ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Προσαρμογή συστημάτων κινητών επικοινωνιών για λειτουργία σε περιβάλλον υπολογιστικού πλέγματος

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΤΣΟΓΚΑ ΠΑΝΑΓΙΩΤΗ

Επιβλέπουσα : Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2009

Η σελίδα αυτή είναι σκόπιμα λευκή.

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

Προσαρμογή συστημάτων κινητών επικοινωνιών για λειτουργία σε περιβάλλον υπολογιστικού πλέγματος

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΤΣΟΓΚΑ ΠΑΝΑΓΙΩΤΗ

Επιβλέπουσα : Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 10η Φεβρουαρίου 2009.

Θεοδώρα Βαρβαρίγου Καθηγήτρια Ε.Μ.Π.	Ελευθέριος Καγιάφας Καθηγητής Ε.Μ.Π.	Μιλτιάδης Αναγνώστου Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2009

(Υπογραφή)

.....

ΠΑΝΑΓΙΩΤΗΣ ΤΣΟΓΚΑΣ

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

Περίληψη

Οι κινητές επικοινωνίες αποτελούν ένα από τους πλέον σημαντικούς τεχνολογικούς κλάδους. Η ραγδαία ανάπτυξη των τελευταίων χρόνων και η μεγάλη έκταση εφαρμογών δημιουργούν την ανάγκη για περαιτέρω αξιοποίηση των αναδυόμενων δυνατοτήτων. Μία από αυτές είναι η εκμετάλλευση των υπολογιστικών πλεγμάτων.

Με την αύξηση των φορητών συσκευών οποιασδήποτε μορφής (κινητά τηλέφωνα, PDA, φορητοί υπολογιστές συνδεδεμένοι σε κάποιο ασύρματο τοπικό δίκτυο), εμφανίζεται η ευκαιρία της επέκτασης του Υπολογιστικού Πλέγματος και σε αυτές τις συσκευές.

Σκοπός της εργασίας αυτής είναι η προσαρμογή συστημάτων κινητών επικοινωνιών για λειτουργία σε περιβάλλον Υπολογιστικού Πλέγματος. Συγκεκριμένα προσομοιώνουμε το περιβάλλον πλέγματος χρησιμοποιώντας το μεσολογισμικό GRIA και ενσωματώνουμε σε αυτό κινητές συσκευές βασισμένες στο λειτουργικό σύστημα Android.

Μέσω εφαρμογών του Android αναπτύσσουμε έναν πελάτη για το πλέγμα που παρέχει στον χρήστη την δυνατότητα αλληλεπίδρασης με το Grid και μάλιστα επιτυγχάνει την ιδία λειτουργικότητα με αυτήν ενός Η/Υ. Ο χρήστης εκμεταλλευόμενος τις ιδιότητες του πλέγματος μπορεί να εκτελεί με παράλληλο τρόπο υπολογιστικά απαιτητικές εργασίες απομακρυσμένα και αποδοτικά, όπως για παράδειγμα να διαχειριστεί κάθε είδους αρχεία να τα υποβάλει στο πλέγμα και να αποκτήσει τα αποτελέσματα της επεξεργασίας, όλα από την κινητή του συσκευή. Υποστηρίζεται μάλιστα η δυνατότητα υποβολής πολλαπλών εργασιών, η επίβλεψη τους αλλά και η διακοπή τους όπως ακριβώς και σε έναν τυπικό πελάτη του πλέγματος.

Λέξεις Κλειδιά: κινητές επικοινωνίες, υπολογιστικό πλέγμα, υπηρεσία, GRIA, android

Η σελίδα αυτή είναι σκόπιμα λευκή.

Abstract

Mobile communications constitute one of the most important technological sectors. The rapid growth and the vast field of applications, create the need for further exploitation of the emerging possibilities. One of them is the utilization of computing grids.

With the increase of portable devices of any form (mobile telephones, PDA, portable computers connected in some wireless local area network), emerges the opportunity to extend the Grid in these devices.

The scope of this thesis was the adaptation of mobile communication systems for operation in a grid environment. In particular we emulate the Grid environment using the GRIA middleware and incorporate, mobile devices running on the Android operating system.

By developing Android applications for the Grid we provide the user the same interaction and functionality as with a typical grid client in a personal computer. The user can utilize the properties of the grid and therefore execute in a parallel manner multiple computational-intensive tasks remotely and efficiently, such as submitting a file to the grid and receiving the results, all through his mobile device. Multiple job submission, monitoring and termination functions are also supported.

Keywords: Mobile communication systems, Grid, Service, GRIA, Android

Η σελίδα αυτή είναι σκόπιμα λευκή.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω την κα Βαρβαρίγου Θεοδώρα, Καθηγήτρια της σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Ηλεκτρονικών Υπολογιστών του ΕΜΠ για την ευκαιρία που μου έδωσε να συνεργαστώ μαζί της, την προθυμία της να παρέχει λύσεις σε ζητήματα που παρουσιάστηκαν και την εμπιστοσύνη που μου έδειξε για τη λήψη αποφάσεων και πρωτοβουλιών. Η διδασκαλία, η καθοδήγηση, η εμπειρία και οι γνώσεις της ήταν άκρως απαραίτητα για τη διεξαγωγή και ολοκλήρωση της διπλωματικής εργασίας.

Επίσης, ευχαριστώ τον υποψήφιο διδάκτορα κ. Μενύχτα Ανδρέα για την άψογη συνεργασία του. Η πολύτιμη καθοδήγηση και συμβολή του, υπήρξαν καθοριστικής σημασίας.

Τέλος ευχαριστώ τους γονείς μου για την στήριξη τους όλα αυτά τα χρόνια.

Ευρετήριο εικόνων

Ευρετήριο πινάκων

Πίνακας περιεχομένων

1

Εισαγωγή

1.1 *Γενικά για τα υπολογιστικά πλέγματα*

Η μεγάλη ανάπτυξη των δικτύων Internet, η τεχνολογική εξέλιξη των προσωπικών υπολογιστών (PC) και η ανάπτυξη του κατάλληλου ενδιάμεσου λογισμικού (*middleware*) και εφαρμογών έχει δημιουργήσει μια νέα δυναμική στην κλασσική έννοια του όρου "υπολογιστικό περιβάλλον". Ο συνδυασμός των παραπάνω δίνει τη δυνατότητα του κατανεμημένου γεωγραφικά διαμοιρασμού πόρων όπως η υπολογιστική ισχύς, ο αποθηκευτικός χώρος, το ψηφιακό περιεχόμενο και άλλα επιστημονικά όργανα (π.χ. αισθητήρες). Ουσιαστικά κάποιος που βρίσκεται συνεχώς συνδεδεμένος σε δίκτυο υψηλών ταχυτήτων, με τη χρήση του κατάλληλου λογισμικού μπορεί να μοιράζεται την υπολογιστική ισχύ των υπολογιστών του, τον αποθηκευτικό του χώρο και τους άλλους πόρους του με χιλιάδες άλλους στον κόσμο. Ο διαμοιρασμός αυτός μπορεί να γίνει με ομοιόμορφο, ασφαλή και κατανεμημένο τρόπο σε παγκόσμιο επίπεδο. Το Grid είναι γενικά ένας πρόσθετος τύπος παράλληλης επεξεργασίας, ο οποίος στηρίζεται σε πλήρεις υπολογιστές (με ΚΜΕ ,αποθηκευτικά μέσα , την παροχή ηλεκτρικού ρεύματος, τη διεπαφή δικτύων κ.λπ.) που συνδέονται με ένα δίκτυο (ιδιωτικό, δημόσιο ή το διαδίκτυο) με μια συμβατική διεπαφή δικτύων, όπως το Ethernet. Έτσι το υπολογιστικό πλέγμα δίνει την αίσθηση ότι είναι ένας υπέρ-υπολογιστής. Αυτό είναι σε αντίθεση με την παραδοσιακή έννοια ενός υπέρ-υπολογιστή ο οποίος συνδέει πολλούς

επεξεργαστές με ένα τοπικό διάδρομο υψηλής ταχύτητας. Έτσι ο χρήστης υποβάλλει εργασίες για εκτέλεση χωρίς να τον ενδιαφέρει που θα εκτελεστούν Το υπολογιστικό πλέγμα είναι μια αναπτυσσόμενη υποδομή που χρησιμοποιείται όπου υπάρχουν απαιτήσεις για μεγάλη υπολογιστική ισχύ ή για επεξεργασία πολύ μεγάλου όγκου δεδομένων. Τα τελευταία χρόνια το πλέγμα έχει ραγδαία ανάπτυξη. Σε αυτό έχουν βοηθήσει οι νέες τεχνικές που χρησιμοποιούνται στο διαδίκτυο, οι οπτικές ίνες και η βελτίωση των ασύρματων ζεύξεων που συμβάλλουν στην αύξηση της ταχύτητας των δικτύων υπολογιστών. Δεν υπάρχει αμφιβολία πώς η πιο άμεση εφαρμογή του Grid αφορά τον επαγγελματικό χώρο.

Οι προσωπικοί υπολογιστές, που λειτουργούν πάντα χαμηλότερα από τις δυνατότητές τους, είναι οι κυριότεροι υπολογιστικοί πόροι που μπορούν να χρησιμοποιηθούν σε ένα εταιρικό δίκτυο για την δημιουργία ενός Grid . Υπάρχει όμως και ο προσανατολισμός της δημιουργίας εταιριών που θα παρέχουν υπηρεσίες υψηλών υπολογιστικών απαιτήσεων, που θα βασίζονται σε τεχνολογίες Grid. Οι εταιρίες αυτές θα καλύπτουν τις ανάγκες οποιονδήποτε οργανισμών οι οποίοι θέλουν να εκτελέσουν τις εφαρμογές τους και να πάρουν τα αποτελέσματά του, με τρόπο οικονομικό, γρήγορο, ασφαλή και αποδοτικό. Διαπιστώνοντας λοιπόν τις ανάγκες των εταιριών σε ισχύ αλλά και των υπαρχόντων τεχνολογικών υποδομών, η δυνατότητα παροχής υπολογιστικής ισχύος καθίσταται αναγκαία αλλά και παράλληλα εφικτή. Είναι προφανές ότι για να λειτουργήσει ένα ουσιώδες Grid είτε σε επίπεδο εταιρίας, που θα το χρησιμοποιεί για τους δικούς της σκοπούς είτε σαν ανεξάρτητος πάροχος υπολογιστικών υπηρεσιών χρειάζεται να πληροί κάποιες προϋποθέσεις. Αρχικά πρέπει να υπάρχει η απαραίτητη προτυποποίηση.

Για να είναι εφικτό το Grid να φτάσει σε κάθε πελάτη και να χρησιμοποιηθεί ευρέως είναι υποχρεωτικό να ακολουθήσει τα πρότυπα ενός ανώτερου οργανισμού. Ο οργανισμός που έχει αναλάβει την εποπτεία και την συνεργασία όλων την ερευνητικών εγχειρημάτων σχετικά με το Grid είναι το GGF(Global Grid Forum). Το εκάστοτε πλέγμα και ειδικά αυτά που προορίζονται για επαγγελματικές εφαρμογές πρέπει να υιοθετούν τα υπάρχοντα πρότυπα και να συνεισφέρουν στην δημιουργία νέων. Αυτός είναι ο μόνος τρόπος να εξασφαλιστεί συμβατότητα ανάμεσα στους οργανισμούς και μεταξύ των εφαρμογών και υλοποιήσεων Grid. Παράλληλα με αυτόν τον τρόπο είναι εύκολο να υιοθετηθούν και νέες τεχνολογικές εξελίξεις που σίγουρα θα πραγματοποιηθούν τα επόμενα χρόνια. Ακόμη η εξέλιξη του Grid έχει προσανατολιστεί στην ενσωμάτωση των νέων τεχνολογιών για μέγιστη χρησιμοποίηση όλων των δυνατών υπολογιστικών πόρων. Η διαδικασία αυτή πρέπει να δημιουργήσει ένα νέο περιβάλλον που θα έχει παράλληλα δυνατότητες διαχείρισης από τους υπευθύνους της εταιρίας αλλά και δυνατότητες χρέωσης των πελατών. Στις μέρες μας οι συναλλαγές μεταξύ των εταιριών γίνονται αποκλειστικά με ηλεκτρονικά μέσα κάτι που φυσικά πρέπει να ενσωματώνει

και το περιβάλλον του επαγγελματικού Grid. Πολύ σημαντικός είναι και ο τομέας της ασφάλειας. Σήμερα έχουμε διαπιστώσει ότι κάθε σύστημα που χρησιμοποιεί το διαδίκτυο είναι ευάλωτο σε επιθέσεις που γίνονται για κερδοσκοπικούς ή όχι σκοπούς. Σε ένα ευαίσθητο σύστημα πλέγματος όπου διακινούνται σημαντικά δεδομένα, όπως για παράδειγμα ανταγωνιστικές εταιρικές εφαρμογές, η ασφάλεια είναι πολύ σημαντικός παράγοντας. Αρχικά είναι υψίστης σημασίας η ταυτοποίηση των χρηστών.

Κάθε εταιρία που παρέχει υπολογιστικές υπηρεσίες πρέπει να έχει μηχανισμούς μέσα από τους οποίους να επιτρέπει μόνο σε εξουσιοδοτημένους – πιστοποιημένους χρήστες να έχουν πρόσβαση και ταυτόχρονα να αποκλείει την είσοδο σε όλους τους άλλους. Ακόμη πρέπει να εξασφαλίζει το απόρρητο ανάμεσα στους χρήστες που καταχωρούν δουλειές και τον απόλυτο διαχωρισμό των δεδομένων. Σε κάθε διαφορετική περίπτωση θα αμφισβητηθεί η τεχνολογία και οι υπηρεσίες του πλέγματος που παρέχονται και τελικά θα απορριφθούν από τον επαγγελματικό χώρο. Τέλος έχει πολύ μεγάλη σημασία και ο τομέας της χρηστικότητας. Από την μέχρι τώρα εμπειρία σε τεχνολογίες Grid έχουμε διαπιστώσει πως οι εφαρμογές που υπάρχουν μέχρι σήμερα είναι αρκετά δύσχρηστες και συνήθως παρουσιάζουν πολλά προβλήματα διότι βρίσκονται ακόμη σε φάση εξέλιξης. Για να μπορεί να λειτουργήσει ένα εταιρικό πλέγμα θα πρέπει να είναι σταθερό, χωρίς προβλήματα που να προϋποθέτουν την απασχόληση ενός ειδικού σε θέματα Grid. Ακόμα θα πρέπει να περιλαμβάνει ένα εύχρηστο γραφικό περιβάλλον. Ο εκάστοτε πελάτης δεν θα πρέπει να έχει καμία υποχρέωση γνώσεων περί πλέγματος. Γνωρίζει μόνο τις ανάγκες και τις απαιτήσεις που έχει η δουλειά του και μέσα από αυτό το περιβάλλον πρέπει να μπορεί να καταχωρεί τις δουλειές του με ευκολία ανάλογη των ηλεκτρονικών αγορών μέσω Internet.

<u>Κατηγορίες υπολογιστικών πλεγμάτων</u>

- ***Υπολογιστικά Grids (Computational Grids):*** Συλλογή κατανεμημένων υπολογιστικών υποδομών οι οποίες λειτουργούν ως ενιαίος επεξεργαστής και πραγματοποιούν επεξεργασία δεδομένων με μεγάλες υπολογιστικές απαιτήσεις. Τα αποτελέσματα είναι ταχύτερα, αποτελεσματικότερα και με μικρότερο κόστος.
- ***Grids Δεδομένων (Data Grids):*** Οι χρήστες και οι εφαρμογές διαχειρίζονται πληροφορίες από βάσεις δεδομένων που βρίσκονται σε κατανεμημένες πλατφόρμες ευκολότερα και αποτελεσματικότερα. Επίσης το κόστος είναι μικρότερο γιατί δεν υπάρχει ανάγκη για μεταφορά, αντιγραφή και συγκέντρωση δεδομένων σε ένα κεντρικό σημείο, καθώς και μεγαλύτερη αξιοπιστία κατά την πρόσβαση στα δεδομένα.
- ***Grids Υπηρεσιών (Service Grids):*** Πραγματοποίηση επεξεργασίας πραγματικού Χρόνου.

1.2 *Χρήση υπολογιστικών πλεγμάτων*

Η τεχνολογία των πλεγμάτων, χρησιμοποιείται εδώ και χρόνια κυρίως στο τομέα της επιστημονικής έρευνας, βοηθώντας στη επίλυση πολύπλοκων και σημαντικών για την ανθρωπότητα προβλημάτων. Σήμερα η συγκεκριμένη τεχνολογία εφαρμόζεται και γίνεται ολοένα και πιο διαδεδομένη στο χώρο των σύγχρονων επιχειρήσεων και του e-business architecture. Το σημερινό ανταγωνιστικό επιχειρησιακό περιβάλλον απαιτεί συνεχή εγρήγορση με σκοπό τη αποτελεσματική διαφοροποίηση προϊόντων και υπηρεσιών. Οι επιχειρήσεις πρέπει να προσαρμόζονται διαρκώς και αποτελεσματικά στις απαιτήσεις της αγοράς και των πελατών τους.

Αυτή η ταχύτητα στην εξέλιξη αλλά και η δυσκολία πρόβλεψης των αλλαγών στις επιχειρηματικές συνθήκες, οδηγούν πολλές επιχειρήσεις στα όρια της διαχειριστικής ικανότητας. Στη προσπάθεια των επιχειρήσεων να αντεπεξέλθουν, πολύ συχνά τα συστήματα πληροφοριών τους είναι αργά για να ανταποκριθούν. Επιπλέον, όλες οι επιχειρήσεις επιδιώκουν τη μέγιστη αποδοτικότητα από τα συστήματα τους με το χαμηλότερο δυνατόν κόστος. Αυτό το πρόβλημα έρχεται να αντιμετωπίσει η τεχνολογία του Grid Computing. Με στόχο να αποτελέσει ένα κρίσιμο συστατικό της σύγχρονης επιχείρησης, δημιουργεί μια αρχιτεκτονική που προσαρμόζεται στις μεταβαλλόμενες επιχειρησιακές ανάγκες, δίνοντας τη δυνατότητα υλοποίησης ενός ισχυρού κέντρου δεδομένων με μια μεταβλητή και ευέλικτη κατανομή δαπανών.

Το Grid Computing είναι ένα υπολογιστικό μοντέλο που παρέχει τη δυνατότητα ανάπτυξης μεγάλης υπολογιστικής ισχύος. Βασισμένο σε ένα εξελιγμένο, ανοικτό σύνολο προτύπων και πρωτοκόλλων επικοινωνίας και ανταλλαγής δεδομένων, που επιτρέπει την επικοινωνία μεταξύ ετερογενών και γεωγραφικά διασκορπισμένων συστημάτων, υλοποιεί μια εικονική αρχιτεκτονική με πολλούς δικτυωμένους υπολογιστές, που είναι σε θέση να διαμοιράσει τις εκτελεστικές εργασίες μέσα σε μία υποδομή παράλληλων συστημάτων. Χρησιμοποιώντας τους πόρους πολλών ξεχωριστών υπολογιστών συστημάτων ενός δικτύου, μπορούν να εξυπηρετηθούν μεγάλης κλίμακας εργασίες και να εκτελεσθούν υπολογισμοί σε μεγάλα σύνολα δεδομένων, είτε διαχωρίζοντας τα σε μικρότερα σύνολα, είτε παρέχοντας τη δυνατότητα να εκτελεσθούν περισσότερες εργασίες ταυτόχρονα, από ότι είναι δυνατόν σε έναν μόνο υπολογιστή. Μια υλοποίηση Grid, αναλύει συνεχώς την ανάγκη για πόρους και τους κατανέμει ανάλογα, με σκοπό την εξυπηρέτηση των αιτημάτων. Δεν είναι απαραίτητο να γνωρίζουμε που βρίσκονται τα δεδομένα ή ποιος υπολογιστής επεξεργάζεται το αίτημα. Η αρχιτεκτονική Grid μπορεί να αξιοποιήσει δημιουργικά υπολογιστικές υποδομές και να βοηθήσει στην προστασία

της πραγματοποιούμενης επένδυσης κυρίως λόγω των εξαιρετικών χαρακτηριστικών επεκτασιμότητας που διαθέτει.

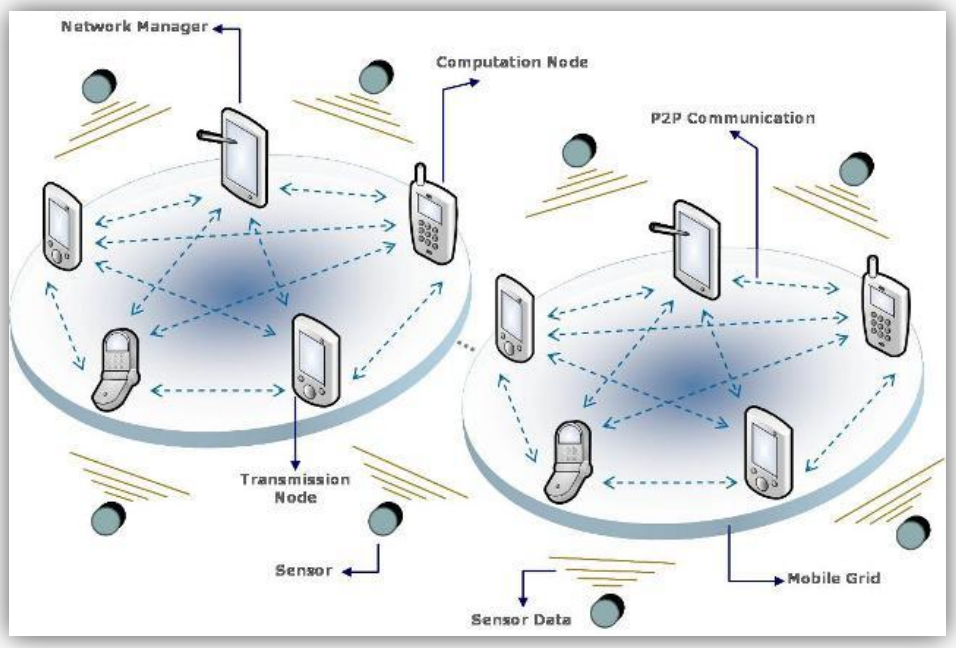
1. Με την υλοποίησης αρχιτεκτονικής Grid Computing παρέχονται σημαντικά οφέλη για έναν οργανισμό, όπως:
2. Ευελιξία στη κάλυψη μεταβαλλόμενων επιχειρησιακών αναγκών
3. Βέλτιστο λόγο επένδυσης / απόδοσης.
4. Προστασία επένδυσης και γρήγορη απόσβεση της
5. Μεγάλα αποθέματα επεξεργαστικής ισχύος
6. Δυνατότητα εξυπηρέτησης εφαρμογών μεγάλης κλίμακα
7. Αυξημένη διαθεσιμότητα των υπολογιστικών υποδομών

1.3 *Κινητό υπολογιστικό πλέγμα*

Το Κινητό πλέγμα (Κ.Π) είναι ένας γενικότερος όρος που περιγράφει την εφαρμογή των αρχών του πλέγματος που εξετάστηκαν νωρίτερα σε μικρές , φορητές και ασύρματες συσκευές καθώς και συσκευές τηλεπικοινωνιών. Σκοπός αυτής της κατηγορίας πλέγματος είναι η παροχή πρόσβασης σε πληροφορίες , επικοινωνίες και υπηρεσίες, οπουδήποτε, οποτεδήποτε και με κάθε δυνατό μέσο. Τα να επιτευχθεί βέβαια κάτι τέτοιο δεν είναι πάντα εύκολο. Στην πραγματικότητα απαιτεί την δημιουργία τηλεπικοινωνιακών υποδομών και παρεμβάσεις στα δίκτυα υπολογιστών , στα λειτουργικά που χρησιμοποιούνται και τις εφαρμογές. Κάτι τέτοιο γίνεται εύκολα αντιληπτό αν αναλογιστούμε ότι οι πόροι σε ένα Κ.Π είναι συνεχώς κινούμενοι ,παρουσιάζονται σαν αυτοδιοργανούμενα πλέγματα που οι κόμβοι τους συνδέονται ασύρματα, σχηματίζουν απρόβλεπτες τοπολογίες και έχουν άλλους περιορισμούς σε σχέση με τους σταθερούς πόρους. Από μια τεχνική σκοπιά το Κ.Π θα πρέπει να μπορεί να χειρίζεται τους ακόλουθους τύπους τελικών χρηστών , την μετακίνηση των συσκευών τους αλλά και τις συνεδρίες τους όπως ορίζονται παρακάτω:

- *Νομαδικός χρήστης* : Ο νομαδικός χρήστης αλλάζει την φυσική του τοποθεσία και στοχεύει να συνδεθεί στο διαδίκτυο και στο Πλέγμα από αυτές τις διαφορετικές τοποθεσίες. Καθώς ο χρήστης αλλάζει την τοποθεσία του η συσκευή του είναι κλειστή.
- *Φορητός χρήστης*: Ο πραγματικός φορητός χρήστης είναι ένας νομαδικός χρήστης που αλλάζει το δίκτυο ενώ είναι συνδεδεμένος. Αυτό προϋποθέτει επιπρόσθετες απαιτήσεις καθώς η αλλαγή ,πχ της διεύθυνσης IP , πρέπει να διαχειριστεί σε πραγματικό χρόνο.

- *Φορητότητα Συσκευής ή συνεδρίας* : Φορητότητα συσκευής σημάνει ότι η συνέδρια μετακινείται από συσκευή σε μια άλλη αυτούσια. Έτσι για παράδειγμα μετά την έναρξη της συνεδρίας από ένα υπολογιστή παλάμης ο χρήστης αποφασίζει να την συνεχίσει από τον σταθερό υπολογιστή του.



Εικόνα 1 : Η διάταξη ενός κινητού πλέγματος.
(http://www.comnets.rwth-aachen.de)

Μια άλλη οπτική της φορητότητας που θα έπρεπε να σκεφτούμε είναι η διαφοροποίηση μεταξύ της ασύρματης δικτύωσης και των φορητών διεπαφών χρήστη. Στην ασύρματη δικτύωση το πρόβλημα έγκειται στο γεγονός ότι η σύνδεση μεταβάλλεται και σε ποιότητα και σε τιμή ενώ οι αποσυνδέσεις είναι απρόσμενες , καθιστώντας έτσι κάθε είδος σύγχρονης αλληλεπίδρασης αβέβαιη. Στην περίπτωση των φορητών διεπαφών χρήστη οι προκλήσεις προέρχονται από το μικρό μέγεθος των συσκευών. Μικρές οθόνες απεικόνισης μπορούν να παρουσιάσουν μόνο περιορισμένες ποσότητες πληροφορίας, ενώ η απουσία πληκτρολογίου κάνει την εισαγωγή δεδομένων δύσκολη. Πρέπει εδώ να τονιστεί ότι η φορητότητα βασίζεται στην σύγχρονη επικοινωνία. Σε ένα συνεργατικό περιβάλλον το πιο δύσκολο είναι η συγχρονισμένη συνεργασία πραγματικού χρόνου ανάμεσα στους συμμετέχοντες. Παρότι η ασύγχρονη επικοινωνία είναι ευκολότερα υλοποιήσιμη σε αβέβαια και πολύπλοκα περιβάλλοντα, η σύγχρονη εισάγει περιορισμούς που από την φύση τους είναι δύσκολοι να χειριστούν. Τα κοινόχρηστα αντικείμενα δεν είναι πλέον επιφάνειες εργασίας και βάσεις δεδομένων όπου δέχονται την επίδραση πολλών χρηστών , αλλά θα πρέπει να χρησιμοποιηθούν μηχανισμοί που

θα επιτρέπουν τον διαμοιρασμό των εφαρμογών καθώς και ταυτόχρονων σχημάτων για τον χειρισμό των συνεδριών.

Στις φορητές συσκευές η επεξεργαστική ισχύς , η αποθηκευτική ικανότητα καθώς και η ενσωματωμένη μνήμη είναι πολύ περιορισμένα σε σχέση με τους υπολογιστές (εξαιρώντας τους φορητούς υπολογιστές). Η διάρκεια της μπαταρίας είναι μικρή ενώ η ικανότητα απεικόνισης είναι αυστηρά περιορισμένη σε μέγεθος αλλά και ποιότητα. Λόγω της συνεχούς μετακίνησης το δίκτυο θεωρείται ασταθές (διακοπτόμενες συνδέσεις και μικρό bandwidth) . Αυτοί ακριβώς οι περιορισμοί είναι που καθιστούν τις φορητές συσκευές ιδανικές να επωφεληθούν του Υπολογιστικού Πλέγματος. Τα κύρια πλεονεκτήματα του Κ.Π είναι η ***Ad – Hoc*** διασύνδεση *κινητού- με- κινητό* και *κινητού- με- υπολογιστή* για την ανακάλυψη και διαμοιρασμό πόρων καθώς και για την βελτίωση της εμπειρίας του χρήστη. Ένα περιβάλλον βασισμένο σε Κ.Π θα επέτρεπε στις φορητές συσκευές να γίνουν τελικά πιο ευέλικτες και αποτελεσματικές, καθότι όλες οι υπολογιστικά απαιτητικές εργασίες θα στέλνονται σε άλλους ισχυρότερους υπολογιστές ή συσκευές.

Το Κ.Π. επιτρέπει αφενός την φορητότητα των χρηστών που επιζητούν πρόσβαση σε ένα σταθερό Πλέγμα αφετέρου την ενσωμάτωση των πόρων καθαυτών σαν μέρη του Πλέγματος. Και οι δύο περιπτώσεις έχουν περιορισμούς που πρέπει να αντιμετωπιστούν. Στην πρώτη οι φορητές συσκευές δρουν σαν διεπαφές για το Πλέγμα με δυνατότητα Job submission, monitoring και διαχείρισης δραστηριοτήτων δίχως χρονικούς/τοπικούς περιορισμούς ενώ το Πλέγμα τους παρέχει υψηλή αξιοπιστία , επιδόσεις και οικονομία. Οι φυσικοί περιορισμοί των φορητών συσκευών κάνουν απαραίτητη την υιοθέτηση των υπηρεσιών που προσφέρει το πλέγμα στους φορητούς χρήστες. Σε αυτές τις περιπτώσεις το Κ.Π έχει την έννοια της ενσωμάτωσης των κινητών πόρων στο Πλέγμα. Οι επιδόσεις μάλιστα των σημερινών φορητών συσκευών βασιζόμενοι στην τελευταία προϋπόθεση έχει αποδεχτεί ότι είναι σημαντικά αυξημένες. Φορητοί Υπολογιστές και υπολογιστές παλάμης (Palmtop) μπορούν , συναθροισμένα, να προσφέρουν υπολογιστική ισχύ όταν συγκεντρώνονται τοπικά. Αυτή και μόνο η δυνατότητα μπορεί να προάγει την χρήση των Πλεγμάτων ,αλλά και των εφαρμογών τους, ακόμα και σε μέρη που αυτό θεωρείται απίθανο. Παρόλα αυτά πάντα θα υπάρχει το ερώτημα αν η υιοθέτηση των Πλεγμάτων για την επίλυση ενός προβλήματος είναι συμφέρουσα σε σύγκριση με μία συμβατική λύση. Είναι μια αναδυόμενη τεχνολογία που έχει την φιλοδοξία να παρέχει περισσότερο αποτελεσματικές λύσεις από ανταγωνιστικές μεθόδους χρησιμοποιώντας την απλή λογική της διασύνδεσης και του διαμοιρασμού. Με αυτό τον τρόπο τα Πλέγματα και τα Κ.Π μπορούν να είναι ιδανικές λύσεις για πολλές μεγάλες εφαρμογές δυναμικής φύσεως που απαιτούν διαφάνεια σε επίπεδο χρήστη. Το πλέγμα θα αυξήσει τις

επιδόσεις των εμπλεκομένων εφαρμογών καθώς και τον ρυθμό χρησιμοποίησης των πόρων εφαρμόζοντας αποτελεσματικούς μηχανισμούς διαχείρισης στον τεράστιο αριθμό πόρων.

1.4 *Αντικείμενο διπλωματικής*

Σκοπός της εργασίας αυτής είναι η προσαρμογή συστημάτων κινητών επικοινωνιών για λειτουργία σε περιβάλλον Υπολογιστικού Πλέγματος. Προσομοιώνουμε το περιβάλλον πλέγματος χρησιμοποιώντας το μεσολογισμικό GRIA και ενσωματώνουμε σε αυτό κινητές συσκευές βασισμένες στο λειτουργικό σύστημα Android.

Μέσω εφαρμογών του Android αναπτύσσουμε έναν πελάτη για το Grid που παρέχει στον χρήστη την ιδία λειτουργικότητα με αυτήν ενός Η/Υ. Υπάρχει δηλαδή η δυνατότητα Υποβολής εργασιών στο Πλέγμα, η διακοπή τους αλλά και η επίβλεψη τους όπως ακριβώς και σε έναν τυπικό πελάτη που εκτελείται σε προσωπικό υπολογιστή. Η δυνατότητα αυτή θα μας επιτρέψει την εκτέλεση εργασιών ιδιαίτερα πολύπλοκων , κάτι που δεν θα μπορούσε να συμβεί βασιζόμενοι εξολοκλήρου στην υπολογιστική ισχύ μιας κινητής συσκευής. Χρησιμοποιώντας ως βασικά εργαλεία μας το Android και το GRIA σχεδιάζουμε και υλοποιούμε το προαναφερθέν σύστημα.

1.5 *Οργάνωση κειμένου*

Το Κεφάλαιο 2 αναφέρεται στην θεωρητική παρουσίαση των συστατικών μερών. Η ανάλυση απαιτήσεων του συστήματος γίνεται στο κεφάλαιο 3, ενώ στο Κεφάλαιο 4 αναπτύσσουμε τον σχεδιασμό των υποσυστημάτων. Στο κεφάλαιο 5 παρουσιάζεται η υλοποίηση που ακολουθήθηκε ενώ στο Κεφάλαιο 6 ελέγχουμε και αξιολογούμε το σύστημα μας. Ο επίλογος και τα συμπεράσματα βρίσκονται στο κεφάλαιο 7.

2

Θεωρητικό υπόβαθρο

Στο κεφάλαιο αυτό, παρουσιάζουμε αρχικά κάποιες γνωστές πλατφόρμες κινητών επικοινωνιών, ενώ ακολουθεί η θεωρητική ανάλυση των συστατικών μερών που χρησιμοποιούμε.

2.1 *Πλατφόρμες κινητών επικοινωνιών*

Μία πλατφόρμα κινητής συσκευής, ή καλύτερα το λειτουργικό σύστημα, είναι το υπεύθυνο λογισμικό για τον έλεγχο και την λειτουργία της, όπως ακριβώς συμβαίνει και στους προσωπικούς υπολογιστές. Η διαφορά έγκειται στο ότι είναι δομικά απλούστερη και ασχολείται περισσότερο με τις ασύρματες εκδόσεις της ευρυζωνικής και τοπικής συνδεσιμότητας, των πολυμέσων, και των διαφορετικών μεθόδων εισαγωγής. Υπάρχουν πλήθος λειτουργικών συστημάτων για κινητές συσκευές με τελευταία προσθήκη το Google Android όπου χρησιμοποιήθηκε στην παρούσα εργασία και παρουσιάζεται αναλυτικά στην επόμενη ενότητα. Παρουσιάζουμε εδώ ωστόσο, κάποιες από τις σημαντικότερες ανταγωνιστικές πλατφόρμες :

2.1.1 Το λειτουργικό Symbian

Το *Symbian OS* , κατέχει αυτήν την στιγμή το μεγαλύτερο μερίδιο αγοράς και χρησιμοποιείται στις περισσότερες κινητές συσκευές. Αποτελεί εξέλιξη του λειτουργικού συστήματος EPOC από την Psion και έχει δημιουργηθεί σε C++. Δημιουργήθηκε με βάση τρείς σχεδιαστικές αρχές: η

ακεραιότητα και η ασφάλεια των στοιχείων των χρηστών να είναι κυρίαρχες, ο χρόνος των χρηστών δεν πρέπει να σπαταλείται, και ότι όλοι οι πόροι είναι ελάχιστοι. Αυτό οδήγησε στο γράψιμο ενός μικρό-πυρήνα (microkernel) , την έμφαση στις υπηρεσίες της μορφής αίτηση-απάντηση και στον διαχωρισμό του UI από την λογική του προγράμματος. Οι εφαρμογές και το ίδιο το λειτουργικό ακολουθούν το αντικειμενοστραφές σχεδιαστικό μοντέλο MVC (Model – View – Controller). Το πρότυπο συστημάτων Symbian OS περιέχει τα ακόλουθα στρώματα, από πάνω προς τα κάτω:

- Στρώμα αλληλεπίδρασης χρήστη (***UI***).
- Στρώμα υπηρεσιών εφαρμογής (***Application Layer***).
    - Java ME (Mobile Edition)
- Στρώμα υπηρεσιών λειτουργικού συστήματος.
    - γενικές υπηρεσίες.
    - υπηρεσίες επικοινωνιών.
    - πολυμέσα και υπηρεσίες γραφικής αναπαράστασης.
    - υπηρεσίες συνδεσιμότητας.
- Στρώμα βασικών υπηρεσιών.
- Υπηρεσίες πυρήνα & στρώμα διεπαφών υλικού.

Το στρώμα βασικών υπηρεσιών είναι το χαμηλότερο επίπεδο στο οποίο δίνεται πρόσβαση στον προγραμματιστή εφαρμογών.

2.1.2 Το λειτουργικό Windows Mobile

Το *Windows Mobile OS* , είναι ένα συμπαγές λειτουργικό σύστημα που συνδυάζεται με μία σουίτα βασικών εφαρμογών βασισμένες στο Win32 API της Microsoft. Στις συσκευές που τρέχουν το Windows Mobile , περιλαμβάνονται υπολογιστές τσέπης, smartphones και ενσωματωμένα συστήματα για οχήματα. Έχει σχεδιαστεί να είναι παρόμοιο με τις εκδόσεις των Windows για προσωπικούς υπολογιστές, τόσο αισθητικά όσο και λειτουργικά. Η ανάπτυξη εφαρμογών από τρίτους μπορεί να γίνει με πολλούς τρόπους, είτε γράφοντας κώδικα στην φυσική γλώσσα του λειτουργικού ,την Visual C++, είτε με την χρήση του .NET Compact Framework, ή κάποιον Server-side κώδικα που χρησιμοποιεί τον Internet Explorer Mobile για να λειτουργήσει. Το .NET Compact Framework είναι στην πραγματικότητα ένα υποσύνολο του .NET Framework και γι αυτό μοιράζεται πολλά στοιχεία από την ανάπτυξη εφαρμογών σε προσωπικούς υπολογιστές, application servers και web servers που έχουν το .NET Framework εγκατεστημένο.

2.2 *Google Android*

Το Android είναι μία πλατφόρμα για κινητές συσκευές βασισμένη σε πυρήνα Linux 2.6. Αναπτύχθηκε αρχικά από την Google και αργότερα από την Open Handset Alliance. Πρόκειται για λογισμικό ανοιχτού κώδικα που επιτρέπει στους προγραμματιστές να γράφουν κώδικα σε γλώσσα Java χρησιμοποιώντας βιβλιοθήκες που έχει αναπτύξει η Google. Η παρουσίαση της πλατφόρμας ανακοινώθηκε μαζί με την ίδρυση της Open Handset Alliance, ένα σύνδεσμο από 34 εταιρίες hardware,λογισμικού και τηλεπικοινωνιών αφιερωμένο στην προώθηση των ανοιχτών standards για τις κινητές συσκευές. Το μεγαλύτερο μέρος της πλατφόρμας είναι διαθέσιμο με Apache free-software and open source license. Οι εφαρμογές που δημιουργούνται χρησιμοποιούν το Android SDK και τρέχουν στον Dalvik , μια ειδική εικονική μηχανή για ενσωματωμένα συστήματα, που τρέχει πάνω από πυρήνα Linux. Τα κύρια χαρακτηριστικά του λειτουργικού συστήματος είναι.

Αξιοποίηση ελεύθερου Κώδικα:

Το Android χτίστηκε από κάτω προς τα πάνω ώστε να επιτρέπει στους προγραμματιστές να δημιουργούν εφαρμογές που αξιοποιούν στον μέγιστο βαθμό όλες τις δυνατότητες μίας κινητής συσκευής. Έχει δημιουργηθεί για να είναι πραγματικά ανοιχτό. Παραδείγματος χάριν, μια εφαρμογή θα μπορούσε να ζητήσει από οποιαδήποτε άλλη λειτουργία πυρήνα όπως η πραγματοποίηση των κλήσεων, η αποστολή των μηνυμάτων κειμένων, ή τη χρησιμοποίηση της φωτογραφικής μηχανής, επιτρέποντας στους υπεύθυνους developers να δημιουργήσουν μια πλουσιότερη και πιο συνεκτική εμπειρία για τους χρήστες. Ο προγραμματιστής μπορεί να κάνει τα πάντα, από την αποστολή σύντομων μηνυμάτων με ακριβώς 2 γραμμές κώδικα, μέχρι την αντικατάσταση ακόμη και της βασικής οθόνης της συσκευής. Κάποιος θα μπορούσε εύκολα να δημιουργήσει ένα πλήρως προσαρμοσμένο λειτουργικό σύστημα εντός σύντομου χρονικού διαστήματος. Το Android στηρίζεται στον ανοικτό πυρήνα Linux. Επιπλέον, χρησιμοποιεί μια εικονική μηχανή που έχει σχεδιαστεί για να βελτιστοποιήσει τους πόρους μνήμης και υλικού σε ένα κινητό περιβάλλον. Το σύστημα θα είναι ανοικτή πηγή που μπορεί να επεκταθεί γενναιόδωρα για να ενσωματώσει τις νέες τεχνολογίες αιχμής όπως προκύπτουν. Η πλατφόρμα συνεχίζει να εξελίσσεται για να χτίσει καινοτόμες κινητές εφαρμογές.

Όλες οι εφαρμογές θεωρούνται ίσες

Το Android δεν διαφοροποιεί μεταξύ των εφαρμογών πυρήνων της κινητής συσκευής και των εφαρμογών τρίτων. Μπορούν όλοι να δημιουργήσουν εφαρμογές και να έχουν ισότιμη πρόσβαση στις ενσωματωμένες εφαρμογές παρέχοντας στους χρήστες ένα ευρύ φάσμα υπηρεσιών. Με τις συσκευές που στηρίζονται στο Android, οι χρήστες θα είναι σε θέση να

προσαρμόσουν πλήρως το τηλέφωνο στα ενδιαφέροντά τους. Μπορούν να ανταλλάξουν την βασική οθόνη, το ύφος του dialer, ή οποιαδήποτε από τις εφαρμογές. Μπορούν ακόμη και να καθοδηγήσουν τα τηλέφωνά τους για να χρησιμοποιήσουν την αγαπημένη εφαρμογή περιήγησης των φωτογραφιών τους να χειριστούν την εξέταση όλων των φωτογραφιών. Οι developers μπορούν να προσαρμόσουν 100% την android-συσκευή τους. Στο Android η επικοινωνία των υποσυστημάτων είναι βασισμένη στις αποκαλούμενες *Προθέσεις (Intents)* , οι οποίες είναι λίγο ως πολύ μια σειρά κινήσεων που καθορίζει πως πρέπει να αντιμετωπιστεί μια δράση. Ένα παράδειγμα για αυτό είναι: "*android.provider.Telephony.SMS_RECEIVED*" κάποιος μπορεί απλά να ακούσει εκείνη την *πρόθεση* με το γράψιμο περίπου 5 γραμμών κώδικα. Το σύστημα έπειτα θα αναγνώριζε ότι υπάρχουν περισσότερες από μια εφαρμογές που θέλουν να χειριστούν εκείνη την *πρόθεση* και να ζητήσουν από το χρήστη να επιλέξει ποια θα επιθυμούσε για να την χειριστεί.

Εφαρμογές δίχως περιορισμούς

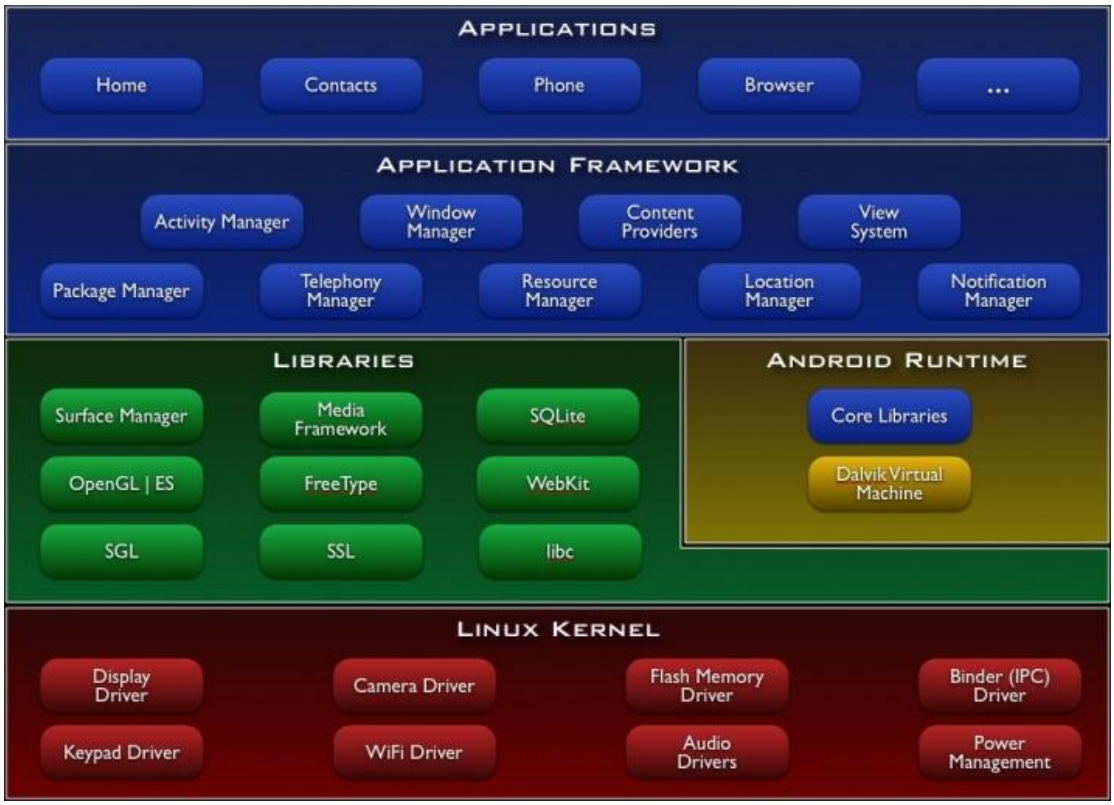
Το Android καταργεί τα εμπόδια στην οικοδόμηση νέων και καινοτόμων εφαρμογών. Παραδείγματος χάριν, ένας υπεύθυνος για την ανάπτυξη μπορεί να συνδυάσει τις πληροφορίες από τον Ιστό με τα στοιχεία που αφορούν το κινητό τηλέφωνο ενός ατόμου όπως οι επαφές του χρήστη, το ημερολόγιο, ή η γεωγραφική θέση - για να παρέχει μια πιο σχετική εμπειρία χρηστών. Με το Android, μπορεί να χτιστεί μια εφαρμογή που επιτρέπει στους χρήστες να δουν τη θέση των φίλων τους και να προειδοποιηθούν όταν είναι σε κοντινή περιοχή.

Γρήγορη & εύκολη ανάπτυξη εφαρμογής

Το Android παρέχει πρόσβαση σε ένα ευρύ φάσμα χρήσιμων βιβλιοθηκών και εργαλείων που μπορούν να χρησιμοποιηθούν για να χτίσουν πλούσιες εφαρμογές. Παραδείγματος χάριν, το Android επιτρέπει στους προγραμματιστές να λάβουν τη θέση της συσκευής, και επιτρέπει στις συσκευές να επικοινωνήσουν η μια με την άλλη επιτρέποντας τις λεγόμενες κοινωνικές εφαρμογές (Social Applications). Επιπλέον, περιλαμβάνει ένα πλήρες σύνολο εργαλείων που έχουν χτιστεί από την αρχή τους παράλληλα με την πλατφόρμα και παρέχει στους προγραμματιστές υψηλή παραγωγικότητα και βαθιά διορατικότητα στις αιτήσεις τους.

2.2.1 Η Αρχιτεκτονική του Android

Το κάτωθι διάγραμμα παρουσιάζει τα βασικότερα μέρη του λειτουργικού συστήματος Android. Κάθε μέρος αναλύεται παρακάτω:



Εικόνα 2: Η αρχιτεκτονική του Android. Από την επίσημη ιστοσελίδα του Android. (http://code.google.com/android)

2.2.1.1 Applications - Εφαρμογές

Το Android είναι εξοπλισμένο με ένα σετ κύριων και θεμελιωδών εφαρμογών όπως πρόγραμμα διαχείρισης ηλεκτρονικής αλληλογραφίας, πρόγραμμα αποστολής γραπτών μηνυμάτων (SMS), ημερολόγιο , χάρτες, περιηγητή ιστού, διαχείριση τηλεφωνικών επαφών και άλλα. Όλες οι εφαρμογές είναι γραμμένες χρησιμοποιώντας την γλώσσα προγραμματισμού Java.

2.2.1.2 Application Framework - Πλαίσιο Εφαρμογών

Οι προγραμματιστές έχουν πλήρη πρόσβαση στο ίδιο framework API που χρησιμοποιείται και από τις κύριες εφαρμογές. Η αρχιτεκτονική έχει σχεδιαστεί για να απλοποιήσει την επαναχρησιμοποίηση συστατικών: κάθε εφαρμογή μπορεί να «δημοσιεύσει» τις δυνατότητες της και οποιαδήποτε άλλη εφαρμογή μπορεί τότε να επαναχρησιμοποιήσει αυτές τις δυνατότητες (πάντα υπό τους περιορισμούς ασφαλείας που επιβάλλονται από το framework). Ο ίδιος αυτός μηχανισμός επιτρέπει σε στοιχεία να αντικαθίστανται από τον χρήστη. Στο

υπόβαθρο τους όλες οι εφαρμογές είναι ένα σύνολο υπηρεσιών και συστημάτων που περιλαμβάνει :

- Ένα πλούσιο και επεκτάσιμο σύνολο από Views (Όψεις, Γραφικά στοιχεία) που μπορούν να χρησιμοποιηθούν για να χτίσουν μία εφαρμογή, όπως lists, grids, text boxes, buttons ακόμα και έναν ενσωματώσιμο περιηγητή ιστού.
- Τους Content Providers που επιτρέπουν στις εφαρμογές να έχουν πρόσβαση σε δεδομένα άλλων εφαρμογών (όπως οι επαφές καταλόγου), ή να μοιράζονται τα δικά τους δεδομένα.
- Έναν Resource Manager που παρέχει πρόσβαση σε μη-προγραμματιστικούς πόρους όπως τοπικές συμβολοσειρές, γραφικά και αρχεία layout.
- Έναν Notification Manager που επιτρέπει σε όλες τις εφαρμογές να δείχνουν προσαρμόσιμες ειδοποιήσεις στην μπάρα κατάστασης του τηλεφώνου
- Έναν Activity Manager που διαχειρίζεται τον κύκλο ζωής των εφαρμογών και παρέχει μία κοινή στοίβα πλοήγησης.

2.2.1.3 Libraries – Βιβλιοθήκες

Το Android περιλαμβάνει ένα σύνολο από C/C++ βιβλιοθήκες που χρησιμοποιούνται από διάφορα στοιχεία της πλατφόρμας. Αυτές οι δυνατότητες εκτίθενται στους προγραμματιστές μέσα από το application framework του Android. Μερικές από τις βασικές βιβλιοθήκες παρατίθεται κάτωθι:

- **Βιβλιοθήκη Συστήματος C** – Μία BSD- παραγόμενη υλοποίηση της standard C system library (libc), προσαρμοσμένη για ενσωματωμένες Linux συσκευές.
- **Βιβλιοθήκες Πολυμέσων –** Βασισμένες στο PacketVideo OpenCORE. Οι βιβλιοθήκες υποστηρίζουν αναπαραγωγή και εγγραφή πολλών δημοφιλών μορφών ήχου και εικόνας , καθώς επίσης και στατικών εικόνων , συμπεριλαμβανομένων των MPEG4, H.264, MP3, AAC, AMR, JPG και PNG.
- **Διαχειριστής Επιφάνειας** – Διαχειρίζεται την πρόσβαση στο υποσύστημα της απεικόνισης δεδομένων και συνθέτει τα 2D και 3D γραφικά στρώματα από πολλαπλές εφαρμογές.
- **LibWebCore –** Μία μοντέρνα μηχανή περιηγητή ιστού που χρησιμοποιείται και από τον Android browser και από την ενσωματώσιμη web view.
- **SGL-** η μηχανή γραφικών 2D.

- **3D Βιβλιοθήκες –** Μια υλοποίηση βασισμένη στο ES 1.0 API, οι βιβλιοθήκες χρησιμοποιούν είτε επιτάχυνση τρισδιάστατων γραφικών (όπου επιτρέπεται) είτε το συμπεριλαμβανόμενο και πλήρως παραμετροποιημένο λογισμικό 3D rasterizer.
- **FreeType-** επεξεργασία bitmap και διανυσματικών γραμματοσειρών.
- **SQLite –** μία ισχυρή και ελαφριά σε υπολογιστικούς πόρους σχεσιακή βάση δεδομένων , διαθέσιμη σε όλες τις εφαρμογές.

2.2.1.4 Android Runtime

Το Android περιλαμβάνει ένα σύνολο βιβλιοθηκών που περιέχουν το μεγαλύτερο μέρος της λειτουργικότητας των βασικών βιβλιοθηκών της γλώσσας Java. Κάθε εφαρμογή στο Android τρέχει στην δική της διεργασία από την εικονική μηχανή Dalvik. Η Dalvik έχει γραφτεί έτσι ώστε η συσκευή να μπορεί να τρέχει περισσότερες εικονικές μηχανές αποδοτικά. Η εικονική μηχανή εκτελεί αρχεία σε μορφή εκτελέσιμων Dalvik (.dex) που έχουν βελτιστοποιηθεί για όσο τον δυνατόν ελάχιστη κατανάλωση μνήμης συστήματος.

Η Dalvik , είναι βασισμένη σε καταχωρητές και τρέχει τις κλάσεις ,πού έχουν περάσει από ένα Java compiler , και έχουν μετασχηματιστεί σε αρχεία .dex από το ενσωματωμένο «dx» εργαλείο. Η Dalvik στηρίζεται στον πυρήνα Linux για την λειτουργικότητα της όπως τα νήματα και η χαμηλού επιπέδου διαχείριση μνήμης.

2.2.1.5 Linux Kernel – Πυρήνας Linux

Το Android βασίζεται στην έκδοση 2.6 του Linux για τις βασικές υπηρεσίες του συστήματος όπως η ασφάλεια , η διαχείριση μνήμης , η διαχείριση διεργασιών , η στοίβα δικτύου και το μοντέλων οδηγών συστήματος. Ο πυρήνας επίσης δρα σαν ένα επίπεδο αφαίρεσης μεταξύ του υλικού και της υπόλοιπης στοίβας λογισμικού.

2.2.2 Μοντέλο Εφαρμογών του Android

Εφαρμογές , Λειτουργίες, Διεργασίες και Νήματα

Στα περισσότερα λειτουργικά συστήματα υπάρχει μια ισχυρή 1-προς-1 συσχέτιση μεταξύ του εκτελέσιμου αρχείου (όπως το αρχείο .exe στα Windows) , της διεργασίας, του εικονιδίου και της εφαρμογής που ο χρήστης αλληλεπιδρά . Στο Android αυτές οι συσχετίσεις είναι πολύ περισσότερο ρευστές και είναι σημαντική η κατανόηση πώς συνδέονται όλα αυτά τα κομμάτια. Λόγω της ευμετάβλητης φύσης των εφαρμογών στο Android , υπάρχει κάποια βασική ορολογία που χρειάζεται κατανόηση όταν υλοποιούνται διάφορα μέρη μίας εφαρμογής: Ένα **android package (ή .apk εν συντομία) είναι το αρχείο που περιέχει τον κώδικα μίας εφαρμογής και τους πόρους της. Αυτή είναι η μορφή που μια εφαρμογή διανέμεται και εγκαθίστανται στην κινητή συσκευή του χρήστη όταν ο ίδιος επιλέξει να την κατεβάσει.**

- Μία **λειτουργία** είναι γενικά αυτό που ο χρήστης αντιλαμβάνεται σαν «εφαρμογή» που μπορεί να εκτελεστεί: συνήθως μια λειτουργία έχει ένα εικονίδιο στην οθόνη υποδοχής μέσω του οποίου δίνεται η πρόσβαση, και είναι διαθέσιμη σαν ένα στοιχείο υψηλού επιπέδου που μπορεί να μεταφερθεί στο προσκήνιο μπροστά από άλλες λειτουργίες.
- Μία **διεργασία** είναι μία low-level διεργασία πυρήνα μέσα στην οποία τρέχει ο κώδικας της εφαρμογής . Κανονικά όλος ο κώδικας μέσα σε ένα .apk τρέχει σε μία , αφιερωμένη διεργασία γι αυτό το .apk, όμως η ετικέτα της διεργασίας μπορεί να χρησιμοποιηθεί οπουδήποτε τρέχει ο κώδικας της , είτε για ολόκληρο το .apk αρχείο είτε για κάποιο από τα στοιχεία activity, receiver, service, ή provider.

2.2.3 Λειτουργίες -Tasks

Ένα σημείο κλειδί εδώ είναι το εξής: Όταν ο χρήστης βλέπει μια «εφαρμογή», αυτό που πραγματικά χειρίζεται είναι ένα task. Μια λειτουργία λοιπόν , ή η εφαρμογή κατά τον χρήστη, είναι από την μεριά του προγραμματιστή μία η περισσότερες δραστηριότητες που ο χρήστης έχει περιηγηθεί μέσα σε αυτή την λειτουργία και ακόμα δεν έχει κλείσει, ή μια στοίβα δραστηριοτήτων.

2.2.4 Συσχέτιση Λειτουργιών

Σε κάποιες περιπτώσεις το Android πρέπει να ξέρει σε ποια λειτουργία ανήκει μία δραστηριότητα ακόμα και αν δεν εκτελείται μέσα σε μία συγκεκριμένη λειτουργία. Αυτό επιτυγχάνεται μέσα από την συσχέτιση των λειτουργιών, που παρέχουν ένα μοναδικό στατικό όνομα για την λειτουργία στην οποία μία η περισσότερες δραστηριότητες πρόκειται να τρέξουν.

Η προεπιλεγμένη συσχέτιση λειτουργίας για μία δραστηριότητα είναι το όνομα του πακέτου .apk που υλοποιείται η δραστηριότητα. Αυτό παρέχει την αναμενόμενη συμπεριφορά όπου όλες οι δραστηριότητες σε ένα συγκεκριμένο .apk είναι μέρος μίας μόνο εφαρμογής στον χρήστη.

2.2.5 Διεργασίες - Processes

Στο Android , οι διεργασίες είναι εξολοκλήρου μία λεπτομέρεια υλοποίησης των εφαρμογών και όχι κάτι που ο χρήστης γνωρίζει συνήθως. Οι κύριες χρήσεις τους είναι απλά:

- Η βελτίωση της σταθερότητας ή της ασφάλειας βάζοντας ασταθή ή αναξιόπιστο κώδικα σε άλλη διεργασία
- Η μείωση της καθυστέρησης που δημιουργείται τρέχοντας τον κώδικα από πολλαπλά .apk στην ίδια διεργασία.
- Να βοηθήσει το σύστημα να διαχειριστεί τους πόρους , βάζοντας τον απαιτητικό σε πόρους κώδικα σε μία ξεχωριστή διεργασία που μπορεί να τερματιστεί ανεξάρτητα από τα αλλά κομμάτια της εφαρμογής.

Όπως προηγουμένως περιγράφηκε, η ετικέτα της διεργασίας χρησιμοποιείται για να ελέγχει την διεργασία στην οποία τρέχουν συγκεκριμένα στοιχεία της εφαρμογής. Άξιο σημείωσης είναι το γεγονός ότι αυτή η ετικέτα δεν μπορεί να χρησιμοποιηθεί για να παραβιαστεί η ασφάλεια του συστήματος: αν δύο .apk που δεν μοιράζονται το ίδιο user ID , προσπαθήσουν να τρέξουν την ίδια διεργασία , αυτό δεν θα επιτραπεί και 2 διακριτές διεργασίες θα δημιουργηθούν για κάθε μία από αυτές.

2.2.6 Νήματα - Threads

Κάθε διεργασία έχει ένα ή περισσότερα νήματα που τρέχουν σε αυτή. Στις περισσότερες περιπτώσεις το Android αποφεύγει την δημιουργία περαιτέρω νημάτων σε μία διεργασία , κρατώντας την εφαρμογή σε ένα νήμα , εκτός αν αυτή δημιουργήσει μόνη της τα δικά της νήματα. Μία σημαντική επίπτωση αυτού είναι ότι όλες οι κλήσεις στα Activity, BroadcastReceiver, και Service γίνονται μόνο από το κυρίως νήμα της διεργασίας στο οποίο τρέχουν.

2.2.7 Ο κύκλος ζωής μίας Android Εφαρμογής

Στις περισσότερες περιπτώσεις, κάθε android εφαρμογή τρέχει στην δική της Linux διεργασία. Αυτή η διεργασία δημιουργείται για την εφαρμογή όταν κάποιο μέρος από τον κώδικά της χρειάζεται να τρέξει και θα συνεχίζει να τρέχει μέχρι να μην χρειάζεται πλέον. Το σύστημα απαιτεί τότε πίσω την μνήμη που παραχωρήθηκε για να συνεχίσει με την εκτέλεση άλλων εφαρμογών. Ένα ιδιαίτερο αλλά και βασικό χαρακτηριστικό του android είναι ότι η διάρκεια ζωής μιας εφαρμογής **δεν εξαρτάται** άμεσα από την ίδια την εφαρμογή. Αντιθέτως, καθορίζεται από το σύστημα μέσω ενός συνδυασμού των μερών της εφαρμογής που το σύστημα γνωρίζει ότι τρέχουν , το πόσο σημαντικά είναι αυτά στον χρήστη και πόση είναι η συνολική διαθέσιμη μνήμη στο σύστημα. Είναι πολύ σημαντική η κατανόηση των διαφορετικών συστατικών που απαρτίζουν μια εφαρμογή και το πως αυτά επηρεάζουν την διάρκεια ζωής μιας διεργασίας. Για να καθορίσει ποιες διεργασίες πρέπει να τερματιστούν όταν βρίσκεται σε κατάσταση χαμηλής μνήμης, το android τοποθετεί κάθε διεργασία σε μία ιεραρχία σημαντικότητας βασιζόμενο στα συστατικά που τρέχουν μέσα σε αυτήν και την κατάσταση αυτών των συστατικών. Οι διεργασίες κατατάσσονται με σειρά σημαντικότητας ως εξής:

1. ***Διεργασία προσκηνίου*** : Απαιτείται για αυτό που κάνει ο χρήστης την τρέχουσα στιγμή. Διάφορα υποσυστήματα εφαρμογών μπορούν να καταδείξουν τις περιεχόμενες σε αυτά διεργασίες σαν προσκηνιακές με διαφορετικούς τρόπους. Μια διεργασία θεωρείται ότι βρίσκεται στο προσκήνιο εάν ισχύει κάποια από τις παρακάτω συνθήκες:

    - Τρέχει μια Activity στην οθόνη με την οποία ο χρήστης αλληλεπιδρά.
    - Έχει ένα BroadcastReceiver που τρέχει
    - Έχει κάποιο Service το οποίο εκτελεί κώδικα σε μία από τις κλήσεις του (Service.onCreate(), Service.onStart(), ή Service.onDestroy()).

    Θα υπάρχουν πάντα ελάχιστες από αυτές τις διεργασίες στο σύστημα , και αυτές θα τερματιστούν μόνο σαν τελευταίο καταφύγιο εάν η μνήμη είναι τόσο χαμηλή που ούτε αυτές οι διεργασίες δεν μπορούν να συνεχίσουν την εκτέλεση τους. Γενικότερα, σε αυτό το σημείο, η συσκευή έχει φτάσει σε κατάσταση σελιδοποίησης της μνήμης, οπότε η πράξη αυτή απαιτείται ώστε να διατηρηθεί η απόκριση του περιβάλλοντος χρήστη.

2. ***Εμφανής Διεργασία*** : μια εμφανής διεργασία είναι κάποια που διατηρεί την activity φανερή στην οθόνη του χρήστη αλλά όχι στο προσκήνιο. Αυτό μπορεί να συμβεί για

παράδειγμα εάν η δραστηριότητα προσκηνίου εμφανίζεται σαν διάλογος που επιτρέπει στην προηγούμενη δραστηριότητα να φαίνεται από πίσω της. Μια τέτοια διεργασία θεωρείται ιδιαίτερα σημαντική και δεν θα τερματιστεί εκτός εάν κάτι τέτοιο απαιτείται για να συνεχίσουν να τρέχουν όλες οι διεργασίες προσκηνίου.

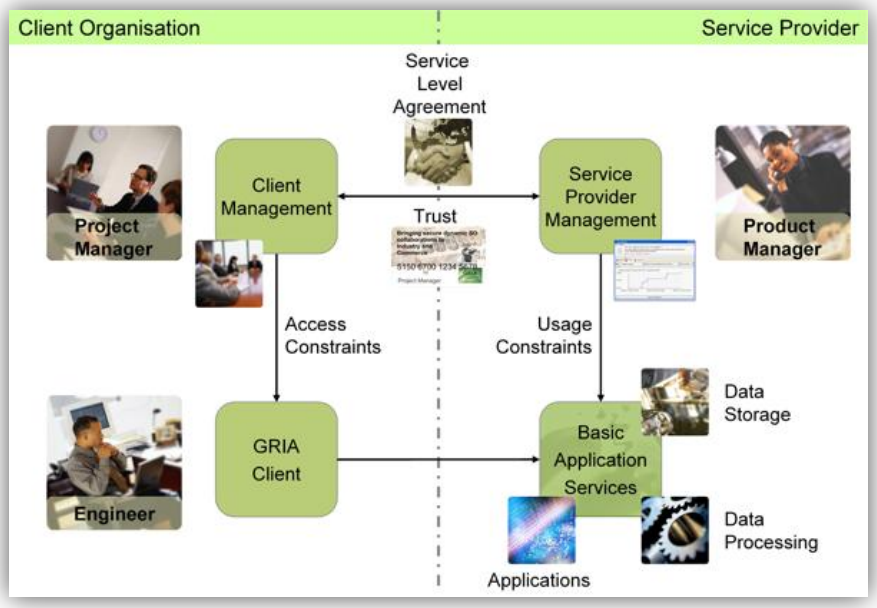
3. ***Διεργασία Υπηρεσίας*** : είναι αυτή που περιέχει ένα Service που ξεκίνησε με την startService() συνάρτηση. Αν και αυτές οι διεργασίες δεν είναι άμεσα ορατές από τον χρήστη, γενικότερα κάνουν πράγματα που τον ενδιαφέρουν (όπως η μεταφορά δεδομένων δικτύου στο παρασκήνιο), ώστε το σύστημα θα κρατάει πάντα αυτές τις διεργασίες να τρέχουν εκτός αν δεν υπάρχει αρκετή μνήμη να διατηρήσει όλες τις διεργασίες τύπου 1 και 2.
4. ***Διεργασία παρασκηνίου*** : είναι αυτή που περιέχει μία Activity που δεν είναι την τρέχουσα στιγμή ορατή από τον χρήστη. Αυτές οι διεργασίες δεν έχουν άμεσο αντίκτυπο στην εμπειρία του χρήστη. Τηρουμένης της σωστής υλοποίησης του κύκλου ζωής τους, το σύστημα μπορεί να σκοτώσει τέτοιες διεργασίες σε οποιαδήποτε στιγμή για να επανακτήσουν μνήμη για κάποια από τις διεργασίες των προηγούμενων τύπων. Συνήθως τρέχουν πολλές από αυτές τις διεργασίες, οπότε διατηρούνται σε μία LRU λίστα για να εξασφαλιστεί ότι η διεργασία που ο χρήστης βλέπει πιο συχνά, τερματίζεται τελευταία σε περίπτωση χαμηλής μνήμης.
5. ***Κενή διεργασία***: αυτή δεν διατηρεί κανένα ενεργό συστατικό . Ο μόνος λόγος για να διατηρείται μία τέτοια διεργασία είναι η χρήση της σαν cache για να βελτιωθεί ο χρόνος έναρξης την επόμενη φορά που κάποιο συστατικό η κάποια εφαρμογή χρειάζεται να τρέξει. Έτσι το σύστημα θα τερματίζει συχνά τέτοιες διεργασίες για να διατηρήσει την ισορροπία στους συνολικούς πόρους του συστήματος μεταξύ αυτών των κενών cached διεργασιών και των υπόγειων cache του πυρήνα.

Όταν αποφασίζει πώς θα κατηγοριοποιηθεί μία διεργασία , το σύστημα θα βασίσει την απόφαση του στο σημαντικότερο επίπεδο που βρέθηκε ανάμεσα στα συστατικά που είναι αυτή την στιγμή ενεργά στην οθόνη. Η προτεραιότητα μία διεργασίας μπορεί επίσης να αυξηθεί βασισμένη σε άλλες εξαρτήσεις που μπορεί να έχει.

2.3 *Το μεσολογισμικό GRIA*

2.3.1 Τι είναι το GRIA

Το GRIA είναι μια υπηρεσιοστρεφής υποδομή σχεδιασμένη να υποστηρίζει συνεργασίες επιχειρήσεων μέσω παροχής υπηρεσιών με έναν ασφαλές, ετερογενή και ευέλικτο τρόπο. Το GRIA χρησιμοποιεί πρωτόκολλα Web Services βασιζόμενα σε αυστηρές προδιαγραφές ώστε να ευνοούν την απροβλημάτιστη συνεργασία ετερογενών συστημάτων. Είναι διαθέσιμο δωρεάν και ανοιχτού κώδικα. Επίσης χρησιμοποιεί business models, διεργασίες και σημασιολογία ώστε να επιτρέπει στους παρόχους υπηρεσιών και στους χρήστες να ανακαλύπτουν ο ένα τον άλλον και να διαπραγματεύονται τους όρους για πρόσβαση σε συστήματα υψηλής τεχνολογικής αξίας. Με την συγκέντρωση στα business processes και στα συσχετιζόμενα semantics, το GRIA επιτρέπει στους χρήστες να ικανοποιούν τις υπολογιστικές απαιτήσεις τους με διαχείριση του κόστους και να αναπτύσσουν νέα επαγγελματικά μοντέλα για τις υπηρεσίες τους. Το GRIA παρέχει διάφορα λογισμικά πακέτα που είναι δωρεάν , καθένα από τα οποία σχεδιάστηκε για συγκεκριμένες επιχειρησιακές ανάγκες.



Εικόνα 3 : Τα λογισμικά πακέτα του GRIA. Από τον ιστότοπο του GRIA (www.gria.org).

- το πακέτο Basic Application Services επιτρέπει σε έναν οργανισμό με εγκαταστάσεις grid computing να παρέχει υπηρεσίες αποθήκευσης δεδομένων και επεξεργασίας (χρησιμοποιώντας εφαρμογές που είναι εγκατεστημένες στο πλέγμα).

- Το πακέτο OGSA-DAI application επιτρέπει σε έναν οργανισμό να δημοσιεύσει πηγές δομημένης πληροφορίας (XML, Σχεσιακές) για εγγραφή στην υπηρεσία.
- Το πακέτο Service Provider Management παρέχει προαιρετική υποστήριξη για SLA βασισμένα στην διαχείριση των υπηρεσιών και σε κοστολόγηση βασισμένη σε ένα απλό πρωτόκολλο διαχείρισης τόσο για το GRIA όσο και για υπηρεσίες τρίτων.
- το πακέτο Client επιτρέπει στους χρήστες να έχουν πρόσβαση στην διαχείριση του GRIA και τις υπηρεσίες του μέσω desktop εφαρμογών, περιλαμβάνει δε και ένα API toolkit που μπορεί να χρησιμοποιηθεί για δημιουργία νέων εφαρμογών.
- το client management πακέτο, παρέχει προαιρετική υποστήριξη για την διαχείριση των υπηρεσιών σε επίπεδο οργανισμού, με κεντρικό έλεγχο και παρακολούθηση της χρήσης των υπηρεσιών.
- το Service developer's Toolkit επιτρέπει την ενσωμάτωση των δομών ασφαλείας και διαχείρισης του GRIA σε ήδη δημιουργημένες υπηρεσίες.

Στις επόμενες ενότητες, περιγράφονται οι ιδέες γύρω από την αρχιτεκτονική του GRIA τόσο από την επιχειρησιακή οπτική , όσο και από την οπτική των εφαρμογών. το GRIA χρησιμοποιεί τα standard Web Service Πρωτόκολλα και μηχανισμούς ασφαλείας και μπορούν να συνεργαστούν με .NET εφαρμογές και υπηρεσίες , παρότι τα πακέτα του GRIA και το API υλοποιήθηκαν χρησιμοποιώντας Java.

2.3.2 Η επιχειρησιακή οπτική

Εκπληρώνοντας τις ανάγκες των επιχειρήσεων

Το GRIA είναι το πρώτο Grid middleware σχεδιασμένο να χρησιμοποιεί υπηρεσιοστρεφή αρχιτεκτονική για να υποστηρίξει επιχειρησιακούς χρήστες του πλέγματος μέσω παροχής υπηρεσιών. Οι υπηρεσίες πλέγματος είναι ένα εξειδικευμένο είδος υπηρεσιών Ιστού που υποστηρίζουν τις μακρόβιες, εντατικές-σε-πόρους δραστηριότητες, οι οποίες μπορούν να απαιτήσουν τους συνδυασμένους πόρους των πολλαπλάσιων φορέων παροχής υπηρεσιών πλέγματος, και μπορούν να περιλάβουν τη διανομή αυτών των πόρων μεταξύ των χρηστών από τις διαφορετικές οργανώσεις. Τέτοιες δραστηριότητες βρίσκονται μέσα στους επιχειρησιακούς τομείς όπως η εφαρμοσμένη μηχανική, η χρηματοδότηση, οι βιολογικές επιστήμες, η παραγωγή μέσων και η υγειονομική περίθαλψη, και στην ακαδημαϊκή έρευνα. Μπορούν να περιλάβουν τους πόρους όλων των ειδών: πηγές στοιχείων, αποθήκευση, εφαρμογές και επεξεργασία. Στα ακαδημαϊκά πλέγματα, η διανομή των πόρων οργανώνεται συνήθως από τους φορείς παροχής υπηρεσιών, οι οποίοι συμφωνούν να μοιραστούν τους πόρους τους για να δημιουργήσουν μια

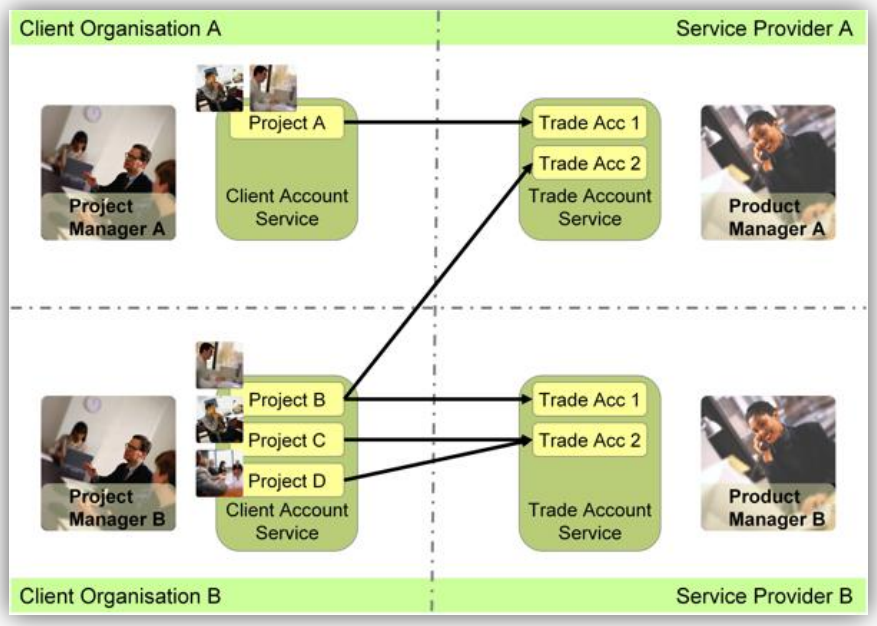
ενιαία "λίμνη των πόρων" που μπορεί να ικανοποιήσει τις ανάγκες μιας κοινής κοινότητας χρηστών. Εντούτοις, αυτό περιλαμβάνει τους φορείς παροχής υπηρεσιών και τους χρήστες που υπογράφουν μια "εικονική οργάνωση" με έναν κοινό στόχο (αυτός της σχετικής κοινότητας χρηστών), και που συμφωνούν με τις ενοποιημένες πολιτικές για τη διαχείριση και τη χορήγηση της πρόσβασης χρηστών στους κοινούς πόρους. Αυτή η ρύθμιση είναι πολύ ακριβή να καθιερωθεί και να λειτουργήσει, δεν ικανοποιεί τις επιχειρησιακές ανάγκες των εμπορικών φορέων παροχής υπηρεσιών ή των καταναλωτών, και δεν υπάρχει κανένα περιθώριο για τους συμμετέχοντες να ανταγωνιστούν ανοιχτά την τιμή ή την ποιότητα της υπηρεσίας. Το GRIA έχει ως σκοπό να υποστηρίξει τις απαιτήσεις πυρήνων για τις επιχειρησιακές εφαρμογές.

- Οι φορείς παροχής υπηρεσιών πρέπει να είναι σε θέση να λειτουργήσουν ανεξάρτητα ο ένας από τον άλλον, και να ανταγωνιστούν ανάλογα με τις ανάγκες για να παρέχουν τις υπηρεσίες στην πληρωμή των πελατών.
- Οι πελάτες πρέπει να είναι σε θέση να ελέγξουν ποιες υπηρεσίες καταναλώνουν, πόσο χρησιμοποιούνται, και από ποιους.
- Οι υπηρεσίες πρέπει να υπόκεινται στις συμφωνίες επιπέδων υπηρεσιών έτσι οι προμηθευτές ξέρουν ποια υπηρεσία πρέπει να παρέχουν, και οι πελάτες ξέρουν ποια υπηρεσία θα λάβουν και σε ποια τιμή.
- η ασφάλεια πρέπει να ακολουθεί τα εμπορικά πρότυπα
- οι φορείς παροχής υπηρεσιών πρέπει να είναι σε θέση λειτουργούν μέσα στους όρους των σχετικών αδειών προγραμμάτων εφαρμογών.
- τα μηνύματα που ανταλλάσσονται πρέπει να προσαρμοστούν στα σχετικά πρότυπα.
- οι λειτουργικές δαπάνες πρέπει να ελαχιστοποιηθούν

Για να συναντήσει αυτούς τους στόχους, το GRIA χρησιμοποιεί μια πλήρως αποκεντρωμένη διοικητική προσέγγιση, με την ελάχιστη εξάρτηση μεταξύ των περιοχών. Κάθε περιοχή που προσφέρει τις υπηρεσίες GRIA λαμβάνει τις επιχειρησιακές αποφάσεις της για τις οποίες οι χρήστες στην εμπιστοσύνη και με ποιους όρους, και είναι αρμόδιοι για την επιβολή των πολιτικών πρόσβασής του και την απόφαση ποιων εφαρμογών να υποστηρίξουν. Οι περιοχές μπορούν να αλληλεπιδράσουν η μια με την άλλη, αλλά αυτό οδηγείται από τους κοινούς καταναλωτές τους, και εκείνοι οι καταναλωτές είναι αρμόδιοι για τη διαχείριση των προκυπτουσών εξαρτήσεων. Δεν υπάρχει καμία σφαιρική συμφωνία που οργανώνεται, και καμία εικονική οργάνωση δεν χρειάζεται να ιδρυθεί, αν και οι χρήστες μπορούν να αλληλεπιδράσουν σύμφωνα με τα εικονικά πρότυπα οργάνωσης εάν θέλουν.

2.3.3 Διαχείριση των Επιχειρησιακών Σχέσεων

Το GRIA υποστηρίζει την διαχείριση των υπηρεσιακών σχέσεων διαμέσου συμβατικών επιχειρησιακών μοντέλων. Όταν ένας καταναλωτής θέλει να αγοράσει τις υπηρεσίες από έναν προμηθευτή, πρέπει πρώτα να ανοίξουν έναν εμπορικό λογαριασμό με το φορέα παροχής υπηρεσιών. Αυτός ο εμπορικός λογαριασμός αντιπροσωπεύει μια σχέση εμπιστοσύνης μεταξύ ενός πελάτη και ενός φορέα παροχής υπηρεσιών, βασισμένη στην προθυμία του πελάτη να πληρώσει για τις παρεχόμενες υπηρεσίες. Οι δύο πλευρές μπορούν να περιορίσουν το επίπεδο εμπιστοσύνης με τη διευκρίνιση ενός πιστωτικού ορίου για κάθε εμπορικό λογαριασμό, ο οποίος αντιπροσωπεύει το μέγιστο ποσό υπηρεσίας που ο προμηθευτής είναι πρόθυμος να παραδώσει πριν πληρωθεί, ή το μέγιστο ποσό υπηρεσίας που ο καταναλωτής είναι πρόθυμος να πληρώσει, οποιοδήποτε είναι μικρότερο. Οι σχέσεις μεταξύ των φορέων παροχής υπηρεσιών και των οργανώσεων καταναλωτών μπορούν να είναι σύνθετες. Μια τυπική οργάνωση καταναλωτών μπορεί να έχει διάφορες σχέσεις με κάθε έναν από διάφορους προμηθευτές, καθένας να χρησιμοποιείται από ένα διαφορετικό τμήμα ή πρόγραμμα, και κάθε ένας να καθιερώνεται και να ρυθμίζεται από τους διαφορετικούς εξουσιοδοτημένους υπαλλήλους (π.χ. διευθυντές προγράμματος), όπως παρουσιάζεται παρακάτω.



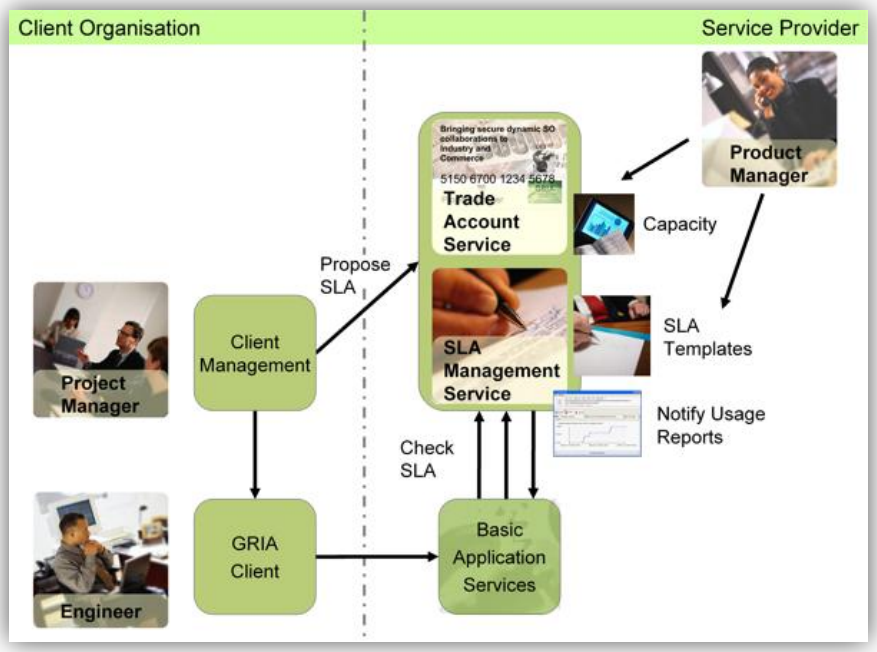
Εικόνα 4: Οι υπηρεσιακές σχέσεις μεταξύ οργανισμών.
Από την ιστοσελίδα του GRIA (www.gria.org).

Είναι σημαντικό να υπογραμμιστεί ότι η υπηρεσία εμπορικής λογιστικής είναι κάτι καθολικό, όχι μια τραπεζική υπηρεσία. Καταγράφει χρήματα που μια οργάνωση οφείλει ή της οφείλονται, όχι

χρήματα που πραγματικά έχει. Δεν εγγυάται (με τον τρόπο που υποχρεούται μια τράπεζα) ότι αυτά τα αρχεία θα είναι 100% πλήρη και ακριβή, το οποίο δεν είναι εφικτό χωρίς αξιόπιστα δίκτυα και εγκαταστάσεις επαναφοράς, όπως λειτουργεί (έναντι αμοιβής) μια τράπεζα. Ένας φορέας παροχής υπηρεσιών δεν μπορεί να πάρει χρήματα από την υπηρεσία λογιστικής, ή να μεταχειριστεί το υπόλοιπο ενός λογαριασμού ως μετρητά που μπορούν να χρησιμοποιηθούν για να πληρωθούν άλλοι, αλλά μπορεί να χρησιμοποιήσει την υπηρεσία λογιστικής για να παραγάγει τα τιμολόγια που στέλνονται στους πελάτες. Ο πελάτης μπορεί να αμφισβητήσει τις λεπτομέρειες ενός τέτοιου τιμολογίου εάν έχουν υπάρξει οποιαδήποτε λειτουργικά ή λάθη δικτύων, αλλά το GRIA μπορεί να βοηθήσει να επιλυθούν τέτοια λάθη επειδή υπογράφει και καταγράφει κάθε μήνυμα που υποβάλλεται προς επεξεργασία από τις υπηρεσίες του. Όταν ο πελάτης πληρώνει το λογαριασμό, αυτό πρέπει να αναφερθεί στον GRIA έτσι δεν τιμολογείται πάλι για τις ίδιες υπηρεσίες.

2.3.4 Συνεργασία υπό όρους

Το GRIA επιτρέπει στους φορείς παροχής υπηρεσιών και τους πελάτες να ανταλλάξουν πόρους (εφαρμογές, στοιχεία, επεξεργασία, αποθήκευση) υπό τον όρο των διμερών συμφωνιών επιπέδων υπηρεσιών (SLAs). Ένα SLA περιγράφει την ποιότητα της υπηρεσίας (QoS) και άλλων υποχρεώσεων από έναν φορέα παροχής υπηρεσιών σε αντάλλαγμα των οικονομικών υποχρεώσεων από έναν πελάτη ενάντια σε ένα συμφωνηθέν πρόγραμμα των τιμών και των πληρωμών. Το GRIA επιτρέπει στους φορείς παροχής υπηρεσιών για να προωθήσει τα πρότυπα SLA που προτείνονται από τους πελάτες κατά τη διάρκεια της διαπραγμάτευσης SLA.



Εικόνα 5: Το SLA απο την μεριά του πελάτη και του παροχου.
Από την ιστοσελίδα του GRIA (www.gria.org).

2.3.5 Διαχείριση SLA

Οι φορείς παροχής υπηρεσιών επεκτείνουν τις υπηρεσίες εφαρμογής αρμόδιες για την επιχειρησιακή λειτουργία τους. Αυτές οι υπηρεσίες παράγουν τις εκθέσεις χρήσης χρησιμοποιώντας τα κριτήρια QoS που μπορούν να είναι ποιοτικά (π.χ. όροι λάθους) ή ποσοτικά (π.χ. χρόνος επεξεργασίας, στοιχεία που μεταφέρονται). Το GRIA χρησιμοποιεί αυτές τις εκθέσεις για να ελέγξει τη χρήση πελατών και το επίπεδο υποχρεώσεων από τις υπάρχουσες συμφωνίες έναντι της διαθέσιμης χωρητικότητας. Το GRIA μπορεί έπειτα αυτόματα να καθορίσει:

- εάν για να εισαγάγει σε ένα νέο SLA (με ένα δεδομένο QoS) όταν ζητείται από έναν πελάτη
- εάν μια απαιτούμενη υπηρεσία καλύπτεται με μια ύπαρξη SLA, και εάν η υπηρεσία μπορεί να παρασχεθεί μέσα στην ικανότητα του προμηθευτή
- ποιο SLA πρέπει να παραβιαστεί όταν η ικανότητα είναι περίπου να ξεπεραστεί, και πώς να μειώσει τα φορτία στην αντίστοιχη υπηρεσία εφαρμογής
- όταν ένας καταναλωτής υπερβαίνει τα όρια ενός SLA, και πώς να αποτρέψει αυτό με τη μείωση της κατανάλωσης υπηρεσιών στην αντίστοιχη υπηρεσία εφαρμογής και
- τι επίπεδο και ποιότητα της υπηρεσίας παραδίδεται πραγματικά, και πόσο να τιμολογήσει για αυτό.

Για να εισαχθεί σε ένα νέο SLA, ένας πελάτης πρέπει να εγκριθεί για να τιμολογήσει τις υπηρεσίες που παρέχονται κάτω από το νέο SLA σε έναν λογαριασμό χρέωσης του GRIA, και ο λογαριασμός πρέπει να παρέχει ικανοποιητική πίστωση για να καλύψει την κλίμακα των υπηρεσιών που θα παρεχόταν κάτω από το SLA. Αυτό σημαίνει ότι ο διευθυντής των υπηρεσιών GRIA μπορεί να ελέγξει πόση υπηρεσία θα παρασχεθεί και σε ποιούς πελάτες, αλλά πρέπει να λάβει μόνο μια επιχειρησιακή απόφαση: ποιο πιστωτικό όριο να θέσει στον εμπορικό λογαριασμό του πελάτη. Μετά από αυτό, όλα είναι αυτοματοποιημένα, κάτι που καθιστά τις υπηρεσίες του GRIA ευέλικτες στις νέες ανάγκες των χρηστών, και παράλληλα ανέξοδες να λειτουργήσουν για τους προμηθευτές. Αυτό είναι κρίσιμο σε ένα επιχειρησιακό πλέγμα όπου το ανθρώπινο δυναμικό μπορεί εύκολα να γίνει το μεγαλύτερο κόστος κατά τρέξιμο των υπηρεσιών - στο GRIA αυτές οι δαπάνες ελαχιστοποιούνται, έτσι οι υπηρεσίες μπορούν να είναι κερδοφόρες για τους προμηθευτές όντας προσιτές για τους πελάτες.

2.3.6 Γενική επισκόπηση του GRIA Basic Application Services

Το GRIA Basic Application Services παρέχει τη βασική λειτουργία για τη διαχείριση εργασιών και δεδομένων. Αποτελείται από:

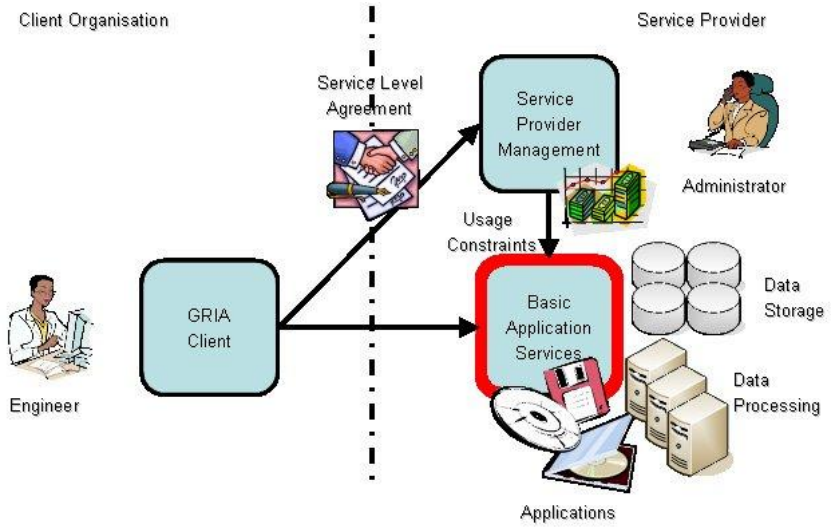
Μια υπηρεσία Δεδομένων – Data Service :

> Αυτό επιτρέπει στους μακρινούς χρήστες να φορτώσουν και να μεταφορτώσουν τα αρχεία δεδομένων στο φορέα παροχής υπηρεσιών, και την μεταφορά δεδομένων μεταξύ των υπηρεσιών που φιλοξενούνται από τους διαφορετικούς φορείς παροχής υπηρεσιών. Η υπηρεσία δεδομένων υποστηρίζει επίσης τη διαχείριση των δικαιωμάτων πρόσβασης (για εγγραφή ή ανάγνωση) που χορηγούνται σε άλλους χρήστες ή φορείς παροχής υπηρεσιών.

Μια υπηρεσία εργασιών – Job Service :

> Αυτό επιτρέπει στους απομακρυσμένους χρήστες να αρχίσουν, να ελέγξουν ή να τερματίσουν τις υπολογιστικές εργασίες, που εκτελούνται από το φορέα παροχής υπηρεσιών. Η υπηρεσία εργασίας θα προσκομίσει την εισαγωγή από και θα γράψει το αποτέλεσμα σε μια τοπική υπηρεσία δεδομένων. Η υπηρεσία εργασίας μπορεί να διαμορφωθεί για να υποστηρίξει τις πολλαπλάσιες εφαρμογές, οι οποίες επιλέγονται από το φορέα παροχής υπηρεσιών.

Οι υπηρεσίες εφαρμογής μπορούν να διαμορφωθούν για να είναι είτε (δωρεάν) είτε διοικούμενες από το GRIA Service Provider Management package, όπως στο διάγραμμα παρακάτω:



Εικόνα 6 : Η διασύνδεση των υπηρεσιών του GRIA.
Απο το ιστότοπο του GRIA (www.gria.org).

2.3.7 Η υπηρεσία δεδομένων - Data Service

Επισκόπηση της υπηρεσίας δεδομένων

Η υπηρεσία δεδομένων του GRIA χρησιμοποιείται για να διαχειριστεί Data Stagers. Data Stager είναι ένα περίβλημα για ένα ενιαίο αρχείο (ή αρχείο Zip). Έχει ένα μοναδικό προσδιοριστικό και ένα σύστημα ελέγχου πρόσβασης για ποιος μπορεί να διαβάσει και να γράψει δεδομένα. Οι πελάτες μπορούν να χρησιμοποιήσουν την υπηρεσία για να δημιουργήσουν νέα stagers, να φορτώσουν και να μεταφορτώσουν δεδομένα, να μεταφέρουν δεδομένα μεταξύ stager, και να ελέγχουν την πρόσβαση.

Βασικό στοιχεία διαμόρφωσης της υπηρεσίας δεδομένων είναι :

- *Η θέση του καταλόγου στοιχείων ρίζας :*
  Η υπηρεσία αποθηκεύει οποιαδήποτε φορτωμένα στοιχεία μέσα σε αυτόν τον κατάλογο. Εάν η υπηρεσία δεδομένων πρόκειται να χρησιμοποιηθεί με μια υπηρεσία εργασίας και οι εργασίες πρόκειται να εκτελεστούν σε ένα πλέγμα έπειτα τα μηχανήματα του πλέγματος πρέπει να είναι σε θέση να διαβάσουν και να γράψουν σε αυτόν τον κατάλογο.

2.3.8 Η υπηρεσία εργασίας - Job Service

Επισκόπηση της υπηρεσίας εργασίας

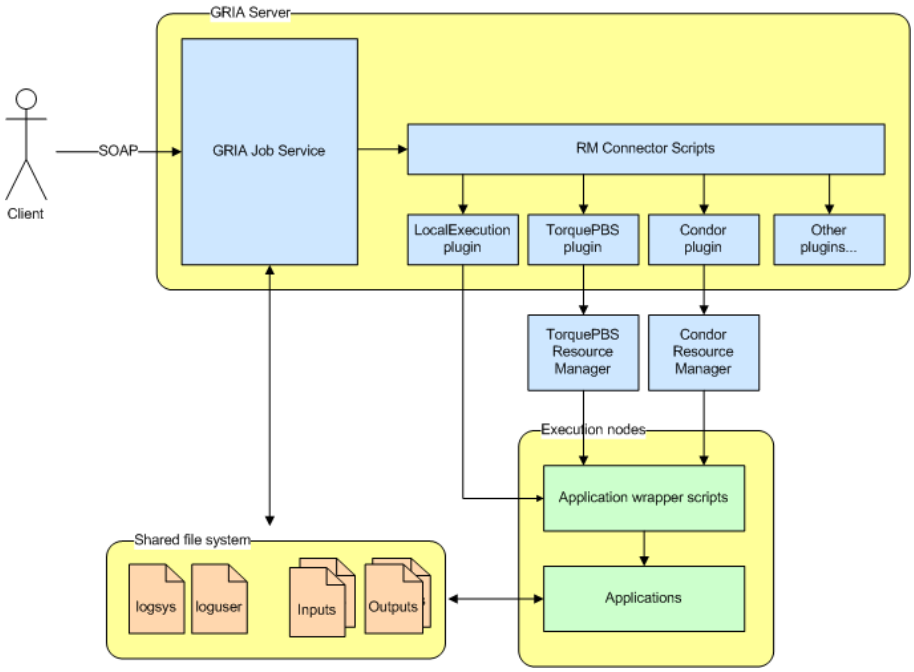
Η υπηρεσία εργασίας του GRIA χρησιμοποιείται για να διαχειριστεί τις εργασίες. Οι πελάτες μπορούν να χρησιμοποιήσουν την υπηρεσία για να δημιουργήσουν τις νέες θέσεις, να

φορτώσουν τα δεδομένα εισόδου, να αρχίσουν την εργασία, να παρακολουθήσουν την πρόοδο, και να μεταφορτώσουν τα αποτελέσματα.

Κάθε εισαγωγή και παραγωγή μιας εργασίας είναι πραγματικά ένας data stager διοικούμενος από την τοπική υπηρεσία δεδομένων (αυτή στο ίδιο .war με την υπηρεσία εργασίας). Επομένως, πρέπει να διαμορφωθεί η υπηρεσία δεδομένων προτού να μπορέσει να χρησιμοποιηθεί η υπηρεσία εργασίας. Οι χρήστες μπορούν να τρέξουν τις εργασίες που παίρνουν την είσοδο ή στέλνουν τα αποτελέσματα σε άλλες υπηρεσίες δεδομένων με τη χρησιμοποίηση των κανονικών χαρακτηριστικών γνωρισμάτων μεταφοράς που παρέχονται από την υπηρεσία δεδομένων.

2.3.9 Η αρχιτεκτονική της υπηρεσίας εργασίας

Η αρχιτεκτονική υπηρεσιών εργασίας του GRIA είναι αρκετά ευέλικτη να χρησιμοποιήσει ποικίλες κρυφές πλατφόρμες υπολογισμού για να τρέξει τις εργασίες π.χ. από υπολογιστές μέχρι συστοιχίες τερματικών σταθμών ή ακόμα και υπερυπολογιστές. Προκειμένου να επιτευχθεί αυτή η ευελιξία, η υπηρεσία εργασίας του GRIA έχει πρόσβαση στους πόρους έμμεσα μέσω των RM connector scripts του, τα οποία αποσυνδέουν την υπηρεσία εργασίας του GRIA από τους διαχειριστές των πόρων και τις εφαρμογές. Τα τμήματα του παρόντος κεφαλαίου δίνουν μια γενική εικόνα των διάφορων συστατικών της υπηρεσίας εργασίας:



Εικόνα 7: Συστατικά της GRIA Job Service .
Από τον ιστότοπο του GRIA (www.gria.org)

2.3.10 Διαχείριση εφαρμογών

Μετά από την εγκατάσταση μιας εφαρμογής στην υπηρεσία εργασίας, διατίθεται για εκτέλεση από τους απομακρυσμένους πελάτες. Σημειώστε, εντούτοις, ότι οι πελάτες πρέπει συνήθως να ικανοποιήσουν τους πρόσθετους επιχειρησιακούς περιορισμούς, όπως η κατοχή μιας κατάλληλης συμφωνίας επιπέδων υπηρεσιών ή ενός λογαριασμού προτού να μπορέσουν να εκτελέσουν τις εγκατεστημένες εφαρμογές. Το GRIA Basic Application Services παρέχεται με ένα σύνολο αρχικών εφαρμογών. Η Web based διαχείριση παρέχει την τοποθεσία τους κατά την εγκατάσταση και οδηγεί τον χρήστη στην ανάπτυξη των εφαρμογών αυτών. Εκτός από τις αρχικές εφαρμογές, είναι απλό να αναπτυχθούν νέες εφαρμογές και να εγκατασταθούν με τον ίδιο τρόπο.

2.3.11 Εγκατάσταση εφαρμογών

Λαμβάνοντας ή δημιουργώντας τα αρχεία και τα scripts που απαιτούνται για την εγκατάσταση , η εφαρμογή μπορεί τώρα να εγκατασταθεί στην υπηρεσία εργασίας. Για να γίνει αυτό, απαιτείται η ύπαρξη του *Apache Tomcat Servlet Container*. Μέσα από την σελίδα διαχείρισης του basic application services και μετά την εισαγωγή των κατάλληλων πιστοποιητικών ασφαλείας μπορούμε να προχωρήσουμε στην διαδικασία της εγκατάστασης.

3

Ανάλυση Απαιτήσεων Συστήματος

Στο κεφάλαιο αυτό θα περιγράψουμε την αρχιτεκτονική και την λειτουργία του συστήματος. Το προς υλοποίηση σύστημα θα παρουσιαστεί αρχικά σε δύο κύρια μέρη. Το μέρος του Εξυπηρετητή (Server Side) και το μέρος του Πελάτη – χρήστη του Grid (Client Side). Μετά την συνοπτική περιγραφή των δυο μερών ακολουθεί ξεχωριστή αναφορά σε κάθε υποσύστημα.

Client Side: Από την πλευρά του πελάτη, Δράστες είναι οι κινητές συσκευές που στην περίπτωση μας θεωρούμε ότι τρέχουν το λειτουργικό σύστημα Android. Το υποσύστημα αυτό το ονομάζουμε Android Application. Ο χρήστης μπορεί να χρησιμοποιήσει εφαρμογές που έχουν αναπτυχθεί για το συγκεκριμένο λειτουργικό και να απευθυνθεί στο Grid με τον ίδιο τρόπο που θα το έκανε από έναν οποιοδήποτε άλλο υπολογιστή. Η διαδικασία είναι διαφανής , διατηρεί όμως ένα επίπεδο αφαίρεσης στην συνολική λειτουργικότητα ώστε ο χρήστης να μην αντιλαμβάνεται διαφορά σε σχέση με την χρήση υπολογιστή. Οι κύριες δυνατότητες που θα δίνονται μέσω της εφαρμογής θα είναι η έναρξη εργασίας στο Πλέγμα, η διακοπή εργασίας που βρίσκεται εν εξελίξει , η παρακολούθηση (monitoring) των εργασιών που τρέχουν καθώς και η αλληλεπίδραση με τους Data Stager για κάθε Εργασία (Φόρτωση Data Stager με δεδομένα προς επεξεργασία από την κινητή Συσκευή προς το Πλέγμα αλλά και επιστροφή του αποτελέσματος της επεξεργασίας από το Πλέγμα στην κινητή Συσκευή.

Server Side: Το μέρος του εξυπηρετητή απαρτίζεται ουσιαστικά από τρία επιμέρους υποσυστήματα όπως φαίνεται και στο σχήμα: Τον Axis Server, τον GRIA Server και τα Resources. Ο Axis Server, παίζει τον ρόλο ενός adapter ανάμεσα στην επικοινωνία τις κινητής συσκευής και του πλέγματος. Η λειτουργικότητα του παρουσιάζεται αναλυτικά παρακάτω, συνοπτικά όμως θα αναφέρουμε ότι χρησιμοποιεί Web Services για να δέχεται τις αιτήσεις που έρχονται από την εφαρμογή στην κινητή συσκευή και τις αναμεταδίδει στον GRIA Server, χρησιμοποιώντας το GRIA Client API. Ο λόγος της ύπαρξης του Axis Server , είναι αφενός η περιορισμένη υπολογιστική ισχύς στις κινητές συσκευές , αφετέρου η μη πλήρης υποστήριξη των Java βιβλιοθηκών από το Android. Οι δυο παραπάνω λόγοι αποτρέπουν την συγγραφή εφαρμογής στο Android, πλήρους λειτουργικότητας και πλήρους διασυνδέσεως με τις βιβλιοθήκες του GRIA.

Ο GRIA Server, είναι ο εξυπηρετητής που φιλοξενεί το GRIA-basic-Application, την εφαρμογή δηλαδή που αναλαμβάνει να διαχειριστεί ολόκληρο το πλέγμα και να εκτελέσει τις εντολές που διέρχονται μέσω του Axis Server. Είναι το κυριότερο μέρος του συστήματος μας αφού εδώ λαμβάνει χώρα η ουσιαστική επεξεργασία, είτε αυτό είναι το γέμισμα των Data Stager είτε είναι η εκτέλεση εργασιών κλπ. Τα Resources, αποτελούν την αποθήκη δεδομένων του συστήματος. Από εδώ το GRIA παίρνει τις εισόδους των εργασιών του και επιστρέφει τα αποτελέσματα της επεξεργασίας.

3.1 *Περιγραφή Λειτουργιών*

Σε αυτήν την ενότητα περιγράφουμε αναλυτικά τις λειτουργίες που απαιτείται να εκτελεί το σύστημα. Αφού έχουμε χωρίσει ήδη το σύστημά μας σε υποσυστήματα, περιγράφουμε το καθένα από αυτά ξεχωριστά.

3.1.1 Android Application

Η εφαρμογή μας στο Android έχει όνομα ***GRIA Manager*** και όπως έχει αναφερθεί, μεταφέρει μέρος της λειτουργικότητας ενός GRID Client στην οθόνη οποιασδήποτε φορητής συσκευής ικανής να τρέχει το λειτουργικό σύστημα Android. Ο χρήστης του GRIA Manager μπορεί :

- Να βλέπει τις διαθέσιμες εργασίες προς εκτέλεση στο Πλέγμα
- Να επιλέγει κάποια από αυτές και να αρχίζει την εκτέλεση της
- Να διακόπτει την εκτέλεση μίας εργασίας
- Να ανεβάζει δεδομένα σε κάποιον Data Stager
- Να κατεβάζει στην κινητή συσκευή τα αποτελέσματα από την εκτέλεση μίας εργασίας.

Για παράδειγμα , υποθέτουμε ότι υπάρχουν 3 εργασίες διαθέσιμες προς υποβολή στο πλέγμα με ονόματα Swirl , Paint και Blend. Στην συγκεκριμένη περίπτωση αναφερόμαστε σε εργασίες επεξεργασίας εικόνας που δέχονται ως ορίσματα δημοφιλή format εικόνων όπως .jpg, .png, gif κ.α. Ένα απλό σενάριο χρήσης θα ήτανε το παρακάτω:

Ο χρήστης βλέπει μέσα από το GRIA Manager τις 3 διαθέσιμες εργασίες και αποφασίζει να επιλέξει την Swirl. Το πρόγραμμα του γνωστοποιεί ότι για να εκκινήσει η swirl χρειάζεται να δημιουργηθεί ένας data stager που να περιέχει μία εικόνα προς επεξεργασία, ή να χρησιμοποιηθεί ένας ήδη υπάρχων data stager. Επιλέγεται η δημιουργία ενός νέου και ζητείται από τον χρήστη να διαλέξει μία εικόνα από την μνήμη της κινητής του συσκευής. Μετά την δημιουργία του data stager υποβάλλεται στον GRIA Server η αίτηση για μία νέα εργασία. Εκκινείτε μία νέα swirl και ο χρήστης μπορεί πλέον να παρακολουθεί την εξέλιξη από την κινητή του συσκευή, με την δυνατότητα να διακόψει την λειτουργία ανά πάσα στιγμή. Μετά το πέρας της swirl μπορούμε να κατεβάσουμε την επεξεργασμένη εικόνα να την δούμε και να την αποθηκεύσουμε. Τονίζουμε σε αυτό το σημείο ότι η λειτουργία της εφαρμογής γίνεται σε non-blocking mode επιτρέποντας την υποβολή πολλών εργασιών στο πλέγμα.

3.1.2 Axis Server

Σε αυτό το υποσύστημα , φιλοξενούνται web services (**WS**) σε Axis που βρίσκονται ενδιάμεσα στην επικοινωνία του Android Application και του GRIA Server. Ο Axis Server δέχεται τις αιτήσεις που προέρχονται από την κινητή συσκευή , οποιαδήποτε από τις λειτουργίες που περιγράφηκαν στην προηγούμενη υποενότητα, καλεί το κατάλληλο web service και αναλαμβάνει την αποστολή των δεδομένων για επεξεργασία στα Resources. Επιγραμματικά η λειτουργία του Axis Server συνοψίζεται στα εξής :

- Δημιουργία και διαχείριση των Data Stager στον GRIA Server.
- Ανέβασμα και κατέβασμα αρχείων από τα Resources.
- Ανακάλυψη ενεργών εργασιών στο πλέγμα.
- Εκκίνηση νέων εργασιών και τερματισμός εκτελούμενων.

Για παράδειγμα , εξετάζοντας το σενάριο εκτέλεσης που παρουσιάστηκε στην προηγούμενη ενότητα από την μεριά αυτού του υποσυστήματος θα έχουμε: Αρχικά ο Axis Server λαμβάνει αίτηση ανακάλυψης ενεργών εργασιών οπότε την προωθεί στον GRIA περιμένοντας απάντηση. Η απάντηση αυτή επιστρέφεται στο Android Application και μία νέα αίτηση καταφθάνει αυτή την φορά για την δημιουργία ενός νέου Data Stager. Η αίτηση αυτή περιέχει και μία επισυναπτόμενη εικόνα που στέλνεται στα Resources. Τέλος λαμβάνεται η εντολή εκκίνησης της swirl που προωθείται και αυτή με την σειρά της στο GRIA. Παρατηρούμε μέσα από αυτό το παράδειγμα ότι ο ρόλος του συγκεκριμένου υποσυστήματος είναι υποστηρικτικός προς τον τελικό GRIA Server και η ύπαρξη του δεν θα ήτανε αναγκαία εάν το Android δεχόταν όλες τις απαραίτητες βιβλιοθήκες για το GRIA API. Παρόλα αυτά η χρήση WS καθιστά τελικά την υλοποίηση μας ανεξάρτητη λειτουργικού και γλώσσα υλοποίησης. Μια νέα εφαρμογή γραμμένη για παράδειγμα σε C++ θα μπορούσε να αναπτυχθεί και να εκμεταλλευτεί τα υπάρχοντα υποσυστήματα χωρίς κάποια αλλαγή στην δομή τους.

3.1.3 GRIA Server

Ο GRIA Server είναι το κεντρικότερο υποσύστημα. Πρόκειται για έναν διαχειριστικό κόμβο στον οποίο καταλήγουν όλες οι αιτήσεις από τον Axis Server ,είτε είναι πυροδοτούμενες από την χρήστη είτε όχι, και αναλαμβάνει την επεξεργασία των αιτήσεων καθώς και μια σειρά αποφάσεων που αφορούν το πλέγμα. Στις αποφάσεις αυτές περιλαμβάνονται η ανακάλυψη υπηρεσιών ,η δέσμευση πόρων , η σύναψη SLA κ.α. πάντα κάτω από τις προδιαγραφές που καθορίζονται από τα WS – Standards. Στον GRIA Server βρίσκεται εγκατεστημένη η εφαρμογή Basic Application Services που παρέχει την κύρια λειτουργικότητα για την διαχείριση των εργασιών και των δεδομένων. Αποτελείται από :

- Την Υπηρεσία Δεδομένων (Data Service)

  Αυτή παρέχει στους απομακρυσμένους χρήστες να ανεβάσουν και να κατεβάσουν δεδομένα στον παροχέα υπηρεσιών και να μεταφέρουν δεδομένα μεταξύ Υπηρεσιών Δεδομένων που φιλοξενούνται σε διαφορετικούς παρόχους υπηρεσιών. Η υπηρεσία δεδομένων υποστηρίζει επίσης την διαχείριση των δικαιωμάτων πρόσβασης (για πρόσβαση ανάγνωσης ή εγγραφής – ανάγνωσης) που παρέχονται σε άλλους χρήστες ή παρόχους υπηρεσιών.

- Την Υπηρεσία Εργασιών (Job Service)

  Αυτή επιτρέπει στους απομακρυσμένους χρήστες να αρχίζουν, να ελέγχουν ή να τερματίσουν υπολογιστικές εργασίες , που εκτελούνται από τον πάροχο υπηρεσιών. Η υπηρεσία εργασιών θα λάβει ως είσοδο και θα γράψει σαν έξοδο σε μία τοπική υπηρεσία δεδομένων. Μπορεί επίσης να διαμορφωθεί ώστε να υποστηρίζει πολλαπλές εφαρμογές , οι οποίες επιλέγονται από τον πάροχο υπηρεσιών

3.1.4 Resources

Το υποσύστημα αυτό είναι ουσιαστικά μέρος του GRIA Basic Application και εδώ αποθηκεύονται όλα τα δεδομένα είτε αφορούν την υπηρεσία εργασιών, είτε αφορούν την υπηρεσία δεδομένων ακόμα και αρχεία ρυθμίσεων. Παρουσιάζεται σαν ξεχωριστό υποσύστημα για λόγους πληρότητας στην σχεδίαση, γεγονός που αποδεικνύεται από την σημασία του ρόλου των συστημάτων αποθήκευσης σε επεκτάσιμα συστήματα. Με αυτόν τον τρόπο καθίσταται δυνατή η ευκολότερη διαχείριση, η μεγαλύτερη απόδοση και η αξιοπιστία . Ο κύριος όγκος δεδομένων των Resources καταλαμβάνεται από αρχεία των Υπηρεσιών εργασιών και δεδομένων. Συγκεκριμένα:

Η Υπηρεσία Εργασιών χρησιμοποιείται για την διαχείριση των «data stager» . Ένας «data stager» είναι ο φλοιός που περικλείει ένα αρχείο (ή .zip αρχείο). Έχει ένα μοναδικό αναγνωριστικό και σύστημα ελέγχου δικαιωμάτων πρόσβασης για να καθορίζει ποιός μπορεί να διαβάσει και να γράψει δεδομένα. Για την σωστή λειτουργία της η υπηρεσία δεδομένων ορίζει την τοποθεσία του root directory μέσα στην αποθήκη Resources. Εκεί αποθηκεύει οποιοδήποτε αρχείο ανεβαίνει στον server και εάν πρόκειται να εκτελεστεί κάποια εργασία από μία συστοιχία υπολογιστών , τότε τα μηχανήματα της συστοιχίας πρέπει να μπορούν να διαβάζουν και να γράφουν σε αυτό το directory. Όσον αφορά την υπηρεσία εργασιών , αυτή αποθηκεύει το δικό της root directory μέσα στα Resources. Η υπηρεσία δημιουργεί ένα υποφάκελο στο root για κάθε εργασία. Ο φάκελος θα πρέπει να είναι στο ίδιο filesystem όπως και του φακέλου της υπηρεσίας δεδομένων , έτσι ώστε να δεδομένα να είναι άρρηκτα και αποδοτικά συνδεδεμένα καθώς οι εργασίες σταματούν και ξεκινούν. Διαφορετικά τα δεδομένα θα πρέπει να αντιγράφονται το οποίο είναι αργό και σπαταλά αποθηκευτικό χώρο.

4

Σχεδίαση Συστήματος

4.1 *Αρχιτεκτονική*

Σε αυτό το κεφάλαιο παρουσιάζουμε τα επιμέρους κομμάτια από τα οποία έχει κτιστεί ο κώδικάς της εφαρμογής σε όλα τα υποσυστήματα. Δεδομένου ότι η γλώσσα ανάπτυξης είναι η Java, κρίνεται σκόπιμη η παρουσίαση των σημαντικότερων κλάσεων για κάθε υποσύστημα. Για κάθε κλάση δίνεται μία συνοπτική περιγραφή, καθώς και ένας πίνακας με τις μεθόδους της.

4.2 *Περιγραφή Κλάσεων – [GRIA Manager]*

Για λόγους συνέπειας με την έως τώρα παρουσίαση θα ακολουθήσουμε και εδώ τον διαχωρισμό Client – Server, ξεκινώντας την περιγραφή της εφαρμογής από το μέρος του χρήστη και καταλήγοντας στις κλάσεις των Εξυπηρετητών.

4.2.1 [class] ClientHttpRequest.java

Η κλάση αυτή δημιουργεί και στέλνει αιτήσεις POST – HTTP με διάφορα δεδομένα συμπεριλαμβανομένων των αρχείων και των Cookies. Η λειτουργία της βασίζεται στην αναπαραγωγή του τρόπου λειτουργίας των html forms υποστηρίζοντας και την multipart/form-data request για την αποστολή αρχείων. Η κλάση αυτή χρησιμοποιείται εξολοκλήρου για την επικοινωνία του GRIA Manager με τον Axis Server. Οι κύριες μέθοδοι που την απαρτίζουν είναι :

Πίνακας 1 : Οι μέθοδοι της ClientHttpRequest κλάσης.

1	`protected void connect() throws IOException`
Δημιουργεί και συνδέεται σε ένα URLConnection, ανοίγοντας ενα OutputStream.	
2	`protected void write(char c) throws IOException` `protected void write(String s) throws IOException`
Γράφει στο ήδη ανοιχτό URLConnection.	

3	`public ClientHttpRequest(URLConnection connection) throws IOException`
	`public ClientHttpRequest(URL url) throws IOException`
	`public ClientHttpRequest(URL urlString) throws IOException`

Δημιουργεί μία καινούργια Multipart POST-HTTP αίτηση στην σύνδεση που καθορίζεται από τα ορίσματα.	
4	`private void postCookies()`
Κάνει POST όλα τα Cookies που έχουν δημιουργηθεί.	
5	`public void setCookie(String name, String value) throws IOException`
Προσθέτει ένα Cookie στην αίτηση.	
6	`public void setCookies(Map cookies) throws IOException` `public void setCookies(Map cookies) throws IOException`
Προσθέτει πολλαπλά Cookies στην αίτηση με "name-to-value" Map.	
7	`public void setParameter(String name, String value) throws IOException`
Προσθέτει μια String παράμετρο στην αίτηση.	
8	`private static void pipe(InputStream in, OutputStream out) throws IOException`
Δημιουργεί το pipe της επικοινωνίας.	
9	`public void setParameter(String name, String filename, InputStream is) throws IOException`

	public void setParameter(String name, File file) **throws** IOException **public void** setParameter(String name, Object object) **throws** IOException **public void** setParameter(String name, Object object) **throws** IOException
Προσθέτει File παραμέτρους στην αίτηση.Τα αρχεία καθορίζονται με έναν από τους τροπους που φαίνονται από τα ορίσματα.	
10	**public void** setParameters(Map parameters) **throws** IOException **public void** setParameters(Object[] parameters) **throws** IOException
Προσθέτει πολλαπλές παραμέτρους στην αίτηση μονομιάς, χρησιμοποιώντας είτε "name-to-value" Map , είτε πίνακα αντικειμένων. Εάν η τιμή της παραμέτρου είναι αρχείο τότε αυτό γίνεται upload, αλλιώς γίνεται String και προστείθεται στην αίτηση.	
11	**public** InputStream post() **throws** IOException **public** InputStream post(Map parameters) **throws** IOException **public** InputStream post(Object[] parameters) **throws** IOException **public** InputStream post(Object[] parameters) **throws** IOException **public** InputStream post(String[] cookies, Object[] parameters) **throws** IOException **public** InputStream post(String name, Object value) **throws** IOException **public** InputStream post(String name1, Object value1, String name2, Object value2) **throws** IOException **public static** InputStream post(URL url, Map parameters) **throws** IOException **public static** InputStream post(URL url, Object[] parameters) **throws** IOException **public static** InputStream post(URL url, Map cookies, Map parameters) **throws** IOException **public static** InputStream post(URL url, String[] cookies, Object[] parameters) **throws** IOException
Κάνει POST τις αιτήσεις στον Server, μαζί με όλα τα Cookies και τις παραμέτρους που δόθηκαν.	

4.2.2 [class] Operations.java

Η κλάση αυτή εξυπηρετεί την συγκέντρωση όλων των συχνά χρησιμοποιούμενων μεθόδων σε ένα μέρος. Περιλαμβάνει κύριες λειτουργίες τις εφαρμογής , αλλά και δευτερεύουσες όπως την συγχώνευση δύο πινάκων κ.α. Οι μέθοδοι που την απαρτίζουν είναι :

Πίνακας 2: Οι μέθοδοι της Operations.java κλάσης.

1	**public static boolean** makeStagerLocal(Context context,String filename)	
Δημιουργεί και συνδέεται σε ένα URLConnection, ανοίγοντας ενα OutputStream.		
2	**public static void** StartJob(Context context,String APPLICATION,String[] inputfiles,String[] outputfiles)	
Στέλνει μια POST αίτηση για εκκίνηση της εργασίας που ορίζεται στις παραμέτρους, με ορίσματα εισόδου στον πίνακα inputfiles και εξόδου στον outputfiles.		
3	**public static boolean** createStager(Context context,String filename,File file)	
Στέλνει μια POST αίτηση για την δημιουργία ένος Data Stager, μαζί με το όνομα και το περιεχόμενο του αρχείου που είναι επιθυμητό να φορτωθέι.		
4	**public static boolean** fileUpload(Context context,String filename,File file)	
Κάνει upload στα Resources του αρχείου που καθορίζεται από τα ορίσματα.		
5	**public static** String[] FilterByExtention(String[] fileList,String file_extension)	**Convinience Methods**
[Βοηθητική Μέθοδος]: Δέχεται σαν έισοδο ένα πίνακα με ονόματα αρχείων (πρόθεμα + κατάληξη) και την κατάληξη στην οποία θέλουμε να εφαρμοστεί το φίλτρο και μας επιστέφει πίνακα που περιέχει μόνο τα αρχεία αυτής της κατάληξης. Χρησιμοποιείται για την επιλογή συγκεκριμένων τύπων αρχείων από τον δίσκο της κινητής συσκευής.		
6	**public static** String[] mergeArrays(String[] array1,String[] array2)	
[Βοηθητική Μέθοδος]: Συγχωνεύει τους 2 πίνακες συμβολοσειρών που δέχεται σαν είσοδο.		

7	`public static byte[] getBytesFromFile(File file) throws IOException`	
[Βοηθητική Μέθοδος]: Μετατρέπει ένα αρχείο σε byte[].		
8	`public static String convertStreamToString(InputStream is)`	
[Βοηθητική Μέθοδος]: Μετατρέπει ένα InputStream σε String.		

4.2.3 [class] UserInterface.java

Η κλάση αυτή επεκτείνει την Activity (κλάση συστήματος) και είναι κεντρική (Γονέας) για την Android εφαρμογή. Ενεργοποιείται κατά την εκκίνηση και αποτελεί σημείο αφετηρίας για τις κλήσεις των υπολοίπων κλάσεων. Είναι επίσης υπεύθυνη για την λειτουργικότητα του κεντρικού μενού. Οι μέθοδοι που την αποτελούν είναι :

Πίνακας 3: Οι μέθοδοι της κλάσης UserInterface.

1	`public void onCreate(Bundle savedInstanceState)`	Override
[Υπερσκέλιση Μεθόδου]: Καλείται όταν η Δραστηριότητα δημιουργείται για πρώτη φορά.		
2	`protected void onActivityResult(int requestCode, int resultCode,Intent data)`	
[Υπερσκέλιση Μεθόδου]: Καλείται κατά την επιστροφή τον υπο-δραστηριοτήτων περιέχοντας το αποτέλεσμα της εκτελεσής τους (Επιτυχής,Αποτυχημένη κλπ).		
3	`public boolean createFile(String filename)`	
Δημιουργεί ένα αρχείο κειμένου (Για δοκιμαστικούς σκοπούς).		
4	`public String download(String address, String localFileName)` `public String download(String address)`	
Κατεβάζει στον τοπικό δίσκο της κινητής συσκευής το αρχείο που βρίσκεται σε μία Web Adress. Για παράδειγμα από την διεύθυνση : http://www.domain.org/files/text.txt θα κατεβάσει το text.txt		

4.2.4 [class] SubActivity_1.java

Η κλάση αυτή είναι η πρώτη υπο – δραστηριότητα της εφαρμογής. Επεκτείνει την Activity και προτρέπει τον χρήστη να επιλέξει μία εικόνα από την βιβλιοθήκη του συστήματος ώστε να δημιουργηθεί ο αντίστοιχος Data Stager που να την περιέχει. Η εικόνα γίνεται upload στα Resources. Οι βασικές μέθοδοι της κλάσης αυτής είναι :

Πίνακας 4: Οι μέθοδοι της κλάσης SubActivity_1.

1	`public void onCreate(Bundle savedInstanceState)`	Override
[Υπερσκέλιση Μεθόδου]: Καλείται όταν η Δραστηριότητα δημιουργείται για πρώτη φορά.		
2	`public void onCreateContextMenu(ContextMenu menu, View v, ContextMenuInfo menuInfo)`	
[Υπερσκέλιση Μεθόδου]: Καλείται κατά την δημιουργία του δευτερεύοντος μενού, που ενεργοποιείται από το συνεχές πάτημα του μεσαίο κουμπιού επιλογής στην συσκευή.		
3	`public boolean onContextItemSelected(MenuItem item)`	
[Υπερσκέλιση Μεθόδου]: Καλείται όταν επιλεχθεί ένα αντικείμενο από το δευτερεύον μενού και αναγνωρίζει ποιο είναι αυτό το αντικείμενο.		

4.2.5 [class] ImageAdapter.java

Η κλάση αυτή εμπεριέχεται στην SubActivity_1 και είναι υπεύθυνη για την δημιουργία μία Gallery φωτογραφιών από αυτές που βρίσκονται στον δίσκο της κινητής συσκευής. Οι κύριες μέθοδοι που την απαρτίζουν είναι :

Πίνακας 5: Οι μέθοδοι της κλάσης ImageAdapter

1	`public ImageAdapter(Context c)`
Αποτελεί τον Constructor της κλάσης.	
2	`public int getCount()`
Βρίσκει τον αριθμό των αρχείων που θα χρησιμοποιηθούν για την Gallery.	
3	`public Object getItem(int position)` `public long getItemId(int position)`
Επιστρέφουν την θέση ενός στοιχείου στην Gallery.	
4	`public View getView(int position, View convertView, ViewGroup parent)`
Εμφανίζει την Gallery με τις ιδιότητες που καθορίζονται .	

4.2.6 *[class] SubActivity_2.java*

Η κλάση αυτή είναι η δεύτερη υπο-δραστηριότητα της Android εφαρμογής. Εδώ ο χρήστης μπορεί να επιλέξει έναν δημιουργημένο Data Stager και να κατεβάσει το αρχείο που αυτός περιέχει. Χρησιμεύει για την επιστροφή των αποτελεσμάτων στον χρήστη. Οι μέθοδοι της κλάσης αυτής είναι :

Πίνακας 6: Οι μέθοδοι της κλάσης SubActivity_2.

1	`protected Dialog onCreateDialog(int id)`	Override
[Υπερσκέλιση Μεθόδου]: Καλείται όταν δημιουργείται ο διάλογος επιλογής των Data Stager.		
2	`public void onCreate(Bundle savedInstanceState)`	
[Υπερσκέλιση Μεθόδου]: Καλείται όταν η Δραστηριότητα δημιουργείται για πρώτη φορά.		

4.2.7 *[class] SubActivity_3.java*

Η κλάση αυτή είναι η τρίτη υπό-δραστηριότητα. Εδώ ο χρήστης επιλέγει μία από τις διαθέσιμες εργασίες στο πλέγμα καθώς και τα ορίσματα που είναι απαραίτητα για την εκκίνηση της. Σημειώνουμε εδώ ότι τα ορίσματα μπορεί να είναι είτε κάποιος ήδη δημιουργημένος Data Stager είτε κάποιο αρχείο απευθείας. Οι βασικές μέθοδοι της κλάσης αυτής είναι :

Πίνακας 7: Οι μέθοδοι της κλάσης SubActivity_3.

1	`public void onCreate(Bundle savedInstanceState)`	Override
[Υπερσκέλιση Μεθόδου]: Καλείται όταν η Δραστηριότητα δημιουργείται για πρώτη φορά.		
2	`protected Dialog onCreateDialog(int id)`	
[Υπερσκέλιση Μεθόδου]: Καλείται όταν δημιουργείται ο διάλογος επιλογής των ορισμάτων.		
3	`private Spinner.OnItemSelectedListener spinnerListener = new Spinner.OnItemSelectedListener()`	
Δημιουργεί ένα αντικείμενο τύπου Spinner υπεύθυνο για την παρουσίαση των ενεργών εργασιών στον χρήστη.		
4	`public void onNothingSelected(AdapterView parent)`	
Ενεργοποιείται σε περίπτωση μη επιλογής εργασίας.		
5	`private void appendRow(TableLayout table,int index)`	

Προσθέτει μία νέα γραμμή στην διάταξη της οθόνης. Στην συγκεκριμένη περίπτωση TableLayout.

4.2.8 [class] TranslucentBlurActivity.java

Η κλάση αυτή επεκτείνει την Activity (κλάση συστήματος) και δίνει εφέ διαφάνειας στις δραστηριότητες που την υπερσκελίζουν. Η μοναδική μέθοδος που περιλαμβάνει είναι η onCreate() , που αναλαμβάνει χρέη Constructor.

4.2.9 [class] TransparentPanel.java

Απλή κλάση που δημιουργεί ένα διαφανές πλαίσιο διαλόγου ώστε να ενημερώνει τον χρήστη για την εξέλιξη των αιτημάτων του. Οι μέθοδοι που περιέχει είναι :

Πίνακας 8: Οι μέθοδοι της κλάσης TransparentPanel

1	**public** TransparentPanel(Context context) **public** TransparentPanel(Context context, AttributeSet attrs)
Κατασκευαστές της κλάσης.	
2	**private void** init()
Μέθοδος με αρχικοποιήσεις για την σχεδίαση του διαλόγου.	
3	**public void** setInnerPaint(Paint innerPaint) **public void** setBorderPaint(Paint borderPaint)
Μέθοδοι για τον καθορισμό του χρωματισμού συγκεκριμένων περιοχών του διαλόγου.	
4	**protected void** dispatchDraw(Canvas canvas)
[Υπερσκέλιση Μεθόδου]: Καλείται για να δημιουργηθεί ο διάλογος στον Canvas που ορίζεται.	

4.3 *Περιγραφή Κλάσεων – [Axis2 WS]*

Σε αυτό το σημείο παρουσιάζουμε τις κλάσεις που απαρτίζουν τον Axis Server, καθώς και τις σημαντικότερες μεθόδους για κάθε κλάση.

4.3.1 [class] GRIAServicesAdapter.java

Η κλάση αυτή είναι η κυριότερη γι αυτό το υποσύστημα. Οι μέθοδοι που την αποτελούν μετατρέπονται και τελικά εκτίθενται σαν Axis Services σε κάποιο URL. Τα services αυτά κάνοντας χρήση του GRIA API μετατρέπουν τις αιτήσεις της κινητής συσκευής σε κλήσεις προς τον GRIA Server. Στον παρακάτω πίνακα παρουσιάζονται οι μέθοδοι της GRIAServicesAdapter :

Πίνακας 9: Οι μέθοδοι της κλάσης GRIAServicesAdapter

1	**public static** String makeStagerLocal(String filename)
Η μέθοδος αυτή ψάχνει στα GRIA Resources για τον Data Stager που να περιέχει το αρχείο του ορίσματος. Αυτό επιτυγχάνεται με τον έλεγχο του αρχείου ***StagerLogFile.xml*** που διατηρείται η απεικόνιση *< ονόματος αρχείου – DataStager id >* Εφόσον η αναζήτηση είναι επιτυχής το αρχείο «κατεβαίνει» στον τοπικό δίσκο του συστήματος (χρήση μεθόδου 7).	
2	**public static** String createStagerAndFill(String filename,String data)
Αποτελεί την αντίστροφη της μεθόδου 1. Το αρχείο που δίνεται σαν όρισμα «ανεβαίνει» στα Resources αφού δημιουργηθεί ο αντίστοιχος Data Stager (χρήση μεθόδου 6). Έπειτα ενημερώνεται το αρχείο ***StagerLogFile.xml*** με την αντιστοιχία *< ονόματος Αρχείου – Stager ID >*.	
3	**public static** String receiveFile(String filename,String data)
Μέθοδος για την αποκωδικοποίηση ενός αρχείου που έχει μετατραπεί σε Base64 String format.	
4	**public static** String saveFile(String filename,**byte**[] data)
Μέθοδος που αποθηκεύει στον τοπικό δίσκο το αρχείο που δίνεται στα ορίσματα.	
5	**public** String[] discoverApplications(String input)
Η μέθοδος αυτή χρησιμοποιείται για την ανακάλυψη των ενεργών εργασιών στο πλέγμα. Επιστρέφονται σε πίνακα String[] τα URI των εργασιών.	
6	**public static** String upload(String filename)
Δέχεται σαν όρισμα το όνομα του αρχείου και δημιουργεί τον Data	

Stager που θα το δεχτεί. Η μέθοδος αυτή χρησιμοποιείται από την μέθοδο 2: *createStagerAndFill()* για το ανέβασμα του αρχείου.	
7	**public static** File download(String conversationId,String filename)
Είναι η αντίστροφη της Upload, κατεβάζει το αρχείο του πρώτου ορίσματος από τον Stager του δεύτερου ορίσματος.Χρήσιμοποιείται από την μέθοδο 1.	
8	**public static void** delete(String conversationId)
Διαγράφει τον Data Stager με το Id που δίνεται σαν όρισμα, μαζι με τα αρχεία που αυτός περιέχει.	
9	**public static boolean** executeFunction(String applicationUrl,String[] inputfiles,String[] outputfiles)
Στέλνει στον GRIA αίτηση εκκίνησης μία εργασίας , το URI της οποίας δίνεται σαν πρώτο όρισμα, με αρχεία είσόδου και εξόδου τα ορίσματα 2 και 3.	
10	**public static boolean** stopFunction(JobConversation jobConv)
Στέλνει στον GRIA αίτηση διακοπής μία εργασίας , το αναγνωριστικό της οποίας δίνεται σαν όρισμα.	

4.3.2 [class] Base64Service.java

Η κλάση αυτή χρησιμοποιείται για την αποκωδικοποίηση αρχείων που δίνονται σε μορφή String.Τα δεδομένα γίνονται πρώτα byte[] και έπειτα σώζονται στον δίσκο στην κατάλληλη μορφή. Οι μέθοδοι της είναι :

Πίνακας 10: Οι μέθοδοι της κλάσης Base64Service.

1	**public static** String getParam(String data)
Μετατρέπει το αρχείο από μορφή String σε Base64 byte[].	
2	**public static** String doBinaryWithByteArray(**byte**[] data)
Αναπαράγει το αρχέιο από τα byte[] δεδομένα και το αποθηκεύει στον δίσκο.	

4.3.3 [class] XmlUtilities.java

Σε αυτή την κλάση περιλαμβάνονται όλες εκείνες οι μέθοδοι που βοηθούν στην επεξεργασία xml αρχείων. Η δομή των xml που χρησιμοποιούνται παρουσιάζονται στην επόμενη ενότητα. Οι μέθοδοι της είναι :

Πίνακας 11: Οι μέθοδοι της κλάσης XmlUtilities.

1	**public static boolean** create(String filename)
Δημιουργεί ένα νέο xml αρχείο με το όνομα που ορίζεται.Η δομή του xml που θα δημιουργηθεί έχει καθοριστεί από πριν και παρουσιάζεται στην επόμενη ενότητα	
2	**public static boolean** isCreated(String filename)
Εξετάζει αν έχει δημιουργηθεί ήδη ένα αρχείο.	
3	**public static** String search(String file,String fileName)
Η συνάρτηση αυτή , αναζητά αν το αρχείο με όνομα fileName υπάρχει στο στο file.xml. Αν επιτύχει επιστρέφει το StagerID που αντιστοιχεί στο fileName.	
4	**public static** String[] getAttributes(String file)
Βρίσκει όλα τα attributes του xml αρχείου.	
5	**public static void** add(String filename,String newStagerID,String name)
Η συναρτηση αυτη προσθετει στο ηδη υπαρχον xml αρχειο το νεο StagerID (εφοσον αυτο δεν υπαρχει).	
6	**public static void** deleteByStagerID(String filename,String StagerID)
Διαγράφουμε τον κόμβο που περιέχει το οριζόμενο StagerID.	
7	**public static void** deleteByFileName(String filename,String FileName)
Διαγράφουμε τον κόμβο που περιέχει το οριζόμενο αρχέιο.	
8	**public static** String getFirst(String filename)
Επιστρέφει το πρώτο κόμβο του xml αρχείου.	

4.4 *Κωδικοποίηση αρχείων – [XML]*

Για την υλοποίηση ορισμένων λειτουργιών της εφαρμογής , κρίθηκε σκόπιμη η χρήση xml αρχείων και η αποφυγή δεύτερης βάσης δεδομένων (η πρώτη βάση διατηρείται από τον GRIA για εσωτερικές λειτουργίες). Τα xml αρχεία που χρησιμοποιούνται παρουσιάζονται αναλυτικά στον παρακάτω πίνακα:

Πίνακας 12: Τα XML αρχεία ρυθμίσεων.

	Όνομα Αρχείου	Χρήση
1	***StagerLog.xml***	Διατηρεί την απεικόνιση <αρχείο – StagerID> που χρησιμοποιείται για την ανεύρεση αρχείων από τα Resources.
2	***jobLog.xml***	Διατηρεί λεπτομέρειες για τις εκτελούμενες εργασίες.
3	***descriptions.xml***	Περιέχει την πλήρη περιγραφή των ενεργών εργασιών στο πλέγμα.

Στις επόμενες ενότητες εξετάζουμε αναλυτικά την μορφή του κάθε αρχείου.

4.4.1 [xml] StagerLog.xml

Σε αυτό το xml διατηρείται το όνομα του αρχείου μαζί με τον Data Stager που το περιέχει . Μας είναι έτσι πολύ εύκολο να ανακτήσουμε τα αποτελέσματα της επεξεργασίας του πλέγματος αφού χρειάζεται να γνωρίζουμε μονάχα το όνομα του αρχείου. Ένα τυπικό παράδειγμα του StagerLog.xml φαίνεται παρακάτω :

```
<?xml version="1.0" encoding="UTF-8"?>
<collection>
    <mapping>
        <StagerID>402880e5-1c106b01-011c-10a6e69d-0006</StagerID>
        <filename>gallery_photo_8.jpg</filename>
    </mapping>
    <mapping>
        <StagerID>402880e5-1c106b01-011c-10a9cb8a-0007</StagerID>
        <filename>gallery_photo_2.jpg</filename>
    </mapping>
    <mapping>
        <StagerID>402880e5-1c106b01-011c-10aa0ae1-0008</StagerID>
```

```
        <filename>gallery_photo_4.jpg</filename>
    </mapping>
    <mapping>
        <StagerID>402880e5-1c106b01-011c-10aa4861-0009</StagerID>
        <filename>gallery_photo_7.jpg</filename>
    </mapping>
</collection>
```

4.4.2 [xml] jobLog.xml

Η διατήρηση λεπτομερειών για τις τρέχουσες εργασίες είναι απαραίτητη προϋπόθεση για το monitoring αλλά και την αλληλεπίδραση του χρήστη με το σύστημα όπως στην περίπτωση διακοπής μίας εργασίας . Όπως φαίνεται και παρακάτω για κάθε εργασία που ξεκινά προσθέτουμε το όνομά της ,το ID της, την χρονική στιγμή έναρξης και λήξης και την τρέχουσα κατάσταση της. Μπορούμε έτσι να γνωρίζουμε ανά πάσα χρονική στιγμή πόσες εργασίες τρέχουν στο πλέγμα, τι τύπου είναι αν τελείωσαν ή όχι κλπ.

```
<?xml version="1.0" encoding="UTF-8"?>
<collection>
    <mapping>
        <JobID>402880e5-1c106b01-011c-10a9cb8a-0012</JobID>
        <jobname>swirl</jobname>
        <started>050708123010</started>
        <finished> </finished>
        <status>active</status>
    </mapping>
    <mapping>
        <JobID>402880e5-1c106b01-011c-10aa0ae1-0020</JobID>
        <jobname>paint</jobname>
        <started>050708123210</started>
        <finished>050708123301</finished>
        <status>finished</status>
    </mapping>
</collection>
```

4.4.3 [xml] descriptions.xml

Το αρχείο αυτό περιέχει την αναλυτική περιγραφή των ενεργών εργασιών στο πλέγμα. Περιλαμβάνει λεπτομέρειες για το όνομα και την περιγραφή της εργασίας , τα ορίσματα που

δέχεται , τον τύπο τους , καθώς και το Service Endpoint της. Ανανεώνεται κάθε φορά που μία νέα εργασία προστίθεται στα JobServices του GRIA και χρησιμοποιείται σαν οδηγός για τον χειρισμό της. Η παρακάτω μορφή παρουσιάζει τις λεπτομέρειες για τις τρείς δοκιμαστικές εργασίες που είχαμε στην διάθεση μας :

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
<GRIAApplicationDescription                                    xmlns="http://www.it-
innovation.soton.ac.uk/2007/grid/application">
       <JobServiceMinVersion>5.2</JobServiceMinVersion>
       <Application>
              <Description>Blends two images together</Description>
              <ApplicationName>http://it-
innovation.soton.ac.uk/grid/imagemagick/blend</ApplicationName>
              <Inputs>2</Inputs>
              <Outputs>1</Outputs>
              <ApplicationVersion>2.0-1</ApplicationVersion>
              <Group>graphics</Group>
              <Keywords>imagemagick, example</Keywords>
       </Application>
       <DataStagers>
              <Data    Stager    defaultSize="2"    maxOccurs="2"    minOccurs="2"
name="inputImage" type="input">
                     <Description>Input      images      to      be      blended
together</Description>
                     <MimeType>image</MimeType>
              </Data Stager>

              <Data Stager name="outputImage" type="output">
                     <Description>Resulting blended image</Description>
                     <MimeType>image</MimeType>
              </Data Stager>
       </DataStagers>
</GRIAApplicationDescription>
<GRIAApplicationDescription                                    xmlns="http://www.it-
innovation.soton.ac.uk/2007/grid/application">
       <JobServiceMinVersion>5.2</JobServiceMinVersion>
       <Application>
```

```
            <Description>Application to render an image into
painting</Description>
            <ApplicationName>http://it-
innovation.soton.ac.uk/grid/imagemagick/paint</ApplicationName>
            <Inputs>1</Inputs>
            <Outputs>1</Outputs>
            <ApplicationVersion>2.0-1</ApplicationVersion>
            <Group>graphics</Group>
            <Keywords>imagemagick, example</Keywords>
      </Application>
      <DataStagers>
            <Data Stager name="inputImage" type="input">
                  <Description>Input image to be painted</Description>
                  <MimeType>image</MimeType>
            </Data Stager>
            <Data Stager name="outputImage" type="output">
                  <Description>Resulting output image</Description>
                  <MimeType>image</MimeType>
            </Data Stager>
      </DataStagers>
</GRIAApplicationDescription>
<GRIAApplicationDescription                              xmlns="http://www.it-
innovation.soton.ac.uk/2007/grid/application">
      <JobServiceMinVersion>5.2</JobServiceMinVersion>
      <Application>
            <Description>Application to swirl an image</Description>
            <ApplicationName>http://it-
innovation.soton.ac.uk/grid/imagemagick/swirl</ApplicationName>
            <Inputs>1</Inputs>
            <Outputs>1</Outputs>
            <ApplicationVersion>2.0-1</ApplicationVersion>
            <Group>graphics</Group>
            <Keywords>imagemagick, example</Keywords>
      </Application>
      <DataStagers>
            <Data Stager name="inputImage" type="input">
                  <Description>Input image to be swirled</Description>
                  <MimeType>image</MimeType>
            </Data Stager>
```

```
            <Data Stager name="outputImage" type="output">
                  <Description>Swirled image</Description>
                  <MimeType>image</MimeType>
            </Data Stager>
      </DataStagers>
</GRIAApplicationDescription>
</root>
```

5

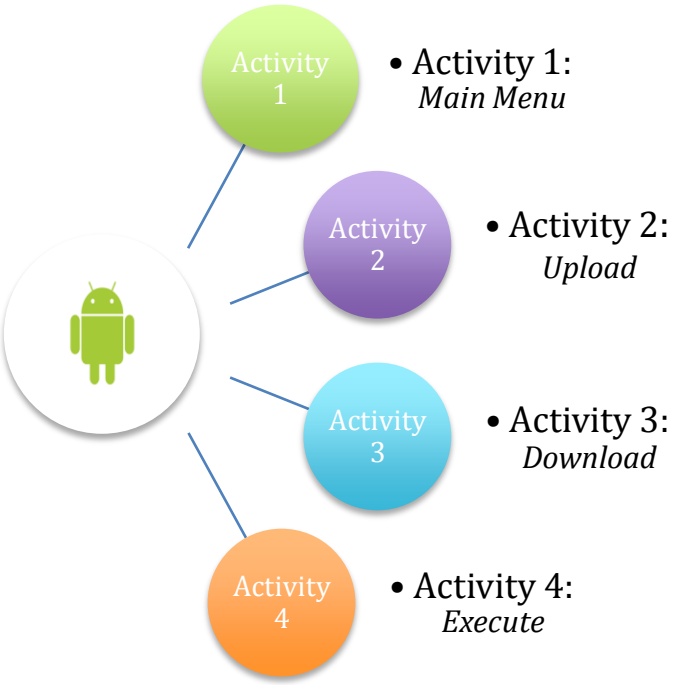
Υλοποίηση

Έχοντας αναλύσει την αρχιτεκτονική δομή προηγουμένως, θα παρουσιάσουμε στο παρόν κεφάλαιο κομμάτια από την υλοποίηση της εφαρμογής. Αναλύουμε τμήματα του κώδικα τόσο από την εφαρμογή στο Android , όσο και από τα web services που φιλοξενούνται στον Axis Server. Σκοπός μας είναι να αναδειχθεί η μεθοδολογία αλλά και οι σχεδιαστικές αποφάσεις που χρησιμοποιήθηκαν για την ανάπτυξη του συστήματος. Ξεκινάμε την ανάλυση κατά τα γνωστά, πρώτα από τον GRIA Manager (Android Application) καταλήγοντας στα Axis Web Services.

5.1 *Λεπτομέρειες υλοποίησης – [GRIA Manager]*

Η Android εφαρμογή έχει σχεδιαστεί εξαρχής με γνώμονα το περιβάλλον εκτέλεσης της. Το συγκεκριμένο λειτουργικό σύστημα, δίνει ιδιαίτερη έμφαση στο γραφικό περιβάλλον κάθε εφαρμογής και προωθεί την δημιουργία κώδικα χαμηλών υπολογιστικών απαιτήσεων. Σεβόμενοι λοιπόν τις αρχές αυτές , θεωρούμε ότι η βασική δομική μονάδα για μια εφαρμογή τέτοιου τύπου είναι η «τρέχουσα απεικόνιση» στην οθόνη ή αλλιώς δραστηριότητα (activity) . Με τον όρο αυτό εννοούμε ότι η δομή του προγράμματος , καθορίζεται από ένα πλήθος οθονών κάθε μία από τις οποίες αντιστοιχεί σε συγκεκριμένη λειτουργία. Το σύνολο λοιπόν των δραστηριοτήτων αλλά και των συνδέσεων τους αποτελούν μία ολοκληρωμένη εφαρμογή. Η βασική λειτουργικότητα του GRIA Manager μπορεί να παρουσιαστεί συγκεντρωτικά σε τέσσερις

βασικές οθόνες (activities). Η πρώτη activity, παρουσιάζεται με την έναρξη της εφαρμογής και περιλαμβάνει το κυρίως μενού επιλογών. Η κάθε μία από αυτές τις επιλογές οδηγεί σε μια νέα οθόνη και κατά συνέπεια σε άλλη λειτουργία. Αξίζει σε αυτό το σημείο να σημειώσουμε ότι υπάρχει άμεση επικοινωνία μεταξύ των activities σε μία εφαρμογή καθώς και ότι στο Android είναι δυνατός ο πολυνηματικός προγραμματισμός (όπως θα δούμε και παρακάτω). Η λειτουργικότητα του κάθε activity φαίνεται στο παρακάτω σχήμα :



Στις παρακάτω ενότητες αναλύουμε ενδεικτικά κομβικές λειτουργίες της εφαρμογής, παρουσιάζουμε σημεία από τον κώδικα και εμβαθύνουμε περαιτέρω στην λειτουργία του.

5.1.1 Δημιουργία HTTP POST/form-multipart data request

Ένα από τα βασικότερα θέματα είναι η επικοινωνία του Android με τον Axis Server. Δεδομένου ότι τα WS δέχονται POST /GET αιτήσεις , μέρος του κώδικα της εφαρμογής αφιερώνεται στην υλοποίηση και αποστολή HTTP POST αιτήσεων προς το URL του Axis. Η μέθοδος POST παρότι δυσκολότερη ως προς την υλοποίηση σε σχέση με την GET, επιλέχθηκε διότι δεν έχει περιορισμούς ως προς τον όγκο της μεταφερόμενης πληροφορίας και κατά συνέπεια μπορεί να χρησιμοποιηθεί για την μεταφορά αρχείων. Μέσω Java είναι δυνατόν να προσομοιώσουμε την αίτηση POST που δημιουργεί κάποιος browser κατά την αποστολή μίας html form. Μια τυπική τέτοια αίτηση θα μοιάζει με την παρακάτω :

```
POST register.jsp HTTP/1.1
Host: hi.iq
```

```
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.2)
Gecko/20021126
Accept:
text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plai
n;q=0.8,
video/x-mng,image/png,image/jpeg,image/gif;q=0.2,text/css,*/*;q=0.1
Accept-Language: en-us, en;q=0.50
Accept-Encoding: gzip, deflate, compress;q=0.9
Accept-Charset: ISO-8859-1, utf-8;q=0.66, *;q=0.66
Keep-Alive: 300Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 36
name=J.Doe&email=abuse%40spamcop.com
```

Η μόνη πληροφορία που μεταφέρεται εδώ είναι 2 παράμετροι με τις τιμές τους, όπως φαίνεται από τον κώδικα από όπου προήλθε η αίτηση :

```
<form method="post" action="hi.iq/register.jsp">
  Name:  <input type="text" name="name" value="J.Doe">
  email: <input type="text" name="email" value="abuse@spamcop.com">
  <input type="submit">
</form>
```

Αν επιπλέον θέλουμε να στείλουμε και κάποιο αρχείο, ο html κώδικας θα περιλαμβάνει την δήλωση `enctype="multipart/form-data"` και η παραγόμενη POST αίτηση θα μοιάζει με την παρακάτω

```
POST register.jsp HTTP/1.1
Host: hi/iq
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.2)
Gecko/20021126
Accept:
text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plai
n;q=0.8,
video/x-mng,image/png,image/jpeg,image/gif;q=0.2,text/css,*/*;q=0.1
Accept-Language: en-us, en;q=0.50
Accept-Encoding: gzip, deflate, compress;q=0.9
Accept-Charset: ISO-8859-1, utf-8;q=0.66, *;q=0.66
Keep-Alive: 300
Connection: keep-alive
Content-Type: multipart/form-data; boundary=---------------------------
29772313742745
Content-Length: 452
-----------------------------29772313742745
Content-Disposition: form-data; name="name"
J.Doe
-----------------------------29772313742745
Content-Disposition: form-data; name="email"
abuse@spamcop.com
-----------------------------29772313742745
Content-Disposition: form-data; name="file-upload"; filename="test.txt"
```

```
Content-Type: text/plain
test data with some high ascii: Â¿Como estÃ¡s?
-----------------------------29772313742745--
```

Σκοπός μας είναι λοιπόν να αναπαράγουμε αιτήσεις αυτού του τύπου. Η υπεύθυνη κλάση για αυτή την λειτουργία είναι η `ClientHttpRequest` και ο κώδικάς της παρουσιάζεται παρακάτω:

Αρχικά ανοίγουμε την σύνδεση URLConnection και ορίζουμε τις μεθόδους για εγγραφή σε αυτή

```
public class ClientHttpRequest {
  URLConnection connection;
  OutputStream os = null;
  Map cookies = new HashMap();

  protected void connect() throws IOException {
    if (os == null) os = connection.getOutputStream();
  }

  protected void write(char c) throws IOException {
    connect();
    os.write(c);
  }

  protected void write(String s) throws IOException {
    connect();
    os.write(s.getBytes());
  }

  protected void newline() throws IOException {
    connect();
    write("\r\n");
  }

  protected void writeln(String s) throws IOException {
    connect();
    write(s);
    newline();
  }

  private static Random random = new Random();

  protected static String randomString() {
    return Long.toString(random.nextLong(), 36);
  }

  String boundary = "---------------------------" + randomString() +
randomString() + randomString();

  private void boundary() throws IOException {
    write("--");
    write(boundary);
  }
```

Δημιουργούμε έπειτα μία νέα multipart POST HTTP request στην ανοιχτή URLConnection:

```
  public ClientHttpRequest(URLConnection connection) throws IOException
{
    this.connection = connection;
    connection.setDoOutput(true);
    connection.setRequestProperty("Content-Type",
                                  "multipart/form-data; boundary=" +
boundary);
  }
```

και υπερφορτώνουμε την μέθοδο `ClientHttpRequest` ώστε να δέχεται πολλές διαφορετικές παραμέτρους. Όπως το URL και URL String .

```
  public ClientHttpRequest(URL url) throws IOException {
    this(url.openConnection());
  }

  public ClientHttpRequest(String urlString) throws IOException {
    this(new URL(urlString));
  }
```

Έχουμε αρχίσει πλέον την δημιουργία της αίτησης οπότε σημπληρώνουμε τα ονόματα για κάθε στοιχείο της

```
private void writeName(String name) throws IOException {
    newline();
    write("Content-Disposition: form-data; name=\"");
    write(name);
    write('"');
  }
```

με την setParameter προσθέτουμε μία String παράμετρο στην αίτηση καθορίζοντας το όνομα και την τιμή της.

```
  public void setParameter(String name, String value) throws IOException
{
    boundary();
    writeName(name);
    newline(); newline();
    writeln(value);
  }
```

Ενώ με την pipe δημιουργούμε την δίοδο ανάμεσα στα δύο άκρα της επικοινωνίας.

```
  private static void pipe(InputStream in, OutputStream out) throws
IOException {
    byte[] buf = new byte[500000];
    int nread;
    int navailable;
    int total = 0;
    synchronized (in) {
      while((nread = in.read(buf, 0, buf.length)) >= 0) {
        out.write(buf, 0, nread);
        total += nread;
```

```
      }
    }
    out.flush();
    buf = null;
  }
```

Με την υπερφορτωμένη μέθοδο setParameter, προσθέτουμε παραμέτρους στην request συμπεριλαμβανομένων και των αρχείων. Χρησιμοποιούνται πολλοί τύποι της setParameter για να καλύψουν μεγάλο εύρος των περιπτώσεων.

```
  /**
   * adds a file parameter to the request
   * @param name parameter name
   * @param filename the name of the file
   * @param is input stream to read the contents of the file from
   * @throws IOException
   */
  public void setParameter(String name, String filename, InputStream is)
throws IOException {
    boundary();
    writeName(name);
    write("; filename=\"");
    write(filename);
    write('"');
    newline();
    write("Content-Type: ");
    String type = connection.guessContentTypeFromName(filename);
    if (type == null) type = "application/octet-stream";
    writeln(type);
    newline();
    pipe(is, os);
    newline();
  }
```

```
  /**
   * adds a file parameter to the request
   * @param name parameter name
   * @param file the file to upload
   * @throws IOException
   */
  public void setParameter(String name, File file) throws IOException {
    setParameter(name, file.getPath(), new FileInputStream(file));
  }
```

```
/**
   * adds a parameter to the request; if the parameter is a File, the
file is uploaded, otherwise the string value of the parameter is passed
in the request
   * @param name parameter name
   * @param object parameter value, a File or anything else that can be
stringified
   * @throws IOException
   */
  public void setParameter(String name, Object object) throws
IOException {
    if (object instanceof File) {
      setParameter(name, (File) object);
    } else {
      setParameter(name, object.toString());
    }
```

```
  }
```

```
  /**
   * adds parameters to the request
   * @param parameters "name-to-value" map of parameters; if a value is
a file, the file is uploaded, otherwise it is stringified and sent in
the request
   * @throws IOException
   */
  public void setParameters(Map parameters) throws IOException {
    if (parameters == null) return;
    for (Iterator i = parameters.entrySet().iterator(); i.hasNext();) {
      Map.Entry entry = (Map.Entry)i.next();
      setParameter(entry.getKey().toString(), entry.getValue());
    }
  }
```

```
  /**
   * adds parameters to the request
   * @param parameters array of parameter names and values
(parameters[2*i] is a name, parameters[2*i + 1] is a value); if a value
is a file, the file is uploaded, otherwise it is stringified and sent in
the request
   * @throws IOException
   */
  public void setParameters(Object[] parameters) throws IOException {
    if (parameters == null) return;
    for (int i = 0; i < parameters.length - 1; i+=2) {
      setParameter(parameters[i].toString(), parameters[i+1]);
    }
  }
```

Τέλος χρησιμοποιούμε την post() για να κάνουμε post τις αιτήσεις στον server με ότι παραμέτρους έχουν προστεθεί. Η μέθοδος αυτή είναι επίσης υπερφορτωμένη:

```
  /**
   * posts the requests to the server, with all the cookies and
parameters that were added
   * @return input stream with the server response
   * @throws IOException
   */
  public InputStream post() throws IOException {
    boundary();
    writeln("--");
    os.close();
    return connection.getInputStream();
  }
```

```
  /**
   * posts the requests to the server, with all the cookies and
parameters that were added before (if any), and with parameters that are
passed in the argument
   * @param parameters request parameters
   * @return input stream with the server response
   * @throws IOException
   * @see setParameters
   */
  public InputStream post(Map parameters) throws IOException {
```

```
    setParameters(parameters);
    return post();
  }
```

```
  /**
   * posts the requests to the server, with all the cookies and
parameters that were added before (if any), and with parameters that are
passed in the argument
   * @param parameters request parameters
   * @return input stream with the server response
   * @throws IOException
   * @see setParameters
   */
  public InputStream post(Object[] parameters) throws IOException {
    setParameters(parameters);
    return post();
  }
```

```
  /**
   * posts the requests to the server, with all the cookies and
parameters that were added before (if any), and with cookies and
parameters that are passed in the arguments
   * @param cookies request cookies
   * @param parameters request parameters
   * @return input stream with the server response
   * @throws IOException
   * @see setParameters
   * @see setCookies
   */
  public InputStream post(Map cookies, Map parameters) throws
IOException {
    setCookies(cookies);
    setParameters(parameters);
    return post();
  }
```

```
  /**
   * posts the requests to the server, with all the cookies and
parameters that were added before (if any), and with cookies and
parameters that are passed in the arguments
   * @param cookies request cookies
   * @param parameters request parameters
   * @return input stream with the server response
   * @throws IOException
   * @see setParameters
   * @see setCookies
   */
  public InputStream post(String[] cookies, Object[] parameters) throws
IOException {
    setCookies(cookies);
    setParameters(parameters);
    return post();
  }
```

```
  /**
   * post the POST request to the server, with the specified parameter
   * @param name parameter name
   * @param value parameter value
   * @return input stream with the server response
   * @throws IOException
```

```
     * @see setParameter
     */
    public InputStream post(String name, Object value) throws IOException
{
      setParameter(name, value);
      return post();
    }
```

```
    /**
     * post the POST request to the server, with the specified parameters
     * @param name1 first parameter name
     * @param value1 first parameter value
     * @param name2 second parameter name
     * @param value2 second parameter value
     * @return input stream with the server response
     * @throws IOException
     * @see setParameter
     */
    public InputStream post(String name1, Object value1, String name2,
Object value2) throws IOException {
      setParameter(name1, value1);
      return post(name2, value2);
    }
```

```
    /**
     * post the POST request to the server, with the specified parameters
     * @param name1 first parameter name
     * @param value1 first parameter value
     * @param name2 second parameter name
     * @param value2 second parameter value
     * @param name3 third parameter name
     * @param value3 third parameter value
     * @return input stream with the server response
     * @throws IOException
     * @see setParameter
     */
    public InputStream post(String name1, Object value1, String name2,
Object value2, String name3, Object value3) throws IOException {
      setParameter(name1, value1);
      return post(name2, value2, name3, value3);
    }
```

```
    /**
     * post the POST request to the server, with the specified parameters
     * @param name1 first parameter name
     * @param value1 first parameter value
     * @param name2 second parameter name
     * @param value2 second parameter value
     * @param name3 third parameter name
     * @param value3 third parameter value
     * @param name4 fourth parameter name
     * @param value4 fourth parameter value
     * @return input stream with the server response
     * @throws IOException
     * @see setParameter
     */
    public InputStream post(String name1, Object value1, String name2,
Object value2, String name3, Object value3, String name4, Object value4)
throws IOException {
      setParameter(name1, value1);
```

```
    return post(name2, value2, name3, value3, name4, value4);
  }
```

```
  /**
   * posts a new request to specified URL, with parameters that are
passed in the argument
   * @param parameters request parameters
   * @return input stream with the server response
   * @throws IOException
   * @see setParameters
   */
  public static InputStream post(URL url, Map parameters) throws
IOException {
    return new ClientHttpRequest(url).post(parameters);
  }
```

```
  /**
   * posts a new request to specified URL, with parameters that are
passed in the argument
   * @param parameters request parameters
   * @return input stream with the server response
   * @throws IOException
   * @see setParameters
   */
  public static InputStream post(URL url, Object[] parameters) throws
IOException {
    return new ClientHttpRequest(url).post(parameters);
  }
```

```
  /**
   * posts a new request to specified URL, with cookies and parameters
that are passed in the argument
   * @param cookies request cookies
   * @param parameters request parameters
   * @return input stream with the server response
   * @throws IOException
   * @see setCookies
   * @see setParameters
   */
  public static InputStream post(URL url, Map cookies, Map parameters)
throws IOException {
    return new ClientHttpRequest(url).post(cookies, parameters);
  }
```

```
  /**
   * posts a new request to specified URL, with cookies and parameters
that are passed in the argument
   * @param cookies request cookies
   * @param parameters request parameters
   * @return input stream with the server response
   * @throws IOException
   * @see setCookies
   * @see setParameters
   */
  public static InputStream post(URL url, String[] cookies, Object[]
parameters) throws IOException {
    return new ClientHttpRequest(url).post(cookies, parameters);
  }
```

```
  /**
```

```
    * post the POST request specified URL, with the specified parameter
    * @param name parameter name
    * @param value parameter value
    * @return input stream with the server response
    * @throws IOException
    * @see setParameter
    */
  public static InputStream post(URL url, String name1, Object value1)
throws IOException {
    return new ClientHttpRequest(url).post(name1, value1);
  }
```

```
  /**
   * post the POST request to specified URL, with the specified
parameters
   * @param name1 first parameter name
   * @param value1 first parameter value
   * @param name2 second parameter name
   * @param value2 second parameter value
   * @return input stream with the server response
   * @throws IOException
   * @see setParameter
   */
  public static InputStream post(URL url, String name1, Object value1,
String name2, Object value2) throws IOException {
    return new ClientHttpRequest(url).post(name1, value1, name2,
value2);
  }
```

```
  /**
   * post the POST request to specified URL, with the specified
parameters
   * @param name1 first parameter name
   * @param value1 first parameter value
   * @param name2 second parameter name
   * @param value2 second parameter value
   * @param name3 third parameter name
   * @param value3 third parameter value
   * @return input stream with the server response
   * @throws IOException
   * @see setParameter
   */
  public static InputStream post(URL url, String name1, Object value1,
String name2, Object value2, String name3, Object value3) throws
IOException {
    return new ClientHttpRequest(url).post(name1, value1, name2, value2,
name3, value3);
  }
```

```
  /**
   * post the POST request to specified URL, with the specified
parameters
   * @param name1 first parameter name
   * @param value1 first parameter value
   * @param name2 second parameter name
   * @param value2 second parameter value
   * @param name3 third parameter name
   * @param value3 third parameter value
   * @param name4 fourth parameter name
```

```
    * @param value4 fourth parameter value
    * @return input stream with the server response
    * @throws IOException
    * @see setParameter
    */
  public static InputStream post(URL url, String name1, Object value1,
String name2, Object value2, String name3, Object value3, String name4,
Object value4) throws IOException {
     return new ClientHttpRequest(url).post(name1, value1, name2, value2,
name3, value3, name4, value4);
  }
}
```

5.1.2 Κεντρικό Μενού Εφαρμογής

Έχοντας καλύψει ένα κρίσιμο κομμάτι στην προηγούμενη ενότητα , είναι σκόπιμο να παρουσιάσουμε εδώ την λειτουργικότητα και τον κώδικα του κεντρικού μενού. Η ανάλυση αυτή διαφωτίζει όχι μόνο την ροή των εργασιών στον GRIA Manager, αλλά θα δώσει περισσότερες λεπτομέρειες για την γενικότερη δομή μίας εφαρμογής στο Android. Όπως ήδη έχουμε περιγράψει, το κεντρικό μενού αποτελεί από μόνο του μία δραστηριότητα – activity στην ορολογία του android. Η δραστηριότητα αυτή ακολουθεί ένα κύκλο ζωής που καθορίζει την δημιουργία, την λειτουργία αλλά και τον τερματισμό της. Επίσης μια δραστηριότητα μπορεί να πυροδοτήσει νέα γεγονότα, όπως μία άλλη υπο-δραστηριότητα (subactivity), είτε για να περιμένει τα αποτελέσματα της εκτέλεσης τους είτε όχι. Στον GRIA Manager η σχεδίαση έχει διαμορφωθεί με πυρήνα την δραστηριότητα `UserInterface` . Αυτή είναι υπεύθυνη για την εκκίνηση των subactivities που φιλοξενούν τις λειτουργίες Upload, Download και Execute. Σημείο άξιο αναφοράς αποτελεί το πλούσιο γραφικό περιβάλλον που συνδέεται με τις εφαρμογές στο android. Έχουν αναπτυχθεί έτοιμα γραφικά στοιχεία, που ενσωματώνονται εύκολα στην εφαρμογή μέσω αρχείων ρυθμίσεων σε xml. Οι ρυθμίσεις αυτές μπορεί να περιλαμβάνουν εκτός από τον τύπο του εκάστοτε γραφικού και την ακριβή διάταξη όλων των στοιχείων στον χώρο της οθόνης. Για το GRIA Manager το layout αρχείο είναι :

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
android:id="@+id/relative_layout"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:padding="10px"
android:background="@drawable/wallpaper1"
xmlns:android="http://schemas.android.com/apk/res/android"
>
      <TextView
      android:id="@+id/welcome_txt"
      android:layout_width="fill_parent"
```

```
      android:layout_height="wrap_content"
      android:padding="2px"
      android:text="Welcome to GRIA Manager"
      android:textStyle="bold"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true"
      >
      </TextView>

      <TextView
      android:id="@+id/select_txt"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="Select your action :"
      android:layout_below="@+id/welcome_txt"
      android:layout_alignParentLeft="true"
      >
      </TextView>

            <LinearLayout
            android:id="@+id/Linear_layout"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:padding="5px"
            android:orientation="vertical"
            android:layout_below="@+id/select_txt"
            android:layout_centerHorizontal="true"
            >

            <gr.ntua.ece.test.gui.TransparentPanel
                        android:id="@+id/transparent_panel"
                  android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
                    android:paddingTop="5px"
                    android:paddingLeft="5px"
                          android:paddingBottom="5px"
                          android:paddingRight="5px"
                          android:layout_margin="2px">

                    <Button android:id="@+id/upload_btn"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:text="Upload"/>

                  <TextView
                            android:id="@+id/upload_tv"
                            android:layout_width="fill_parent"
                            android:layout_height="wrap_content"
                            android:paddingLeft="5px"
                            android:text="Upload File to Remote Location"
                            >
                      </TextView>

            </gr.ntua.ece.test.gui.TransparentPanel>

            <gr.ntua.ece.test.gui.TransparentPanel
                        android:id="@+id/transparent_panel"
                  android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
                    android:paddingTop="5px"
                    android:paddingLeft="5px"
```

```
                        android:paddingBottom="5px"
                        android:paddingRight="5px"
                        android:layout_margin="2px">

                  <Button android:id="@+id/download_btn"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Download"/>

              <TextView
                        android:id="@+id/download_tv"
                        android:layout_width="fill_parent"
                        android:layout_height="wrap_content"
                        android:paddingLeft="5px"
                        android:text="Download File from Remote
Location"
                        >
                    </TextView>

            </gr.ntua.ece.test.gui.TransparentPanel>

          <gr.ntua.ece.test.gui.TransparentPanel
                        android:id="@+id/transparent_panel"
                      android:layout_width="fill_parent"
                        android:layout_height="wrap_content"
                        android:paddingTop="5px"
                        android:paddingLeft="5px"
                        android:paddingBottom="5px"
                        android:paddingRight="5px"
                        android:layout_margin="2px">

                  <Button android:id="@+id/execute_btn"
                     android:layout_width="wrap_content"
                     android:layout_height="wrap_content"
                     android:text="Execute"/>

                     <TextView
                          android:id="@+id/execute_tv"
                          android:layout_width="fill_parent"
                          android:layout_height="wrap_content"
                          android:paddingLeft="5px"
                          android:text="Execute Function"
                          >
                    </TextView>

            </gr.ntua.ece.test.gui.TransparentPanel>
            <TextView
                  android:id="@+id/powered_tv"
                  android:layout_width="wrap_content"
              android:layout_height="wrap_content"
                  android:layout_marginTop="5px"
            android:layout_gravity="center_horizontal"
            android:text="Powered by:"
                  >
                  </TextView>

                    <!-- 2. Limit to at most 50x50 -->
        <ImageView
            android:src="@drawable/ntua"
            android:adjustViewBounds="true"
            android:maxWidth="50dip"
```

```
            android:maxHeight="50dip"
            android:layout_gravity="center_horizontal"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />


            </LinearLayout>
</RelativeLayout>
```

Η τελική εικόνα που παράγεται από το παραπάνω layout είναι :

Εικόνα 8: Το αρχικό μενού που δημιουργείται βάση του XML αρχείου.

Το κάθε ένα από τα παραπάνω κουμπιά εκκινούν την αντίστοιχη δραστηριότητα ενώ η UserInterface περιμένει τον τερματιμό τους. Η διαδικασία εξηγείται αναλυτικότερα παρακάτω. Ορίζουμε αρχικά ένα αναγνωριστικό κωδικό για κάθε subactivity ώστε να μπορούμε να το αναγνωρίσουμε κατά την επιστροφή του.

```
    protected static final int SUB_ACTIVITY_REQUEST_CODE_1 = 1;
    protected static final int SUB_ACTIVITY_REQUEST_CODE_2 = 2;
    protected static final int SUB_ACTIVITY_REQUEST_CODE_3 = 3;
```

Η συνάρτηση onCreate ενεργοποιείται κατα την εκκίνηση της εφαρμογής και μέσα σε αυτήν δηλώνουμε τα buttons που δημιουργήσαμε με το layout, αναθέτοντας παράλληλα τους αντίστοιχους listeners.

```
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

     // Find the button defined in the main.xml
        Button upload_btn = (Button)findViewById(R.id.upload_btn);
        // Add an OnClickListener to it, that will open the SubActivity
        upload_btn.setOnClickListener(mGetListener_upload);

     // Find the button defined in the main.xml
        Button download_btn = (Button)findViewById(R.id.download_btn);
        // Add an OnClickListener to it, that will open the SubActivity
        download_btn.setOnClickListener(mGetListener_download);

     // Find the button defined in the main.xml
        Button execute_btn = (Button)findViewById(R.id.execute_btn);
        // Add an OnClickListener to it, that will open the SubActivity
        execute_btn.setOnClickListener(mGetListener_execute);
    }
```

Στον κώδικα των listener βλέπουμε πως καλείται μία νέα subactivity περιμένοντας το αποτέλεσμα της. Δημιουργούμε το intent με την activity στην οποία θέλουμε να μεταφερθούμε και καλούμε την startActivityForResult με παραμέτρους το intent που ορίσαμε και τον αναγνωριστικό κωδικό της δραστηριότητας:

```
    private OnClickListener mGetListener_upload = new OnClickListener()
{
        public void onClick(View v) {
            // Start the activity whose result we want to retrieve.  The
            // result will come back with request code .
            Intent i = new Intent(UserInterface.this,
                        SubActivity_1.class);
            // We use SUB_ACTIVITY_REQUEST_CODE as an 'identifier'
            startActivityForResult(i, SUB_ACTIVITY_REQUEST_CODE_1);

        }
    };
    private OnClickListener mGetListener_download = new
OnClickListener() {
        public void onClick(View v) {
            // Start the activity whose result we want to retrieve.  The
            // result will come back with request code
            Intent i = new Intent(UserInterface.this,
                        SubActivity_2.class);
            // We use SUB_ACTIVITY_REQUEST_CODE as an 'identifier'
            startActivityForResult(i, SUB_ACTIVITY_REQUEST_CODE_2);

        }
    };
```

```
    private OnClickListener mGetListener_execute = new OnClickListener()
{
        public void onClick(View v) {
            // Start the activity whose result we want to retrieve.  The
            // result will come back with request code
            Intent i = new Intent(UserInterface.this,
                        SubActivity_3.class);
            // We use SUB_ACTIVITY_REQUEST_CODE as an 'identifier'
            startActivityForResult(i, SUB_ACTIVITY_REQUEST_CODE_3);
        }
    };
```

Η onActivityResult, όταν επιστρέψει κάποια από τις υπο-δραστηριότητες. Χρησιμοποιεί τον αναγνωριστικό κωδικό για να αντιστοιχήσει λειτουργία – activity. Παρατηρούμε ότι για την λειτουργία upload, αφού επιστρέψει η sub_activity_1 ξεκινούμε την διαδικασία ανεβάσματος του αρχείου με την εντολή

```
Operations.createStager(UserInterface.this,mImageName,mImageFile);
```

που δημιουργούμε σε ένα νέο νήμα εκτέλεσης. Το νήμα αυτό μάλιστα παρακολουθείται από μία μπάρα προόδου (ProgressDialog) που σταματά όταν τερματιστεί. Οι subActivities 2 και 3 δεν έχουν κάποιο κώδικα να εκτελείται κατα την επιστροφή τους γι αυτό και οι αντίστοιχες θέσεις είναι κενές.

```
    @Override
    protected void onActivityResult(int requestCode, int resultCode,
            Intent data) {
      // You can use the requestCode to select between multiple child
      // activities you may have started.  Here there is only one thing
      // we launch.
      switch (requestCode){
      case SUB_ACTIVITY_REQUEST_CODE_1:
            // This is a standard resultCode that is sent back if the
            // activity doesn't supply an explicit result.  It will also
            // be returned if the activity failed to launch.
            if (resultCode == RESULT_CANCELED) {
                  // Our protocol with the sending activity is that it
will send
                  // text in 'data' as its result.
            } else {
                  //set the image file name +path we want to upload
                  //make it final to run into a thread
                  final String mImageName=data.getAction();
                  final File
mImageFile=getFileStreamPath(data.getAction());
                  Toast.makeText(this, "filepath : " +
mImageFile.toString(), Toast.LENGTH_SHORT).show();

                  //Image Selected begin upload in a new thread
                  //Display an indeterminate Progress-Dialog
```

```
                  myProgressDialog =
ProgressDialog.show(UserInterface.this,
                              null, "Creating Data Stager...", true);

                  new Thread() {
                        public void run() {
                              try{

      Operations.createStager(UserInterface.this,mImageName,mImageFile);

                              } catch (Exception e) {  }
                              // Dismiss the Dialog
                              myProgressDialog.dismiss();
                        }
                  }.start();

            }
            break;
      case SUB_ACTIVITY_REQUEST_CODE_2:
            // This is a standard resultCode that is sent back if the
            // activity doesn't supply an explicit result.  It will also
            // be returned if the activity failed to launch.
            if (resultCode == RESULT_CANCELED) {


            } else {
            }
            break;
      case SUB_ACTIVITY_REQUEST_CODE_3:
            // This is a standard resultCode that is sent back if the
            // activity doesn't supply an explicit result.  It will also
            // be returned if the activity failed to launch.
            if (resultCode == RESULT_CANCELED) {


            } else {

            }
            break;
      }
   }
```

Το δείγμα αυτό κώδικα έιναι αντιπροσωπευτικό όσον αφορα την γενική λειτουργία της δραστηριότητας UserActivity. Με παρόμοια λογική η εφαρμογή επεκτείνεται στις υπόλοιπες δραστηριότητες ώστε να επιτύχει την προσδοκούμενη λειτουργικότητα. Στην επόμενη ενότητα εξετάζουμε κομμάτια κώδικα απο το υποσύστημα του Axis Server.

5.2 *Λεπτομέρειες υλοποίησης – [Axis2 Server]*

Έχοντας εξετάσει λεπτομερώς κομμάτια από την υλοποίηση της android εφαρμογής, μπορούμε πλέον να επεκταθούμε στο άμεσα συνδεόμενο υποσύστημα, αυτό του Axis Server. Τα Web Services που δημιουργούμε εδώ εκτίθενται σε κάποιο URL της μορφής :

```
http://<domain>/axis2/services/GRIAServices/<service>
```

όπου κάθε <service> απαιτεί τα δικά του ορίσματα. Από το GRIAManager στην κινητή συσκευή, δημιουργούμε POST αιτήσεις με τις κατάλληλες παραμέτρους και καλούμε το URL που περιέχει το επιθυμητό service. Εκεί ο Axis Server απαντά δεχόμενος την αίτηση μαζί με τα ορίσματα και χρησιμοποιεί το GRIA API για να υλοποιήσει τις λειτουργίες προς το πλέγμα. Δύο βασικά URL που ορίζουμε είναι το Job Service Endpoint και το Data Service Endpoint , που ορίζουν την θέση επικοινωνίας με τα Job Service και Data Service αντίστοιχα.

```
private static String JOB_SERVICE_ENDPOINT = "https://192.168.0.100:8443/GRIA-
basic-app-services/services/JobService";
private static String DATA_SERVICE_ENDPOINT = "https://192.168.0.100:8443/GRIA-
basic-app-services/services/DataService";
```

Παρακάτω εξετάζουμε ενδεικτικά τρία services όπου παρουσιάζεται το GRIA API.

5.2.1 Ανακάλυψη ενεργών εργασιών στο πλέγμα

Η συνάρτηση αυτή δημιουργώντας ένα αντικείμενο RemoteJobService αναζητά στο URL του Job Service όλα τα ID των ενεργών εργασιών. Με την χρήση της getApplications() επιστρέφει ένα array με τα ονόματα εργασιών αυτών.

```
/** Discover active applications and return a String[] with their URIs
**/
      public String[] discoverApplications(String input) throws
RemoteException {
            StateRepository repository = new MemoryStateRepository();
            RemoteJobService jobService =

      (RemoteJobService)repository.getOrCreateObject(RemoteJobService.cl
ass,ConversationID.getEPR(JOB_SERVICE_ENDPOINT));

            String[] applicationList = jobService.getApplications();

            if(applicationList.length > 0){
                  return applicationList;
            }
            else{
                  return null;
            }
      }
```

5.2.2 *Δημιουργία Data Stager*

Η χρήση αρχείων από τις εργασίες του πλέγματος, επιβάλλει την δημιουργία των Data Stager που τα περιέχουν. Η συνάρτηση Upload αναλαμβάνει αυτή την διαδικασία ως εξής :

1. Με την getOrCreateObject() δημιουργεί μία σύνδεση προς το Data Service Endpoint
2. Δημιουργεί νέο Data Stager χρησιμοποιώντας την createStagingArea()
3. Αποθηκεύει το DataStagerID με την getConversationFromEPR()
4. Ανεβάζει το αρχείο που δίνεται σαν όρισμα στον νέο Data Stager.
5. Τέλος επιστρέφει το DataStagerID ώστε η συνάρτηση που την κάλεσε να ενημερώσει το αντίστοιχο xml αρχείο.

```
/** Upload to GRIA the given file **/
public static String upload(String filename) throws RemoteException
      {
      StateRepository   repository  = new MemoryStateRepository();
      RemoteDataService dataService       = (RemoteDataService)
repository.getOrCreateObject(RemoteDataService.class,ConversationID.getE
PR(DATA_SERVICE_ENDPOINT));
            DataConversation  data  =
dataService.createStagingArea(filename);
            String DataStagerID =
ConversationID.getConversationFromEPR(data.getEndpointRef());
            DataHandler inputHandler = new DataHandler(new
FileDataSource(uploadFolder+filename));
            data.save(inputHandler);

            return DataStagerID;
      }
```

5.2.3 *Εκτέλεση εργασίας*

Με την execute εκτελούμε μία εργασία στο πλέγμα με τα ορίσματα που έχουν δοθεί. Τα ορίσματα καθορίζουν τα δεδομένα προς επεξεργασία αλλά και την θέση αποθήκευσης των αποτελεσμάτων. Συνοπτικά η execute δημιουργεί πρώτα όλες τις απαραίτητες συνδέσεις με το Job Service και ύστερα αναθέτει handlers για τα ορίσματα που κλήθηκε. Ξεκινά την εργασία και όταν αυτή τελειώσει αποθηκεύει τα αποτελέσματα της και σταματά . Σε περίπτωση αποτυχίας σταματά με μήνυμα λάθους και ενημερώνει το καταγραφών. Αναλυτικά:

Δημιουργία σύνδεσης με το Job Service Endpoint :

Με την getOrCreateObject() δημιουργείται σύνδεση προς το Job Service Endpoint και με την createJob() δημιουργούμε μία νέα εργασία τύπου applicationUrl (αυτή που δώσαμε εμείς σαν όρισμα).

```
/** Execute the function which is in the given url ( we give the input
and the name for the output file and we will wait until the function
```

```
ends.)**/
      public static boolean execute(String applicationUrl,String[]
inputfiles,String[] outputfiles)
      {
            String help = null;
            try
            {
                  StateRepository repository = new
MemoryStateRepository();
                  RemoteJobService jobService =
(RemoteJobService)repository.getOrCreateObject(RemoteJobService.class,Co
nversationID.getEPR(JOB_SERVICE_ENDPOINT));
                  JobConversation jobConv =
jobService.createJob(applicationUrl, null, "");
```

Ανάκτηση ορισμάτων εισόδου :

Αναθέτουμε DataHandlers στα αρχεία εισόδου.

```
                  DataConversation[] inputs = jobConv.getInputs();
                  DataConversation[] outputs = jobConv.getOutputs();

                  // Give inputs //
                  for (int i=0; i<inputfiles.length;i++)
                  {
/*removes the path from file name and appends ours eg:
* filename=C:/test/file.jpg becomes file.jpg and then the uploadFolder
* path is appended */
String filepath =
uploadFolder+inputfiles[i].substring(inputfiles[i].lastIndexOf("/")+1);
                        inputs[i].save(new DataHandler(new
FileDataSource(filepath)));
                  }
```

Εκτέλεση της εργασίας:

Καλώντας την submitJob ξεκινά η εκτέλεση της εργασίας σε νέο Thread. Με την checkJob() ελέγχουμε το status και αν όλα τερματίσουν ομαλά προχωρούμε στο επόμενο στάδιο.

Διαφορετικά προκαλείται Runtime Exception και ενημερώνεται ο χρήστης.

```
jobConv.submitJob(null, new String[] {});
                  while(jobConv.checkJob().getInProgress())
                  {
                        Thread.sleep(500);
                  }

                  JobStatus jobStatus = jobConv.checkJob();

                  if(jobStatus.getExitStatus() != 0)
                  {
                        // handle the failure
                        throw new RuntimeException("job failed. Here is
the log:\n" + jobStatus.getLogText());
                  }
```

Αποθήκευση των αποτελεσμάτων και κλείσιμο :

Αν όλα τελειώσουν ομαλά, αποθηκεύουμε τα επεξεργασμένα αρχεία προσθέτοντας στο αρχικό τους όνομα το πρόθεμα "modified_" για να τα διαχωρίσουμε από τα αυθεντικά , ενώ ενημερώνουμε και το *StagerLogFile* που είναι υπεύθυνο για την αντιστοιχία των Stager με τα αρχεία. Τέλος καλούμε την destroy() για να καθαρίσουμε υπολείμματα της εργασίας στον server.

```
                    // Take outputs //
                    for (int i=0;i<outputfiles.length;i++)
                    {
/*removes the path from file name and appends ours eg:
* filename=C:/test/file.jpg becomes file.jpg and then the uploadFolder
* path is appended */
String filepath =
downloadFolder+"modified_"+outputfiles[i].substring(outputfiles[i].lastI
ndexOf("/")+1);
                              outputs[i].read(new File(filepath));
                              //Create a new Stager and load the results
                              //Dont forget to update the xml file!
                              //or create it if it doesnt exist
                              if(!XmlUtilities.isCreated(StagerLogFile)){
                                    if(!XmlUtilities.create(StagerLogFile)){
                                          //error creating file
                                    }
                              }
                              XmlUtilities.add(StagerLogFile,
            upload(filepath.substring(filepath.lastIndexOf("/")+1)),
                              filepath.substring(filepath.lastIndexOf("/")+1)
                                          );
                    }
                    // clean up resources at the server
                    jobConv.destroy();
                    return true;
              }
              catch(Exception e)
              {
                    return false;
              }
        }
```

5.3 *Πλατφόρμες και προγραμματιστικά εργαλεία*

Για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκε η πλατφόρμα ***Eclipse*** πάνω σε λειτουργικό σύστημα ***Windows XP SP3.*** Η γλώσσα ανάπτυξης ήτανε εξολοκλήρου η ***Java.*** Ο παρακάτω πίνακας δίνει αναλυτικά τις εκδόσεις των προγραμματιστικών εργαλείων που χρησιμοποιήθηκαν :

Πίνακας 13: Τα προγραμματιστικά εργαλεία

Λειτουργικό Σύστημα	Microsoft Windows XP Professional version 2002 Service Pack 3
Java Development Kit	JDK Version 6 Update 6
Eclipse Platform	Eclipse Platform Version 3.4.0 Ganymede
Android Software Development Kit	SDK Version 0.9 Beta
Android Development Tools (Eclipse Plugin)	0.7.1v2008
Apache Tomcat	Version 6.0.16
Apache Axis 2 Java	Version 1.4.0
GRIA Software	GRIA Basic Application Services v5.2
ImageMagick	Version 6.4.1 Q8

Τέλος το hardware που αναπτύχθηκε και δοκιμάστηκε η εφαρμογή :

Πίνακας 14 : Το υλικό ανάπτυξης και δοκιμής.

Επεξεργαστής :	AMD Athlon 64 x 3000+
Μνήμη (RAM) :	1Gb
Σκληρός Δίσκος :	120Gb
Κάρτα γραφικών :	ATI x700 pro 256mb

6

Έλεγχος

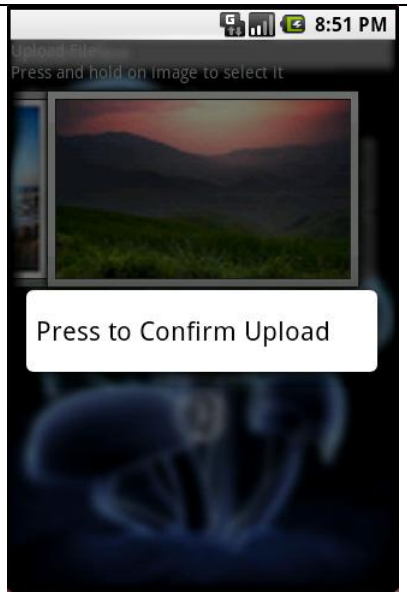
Στο κεφάλαιο αυτό θα ακολουθήσει η αξιολόγηση του συστήματος.

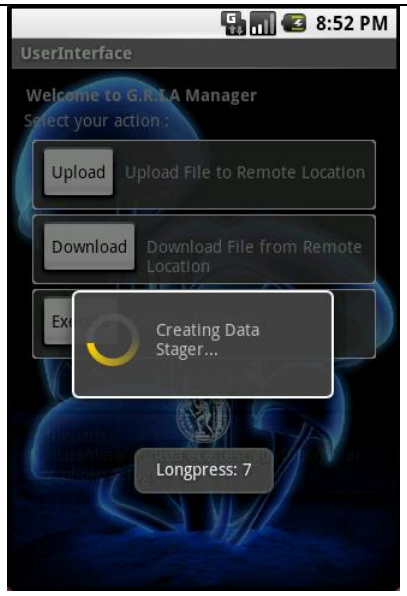
6.1 *Μεθοδολογία ελέγχου*

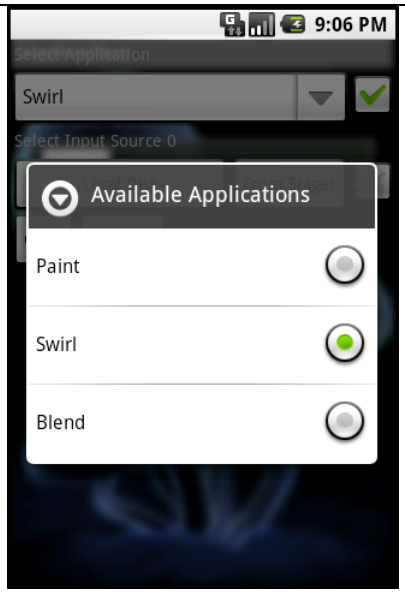
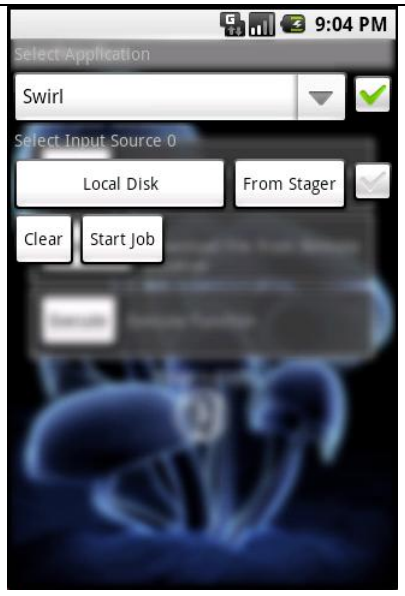
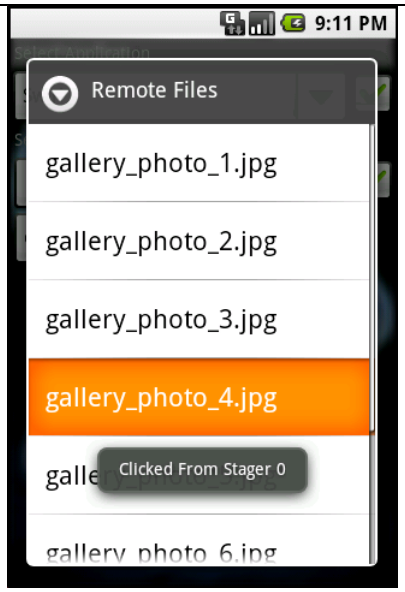
Ο έλεγχος θα πραγματοποιηθεί χρησιμοποιώντας ένα απλό σενάριο λειτουργίας : Το σενάριο περιλαμβάνει αρχικά την επιλογή ενός αρχείου από τον τοπικό δίσκο της κινητής συσκευής και την δημιουργία ενός νέου Data Stager. Έπειτα επιλέγουμε μία διαθέσιμη εργασία για εκτέλεση και το αρχείο που ανεβάσαμε πριν. Μετά την επιτυχή εκτέλεση κατεβάζουμε στο κινητό τα αποτελέσματα της επεξεργασίας. Στην ενότητα 6.3 παρουσιάζουμε κάποιες ενδεικτικές μετρήσεις για τους χρόνους που κάνουν να εκτελεστούν οι λειτουργίες της εφαρμογής.

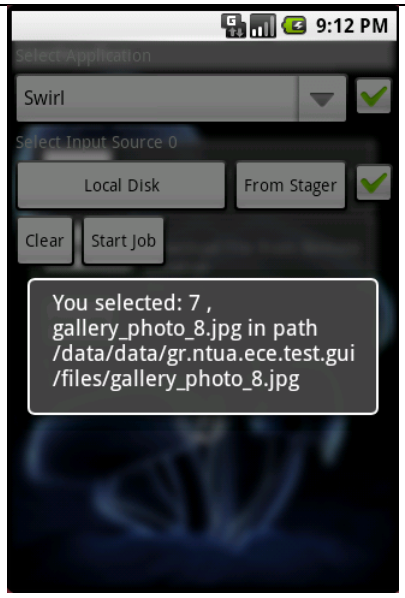
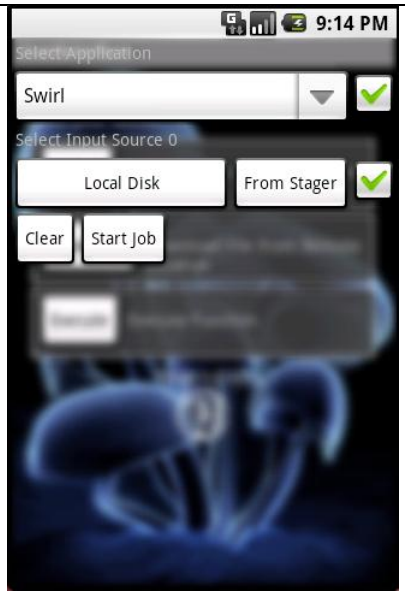
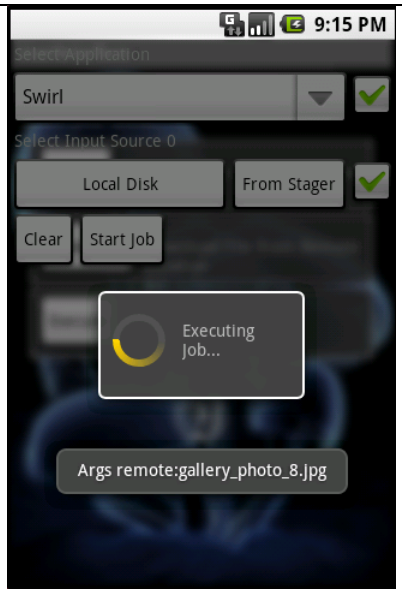
6.2 *Αναλυτική παρουσίαση ελέγχου*

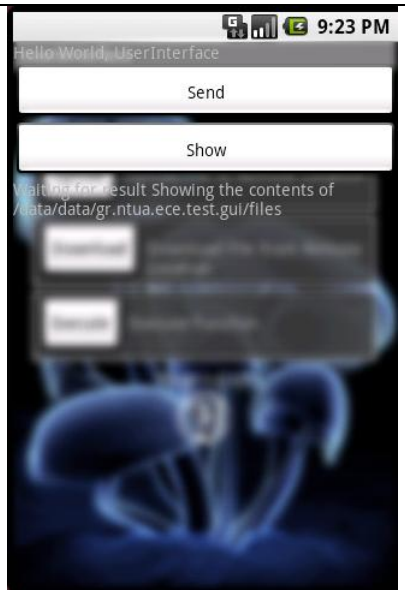
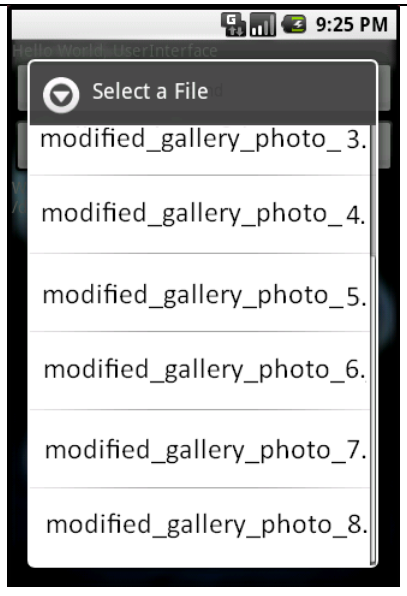
Στην ενότητα αυτή παρουσιάζουμε αναλυτικά τον έλεγχο του συστήματος σύμφωνα με το σενάριο που περιγράφηκε στην προηγούμενη ενότητα. Σε κάθε στάδιο παρουσιάζουμε και τις αντίστοιχες screenshots από την οθόνη του κινητού. Το κεφάλαιο αυτό, λειτουργεί και ως εγχειρίδιο χρήσης του προγράμματος.

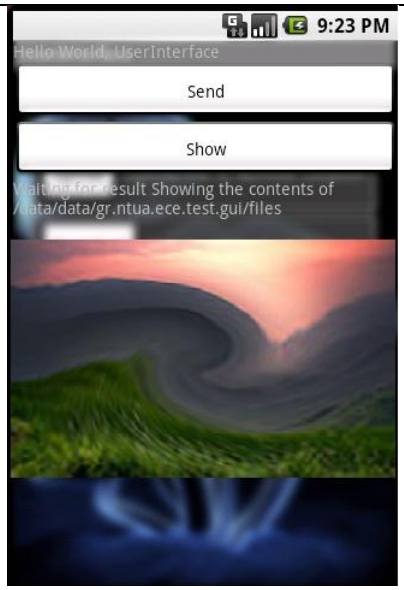
1	Εκκίνηση της εφαρμογής στο Android. Παρουσιάζεται στον χρήστη η αρχική οθόνη με τις τρείς βασικές επιλογές : ***Upload***, ***Download*** και **Execute**. Για την δημιουργία νέου Data Stager επιλέγουμε το πρώτο button.	
2	Στην οθόνη της λειτουργίας Upload, παρουσιάζεται η Gallery από τις φωτογραφίες του κινητού, ενώ αν επιλέξουμε κάποια από αυτές ενημερωνόμαστε για το πλήρες path της.	
3	Επιλέγουμε την εικόνα 8 και κρατώντας πατημένο το δείκτη στην εικόνα , ζητείται η επιβεβαίωση μας για την εκκίνηση της δημιουργίας Data Stager.	

4	Καθώς ανεβαίνει η εικόνα στα Resources ενημερωνόμαστε για την πρόοδο της με τον εμφανιζόμενο διάλογο	
5	Μετά από σύντομο χρονικό διάστημα τελειώνει το upload και επιστρέφουμε αυτόματα στο αρχικό μενού επιλογών. Αυτή την φορά επιλέγουμε το Execute.	
6	Η οθόνη που μας παρουσιάζεται δείχνει την αρχικά την επιλεγμένη εργασία ενώ ακολουθούν δύο κουμπιά που βοηθούν τον χρήστη να επιλέξει το κάθε όρισμα. Παρατηρούμε ότι η Blend που παρουσιάζεται εδώ έχει δύο ορίσματα και για κάθε ένα από αυτά μπορούμε να διαλέξουμε διαφορετική πηγή προέλευσης: Είτε από κάποιο αρχείο στον δίσκο, είτε από κάποιον ήδη δημιουργημένο Data Stager.	

7	Αν επιλέξουμε να δούμε τις ενεργές εργασίες στο πλέγμα μας παρουσιάζεται ο αντίστοιχος διάλογος. Επιλέγουμε την swirl .	
8	Η οθόνη διαμορφώνεται κατάλληλα παρουσιάζοντας μόνο τις επιλογές για ένα όρισμα. Βλέπουμε ότι η πράσινη ένδειξη δίπλα από κάθε στάδιο είναι ανενεργή , κάτι που σημαίνει ότι δεν έχουμε επιλέξει προέλευση για το όρισμα μας. Επιλέγουμε From Stager.	
9	Ανοίγει ο διάλογος Remote Files που περιέχει όλα τα αρχεία ανεβασμένα σε Data Stagers. Επιλέγουμε το gallery_photo_8.jpg αφού αυτό ανεβάσαμε στο πρώτο βήμα.	

10	Η επιλογή μας επιβεβαιώνεται από τον διάλογο που εμφανίζει το όνομα το αρχείου και το πλήρες path του.	
11	Μετά την επιλογή του αρχείου παρατηρούμε ότι πλέον η ένδειξη στην δεξιά πλευρά ενεργείται επισημαίνοντας την επιτυχή επιλογή αρχείου. Το μόνο που απομένει είναι η εκκίνηση της εργασίας πατώντας Start Job.	
12	Αρχίζει η εκτέλεση της εργασίας ενώ εμφανίζεται ο διάλογος προόδου. Ταυτόχρονα ενημερωνόμαστε ότι το όρισμα που δώσαμε ήτανε από κάποιον Data Stager (remote) και ποιο είναι αυτό.	

13	Αφού τερματίσει με επιτυχία η εργασία μας επιστρέφουμε στο αρχικό μενού επιλέγοντας Download αυτή την φορά για να δούμε το αποτέλεσμα της swirl.	
14	Στο αρχικό μενού της Download επιλέγουμε Show για να διαλέξουμε το αρχείο που θέλουμε να κατεβάσουμε.	
15	Εμφανίζονται τα επεξεργασμένα αρχεία που βρίσκονται σε κάποιον Data Stager στα Resources. Επιλέγουμε το *modified_gallery_photo_8* , αφού αυτό είναι το αποτέλεσμα της swirl που εκτελέσθηκε.	

16	Αφού κατέβει στην συσκευή , εμφανίζεται η εικόνα που αρχικά είχαμε επιλέξει μετά την επεξεργασία της με το εφέ swirl.	

Η χρήση της εφαρμογής αποδεικνύεται ιδιαίτερα απλή υπόθεση, λόγω των γραφικών βοηθημάτων που παρέχει το Android στον προγραμματιστή. Ολόκληρη η διαδικασία που περιγράφηκε ολοκληρώνεται σε μικρό χρονικό διάστημα, ενώ σε περίπτωση λάθους ο χρήστης ενημερώνεται κατάλληλα. Τέλος, κάτι που δεν μπορεί να γίνει άμεσα αντιληπτό , είναι η δυνατότητα εκκίνησης πολλαπλών εργασιών προς το πλέγμα.

6.3 *Μετρήσεις*

Παρουσιάζουμε κάποιες ενδεικτικές μετρήσεις για διαφορετικούς τύπους αρχείων, αρχικά συγκρίνοντας τις λειτουργίες Upload και Download , ενώ τέλος δίνουμε τους χρόνους για τις ενεργές εργασίες στο πλέγμα.

6.3.1 Σύγκριση χρόνων Upload - Download

Δίνοντας το όνομα και το μέγεθος των εξεταζόμενων αρχείων παρουσιάζουμε τον χρόνο που κάνουν κατά τις διαδικασίες upload και download.

Πίνακας 15: Χρόνοι της λειτουργίας Upload.

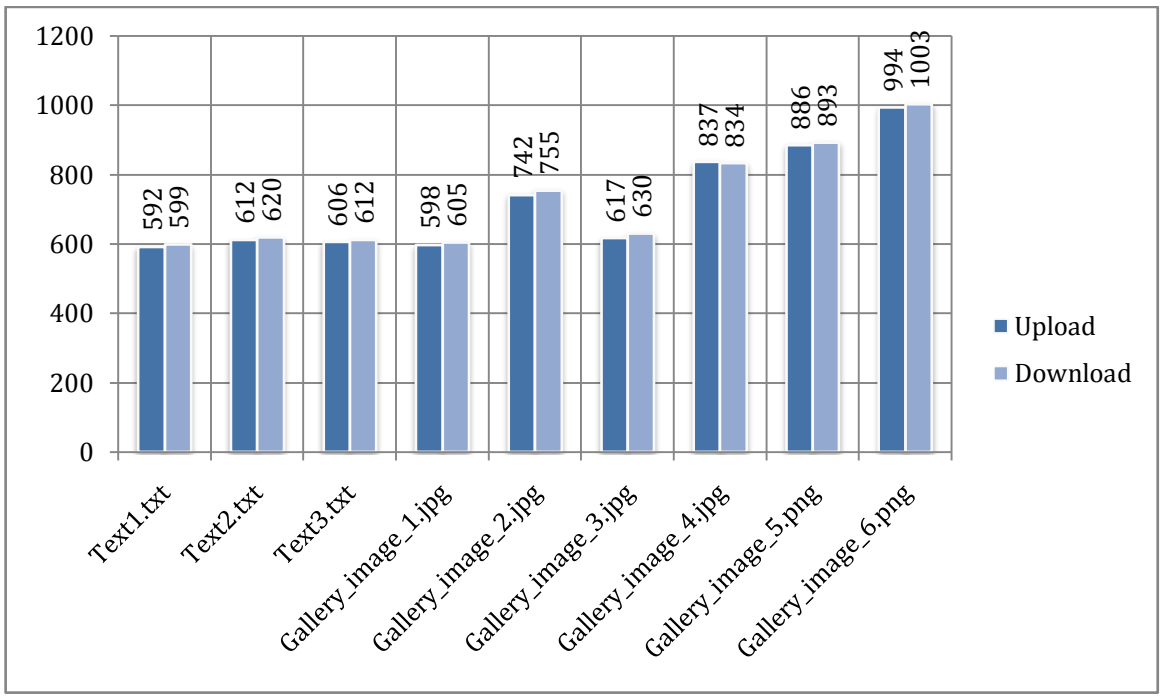
Upload		
Όνομα αρχείου	**Μέγεθος (bytes)**	**Χρόνος (milliseconds)**
Text1.txt	176	**592**
Text2.txt	342	**612**
Text3.txt	479	**606**
Gallery_image_1.jpg	571	**598**
Gallery_image_2.jpg	5474	**742**
Gallery_image_3.jpg	11473	**617**
Gallery_image_4.jpg	12890	**837**
Gallery_image_5.png	13252	**886**
Gallery_image_6.png	81898	**994**

Πίνακας 16: Χρόνοι της λειτουργίας Download.

Download		
Όνομα αρχείου	**Μέγεθος (bytes)**	**Χρόνος (milliseconds)**
Text1.txt	176	**599**
Text2.txt	342	**620**
Text3.txt	479	**612**
Gallery_image_1.jpg	571	**605**
Gallery_image_2.jpg	5474	**755**
Gallery_image_3.jpg	11473	**630**
Gallery_image_4.jpg	12890	**834**

Gallery_image_5.png	13252	**893**
Gallery_image_6.png	81898	**1003**

Διαγραμματικά τα παραπάνω συνοψίζονται ως εξής:



Εικόνα 9 : Διάγραμμα χρονικής απεικόνισης Upload - Download.

Παρατηρούμε ελάχιστα αυξημένο τον χρόνο του Download. Πιθανόν ο χρόνος να επηρεάζεται από την ενημέρωση του αρχείου *StagerLogfile.xml*. Δεν υπάρχει όμως σημαντική απόκλιση μεταξύ των 2 λειτουργιών.

6.3.2 Εκτέλεση Εργασιών

Οι ενεργές υπηρεσίας που προσφέρει το πλέγμα, είναι οι Paint, Swirl και Blend. Τις εκτελούμε για διαφορετικά αρχεία και παρουσιάζουμε τους χρόνους εκτέλεσης.

6.3.2.1 Υπηρεσία Swirl

Πίνακας 17: Η χρονική διάρκεια της Swirl.

Swirl		
Όνομα αρχείου	**Μέγεθος (bytes)**	**Χρόνος (milliseconds)**
Image1.png	11574	**16154**
Image2.png	13232	**17467**
Image3.png	81762	**18603**

6.3.2.2 Υπηρεσία Paint

Πίνακας 18: Η χρονική διάρκεια της Paint.

Paint		
Όνομα αρχείου	**Μέγεθος (bytes)**	**Χρόνος (milliseconds)**
Image1.png	11574	**17156**
Image2.png	13232	**18467**
Image3.png	81762	**19603**

6.3.2.3 Υπηρεσία Blend

Στην Blend τα ορίσματα δίνονται ανά δύο :

Πίνακας 19: Η χρονική διάρκεια της Blend.

Blend		
Όνομα αρχείου	**Μέγεθος (bytes)**	**Χρόνος (milliseconds)**
Image1.png Image2.png	11574 13232	**19600**
Image4.png Image5.png	5474 12890	**19391**
Image6.png Image1.png	12890 11574	**18834**

6.4 *Αξιολόγηση Συστήματος*

Σύμφωνα με τις προηγούμενες ενότητες, μπορούμε να αξιολογήσουμε πλέον το σύστημα μας. Ξεκινώντας από τον σχεδιασμό, η χρήση των ενδιάμεσων υποσυστημάτων καθιστά τον πυρήνα της εφαρμογής ιδιαίτερα ανεξάρτητο. Συγκεκριμένα η ύπαρξη του Axis Server ως μεσάζοντα της επικοινωνίας Android και Πλέγματος απαλλάσσει την εφαρμογή στην κινητή συσκευή από την χρήση των βιβλιοθηκών του *GRIA* για να εκμεταλλευτεί το API του. Με αυτόν τον τρόπο αίρονται οι περιορισμοί γλώσσας προγραμματισμού αλλά και λειτουργικού συστήματος. Αντίστοιχη εφαρμογή με την δική μας (***GRIAManager***) μπορεί να αναπτυχθεί με μεγάλη ευκολία σε ευρεία λειτουργικών συστημάτων και γλωσσών προγραμματισμού , χωρίς κάποια αλλαγή από μέρους μας στα μέρη του Axis Server και του GRIA Server. Η ιδιότητα αυτή , είναι ιδιαίτερα ωφέλιμη γνωρίζοντας την πληθώρα λειτουργικών συστημάτων για κινητές συσκευές αλλά και την αναμενόμενη αύξηση τους στο μέλλον.

Μία άλλη ευεργετική ιδιότητα είναι ότι η υλοποίηση των εργασιών στο πλέγμα δεν επηρεάζει στο ελάχιστο την γενικότερη λειτουργία του συστήματος. Ο γενικευμένος σχεδιασμός των εργασιών, επιτρέπει στον εξωτερικό χρήστη του πλέγματος είτε αυτός είναι προγραμματιστής , είτε πελάτης να επιλέγει και να χρησιμοποιεί εργασίες γνωρίζοντας μόνο τις απαιτούμενες διεπαφές (interfaces). Μια αλλαγή λοιπόν στην προγραμματιστική ή την επιχειρηματική λογική της εργασίας, όσο μεγάλη και αν είναι αυτή, δεν επιφέρει καμία παρενέργεια στα εξαρτώμενα μέρη.

Η ευκολία προσθήκης νέων υπηρεσιών μέσω του ***GRIA Basic Application*** αλλά και η αυτόματη ενημέρωση μας για τον τρόπο χρήσης των υπηρεσιών αυτών είναι ένα ιδιαίτερα ισχυρό πλεονέκτημα του συστήματος. Είναι ξεκάθαρο, ότι η ευελιξία που παρέχεται με την χρήση του πλέγματος δίνει την δυνατότητα στον χρήστη μία κινητής συσκευής να εκτελεί εργασίες πολύπλοκες και υπολογιστικά απαιτητικές. Κάτι τέτοιο δεν θα ήτανε εφικτό χωρίς την χρήση του GRID , δεδομένου της χαμηλής υπολογιστικής ισχύος των συσκευών αυτών. Η αγορά επίσης υπηρεσιών του πλέγματος για συγκεκριμένο χρονικό διάστημα, η ενοικίαση τους στην ουσία, ευνοεί τον περιστασιακό χρήστη που χρειάζεται μεγάλη υπολογιστική ισχύ και εξειδικευμένο λογισμικό για ορισμένο χρόνο. Σε αυτήν την περίπτωση δεν θα είχε νόημα η αγορά ολόκληρης της υπηρεσίας αλλά η χρήση της έναντι του αντιστοίχου κόστους.

Κάτι που δεν φαίνεται με την περιγραφή της λειτουργικότητας του GRIA Manager αλλά ισχύει, είναι η υποστήριξη για πολυνηματική επεξεργασία. Από την στιγμή που θα ξεκινήσουμε την εκτέλεση μιας εργασίας, δεν χρειάζεται να περιμένουμε μέχρι την ολοκλήρωσή της αλλά έχουμε την δυνατότητα να ξεκινήσουμε την εκτέλεση και άλλων εργασιών ή ακόμη να βγούμε

από την εφαρμογή μας και να κλείσουμε το κινητό. Οι εργασίες αυτές θα συνεχίσουν να εκτελούνται και όταν ολοκληρωθούν θα τοποθετήσουν τα αποτελέσματα στα Resources.

7

Επίλογος

Στο κεφάλαιο αυτό θα συνοψίσουμε την παρουσίαση της διπλωματικής παρουσιάζοντας τα συμπεράσματα που προκύπτουν από την ως τώρα ανάλυση.

7.1 *Σύνοψη και συμπεράσματα*

Όπως είδαμε , το σύστημα που υλοποιήθηκε μας επιτρέπει να εκμεταλλευτούμε όλη την δύναμη του πλέγματος στις κινητές συσκευές. Με την χρήση εφαρμογών που εκμεταλλεύονται τις ιδιότητες του GRID, μπορούμε να εκτελούμε με παράλληλο τρόπο υπολογιστικά απαιτητικές εργασίες απομακρυσμένα και αποδοτικά, αποδεσμεύοντας την κινητή συσκευή από την χρονοβόρα και επίπονη αυτή διαδικασία. Η ενσωμάτωση του πλέγματος δίνει στις φορητές συσκευές επεξεργαστική ισχύ κατ' απαίτηση χωρίς τα μειονεκτήματα που συνήθως αυτή συνοδεύεται.

Ο σχεδιασμός και η κατανομή του συστήματος σε επιμέρους υποσυστήματα, είναι σημείο κλειδί για την ομαλή και ανεξαρτητοποιημένη λειτουργία κάθε κόμβου. Ειδικότερα η ύπαρξη web services για την επικοινωνία μεταξύ της κινητής συσκευής και του πλέγματος , εξασφαλίζει υψηλή συμβατότητα με εφαρμογές που αναπτύσσονται με διαφορετική γλώσσα προγραμματισμού σε άλλο λειτουργικό σύστημα.

Τέλος η απόκρυψη της υλοποίησης κάθε εργασίας και η χρήση μόνο της διεπαφής της, επιφέρουν αφενός ευελιξία και αφετέρου υψηλή μελλοντική επεκτασιμότητα.

Βιβλιογραφία

[1] A. Menychtas ,D. Kyriazis, K.Tserpes, “Real-time Reconfiguration for Guaranteeing QoS Provisioning Levels in Grid Environments”.

[2] Debusmann, M.; Keller, A., "SLA-driven management of distributed systems using the common information model," Integrated Network Management, 2003. IFIP/IEEE Eighth International Symposium on , vol., no., pp. 563-576, 24-28 March 2003.

[3] Web Services Agreement Specification (WS-Agreement), OGF, available online: http://www.gridforum.org/Meetings/GGF11/Documents/draft-ggf-graap-agreement.pdf

[4] D. Kyriazis, K. Tserpes, A. Menychtas, I. Sarantidis and T. Varvarigou, “Service Selection and Workflow Mapping for Grids: An approach exploiting Quality of Service Information”, on Concurrency and Computation: Practice and Experience, Willey Interscience, 2008.

[5] L. Cheng, A. Wanchoo, I. Marsic, “*Hybrid Cluster Computing with Mobile Objects*”, Proc. of Fourth International Conference on High-Performance Computing in the Asia-Pacific Region, 2000.

[6] T.Phan, L. Huang, C. Dulan, “*Challenge: Integrating Mobile Wireless Devices into the Computational Grid*”, ACM MOBICOM, 2002

[7] G. H. Forman, J. Zahorjan, “*The Challenges of Mobile Computing*”, IEEE Computing Milieux, 1994

[8] Chlamtac, J. Redi, “*Mobile Computing: Challenges and Potential*”, Encyclopedia of Computer Science, 4th edition, 1998

[9] M. Franz, “*A Fresh Look at Low Power Mobile Computing*”, http://research.ac.upc.es/pact01/colp/paper15.pdf

[10] The Windows Mobile Platform : http://www.microsoft.com/windowsmobile/

[11] B. Chen, C. H. Chang, “*Building Low Power Wireless Grids*”, http://www.ee.tufts.edu/~brchen/pub/LowPower_WirelessGrids_1201.pdf

[12] D. Bruneo, M. Scarpa, A. Zaia, A. Puliafito, “*Communication Paradigms for Mobile Grid Users*”, IEEE/ACM International Symposium on Cluster Computing and the Grid, 2003

[13] K. Czajkowski, D. Ferguson, I. Foster, J. Frey, S. Graham, T. Maguire, D.Snelling, S. Tuecke, "*From Open Grid Services Infrastructure to WS-Resource Framework: Refac-toring & Evolution*", http://www.ibm.com/developerworks/library/ws-resource/ogsi_to_wsrf_1.0.pdf, 2004

[14] The *Android* project: http://code.google.com/android/index.html

[15] The *Axis2* project: http://ws.apache.org/axis2

[16] J. Hwang, P. Aravamudham, "*Proxy-based Middleware Services for Peer-to-Peer Computing in Virtually Clustered Wireless Grid Networks"*, in Proceedings of International Conference on Computer, Communication and Control Technologies, 2003

[17] The Symbian OS mobile platform : http://www.symbian.com

[18] D. Chu, M. Humphrey, "*Mobile OGSI.NET: Grid Computing on Mobile Devices*", http://www.cs.virginia.edu/~humphrey/papers/MobileOGSI.pdf , 2004

[19] The *GRIA* Project, http://www.gria.org