") && line.contains
("STATE")
                              && line.contains ("SERVICE"))
                        flag = true;
                if (line.contains ("finished"))
                        flag = false;

                if (flag == true) {
                        StringTokenizer data2 = new
StringTokenizer (line);

                        if (data2.hasMoreTokens ()) {
                              ports[i] = data2.nextToken ();
                              data2.nextToken ();
                              proto_names[i] = data2.nextToken
();
                              i++;
                        }
                }
          }//end while

                Hashtable table = new Hashtable ();
                table.put ("ports",ports);
                table.put ("protocols",proto_names);
                table.put ("count",new Integer (i));
                return table;

          }//end else
            }
          catch (Throwable t)
           {
        System.out.println ("UDP portpam service:");
             System.out.println ("Runtime Error :
"+t.getMessage ());
```

```
        return null;
      }
   }
   public String Ping (String ip,String packets) throws
java.io.IOException {

   String[] cmd={"ping","-s 1000", "-c", packets, "-q",
ip};
         String line = "";
         String result = "";
         Process prcs;
        try
        {
         prcs = Runtime.getRuntime ().exec (cmd);
         InputStreamReader inp2 = new InputStreamReader
(prcs.getInputStream ());
         InputStreamReader inp3 = new InputStreamReader
(prcs.getErrorStream ());
         BufferedReader in3 = new BufferedReader (inp3);
         prcs.waitFor ();
         if (in3.readLine () != null){
               System.out.println ("Ping service:");
               System.out.println ("Syntax Error :
"+in3.readLine ().toString ());
               return null;

         }else{
         in3 = new BufferedReader (inp2);
         while ( (line = in3.readLine ()) != null) {
         if (line.contains ("packets") && line.contains
("received"))
                     {
                           result=line+" : "+in3.readLine
();
                           return result;
```

```
                    }
                    }
          }
            }
          catch (Throwable t)
           {
        System.out.println ("Ping service:");
            System.out.println ("Runtime Error :
"+t.getMessage ());
        return null;
      }
    return null;
    }
public String BandWidth (String ip) throws
java.io.IOException {

          String[] cmd = { "iperf", "-c",ip,"-p","3000" };
          String line = "";
          Process prcs;
    try
             {
          prcs = Runtime.getRuntime ().exec (cmd);
          InputStreamReader inp2 = new InputStreamReader
(prcs.getInputStream ());
          InputStreamReader inp3 = new InputStreamReader
(prcs.getErrorStream ());
          BufferedReader in3 = new BufferedReader (inp3);
          prcs.waitFor ();
          if (in3.readLine () != null){
                System.out.println ("BandWidth service:");
                System.out.println ("Syntax Error :
"+in3.readLine ().toString ());
                return null;
          }else{
          in3 = new BufferedReader (inp2);
```

```
        while ( (line = in3.readLine ()) != null) {

        if (line.contains ("sec") )

              return line.toString ();
              }
           }
   }
         catch (Throwable t)
          {
       System.out.println ("BandWidth service:");
           System.out.println ("Runtime Error :
"+t.getMessage ());
       return null;

     }
      return null;
   }
public String test (){
         String[] cmd={"ping", "-c", "10", "-q",
"192.168.1.1"};
         String line = "";
         String result = "";
         Process prcs;
        try
        {
         prcs = Runtime.getRuntime ().exec (cmd);
         InputStreamReader inp2 = new InputStreamReader
(prcs.getInputStream ());
         InputStreamReader inp3 = new InputStreamReader
(prcs.getErrorStream ());
         BufferedReader in3 = new BufferedReader (inp3);
         prcs.waitFor ();
         if (in3.readLine () != null){
```

```
                System.out.println ("Test service:");
                System.out.println ("Syntax Error :
"+in3.readLine ().toString ());
                return null;
                }else{
          in3 = new BufferedReader (inp2);
          while ( (line = in3.readLine ()) != null) {
          if (line.contains ("packets") && line.contains
("received"))
                     {
                           result=line+" : "+in3.readLine
();
                           return result;
                     }
                     }
          }
            }
          catch (Throwable t)
           {
             System.out.println ("Runtime Error :
"+t.getMessage ());
        return null;
      }
    return null;
    }


}
```

Admin.java

```
import java.io.File;
import java.sql.*;

public class admin {

public Connection con;

public void add_pc (String hostname,String ip) throws Exception{

      admin db = new admin ();
      String sqlString = "insert into pc_pool values
(null,'"+hostname+"','"+ip+"')";

      try{
      db.myConnect ();
      db.Insert (sqlString);
      db.myDisConnect ();

      }catch (Exception ex){System.out.println ("Add pc error : " +
ex.getMessage () );}

}

public void delete_pc (String id) throws Exception{

      admin db = new admin ();
      String sqlString = "delete from pc_pool where id="+id;

      try{
      db.myConnect ();
      db.Insert (sqlString);
      db.myDisConnect ();

      }catch (Exception ex){System.out.println ("Delete pc error : "
+  ex.getMessage () );}

}

private void myConnect () {
            String databaseDriver = "com.mysql.jdbc.Driver";
            String databaseURL = "jdbc:mysql://localhost:3306/test";

            try {
                  Class.forName (databaseDriver);
            } catch (ClassNotFoundException e) {
                  System.out.println ("e: " + e.getMessage ());
            }

            try {
                  con = DriverManager.getConnection (databaseURL,
"root", "");
            } catch (SQLException ex) {
                  System.out.println ("ex: " + ex.getMessage ());
            }

      }

      private void myDisConnect () {
```

```
            try {
                  con.close ();
            } catch (Exception e) {
                  System.out.println ("e: " + e.getMessage ());
            }

      }
      private boolean Insert (String sqlString) {
            boolean result = false;
            Statement stmt = null;

            try {
                  stmt = con.createStatement ();
                  result = stmt.execute (sqlString);
            } catch (SQLException ex) {
                  System.out.println ("InsertException: " +
ex.getMessage ());
            }
            return result;
      }


}
```

Client Side

Client.java

```
import java.util.HashMap;
import java.util.Hashtable;
import java.io.File;
import javax.xml.rpc.ParameterMode;
import java.sql.*;
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import org.apache.axis.encoding.XMLType;
import java.text.SimpleDateFormat;
import java.util.Date;
import org.apache.soap.rpc.Response;
import org.apache.soap.Fault;
import org.apache.axis.*;
import java.util.StringTokenizer;
import java.util.Timer;
import java.util.TimerTask;


public class Client {


      Client global;
      static int counter_disk=0;
      static int counter_port=0;
      static int counter_bandwidth=0;
      static int counter_list=0;

      Timer timer;
```

```
      public static void main (String[] args) throws Exception {

      Integer parser = 0;
      counter_disk = parser.parseInt (args[0]);
      counter_port = parser.parseInt (args[1]);
      counter_bandwidth = parser.parseInt (args[2]);
      counter_list = parser.parseInt (args[3]);

      Client start =  new Client ();
      start.timer ();

      }

      public void timer (){


            database db = new database ();
            ResultSet result = null;
            db.myConnect ();
            String sqlString = "SELECT id,ip_address,hostname  from
pc_pool";
            result = db.Select (sqlString);
                  try{
                  timer= new Timer ();
                  if (result != null)
                  if (result.next ())     //antikatastasi if me while
gia polla threads
                  {
                  ExecTask Task = new ExecTask ();

                  Task.ip = result.getString ("ip_address");
                  Task.hostname = result.getString ("hostname");
                  Task.pc_id = result.getInt ("id");
                  int time = 10000;
                  timer.schedule (Task,
                              0,        //initial delay
                              time);  //subsequent rate
                  global=this;

                  }
                  }catch (Exception e){}
                  db.myDisConnect ();
      }
//=================================================================
=========================
//=================================================================
=========================
//Task Timer Class
// methods included run () and methodsrun ()
//=================================================================
=========================
//=================================================================
=========================
class ExecTask extends TimerTask {


      public String ip = "";
      public String hostname = "";
      public int pc_id = 0;
```

```
        public void run () {

            try{

            ResultSet result = null;
            database dbc = new database ();
            String[] methods = new String[5];
            int i = 0;
            String sqlString = "";


                  dbc.myConnect ();

                  sqlString = "SELECT
diskcapacity,port_map,list_path,bandwidth,ping_connection  from
scheduler where pc_id="+this.pc_id;
                  result=dbc.Select (sqlString);
                  if ( result.next ())
                  {

                  if (result.getInt ("diskcapacity") == 1)
                  {     methods[i]="d";
                        i++;
                        sqlString = "UPDATE scheduler set
diskcapacity=0 where pc_id="+this.pc_id;
                        dbc.Update (sqlString);

                  }
                  if (result.getInt ("port_map") == 1)
                  {     methods[i]="p";
                        i++;
                        sqlString = "UPDATE scheduler set port_map=0
where pc_id="+this.pc_id;
                        dbc.Update (sqlString);

                  }
                  if (result.getInt ("list_path") == 1)
                  {     methods[i]="l";
                        i++;
                  }
                  if (result.getInt ("bandwidth") == 1)
                  {     methods[i]="b";
                        i++;
                        sqlString = "UPDATE scheduler set bandwidth=0
where pc_id="+this.pc_id;
                        dbc.Update (sqlString);

                  }
                  if (result.getInt ("ping_connection") == 1)
                  {     methods[i]="c";
                        i++;
                  }
                  global.counter_disk--;
                  global.counter_port--;
                  global.counter_bandwidth--;
                  global.counter_list--;
                  System.out.println ();
                  System.out.println ("
====================");
                  System.out.println ("
hostname : "+hostname);
```

```
                  System.out.println ("                              ip
: "+ip);
                  System.out.println ("                              pc
id : "+pc_id);
                  System.out.println ("
====================");
                  System.out.print ("Call methods are : ");
                  for (int j=0;j<i;j++)
                  System.out.print (methods[j]+" ");
                  System.out.println ();

//====================== run () method call ======================
                  if (methods[0] != null)
                        this.runmethods (methods,pc_id,hostname,ip);


//================================================================


                  //scheduling
                  this.schedule (dbc);

                  }

                  dbc.myDisConnect ();

      }catch (Exception e){}

}

public void schedule (database dbc){


String sqlString="";

                  if (global.counter_disk == 0)
                        {
                        global.counter_disk=4;
                        sqlString = "UPDATE scheduler set
diskcapacity=1 where pc_id="+this.pc_id;
                        dbc.Update (sqlString);
                        }

                  if (global.counter_port == 0)
                        {
                        global.counter_port=6;
                        sqlString = "UPDATE scheduler set port_map=1
where pc_id="+this.pc_id;
                        dbc.Update (sqlString);
                        }

                  if (global.counter_bandwidth == 0)
                        {
                        global.counter_bandwidth=2;
                        sqlString = "UPDATE scheduler set bandwidth=1
where pc_id="+this.pc_id;
                        dbc.Update (sqlString);
                        }
                  if (global.counter_list == 0)
                        {
                        global.counter_list=90;
```

```
                        sqlString = "UPDATE scheduler set list_path=1
where pc_id="+this.pc_id;
                        dbc.Update (sqlString);
                        }

                  //scheduling

}
//*****************************************************************
************************
//*****************************************************************
************************
//                              Run Procedure
//*****************************************************************
************************
//*****************************************************************
************************
      public void runmethods (String[] args,int pc_id,String
hostname,String ip) throws Exception {



                  // input Parameters
                  String num_of_packets = "20";
                  String target_ip = "192.168.1.1";

                  HashMap h = new HashMap ();
                  String method = "";
                  int parameter=0;
                  String id = "1";
                  String[] return_value = new String[2];

                  String sqlString ="";
                  ResultSet rs = null;





                  String endpoint =
"http://"+ip+":5555/axis/services/GregService";

                  System.out.println
("=================================================================");
                  System.out.println ("QoS Service Call to :
"+endpoint);
                  System.out.println
("=================================================================");



                  while (args[parameter] != null) {

                        method = args[parameter];
                        parameter++;
                        methods run_method =new  methods ();

                        if (method.equalsIgnoreCase ("d")) {
                        run_method.diskcapacity
(endpoint,hostname,ip);
```

```
                        } else if (method.equalsIgnoreCase ("l")) {

                        run_method.libraries (endpoint,hostname,ip);

//=================================================================
==================
                        } else if (method.equalsIgnoreCase ("b")) {

                        run_method.bandwidth
(endpoint,hostname,pc_id,target_ip);


                        } else if (method.equalsIgnoreCase ("c")) {

                        run_method.ping
(endpoint,hostname,pc_id,target_ip,num_of_packets);
                        } //end if

                        else if (method.equalsIgnoreCase ("p")) {

                        run_method.TCPportscan
(endpoint,hostname,ip);
                        run_method.UDPportscan
(endpoint,hostname,ip);

                        } //end if method "p"


            else if (method.equalsIgnoreCase ("s")){


            }// end if method "s"

                  else
                  System.out.println (method + ": wrong parameter!");



      }

      }//end run



}
//=================================================================
==========================
//=================================================================
==========================
//Task Timer Class End
//=================================================================
==========================
//=================================================================
==========================

}// client class end
```

Methods.java

```
import java.util.HashMap;
import java.util.Hashtable;
import java.io.File;
import javax.xml.rpc.ParameterMode;
import java.sql.*;
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import org.apache.axis.encoding.XMLType;
import org.apache.soap.rpc.Response;
import org.apache.soap.Fault;
import org.apache.axis.*;
import java.util.StringTokenizer;

public class methods{

public void diskcapacity (String endpoint,String hostname, String ip)
throws Exception{

                                String sqlString="";
                                database db = new database ();
                                String[] avail = new String[50];
                                String[] mount = new String[50];
                                HashMap h = new HashMap ();

                                System.out.println ();
                                System.out

                                .println ("========================
DISK-CAPACITY at "+hostname+" ========================");
                                System.out.println ();



                                Service service = new Service ();
                                Call call = (Call) service.createCall
();
                                call.setTargetEndpointAddress (new
java.net.URL (endpoint));
                                call.setOperationName ("DiskCapacity");

                                try{
                                db.myConnect ();
                                h = (HashMap) call.invoke (new Object[]
{});
                                if (h == null)
                                System.out.println ("Web Service call
Error!!!");
                                else{
                                Hashtable table = new Hashtable (h);

                                avail = (String[]) table.get
("available");
                                mount = (String[]) table.get ("mount");
                                Integer ii = (Integer) table.get
("count");


                                int i = ii.intValue ();
                                int space = 0;
```

```
                                for (int j = 1; j < i; j++) {

                                space = ii.parseInt (avail[j]);

                                sqlString = "INSERT INTO diskcapacity
(id,timestamp, hostname, ip,kbfree,mounted_on) "
                                + "VALUES (\""
                                + null
                                + "\","
                                + "NOW ()"
                                + ",\""
                                + hostname
                                + "\", \""
                                + ip
                                + "\", \""
                                + space
                                + "\", \""
                                + mount[j]
                                + "\")";


                                db.Insert (sqlString);

                                      System.out.println ("There are "
+ avail[j]
                                                  + "KB available
mounted on " + mount[j]);
                                }//end for
                           }//end if (for call Error)
                           }catch (AxisFault ex) { System.out.println
("DiskCapacity Call Exception : " +  ex.getFaultString () ); }

}

public void bandwidth (String endpoint,String hostname,int
pc_id,String target_ip)throws Exception{

                                String sqlString="";
                                database db = new database ();
                                String response = "";
                                String bandwidth_server="192.168.1.6";
                                System.out.println ();
                                System.out.println
("========================= BANDWIDTH METER at "+hostname+"
=========================");
                                System.out.println ();
                                try{
                                db.myConnect ();
                                Service service = new Service ();
                                Call call = (Call) service.createCall
();
                                call.setTargetEndpointAddress (new
java.net.URL (endpoint));
                                call.setOperationName ("BandWidth");
                                call.setReturnType ( XMLType.XSD_ANY );
                                call.addParameter ("ip",
XMLType.XSD_STRING,
                                            ParameterMode.IN);
                                response = (String) call.invoke (new
Object[] { bandwidth_server });

                                if (response == null)
```

```
                                    System.out.println ("Web Service
call Error!!");
                              else{

                                    StringTokenizer data = new
StringTokenizer (response);
                                    for (int counter = 0; counter <
6; counter++) {
                                    data.nextToken ();
                              }
                              String bandwidth = data.nextToken ();
                              String speed = data.nextToken ();

                                    System.out.println (bandwidth+"
"+speed);
                              sqlString = "INSERT INTO bandwidth
(id,timestamp,pc_id,target_ip,bandwidth) "
                              + "VALUES (\""
                              + null
                              + "\","
                              + "NOW ()"
                              + ",\""
                              + pc_id
                              + "\", \""
                              + bandwidth_server
                              + "\", \""
                              + bandwidth+speed
                              + "\")";

                              db.Insert (sqlString);

                              }

                              }catch (AxisFault ex) {
System.out.println ("BandWidth Call Exception : " +
ex.getFaultString () ); }

}

public void ping (String endpoint,String hostname,int pc_id,String
target_ip,String num_of_packets) throws Exception {

                              String response = "";
                              String sqlString="";
                              database db = new database ();
                              System.out.println ();
                              System.out.println
("======================== PING CONNECTION at "+hostname+"
========================");
                              System.out.println ();
                              try{
                              db.myConnect ();
                              Service service = new Service ();
                              Call call = (Call) service.createCall
();
                              call.setTargetEndpointAddress (new
java.net.URL (endpoint));
                              call.setOperationName ("Ping");
                              response = (String) call.invoke (new
Object[] { target_ip,num_of_packets });
                              if (response == null)
```

```
                                System.out.println ("Web Service call
Error");
                                else{



                                Integer parser = new Integer (0);
                                String[] subtokens1=new String[20];
                                String[] subtokens2=new String[10];

                                subtokens1 = response.split (",");
                                subtokens2 = response.split (" ");
                                int packets_sent = parser.parseInt
(subtokens2[0]);
                                int packets_received = parser.parseInt
( (subtokens1[1].split (" "))[1]);
                                subtokens1 = ( (response.split
(":")[1]).split ("="));
                                Float avg = new Float (
(subtokens1[1].split ("/"))[1]);

                                double bandwidth =16128/avg;
                                System.out.println ("target ip
:"+target_ip);
                                System.out.println ("packets sent :
"+packets_sent);
                                System.out.println ("packets received :
"+packets_received);
                                System.out.println ("Average time :
"+avg+"ms");
                                System.out.println ("Bandwidth
measured:"+bandwidth+"Mbps");



                                sqlString = "INSERT INTO
ping_connection
(id,timestamp,pc_id,target_ip,packets_sent,packets_received,avg_time_
ms,bandwidth_used_Mbps) "
                                + "VALUES (\""
                                + null
                                + "\","
                                + "NOW ()"
                                + ",\""
                                + pc_id
                                + "\", \""
                                + target_ip
                                + "\", \""
                                + packets_sent
                                + "\", \""
                                + packets_received
                                + "\", \""
                                + avg
                                + "\", \""
                                + bandwidth
                                + "\")";

                                db.Insert (sqlString);

                        }
```

```
                        }catch (AxisFault ex) { System.out.println
("Ping Connection Call Exception : " +  ex.getFaultString () ); }


}


public void TCPportscan (String endpoint,String hostname,String
ip)throws Exception{

                              database db = new database ();
                              HashMap h = new HashMap ();
                              String sqlString="";
                              String[] ports = new String[50];
                              String[] protocols = new String[50];
                              System.out.println ();
                              System.out.println
("========================= TCP PORT SCANING at "+hostname+"
=========================");
                              System.out.println ();
                              try{
                              db.myConnect ();
                              Service service = new Service ();
                              Call call = (Call) service.createCall
();
                              call.setTargetEndpointAddress (new
java.net.URL (endpoint));
                              //-----------tcp scan-----------------
-
                              call.setOperationName ("TCPportmap");

                              h = (HashMap) call.invoke (new Object[]
{});
                              if (h == null)
                              System.out.println ("Web Service call
Error");
                              else{

                              Hashtable table = new Hashtable (h);

                              ports = (String[]) table.get ("ports");
                              protocols = (String[]) table.get
("protocols");
                              Integer ii = (Integer) table.get
("count");

                              int i = ii.intValue ();

                              for (int j = 1; j < i; j++) {
                                    ports[j] = ports[j].substring (0,
ports[j].length () - 4);


                              sqlString = "INSERT INTO port_map
(id,timestamp, hostname, ip,port,protocol,service) "
                              + "VALUES (\""
                              + null
                              + "\","
                              + "NOW ()"
                              + ",\""
                              + hostname
                              + "\", \""
```

```
                              + ip
                              + "\", \""
                              + ports[j]
                              + "\", \""
                              + "TCP"
                              + "\", \""
                              + protocols[j]
                              + "\")";

                                    db.Insert (sqlString);

                                    System.out.println (ports[j] + "
TCP protocol, service : "     + protocols[j]);

                              }// end for
                              }//end if (for call Error)
                        }catch (AxisFault ex) { System.out.println
("Port Scan Call Exception : " +  ex.getFaultString () ); }

}

public void UDPportscan (String endpoint,String hostname,String
ip)throws Exception{

                              database db = new database ();
                              HashMap h = new HashMap ();
                              String sqlString="";
                              String[] ports = new String[50];
                              String[] protocols = new String[50];
                              System.out.println ();
                              System.out.println
("========================= UDP PORT SCANING at "+hostname+"
=========================");
                              System.out.println ();
                              try{
                              db.myConnect ();
                              Service service = new Service ();
                              Call call = (Call) service.createCall
();
                              call.setTargetEndpointAddress (new
java.net.URL (endpoint));
                              call.setOperationName ("UDPportmap");
                              h = (HashMap) call.invoke (new Object[]
{});
                              if (h == null)
                              System.out.println ("Web Service call
Error");
                              else{

                              Hashtable table = new Hashtable (h);
                              ports = (String[]) table.get ("ports");
                              protocols = (String[]) table.get
("protocols");
                              Integer ii = (Integer) table.get
("count");


                              int i = ii.intValue ();

                              for (int j = 1; j < i; j++) {
                                    ports[j] = ports[j].substring (0,
ports[j].length () - 4);
```

```
                              sqlString = "INSERT INTO port_map
(id,timestamp, hostname, ip,port,protocol,service) "
                              + "VALUES (\""
                              + null
                              + "\","
                              + "NOW ()"
                              + ",\""
                              + hostname
                              + "\", \""
                              + ip
                              + "\", \""
                              + ports[j]
                              + "\", \""
                              + "UDP"
                              + "\", \""
                              + protocols[j]
                              + "\")";

                                    db.Insert (sqlString);

                                    System.out.println (ports[j] + "
UDP protocol, service : "     + protocols[j]);

                              }// end for
                         }// end if (for call Error)
                         }catch (AxisFault ex) { System.out.println
("Port Scan Call Exception : " +  ex.getFaultString () ); }

}


public void libraries (String endpoint, String hostname,String
ip)throws Exception{


                              database db = new database ();
                              HashMap h = new HashMap ();
                              String sqlString="";
                              ResultSet rs = null;
                              String[] files = new String[50];
                              String[] paths=new String [100];
                              String[] service_name=new String [100];
                              String[] installed_pc_id=new String
[100];
                              int[] service_id=new int[100];
                              int k=0;
                              int n=0;
                              int i = 0;
                              System.out.println ();
                              System.out.println
("========================= LISTPATH at "+hostname+"
=========================");
                              System.out.println ();
                              db.myConnect ();

                         //          find paths per service

                         sqlString = "select
S.id,lib_paths,servicename,installed_pc_id from services_pool as S,
pc_pool as P where P.ip_address = '"+ip+"' and S.installed_pc_id =
P.id";
                         rs = null;
```

```
                        rs=db.Select (sqlString);
                        if (rs.next ())
                        {
            try {
                  i = 0;


                  do  {

                        service_id[i] = rs.getInt ("id");
                        paths[i] = rs.getString ("lib_paths");
                        service_name[i] = rs.getString
("servicename");
                        installed_pc_id[i] = rs.getString
("installed_pc_id");
                        i++;

                        }while (rs.next ());



            } catch (SQLException ex2) {
                  System.out.println ("ex2 : " + ex2.getMessage ());
            }


//==================find files per path==============================
                                    String[] subpaths=new String[10];

                                    while (paths[k] != null)
                                    {

                                    subpaths=null;
                                    subpaths = paths[k].split (":");

                                    int count =
java.lang.reflect.Array.getLength (subpaths);

                                    while (count > 0)
                                    {
                                    try{
                                    count--;
                                    Service service = new Service ();
                                    Call call = (Call)
service.createCall ();
                                    call.setTargetEndpointAddress
(new java.net.URL (endpoint));
                                    call.setOperationName
("listPath");
                                    call.setReturnType (
XMLType.XSD_ANY );
                                    call.addParameter ("pathString",
XMLType.XSD_STRING,ParameterMode.IN);
                                    h = (HashMap) call.invoke (new
Object[] { subpaths[n] });
                                    if (h == null)
                                    System.out.println ("Web Service
call Error!!!");
                                    else{
                                    Hashtable table = new Hashtable
(h);
```

```
                                    files = (String[]) table.get
("files");
                                    Integer ii = (Integer) table.get
("count");
                                    String file="";
                                    i = ii.intValue ();
                                    System.out.println ();
                                    System.out.println ("In the path
: '"+subpaths[n]+"' of service : '"+service_name[k]+"' there were
:");
                                    System.out.println ();

                                    for (int j = 0; j < i; j++) {
                                          rs=null;
                                          file=files[j].replaceAll
(subpaths[n]+"/","");
                                          System.out.println (file);

sqlString = "select id from service_libraries where filename =
'"+file+"' and path= '"+subpaths[n]+"' and service_id =
'"+service_id[k]+"' and pc_id  = '"+installed_pc_id[k]+"'";

rs=db.Select (sqlString);
if (!rs.next ()){
//---------------------------------------insert ------------------
---------------------------------

            sqlString = "INSERT INTO service_libraries (id,timestamp,
service_id,pc_id, filename,path) "
                              + "VALUES (\""
                              + null
                              + "\","
                              + "NOW ()"
                              + ",\""
                              + service_id[k]
                              + "\", \""
                              + installed_pc_id[k]
                              + "\", \""
                              + file
                              + "\", \""
                              + subpaths[n]
                              + "\")";

                              db.Insert (sqlString);
}//endif
else{



            sqlString = "update service_libraries set timestamp = NOW
() where filename = '"+file+"' and path= '"+subpaths[n]+"' and
service_id = '"+service_id[k]+"' and pc_id  =
'"+installed_pc_id[k]+"'";
            db.Update (sqlString);


}//endelse
}
                  n++;

}//end if (for call Error)
```

```
}catch (AxisFault ex) { System.out.println ("ListPath Call Exception
: " +  ex.getFaultString () ); }
}

                  k++;
                  n=0;

}//end while
}else {System.out.println ("There are no installed service to pc :
"+hostname);}
}

}
```

Database.java

```
import java.io.File;
import java.sql.*;

public class database {
public Connection con;
public void myConnect () {
            String databaseDriver = "com.mysql.jdbc.Driver";
            String databaseURL = "jdbc:mysql://localhost:3306/test";

            try {
                  Class.forName (databaseDriver);
            } catch (ClassNotFoundException e) {
                  System.out.println ("e: " + e.getMessage ());
            }

            try {
                  con = java.sql.DriverManager.getConnection
(databaseURL, "root", "");
            } catch (SQLException ex) {
                  System.out.println ("ex: " + ex.getMessage ());
            }

      }

      public void myDisConnect () {

            try {
                  con.close ();
            } catch (Exception e) {
                  System.out.println ("e: " + e.getMessage ());
            }

      }

      public boolean Insert (String sqlString) {
            boolean result = false;
            Statement stmt = null;

            try {
                  stmt = con.createStatement ();
                  result = stmt.execute (sqlString);
            } catch (SQLException ex) {
```

```
                  System.out.println ("InsertException: " +
ex.getMessage ());
            }
            return result;
      }

      public ResultSet Select (String sqlString) {


            ResultSet rs = null;

            try {
                  Statement stmt = con.createStatement ();
                  rs = stmt.executeQuery (sqlString);
            } catch (SQLException ex) {
                  System.out.println ("SelectException: " +
ex.getMessage ());
            }

                  return rs;

      }
      public boolean Update (String sqlString) {
            boolean result = false;
            Statement stmt = null;

            try {
                  stmt = con.createStatement ();
                  result = stmt.execute (sqlString);
            } catch (SQLException ex) {
                  System.out.println ("UpdateException: " +
ex.getMessage ());
            }
            return result;
}
}
```

Validate.jsp

```
<html>
<body>
<head>
<title></title>
</head>
<body BGCOLOR=#818080>


  <%@ page
import="availabilitylib.*,org.apache.axis.client.*,org.apache.axis.Ax
isFault,java.sql.*"%>

<%

      String username = "";
      String password = "";
      database db = new database ();
      String sqlString = "";
      ResultSet rs;
```

```
      if (request != null)
      {
      username = request.getParameter ("userName");
      password = request.getParameter ("password");
      db.myConnect ();
      sqlString = "select password from users where
username=\'"+username+"\'";
      rs = db.Select (sqlString);
      if (rs.next ())
            if (password.equals (rs.getString ("password")))
                  {
                  %>
                  <h3>Welcome to QoS service administration
page!</h3>
                  <br><br><br><h5>
                  <h3>Resource management</h3>
<P>


<div align='center'>
            <br> <br><br><br>Resource table<br>
            <table border='1'>
            <tr align='center'>
                  <td width='30'> <strong> id  </strong></td>
                  <td width='100'><strong>  hostname  </strong></td>
                  <td width='80'> <strong> ip  </strong></td>
            </tr>

<%
      String
endpoint="http://localhost:8080/axis/services/availability_admin";
      String add_hostname = "";
      String ip_address = "";
      String id = "";
      String del_hostname = "";
      String method = "";


      sqlString = "select * from pc_pool";

      rs = db.Select (sqlString);

      while (rs.next ())
      {
out.write ("      <tr align=\"center\"><td>"+rs.getInt
("id")+"</td><td>"+rs.getString ("hostname")+"</td><td>"+rs.getString
("ip_address")+"</td></tr>");
      }
out.write ("</table></div>");

      db.myDisConnect ();




%>

<br>
<h4>Complete the Hostname and the IP address of the resource you want
to add :</h4>
```

```
<form action="methods.jsp" target="_blank" method=POST>
Hostname : <input type=text size=20 name=hostname> IP address :
<input type=text size=20 name=ip>
<p>
<input type=hidden size=20 name=method value="add">
<input type=submit value="Add">
</form>

<br>
<br>
<h4>Complete the Id or the Hostname of the resource you want to
remove form the pool.</h4>
<form action="methods.jsp"  target="_blank" method=POST>
ID :
 <input type=text size=20 name=id> or Hostname : <input type=text
size=20 name=del_hostname>
<input type=hidden size=20 name=method value="remove">
<p>
<input type=submit value="Remove">
</form>

<br><br><br><br>
<h3>
                  Press<a href = "./Client.jsp"> here </a> to execute
QoS service.<br>
                  Press<a href =
"http://gregorious.homeip.net/availability/"> here </a> to view the
QoS result tables.<br></h3>
                  <%
                  }
            else
                  out.write ("<h3>Incorect password!<br><a href =
\"login.html\">  Back to Try again!</a></h3>");

      }
%>


</body>
</html>
```

Client.jsp

```
<html>
<body BGCOLOR=#808080>
<%


      String sqlString = "SELECT id,ip_address,hostname  from
pc_pool";
      java.sql.ResultSet result =null;


      String[] ip=new String[100];
      String[] hostname=new String[100];
      int[] pc_id=new int[100];
      int j=0;
      int i=0;
```

```
      String databaseDriver = "com.mysql.jdbc.Driver";
      String databaseURL = "jdbc:mysql://localhost:3306/test";
      java.sql.Connection con ;
      try {
                  Class.forName (databaseDriver);
            } catch (ClassNotFoundException e) {
                  System.out.println ("e: " + e.getMessage ());
            }

            try {
                  con = java.sql.DriverManager.getConnection
(databaseURL, "root", "");
                  java.sql.Statement stmt = con.createStatement ();
                  result = stmt.executeQuery (sqlString);

                  while (result.next ())
                  {ip[j] = result.getString ("ip_address");
                   hostname[j] = result.getString ("hostname");
                   pc_id[j] = result.getInt ("id");
                  j++;
                  }

            } catch (java.sql.SQLException ex) {
                  System.out.println ("ex: " + ex.getMessage ());
            }


%>
<br>
<h3>Available pc in the Grid.</h3>
<br>
<FORM ACTION="" method="post">
<% for (i=0;i<j;i++)
{
out.write ("<INPUT TYPE=CHECKBOX NAME=pc value=\""+hostname[i]+"\">
<strong>"+hostname[i]+"</strong> / ip address : "+ip[i]+"<P>");
}
%>
<INPUT TYPE=SUBMIT VALUE="submit">
</FORM>
<br>
<br>

<%

String[] pc_pool = new String[100];
String service_ip="";
pc_pool = request.getParameterValues ("pc");
      if (pc_pool != null)
            if (pc_pool.length > 1)
                  out.write ("Please choose only one!!");
            else
            {
for (i=0;i<j;i++)
{
       if (pc_pool[0].equals (hostname[i]))
            {
            out.write ("<h4>You chose to run the QoS service to :
<I>"+hostname[i]+"</I> with ip address : <I>"+ip[i]+"</I></h4>");
            service_ip=ip[i];
            }
}
```

```
%>
<FORM ACTION="" METHOD="post">
<INPUT TYPE=SUBMIT VALUE="RESET">
</FORM>

<br>
<FORM ACTION="results.jsp" METHOD="post">

<INPUT TYPE=CHECKBOX NAME="diskcapacity">diskcapacity<BR>
<INPUT TYPE=CHECKBOX NAME="bandwidth">bandwidth<BR>
<INPUT TYPE=CHECKBOX NAME="ping">ping<BR>
<INPUT TYPE=CHECKBOX NAME="TCPportscan">TCPportscan<BR>
<INPUT TYPE=CHECKBOX NAME="UDPportscan">UDPportscan<BR>
<INPUT TYPE=CHECKBOX NAME="libraries">libraries<BR>
<INPUT TYPE=CHECKBOX NAME="service">service --   URL : <INPUT
TYPE=TEXT NAME="service_URL">  Method: <INPUT TYPE=TEXT
NAME="service_method">
<% out.write ("<INPUT TYPE=HIDDEN NAME=\"IP\"
VALUE=\""+service_ip+"\"><P>"); %>
<INPUT TYPE=SUBMIT VALUE="RUN" >
</FORM>
<br>
<br>

<%}%>
<a href = "javascript:history.back ()"> Back to administartion page
</a>
</body>
</html>
```

Results.jsp

```
<html>
<body BGCOLOR=#818080>

      <%


      String diskcapacity = "";
      String bandwidth = "";
      String ping = "";
      String TCP = "";
      String UDP = "";
      String lib = "";
      String service_ip="";
      String service = "";
      String service_URL = "";
      String service_method = "";

      String hostname = "hal4000";
      String ip = "192.168.1.6";
      int pc_id = 1;
      String target_ip = "192.168.1.1";
      String iperf_ip = "192.168.1.6";
      String num_of_packets = "30";
      boolean[] parameter=new boolean[7];
      parameter[0]=false; //diskcapacity
```

```
      parameter[1]=false; //bandwidth
      parameter[2]=false; //ping
      parameter[3]=false; //TCP scan
      parameter[4]=false; //UDP scan
      parameter[5]=false; //library files
      parameter[6]=false; //service


if (request != null)
{
      service_ip = request.getParameter ("IP");
      diskcapacity = request.getParameter ("diskcapacity");
      bandwidth = request.getParameter ("bandwidth");
      ping = request.getParameter ("ping");
      TCP = request.getParameter ("TCPportscan");
      UDP = request.getParameter ("UDPportscan");
      lib = request.getParameter ("libraries");
      service = request.getParameter ("service");
      service_URL = request.getParameter ("service_URL");
      service_method = request.getParameter ("service_method");

if (diskcapacity != null && diskcapacity.equalsIgnoreCase ("on"))
      parameter[0]=true;

if (bandwidth != null &&  bandwidth.equalsIgnoreCase ("on"))
      parameter[1]=true;

if (ping != null && ping.equalsIgnoreCase ("on"))
      parameter[2]=true;

if (TCP != null && TCP.equalsIgnoreCase ("on"))
      parameter[3]=true;

if (UDP != null && UDP.equalsIgnoreCase ("on"))
      parameter[4]=true;

if (lib != null && lib.equalsIgnoreCase ("on"))
      parameter[5]=true;

if (service != null && service.equalsIgnoreCase ("on"))
      parameter[6]=true;

if ( (parameter[0] == true) || (parameter[1] == true) ||
(parameter[2] == true) || (parameter[3] == true) || (parameter[4] ==
true) || (parameter[5] == true) || (parameter[6] == true))
{
      out.write ("<h4>Welcome to QoS service!<p>You demanded service
call at : <I>"+service_ip+"</I></h4><br><h5>The requested QOS
parameters are : </h5><FONT COLOR=#FF0000>");

      if (parameter[0] == true) //diskcapacity
            {
            out.write ("<I> diskcapacity </I><br>");
            }
      if (parameter[1] == true) //bandwidth
            {
            out.write ("<I> bandwidth </I><br>");
            }
      if (parameter[2] == true) //ping
            {
            out.write ("<I> ping request </I><br>");
            }
```

```
      if (parameter[3] == true) //TCP scan
            {
            out.write ("<I> TCP ports </I><br>");
            }
      if (parameter[4] == true) //UDP scan
            {
            out.write ("<I> UDP ports </I><br>");
            }
      if (parameter[5] == true) //libraries
            {
            out.write ("<I> library files </I><br>");
            }
      if (parameter[6] == true) //service
            {
            out.write ("<I> service </I><br>");
            }

out.write ("</FONT>");
}
else
out.write ("<h3>Service requirements not set!</h3><br>");


%>
<%-- Print out the variables. --%>



<%          int i = 0;

            availabilitylib.methods methods = new
availabilitylib.methods ();
            String endpoint =
"http://"+service_ip+":8080/axis/services/GregService";


if ( (parameter[0] == true) || (parameter[1] == true) ||
(parameter[2] == true) || (parameter[3] == true) || (parameter[4] ==
true) || (parameter[5] == true) || (parameter[6] == true))
{
      if (parameter[0] == true) //diskcapacity
            {
            out.write (methods.diskcapacity (endpoint,hostname,ip));
            }
      if (parameter[1] == true) //bandwidth
            {
            out.write (methods.bandwidth
(endpoint,hostname,pc_id,iperf_ip));
            }
      if (parameter[2] == true) //ping
            {
            out.write (methods.ping
(endpoint,hostname,pc_id,target_ip,num_of_packets));
            }
      if (parameter[3] == true) //TCP scan
            {
            out.write (methods.TCPportscan (endpoint,hostname,ip));

            }
      if (parameter[4] == true) //UDP scan
            {
```

```
            out.write (methods.UDPportscan (endpoint,hostname,ip));

            }
      if (parameter[5] == true) //libraries
            {
            out.write (methods.libraries (endpoint,hostname,ip));
            }
      if (parameter[6] == true) //service
            {
            out.write (methods.serviceCall
(endpoint,service_URL,service_method,hostname));
            }
}
%>

<br>
<a href = "javascript:history.back ()">  Run another service
parameter!  </a>
<% } %>
<br>

<a href = "Client.jsp">  Back to Start Page!  </a>
</body>
</html>
```

Methods.jsp

```
<html>
<body>
<head>
<title></title>
</head>
<body bgcolor="red">


 <%@ page
import="availabilitylib.*,org.apache.axis.client.*,org.apache.axis.Ax
isFault,java.sql.*"%>

<%
      String
endpoint="http://localhost:8080/axis/services/availability_admin";
      String add_hostname = "";
      String ip_address = "";
      String id = "";
      String del_hostname = "";
      String method = "";



      if (request != null)
      {
      add_hostname = request.getParameter ("hostname");
      ip_address = request.getParameter ("ip");
      del_hostname = request.getParameter ("del_hostname");
      id = request.getParameter ("id");
      method = request.getParameter ("method");
```

```
      if ( (method != null) && ( method.equals ("add")))
      {
      if ( (add_hostname != null) && (ip_address != null))
      try{
            Service service = new Service ();
            Call call = (Call) service.createCall ();
            call.setTargetEndpointAddress (new java.net.URL
(endpoint));
            call.setOperationName ("add_pc");
            call.invoke (new Object[] {add_hostname, ip_address});
            out.write ("<h3>Pc adding completed!!</h3>");
            out.write ("<script
language=\"JavaScript\">window.opener.location.reload ();self.close
();</script>");

            }catch (AxisFault ex){}

      }else
      if ( (method != null) && ( method.equals ("remove")))
      {
      if (! (del_hostname.equals ("")) || ! (id.equals ("")))
      try{
            Service service = new Service ();
            Call call = (Call) service.createCall ();
            call.setTargetEndpointAddress (new java.net.URL
(endpoint));
            call.setOperationName ("delete_pc");

            if (del_hostname.equals (""))
            call.invoke (new Object[] {id});
            else
            {

            String sqlString = "select id from pc_pool where
hostname=\'"+del_hostname+"\'";
            database db = new database ();
            db.myConnect ();
            ResultSet rs = db.Select (sqlString);
            if (rs.next ())
                  {
                  id=rs.getString ("id");
                  call.invoke (new Object[] {id});
                  }



            }

            out.write ("<script
language=\"JavaScript\">window.opener.location.reload ();self.close
();</script>");

            }catch (AxisFault ex){}
      }
      }
%>


</body>
</html>
```