ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Προηγμένες Τεχνικές Παράλληλου Προγραμματισμού και Πλέγματος για Συστήματα Ασύρματων Επικοινωνιών

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Θεόδωρος Ε. Αθαναηλέας

Διπλωματούχος Ηλεκτρολόγος Μηχανικός
και Μηχανικός Υπολογιστών ΕΜΠ

Αθήνα, Ιούνιος 2009

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Προηγμένες Τεχνικές Παράλληλου Προγραμματισμού και Πλέγματος για Συστήματα Ασύρματων Επικοινωνιών

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Θεόδωρος Ε. Αθαναηλέας

Συμβουλευτική Επιτροπή : Δήμητρα Ι. Κακλαμάνη
Νικόλαος Κ. Ουζούνογλου
Ιάκωβος Σ. Βενιέρης

Εγκρίθηκε από την επταμελή εξεταστική επιτροπή την 12η Ιουνίου 2009.

.....	.....	.....
Δήμητρα Κακλαμάνη	Νικόλαος Ουζούνογλου	Ιάκωβος Βενιέρης
Καθηγήτρια ΕΜΠ, ΣΗΜΜΥ	Καθηγητής ΕΜΠ, ΣΗΜΜΥ	Καθηγητής ΕΜΠ, ΣΗΜΜΥ
.....	.....	.....
Γεώργιος Τσούλος	Μιχαήλ Θεολόγου	Γεωργία Αθανασιάδου
Επίκ. Καθηγητής Πανεπ. Πελοποννήσου	Καθηγητής ΕΜΠ, ΣΗΜΜΥ	Επίκ. Καθηγήτρια Πανεπ. Πελοποννήσου
	.....	
	Θεοδώρα Βαρβαρίγου	
	Καθηγήτρια ΕΜΠ, ΣΗΜΜΥ	

Αθήνα, Ιούνιος 2009

.....
Θεόδωρος Ε. Αθαναηλέας

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

Copyright © Θεόδωρος Ε. Αθαναηλέας, 2009
Με επιφύλαξη παντός δικαιώματος. All rights reserved.

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

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


Γ' ΚΟΙΝΟΤΙΚΟ ΠΛΑΙΣΙΟ ΣΤΗΡΙΞΗΣ
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΑΝΤΑΓΩΝΙΣΤΙΚΟΤΗΤΑ
ΕΥΡΩΠΑΪΚΗ ΕΝΩΣΗ
ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ


3rd COMMUNITY SUPPORT FRAMEWORK
OPERATIONAL PROGRAMME
COMPETITIVENESS
EUROPEAN UNION
HELLENIC REPUBLIC

ΠΕΡΙΛΗΨΗ

Αντικείμενο της παρούσας διδακτορικής διατριβής είναι η αντιμετώπιση των αυξημένων υπολογιστικών απαιτήσεων συγκεκριμένων προβλημάτων που σχετίζονται με την προσομοίωση, βελτιστοποίηση και σχεδίαση σύγχρονων συστημάτων ασύρματων και κινητών επικοινωνιών, με χρήση τεχνικών παράλληλης και κατανεμημένης επεξεργασίας και με χρήση τεχνολογιών υπολογιστικού πλέγματος.

Καθώς η χρήση προηγμένων και όλο και πολυπλοκότερων τεχνικών και μοντέλων (για παράδειγμα έξυπνες κεραίες, τεχνικές MIMO, στοχαστικές μέθοδοι βελτιστοποίησης, ντετερμινιστική πρόβλεψη διάδοσης) γίνεται συχνότερη στα συστήματα ασύρματων επικοινωνιών, οι υπολογιστικές απαιτήσεις των προβλημάτων που σχετίζονται με αυτά αυξάνονται με γρήγορους ρυθμούς. Η επίλυση των προβλημάτων αυτών μπορεί να γίνει εξαιρετικά απαιτητική σε υπολογιστικούς χρόνους, ή ακόμη και πρακτικά αδύνατη σε ορισμένες περιπτώσεις. Στα πλαίσια της διατριβής, πραγματοποιήθηκε μελέτη τριών συγκεκριμένων προβλημάτων που σχετίζονται με την προσομοίωση, βελτιστοποίηση και σχεδίαση συστημάτων ασύρματων και κινητών επικοινωνιών, και επιχειρήθηκε η ανάπτυξη και υλοποίηση κατάλληλων μεθόδων για την αντιμετώπιση των αυξημένων υπολογιστικών αναγκών τους. Υλοποιήθηκε παράλληλο μοντέλο πρόβλεψης διάδοσης σε ραδιοκανάλι με βάση την τεχνική ανίχνευσης ακτίνων και την ηλεκτρομαγνητική θεωρία των ειδώλων. Η υλοποίηση έγινε χρησιμοποιώντας το υπολογιστικό παράδειγμα της διεπαφής προώθησης μηνυμάτων. Παρουσιάζονται πειραματικά αποτελέσματα από την εφαρμογή του παράλληλου μοντέλου σε προβλήματα διαφορετικού μεγέθους, τα οποία δείχνουν ότι ο παράλληλος αλγόριθμος επιτυγχάνει σχεδόν ιδανική κλιμάκωση σε όλες τις περιπτώσεις.

Επίσης, αναπτύχθηκε κατανεμημένη πλατφόρμα βελτιστοποίησης νέφους σωματιδίων με βάση το υπολογιστικό παράδειγμα των κινητών πρακτόρων λογισμικού. Ο αλγόριθμος βελτιστοποίησης νέφους σωματιδίων περιλαμβάνει ένα μεγάλο πλήθος εκτιμήσεων μίας συνάρτησης κόστους, η οποία για τα προβλήματα ασύρματων επικοινωνιών είναι συχνά μία χρονοβόρα και μνημοβόρα διαδικασία. Η προσέγγιση που ακολουθήθηκε επιτρέπει την παράλληλη και κατανεμημένη εκτέλεση του αλγορίθμου σε ένα σύνολο ετερογενών υπολογιστών που συνδέονται μέσω δικτύου, παρέχοντας ευκολία και ευελιξία στην εγκατάσταση, καθώς και δυνατότητα για εύκολη επέκταση. Παρουσιάζονται πειραματικά αποτελέσματα που προέκυψαν από την εφαρμογή του συστήματος στην επίλυση προβλήματος βελτιστοποίησης διαγράμματος ακτινοβολίας στοιχειοκεραίας.

Τέλος, αναπτύχθηκε δικτυακό περιβάλλον με βάση τις τεχνολογίες ιστού για την προσομοίωση συστημάτων ασύρματων επικοινωνιών σε υποδομή πλέγματος. Το περιβάλλον παρέχει μια ασφαλή δικτυακή πύλη με βάση τα πρότυπα των portlets ως διεπαφή στο μεσισμικό της υποδομής πλέγματος, η οποία αποκρύπτει από τον τελικό χρήστη την πολυπλοκότητα χρήσης του πλεγματικού μεσισμικού. Το περιβάλλον προσομοίωσης χρησιμοποιήθηκε για την εκτέλεση μιας σειράς προσομοιώσεων συστημάτων ασύρματων επικοινωνιών, ενώ παρουσιάζονται στατιστικά από την εκτέλεση στο πλέγμα ορισμένων σεναρίων για συστήματα WCDMA.

Η παρούσα διδακτορική διατριβή υποστηρίχθηκε από το Πρόγραμμα Ενίσχυσης Ερευνητικού Δυναμικού (ΠΕΝΕΔ) 2003, το οποίο συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (80%) και εθνικούς πόρους (20%).

Λέξεις Κλειδιά: παράλληλα και κατανεμημένα συστήματα, συστήματα ασύρματων επικοινωνιών, πρόβλεψη διάδοσης, ανίχνευση ακτίνων, θεωρία ειδώλων, μεσισμικό, MPI, κινητοί πράκτορες, βελτιστοποίηση σμήνους σωματιδίων, υπολογιστικό πλέγμα, πύλη πλέγματος, portlets

ABSTRACT

The goal of this doctoral thesis is to deal with the increased computational requirements of specific problems concerning the simulation, optimization and planning of modern wireless and mobile communication systems, by employing parallel and distributed processing techniques and computational grid technologies.

While the use of advanced and increasingly complex techniques and models (for example smart antennas, MIMO techniques, stochastic optimization methods, deterministic propagation prediction) becomes more frequent for wireless communication systems, the computational requirements of the concerning problems increase rapidly. The computation of these problems may become very time consuming or even practically infeasible in certain conditions. In the scope of the thesis, three specific problems regarding simulation, optimization and planning of wireless and mobile communication systems have been studied and the development and implementation of suitable methods for dealing with their increased computational needs has been attempted. A parallel radio-wave propagation prediction model based on ray tracing techniques and the electromagnetic theory of images has been developed. The implementation has been based on the computational paradigm of the message passing interface. Experimental results from utilizing the parallel model for problems of different sizes are presented, which demonstrate that the parallel algorithm achieves almost ideal scalability in all cases.

Also, a distributed particle swarm optimization platform has been developed, based on the computational paradigm of mobile software agents. The particle swarm optimization algorithm entails a large number of computations of a cost function, which in the case of wireless communication systems can become time and memory consuming. The approach adopted enables the parallel and distributed processing of the algorithm on a number of heterogeneous computational nodes connected to each other via a network, providing ease and flexibility at deployment, as well as easy extension. Experimental results are presented by exploiting the system for the solution of an antenna array radiation pattern optimization.

Finally, a networked environment for the simulation of wireless communication systems in a grid infrastructure has been developed, based on web technologies. The environment provides a safe web portal based on portlet specifications as the interface to the grid middleware, which hides from the end user the complexity of using the grid middleware. The simulation environment has been employed for executing a number of simulations and

statistics from the execution of several WCDMA system simulation scenarios on the grid are presented.

This doctoral dissertation has been supported by PENED 2003 project. The project is co-funded by the European Union (80%) and national resources (20%).

Keywords: parallel and distributed systems, wireless communication systems, propagation prediction, ray tracing, image theory, middleware, MPI, mobile agents, particle swarm optimization, computational grid, grid portal, portlets

Η παρούσα διατριβή εκπονήθηκε στο Εργαστήριο Μικροκυμάτων και Οπτικών Ινών της Σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Η/Υ του Εθνικού Μετσόβιου Πολυτεχνείου.

Θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες στην επιβλέπουσα καθηγήτριά μου, Δήμητρα Κακλαμάνη, για τη συνεισφορά της σε όλη τη διάρκεια της εκπόνησης της διδακτορικής διατριβής, καθώς και για την εμπιστοσύνη που επέδειξε στο πρόσωπό μου.

Θα ήθελα ακόμη να ευχαριστήσω ιδιαίτερα τον διευθυντή του εργαστηρίου, καθηγητή Νικόλαο Ουζούνογλου, καθώς και τον καθηγητή Ιάκωβο Βενιέρη, για τις πολύτιμες ιδέες και συμβουλές που παρείχαν σε πολλά προβλήματα που συνάντησα.

Θερμές ευχαριστίες οφείλω στους καθηγητές του Πανεπιστημίου Πελοποννήσου Γιώργο Τσούλο και Γεωργία Αθανασιάδου, για την εξαίρετη και αποδοτική συνεργασία μας στα χρόνια εκπόνησης της διδακτορικής διατριβής.

Ακόμη, θα ήθελα ιδιαίτερα να ευχαριστήσω τους φίλους και συνεργάτες Δρ. Παναγιώτη Γκόνη, Δρ. Δήμητρα Ζαρμπούτη, Δρ. Αθηνά Οικονόμου, Κώστα Παπαδόπουλο, Δρ. Δημήτρη Τσιλιμαντό, Δρ. Δημήτρη Κατέρο, Χρήστο Παππά, Δρ. Γεωργία Καπιτσάκη, Γιάννη Στειακογιαννάκη, Δρ. Νίκο Τσελίκα, Δρ. Δήμητρα Διονυσίου, Δρ. Αντρέα Μενύχτα και Δρ. Γιώργο Σταματάκο για την άριστη συνεργασία όλα αυτά τα χρόνια.

Τέλος, θα ήθελα να ολοκληρώσω τις αναφορές στους ανθρώπους, που με στήριξαν, ευχαριστώντας την οικογένειά μου για την αδιάλειπτη ηθική και ψυχολογική συμπαράσταση.

Θεόδωρος Ε. Αθαναηλέας

Αθήνα, Ιούνιος 2009

Κατάλογος περιεχομένων

1 Εισαγωγή - Γενική Επισκόπηση 23
2 Τεχνολογίες Παράλληλης και Κατανεμημένης Επεξεργασίας 27
2.1 Αρχιτεκτονικές Παράλληλων Συστημάτων 29
2.1.1 Κατηγοριοποίηση κατά Flynn 29
2.1.2 Κατηγοριοποίηση με βάση την Αρχιτεκτονική Επικοινωνίας 33
2.2 Μοντέλα Παράλληλου Προγραμματισμού 36
2.2.1 Πολυνηματικός Προγραμματισμός 36
2.2.2 Παράλληλη Εικονική Μηχανή (Parallel Virtual Machine – PVM) 39
2.2.3 Διεπαφή Προώθησης Μηνυμάτων (Message Passing Interface - MPI) 40
2.3 Πλατφόρμες και Τεχνολογίες Κατανεμημένων Αντικειμένων 45
2.3.1 Απομακρυσμένη Κλήση Μεθόδου (Remote Method Invocation – RMI) 45
2.3.2 Jini 46
2.3.3 Common Object Request Broker Architecture (CORBA) 47
2.3.4 Κινητοί Πράκτορες 49
2.4 Απόδοση Παράλληλων Συστημάτων – Δείκτες Απόδοσης 52
2.4.1 Βασικοί Δείκτες Απόδοσης 53
2.4.2 Παράγοντες που Επηρεάζουν την Κλιμάκωση 55
3 Παράλληλος Αλγόριθμος Υπολογισμού Πρόβλεψης Διάδοσης σε Ραδιοκανάλι με βάση την Τεχνική Ανίχνευσης Ακτίνων 59
3.1 Εισαγωγή 60
3.2 Σχετικές Εργασίες 62
3.3 Σειριακό Μοντέλο Ray Tracing 64
3.3.1 Βασική Λειτουργία 65
3.3.2 Δέντρο Ειδώλων 66
3.3.3 Ζώνες Φωτισμού Ειδώλων 67
3.3.4 Σειριακός Αλγόριθμος 68
3.4 Παράλληλο Μοντέλο Ray Tracing 70
3.4.1 Αποσύνθεση Υπολογιστικού Φορτίου - Σχηματισμός Επιμέρους Εργασιών 71
3.4.2 Κατανομή Εργασιών και Ισοκατανομή Υπολογιστικού Φορτίου 73
3.4.3 Αλληλοεπικάλυψη Επικοινωνίας- Επεξεργασίας 75
3.4.4 Υλοποίηση 75
3.5 Πειραματικά Αποτελέσματα - Απόδοση Παράλληλης Υλοποίησης 77
3.5.1 Ανάλυση Απόδοσης Παράλληλης Υλοποίησης 80
3.5.2 Σύγκριση Σχημάτων Ανάθεσης Εργασιών 90
4 Σύστημα Πολλαπλών Πρακτόρων για την Κατανεμημένη Εκτέλεση του Αλγορίθμου Βελτιστοποίησης Σμήνους Σωματιδίων 93
4.1 Εισαγωγή 93
4.2 Βελτιστοποίηση Σμήνους Σωματιδίων 95
4.2.1 Βασική Ορολογία 96
4.2.2 Αλγόριθμος 97
4.3 Παράλληλος Αλγόριθμος 100
4.4 Υλοποίηση Συστήματος 101
4.4.1 Η Πλατφόρμα JADE 102
4.4.2 Κατανεμημένες Οντότητες 109
4.5 Υλοποίηση 113
4.6 Εγκατάσταση – Ρυθμίσεις 114

4.7 Πειράματα - Χρόνοι Εκτέλεσης 117
5 Τεχνολογία Υπολογιστικού Πλέγματος 121
5.1 Ιστορικά Στοιχεία 122
5.2 Εικονικοί Οργανισμοί 124
5.3 Αρχιτεκτονικές Πλέγματος 126
5.3.1 Γενική Προσέγγιση Πλεγματικής Υποδομής 128
5.3.2 Γενική Αρχιτεκτονική Υπολογιστικού Πλέγματος 129
5.3.3 Open Grid Services Architecture (OGSA) 132
5.4 Μεσισμικά Πλέγματος 138
5.5 Χρησιμότητα Τεχνολογίας Πλέγματος - Οφέλη 138
6 Περιβάλλον Προσομοίωσης Συστημάτων Ασύρματων Επικοινωνιών σε Πλέγμα 141
6.1 Εισαγωγή 142
6.2 Η Υποδομή Πλέγματος 143
6.2.1 Υποδομή Ασφάλειας 146
6.2.2 Σύστημα Διαχείρισης Υπολογιστικού Φορτίου (Workload Management System - WMS) 148
6.2.3 Γλώσσα Περιγραφής Εργασίας 151
6.2.4 Σύστημα Διαχείρισης Δεδομένων (Data Management System – DMS) 152
6.2.5 Σύστημα Πληροφοριών (Information System - IS) 152
6.2.6 Ροή Εργασιών 153
6.3 Απαιτήσεις για το Περιβάλλον Προσομοίωσης 156
6.4 Χρησιμοποιηθείσες Τεχνολογίες 157
6.4.1 JSR-168 Portlets 157
6.4.2 GridSphere 159
6.4.3 MyProxy 160
6.5 Αρχιτεκτονική του Περιβάλλοντος Προσομοίωσης - Υλοποίηση 161
6.5.1 Portlets 164
6.5.2 Υπηρεσίες Πύλης 167
6.5.3 Διαδικασία Εκτέλεσης Προσομοίωσης στο Πλέγμα 169
6.6 Στατιστικά Εκτέλεσης Προσομοιώσεων στο Πλέγμα 170
7 Σύνοψη και Θέματα Μελλοντικής Έρευνας 175
7.1 Σύνοψη και Συμπεράσματα της Ερευνητικής Δραστηριότητας 175
7.2 Θέματα Μελλοντικής Έρευνας 177
7.2.1 Ανάπτυξη Υβριδικής Παράλληλης Υλοποίησης για τον Αλγόριθμο Πρόβλεψης Διάδοσης 177
7.2.2 Ανάπτυξη Παράλληλου Μοντέλου Πρόβλεψης Διάδοσης σε Κάρτες Γραφικών Γενικής Χρήσης 177
7.2.3 Υλοποίηση και Μελέτη Κατανεμημένων Υλοποιήσεων Διαφορετικών Εκδοχών του Αλγορίθμου PSO 178
7.2.4 Ανάπτυξη Ολοκληρωμένου Περιβάλλοντος για τη Βελτιστοποίηση και Σχεδίαση Συστημάτων Ασύρματων Επικοινωνιών στο Πλέγμα 178
8 Βιβλιογραφία 181
9 Παραρτήματα 191
9.1 Κατάλογος Δημοσιεύσεων 191
9.1.1 Διεθνή Επιστημονικά Περιοδικά 191
9.1.2 Επιστημονικά Συνέδρια 191
9.1.3 Δημοσιεύσεις σε Διαδικασία Κρίσης 192
9.1.4 Αναφορές από Τρίτους 193

Κατάλογος σχημάτων

Σχήμα 1. Κατηγοριοποίηση κατά Flynn 30
Σχήμα 2. Αρχιτεκτονική SIMD 31
Σχήμα 3. Αρχιτεκτονική MIMD....32
Σχήμα 4. Σύστημα Διαμοιραζόμενης Μνήμης....34
Σχήμα 5. Σύστημα Επικοινωνίας μέσω Δικτύου Διασύνδεσης....35
Σχήμα 6. Αρχιτεκτονική Συστήματος RMI....47
Σχήμα 7. Αρχιτεκτονική Συστήματος CORBA....49
Σχήμα 8. Το υπολογιστικό παράδειγμα του κινητού αντιπρόσωπου 50
Σχήμα 9. Αναπαράσταση διαδρομής λόγω ανακλάσεων στον αλγόριθμο 65
Σχήμα 10. Αναπαράσταση διαδρομών με εμπόδια στον αλγόριθμο 66
Σχήμα 11. Απλό παράδειγμα δέντρου ειδώλων 67
Σχήμα 12. Τεχνική Ζωνών Φωτισμού 68
Σχήμα 13. Αποσύνθεση Υπολογισμών 72
Σχήμα 14. Τοποθεσίες κτιρίων, φυλλωμάτων, και θέσης BS στο τμήμα της περιοχής του Cambridge που θεωρήθηκε 77
Σχήμα 15. Αριθμός υπο-περιοχών, ειδώλων πρώτης τάξης και ειδώλων δεύτερης τάξης που υπολογίστηκαν από τον κάθε επεξεργαστή για 17 επεξεργαστές και για την Περιοχή A 83
Σχήμα 16. Χρόνοι επεξεργασίας υπο-περιοχών, ειδώλων πρώτης τάξης και ειδώλων δεύτερης τάξης που υπολογίστηκαν από τον κάθε επεξεργαστή για 17 επεξεργαστές και για την Περιοχή A 84
Σχήμα 17. Αριθμός υπο-περιοχών, ειδώλων πρώτης τάξης και ειδώλων δεύτερης τάξης που υπολογίστηκαν από τον κάθε επεξεργαστή για 17 επεξεργαστές και για την Περιοχή B 84
Σχήμα 18. Χρόνοι επεξεργασίας υπο-περιοχών, ειδώλων πρώτης τάξης και ειδώλων δεύτερης τάξης που υπολογίστηκαν από τον κάθε επεξεργαστή για 17 επεξεργαστές και για την Περιοχή B 84
Σχήμα 19. Αριθμός υπο-περιοχών, ειδώλων πρώτης τάξης και ειδώλων δεύτερης τάξης που υπολογίστηκαν από τον κάθε επεξεργαστή για 17 επεξεργαστές και για την Περιοχή C 85
Σχήμα 20. Χρόνοι επεξεργασίας υπο-περιοχών, ειδώλων πρώτης τάξης και ειδώλων δεύτερης τάξης που υπολογίστηκαν από τον κάθε επεξεργαστή για 17 επεξεργαστές και για την Περιοχή C 85
Σχήμα 21. Επιτάχυνση για τις περιοχές A,B και C και για 3 τάξεις ανακλάσεων....86
Σχήμα 22. Αποδοτικότητα για τις περιοχές A, B και C και 3 τάξεις ανακλάσεων....87
Σχήμα 23. Επιτάχυνση για 3, 4 και 5 τάξεις ανακλάσεων και για την περιοχή B 88
Σχήμα 24. Αποδοτικότητα για 3, 4 και 5 τάξεις ανακλάσεων και για την περιοχή B...88
Σχήμα 25. Παράγοντας Επέκτασης Υπολογιστικού Φορτίου για τις Περιοχές A, B και C και για 3 τάξεις ανακλάσεων....89
Σχήμα 26. Βαθμός Αξιοποίησης Πόρων για τις Περιοχές A, B και C και για 3 τάξεις ανακλάσεων 90
Σχήμα 27. Λαμβανόμενη ισχύς για την περιοχή C και για 3 τάξεις ανακλάσεων 91
Σχήμα 28. Περιβάλλον Εκτέλεσης του JADE 104
Σχήμα 29. Κύκλος Ζωής Πρακτόρων κατά FIPA 106
Σχήμα 30. Διάγραμμα Ροής Εκτέλεσης Πράκτορα 107
Σχήμα 31. Κύκλος Ζωής του CoordinatorAgent....111
Σχήμα 32. Κύκλος Ζωής του ParticleAgent....112

Σχήμα 33. UML διάγραμμα των κλάσεων πρακτόρων και συμπεριφορών της πλατφόρμας 114
Σχήμα 34. Χρόνοι εκτέλεσης βελτιστοποίησης για διαφορετικό αριθμό κόμβων.....118
Σχήμα 35. Η διαστρωματωμένη αρχιτεκτονική του υπολογιστικού πλέγματος και η αντιστοιχία της με την αρχιτεκτονική του διαδικτύου.....131
Σχήμα 36. Η αρχιτεκτονική OGSA 134
Σχήμα 37. Ροή εργασίας κατά την υποβολή της στο πλέγμα 155
Σχήμα 38. Λειτουργία του συστήματος MyProxy σε εφαρμογή πύλης πλέγματος....162
Σχήμα 39. Συνολική Αρχιτεκτονική Εφαρμογής 163
Σχήμα 40. Σελίδα διαχείρισης μακροπρόθεσμων πιστοποιητικών στην πύλη 165
Σχήμα 41. Σελίδα διαχείρισης πληρεξούσιων πιστοποιητικών στον εξυπηρετητή MyProxy 163
Σχήμα 42. Σελίδα διαχείρισης πληρεξούσιων πιστοποιητικών στην πύλη.....165
Σχήμα 43. Σελίδα διαχείρισης κωδίκων προσομοίωσης και αρχείων εισόδου στην πύλη 166
Σχήμα 44. Σελίδα υποβολής νέας προσομοίωσης.....167
Σχήμα 45. Σελίδα επίβλεψης και διαχείρισης υποβληθείσας προσομοίωσης.....167
Σχήμα 46. Διάγραμμα καταστάσεων κατά την εκτέλεση προσομοίωσης στο πλέγμα171

Κατάλογος Πινάκων

Πίνακας 1. Στάδια του Παράλληλου Μοντέλου Πρόβλεψης Διάδοσης.....76
Πίνακας 2. Set-up πειραματικών εκτελέσεων.....79
Πίνακας 3. Στατιστικά για την Περιοχή Α (500mx500m).....80
Πίνακας 4. Στατιστικά για την Περιοχή Β (1000mx1000m) 81
Πίνακας 5. Στατιστικά για την Περιοχή C (1500mx1500m) 82
Πίνακας 6. Καλύτερο και χειρότερο σχήμα ανάθεσης για 17 επεξεργαστές.....92
Πίνακας 7. Καλύτερο και χειρότερο σχήμα ανάθεσης για 10 επεξεργαστές.....92
Πίνακας 8. Καλύτερο και χειρότερο σχήμα ανάθεσης για 3 επεξεργαστές.....92
Πίνακας 9. Χρόνοι Εκτέλεσης και επιταχύνσεις για διαφορετικά σενάρια.....118
Πίνακας 10. Στατιστικά Εκτέλεσης στο πλέγμα (1).....172
Πίνακας 11. Στατιστικά Εκτέλεσης στο Πλέγμα (2).....173

Listings Κώδικα

Listing 1. Βασικός Σκελετός Προγράμματος MPI.....42
Listing 2. Δρομολόγηση εργασιών με χρήση της cpu affinity mask.....78
Listing 3. Ψευδοκώδικας παράλληλου σύγχρονου αλγορίθμου PSO 100
Listing 4. Λειτουργικότητα του CoordinatorAgent.....113
Listing 5. Λειτουργικότητα του ParticleAgent 114
Listing 6. Ορισμός Προβλήματος Στοιχειοκεραίας.....115
Listing 7. Αρχείο Ιδιοτήτων για την αρχικοποίηση της κατανεμημένης πλατφόρμας 116
Listing 8. XML Αρχείο Παραμέτρων του PSO 117
Listing 9. Παράδειγμα αρχείου περιγραφής εργασίας.....151

Κατάλογος Συντομογραφιών

MIMO – Multiple Input, Multiple Output
SISD – Single Instruction, Single Data
SIMD – Single Instruction, Multiple Data
MISD – Multiple Instruction, Single Data
MIMD – Multiple Instruction, Multiple Data
SPMD – Single Process, Multiple Data
MPMD – Multiple Process, Multiple Data
UMA – Uniform Memory Access
SMP – Symmetric Multi-Processor
PVM – Parallel Virtual Machine
MPI – Message Passing Interface
RMI – Remote Method Invocation
JVM – Java Virtual Machine
CORBA – Common Object Request Broker Architecture
OMG – Object Management Group
IDL – Interface Description Language
ORB – Object Request Broker
GIOP – General Inter-Orb Protocol
IIOP – Internet Inter-Orb Protocol
FIPA – Foundation for Intelligent Physical Agents
ACL – Agent Communication Language
SOAP – Simple Object Access Protocol
XML – Extensible Markup Language
FWA – Fixed Wireless Access
SBR – Shoot and Bouncing Ray
BS – Base Station
MS – Mobile Station
LOS – Line of Sight
PSO – Particle Swarm Optimization
CT – container table
GADT – global agent descriptor table
AMS – Agent Management System
DF – Directory Facilitator
LADT – local agent descriptor table
AID – agent identifier
MTP – Message Transport Protocol
IMTP – Internal Message Transport Protocol
API – Application Programming Interface
JAR – Java Archive
OGSA – Open Grid Services Architecture)
OGSI – Open Grid Services Infrastructure)
WSRF – Web Services Resource Framework
VO – Virtual Organization
QoS – Quality of Service
SDK – Software Development Kit
SOA – Service Oriented Architecture
WSDL – Web Services Description Language

WSA – Web Services Architecture
SOAP – Simple Object Access Protocol
RPC – Remote Procedure Call
URL – Uniform Resource Locator
UDDI – Universal Description, Discovery and Integration
GWSDL – Grid Web Services Description Language
WSRF-RP – Web Services Resource Framework – Resource Properties
WSRF-RL – Web Services Resource Framework – Resource Lifetime
WSRF-SG – Web Services Resource Framework – Service Group
WSRF-BF – Web Services Resource Framework – Base Faults
EGEE – Enabling Grids for E-Science
WCDMA – Wideband Code Division Multiple Access
CA – Certification Authority
UI – User Interface
CE – Computing Element
GG – Grid Gate
LRMS – Local Resource Management System
WN – Worker Node
SE – Storage Element
MSS – Mass Storage System
IS – Information System
MDS – Monitoring and Discovery System
R-GMA – Relational Grid Monitoring Architecture
DMS – Data Management System
GUID – Grid Unique Identifier
LFN – Logical File Name
SURL – Storage URL
TURL – Transport URL
WMS – Workload Management System
JDL – Job Description Language
DLI – Data Location Interface
LB – Logging and Bookkeeping
SN – Subject Name
VOMS – VO Management Service
AC – Attribute Certificate
RB – Resource Broker
WM – Workload Manager
RLS – Replica Location Service
LRC – Local Replica Catalog
RMC – Replica Metadata Catalog
GRIS – Grid Resource Information Service
GIIS – Grid Information Index Service
BDII – Berkeley Database Information Index
SSH – Secure Shell
RNG – Random Number Generators
FGoB – Fixed Grid of Beams

1 Εισαγωγή - Γενική Επισκόπηση

Την τελευταία δεκαετία έχει παρουσιαστεί μια εκρηκτική εξάπλωση των συστημάτων ασύρματων και κινητών επικοινωνιών. Τα συστήματα αυτά έχουν διεισδύσει στην καθημερινή ζωή της συντριπτικής πλειοψηφίας του πληθυσμού, ενώ παράλληλα παρατηρείται μια συνεχώς αυξανόμενη ζήτηση για υπηρεσίες υψηλών ρυθμών. Η διαρκής αυτή ζήτηση για ολοένα και υψηλότερους ρυθμούς μετάδοσης έχει δώσει ώθηση στην ερευνητική δραστηριότητα για αναζήτηση τεχνικών μετάδοσης με μεγαλύτερη απόδοση. Πλέον γίνεται ολοένα και πιο συνήθης η χρήση τεχνικών όπως οι έξυπνες κεραίες και τα συστήματα πολλαπλής εισόδου – πολλαπλής εξόδου (Multiple Input Multiple Output - MIMO) [1]. Ένα φυσικό συνεπακόλουθο από τη χρησιμοποίηση ολοένα και πολυπλοκότερων τεχνικών είναι η αύξηση των υπολογιστικών απαιτήσεων των προβλημάτων προσομοίωσης, βελτιστοποίησης και σχεδίασης των σύγχρονων συστημάτων ασύρματων και κινητών επικοινωνιών. Οι χρόνοι που απαιτούνται για την εκτέλεση των προβλημάτων μπορούν να γίνουν αρκετά μεγάλοι (όταν για παράδειγμα λαμβάνονται υπόψη αυξημένες τάξεις τομεοποίησης σε ένα πολυκυψελωτό σύστημα ή όταν πραγματοποιείται συνδυασμός προσομοίωσης σε επίπεδο συστήματος και σε επίπεδο ζεύξης), με αποτέλεσμα να γίνεται η επίλυση του αντίστοιχου προβλήματος εξαιρετικά χρονοβόρα. Το πρόβλημα αυτό γίνεται ακόμη πιο έντονο όταν χρησιμοποιούνται στοχαστικές μέθοδοι για τη βελτιστοποίηση και σχεδίαση των δικτύων. Τεχνικές όπως οι γενετικοί αλγόριθμοι και η βελτιστοποίηση σμήνους σωματιδίων χρησιμοποιούνται πλέον ερευνητικά όλο και σε μεγαλύτερο βαθμό για την επίλυση προβλημάτων βελτιστοποίησης που σχετίζονται με συστήματα ασύρματων επικοινωνιών. Οι τεχνικές αυτές μπορούν να παρέχουν λύση σε πολυδιάστατα προβλήματα βελτιστοποίησης και στηρίζονται στην επαναλαμβανόμενη εκτέλεση του βασικού προγράμματος επίλυσης για την αποτίμηση των

αποτελεσμάτων μιας αντικειμενικής συνάρτησης. Οι απαιτούμενες επαναλήψεις είναι συχνά αρκετά μεγάλες σε αριθμό μέχρι να επιτευχθεί σύγκλιση και κατ' επέκταση αυξάνεται και ο απαιτούμενος υπολογιστικός χρόνος.

Επιπλέον, η σχεδίαση των σύγχρονων δικτύων ασύρματων επικοινωνιών απαιτεί τη χρήση κατάλληλων εργαλείων που παρέχουν την απαραίτητη πληροφορία για αυτό το σκοπό. Ανάμεσα σε αυτά τα εργαλεία συγκαταλέγονται και μοντέλα που παρέχουν ακριβή δεδομένα διάδοσης σε κάποια συγκεκριμένη περιοχή με βάση ντετερμινιστικές τεχνικές. Τα σύγχρονα συστήματα ασύρματων επικοινωνιών εξαρτώνται σε μεγάλο βαθμό από τα συγκεκριμένα χαρακτηριστικά της κάθε περιοχής, συνεπώς τα μοντέλα διάδοσης απαιτείται να λαμβάνουν υπόψη τους την ακριβή θέση, τον προσανατολισμό, καθώς και τις ηλεκτρικές ιδιότητες του κάθε κτιρίου ή άλλου εμποδίου ξεχωριστά. Μια τεχνική που παρέχει ντετερμινιστικά μοντέλα πρόβλεψης διάδοσης ραδιοκαναλιού είναι η ανίχνευση ακτίνων, η οποία εφαρμόζεται σε συγκεκριμένα περιβάλλοντα-περιοχές που παρέχονται από το χρήστη. Το βασικό μειονέκτημα των ντετερμινιστικών μοντέλων του ραδιοκαναλιού είναι οι κατά πολύ αυξημένες υπολογιστικές απαιτήσεις σε σχέση με τα αντίστοιχα στατιστικά. Αυξάνοντας τη ζητούμενη ακρίβεια και θεωρώντας μεγάλες περιοχές, ο υπολογιστικός χρόνος που απαιτείται για την εξαγωγή αποτελεσμάτων μπορεί να γίνει εξαιρετικά μεγάλος.

Αντικείμενο της παρούσας διδακτορικής διατριβής είναι η αντιμετώπιση των αυξημένων υπολογιστικών απαιτήσεων συγκεκριμένων προβλημάτων που σχετίζονται με την προσομοίωση, βελτιστοποίηση και σχεδίαση συστημάτων ασύρματων και κινητών επικοινωνιών, με χρήση τεχνικών παράλληλης και κατανεμημένης επεξεργασίας και με χρήση των τεχνολογιών υπολογιστικού πλέγματος. Στο Κεφάλαιο 2 πραγματοποιείται μια εισαγωγή στην παράλληλη και κατανεμημένη επεξεργασία. Γίνεται περιγραφή των σημαντικότερων τεχνολογιών και τεχνικών, ενώ παρουσιάζονται ορισμένα θέματα σχετικά με την απόδοση των παράλληλων και κατανεμημένων συστημάτων. Στο Κεφάλαιο 3 παρουσιάζεται η υλοποίηση παράλληλου αλγορίθμου που υλοποιήθηκε στα πλαίσια της διατριβής για την πρόβλεψη διάδοσης σε ραδιοκανάλι με βάση την τεχνική ανίχνευσης ακτίνων. Το μοντέλο πρόβλεψης διάδοσης που χρησιμοποιήθηκε βασίζεται στην ηλεκτρομαγνητική θεωρία των ειδώλων και χρησιμοποιείται για την μοντελοποίηση περιβαλλόντων για συστήματα σταθερής ασύρματης πρόσβασης. Στα πλαίσια της διατριβής αναπτύχθηκε μια τεχνική παραλληλοποίησης που συνίσταται στη σταδιακή κατασκευή του δέντρου των ειδώλων. Κατά την υλοποίηση του παράλληλου αλγορίθμου, η οποία πραγματοποιήθηκε με χρήση βιβλιοθηκών MPI, λήφθηκαν υπόψη ένα σύνολο από παράγοντες για τη βελτιστοποίηση της απόδοσης, όπως η μείωση της επιβάρυνσης

επικοινωνίας και η σύσταση και κατανομή των εργασιών. Παρουσιάζονται αναλυτικά αποτελέσματα από την εκτέλεση μας σειράς πειραμάτων για διάφορα μεγέθη προβλημάτων, θεωρώντας διαφορετικές περιοχές και διαφορετικές απαιτήσεις στην ακρίβεια των αποτελεσμάτων πρόβλεψης διάδοσης. Τα αποτελέσματα που προκύπτουν από τα πειράματα δείχνουν ότι η απόδοση του αλγορίθμου είναι πολύ κοντά στη θεωρητικά βέλτιστη σε όλες τις περιπτώσεις των διαφορετικών προβλημάτων.

Στο Κεφάλαιο 4 παρουσιάζεται ένα κατανεμημένο σύστημα που υλοποιήθηκε στα πλαίσια της διατριβής για την εκτέλεση του αλγορίθμου βελτιστοποίησης νέφους σωματιδίων, με βάση το υπολογιστικό παράδειγμα των κινητών πρακτόρων. Ο συγκεκριμένος αλγόριθμος στηρίζεται σε διαδοχικούς επαναληπτικούς υπολογισμούς μιας αντικειμενικής συνάρτησης, οι οποίοι είναι ανεξάρτητοι μεταξύ τους και μπορούν να εκτελεστούν παράλληλα. Το σύστημα πολλαπλών πρακτόρων που αναπτύχθηκε επιτυγχάνει τη μείωση του χρόνου εκτέλεσης του αλγορίθμου, επιτρέποντας την εκτέλεσή του σε ένα σύνολο ετερογενών υπολογιστών που συνδέονται μέσω δικτύου, παρέχοντας ευκολία και ευελιξία στην εγκατάσταση, καθώς και δυνατότητα για εύκολη επέκταση. Παρουσιάζονται πειραματικά αποτελέσματα που προέκυψαν από την εφαρμογή του συστήματος στην επίλυση προβλήματος βελτιστοποίησης διαγράμματος ακτινοβολίας στοιχειοκεραίας.

Στο Κεφάλαιο 5, πραγματοποιείται μια επισκόπηση της τεχνολογίας του υπολογιστικού πλέγματος, η οποία έχει καθιερωθεί τα τελευταία χρόνια ως αποτελεσματική λύση για την αντιμετώπιση των ολοένα και αυξανόμενων υπολογιστικών απαιτήσεων των επιστημονικών εφαρμογών. Στο Κεφάλαιο 6 παρουσιάζεται ένα περιβάλλον που αναπτύχθηκε για την προσομοίωση συστημάτων ασύρματων επικοινωνιών σε υπολογιστικό πλέγμα. Τα συστήματα αυτά είναι στοχαστικά και για την ανάλυσή τους χρησιμοποιούνται ευρέως στοχαστικές τεχνικές όπως οι Monte Carlo αλγόριθμοι, οι οποίοι στηρίζονται στην επαναληπτική εκτέλεση ενός μεγάλου αριθμού ανεξάρτητων μεταξύ τους προσομοιώσεων και στην εξαγωγή στατιστικών συμπερασμάτων. Οι αυξημένες απαιτήσεις των συγκεκριμένων προβλημάτων σε υπολογιστικούς πόρους καθιστούν την επίλυσή τους εξαιρετικά χρονοβόρα ή ακόμη και πρακτικά αδύνατη σε ορισμένες περιπτώσεις. Παρόλα αυτά, λόγω του ότι μια μέθοδος Monte Carlo μπορεί να διασπαστεί σε ένα σύνολο εντελώς ανεξάρτητων τμημάτων, η επεξεργασία των αποτελεσμάτων των οποίων πραγματοποιείται μόνο μια φορά μετά την ολοκλήρωση όλων αυτών των τμημάτων, είναι δυνατή η εκτέλεσή της σε υποδομή πλέγματος με αποδοτικό τρόπο. Στα πλαίσια της διατριβής αναπτύχθηκε μια ασφαλής δικτυακή πύλη με βάση τα πρότυπα των portlets ως διεπαφή στο μεσισμικό της υποδομής πλέγματος EGEE, η οποία αποκρύπτει από τον τελικό χρήστη την

πολυπλοκότητα χρήσης του πλεγματικού μεσισμικού. Το περιβάλλον προσομοίωσης χρησιμοποιήθηκε για την εκτέλεση μιας σειράς προσομοιώσεων συστημάτων ασύρματων επικοινωνιών, ενώ παρουσιάζονται στατιστικά από την εκτέλεση στο πλέγμα ορισμένων σεναρίων για συστήματα WCDMA.

Τέλος, στο Κεφάλαιο 7 παρατίθενται τα συμπεράσματα που προέκυψαν από τη διατριβή, καθώς και προτάσεις για μελλοντική έρευνα.

2 Τεχνολογίες Παράλληλης και Κατανεμημένης Επεξεργασίας

Οι εξελίξεις που έχουν παρατηρηθεί τα τελευταία χρόνια στις επιδόσεις και τις δυνατότητες των υπολογιστικών συστημάτων είναι πολύ μεγάλες. Οι σύγχρονες αρχιτεκτονικές υπολογιστών παρέχουν αυξημένες επιδόσεις κυρίως μέσω της αύξησης της πυκνότητας των κυκλωμάτων και των αποδοτικότερων packaging τεχνικών. Στις μέρες μας έχουν κατασκευαστεί υπερ-υπολογιστές ενός επεξεργαστή που επιτυγχάνουν πολύ μεγάλες ταχύτητες, χρησιμοποιώντας τεχνολογίες υλικού οι οποίες πλέον πλησιάζουν σε μεγάλο βαθμό τα φυσικά όρια της κατασκευής ολοκληρωμένων κυκλωμάτων. Για αυτό το λόγο θέτονται κάποιοι περιορισμοί στις δυνατότητες εξέλιξης και περαιτέρω βελτίωσης των υπολογιστικών δυνατοτήτων που μπορούν να επιτευχθούν με συστήματα ενός επεξεργαστή.

Παράλληλα, ο ρυθμός αύξησης των υπολογιστικών απαιτήσεων των σύγχρονων εφαρμογών είναι επίσης ραγδαίος, πράγμα που καθιστά τα τυπικά σύγχρονα υπολογιστικά συστήματα αδύνατα να επιτύχουν την εκτέλεση ορισμένων εφαρμογών σε ικανοποιητικά μεγάλες κλίμακες και σε πρακτικά ανεκτούς χρόνους. Μια λύση σε αυτό το πρόβλημα είναι η παράλληλη επεξεργασία: η μέθοδος διάσπασης μεγάλων υπολογιστικών προβλημάτων σε μικρότερες εργασίες που μπορούν να λυθούν παράλληλα. Η παράλληλη επεξεργασία έχει αναδειχθεί σε σημαντικό τομέα της επιστήμης των υπολογιστών και τα τελευταία χρόνια έχει τύχει σημαντικής και συνεχώς αυξανόμενης αποδοχής, τόσο για εξειδικευμένες επιστημονικές εφαρμογές, όσο και για εφαρμογές γενικότερου ενδιαφέροντος, λόγω του σχετικά χαμηλού κόστους σε σχέση με την προσέγγιση χρήσης υπερ-υπολογιστών και των δυνατοτήτων που παρέχει για αυξημένη παραγωγικότητα.

Τα παράλληλα υπολογιστικά συστήματα είναι συστήματα που αποτελούνται από πολλαπλές μονάδες επεξεργασίας που συνδέονται μεταξύ τους μέσω κάποιου δικτύου

διασύνδεσης και από κάποιο ειδικό λογισμικό που απαιτείται για την ενορχήστρωση της λειτουργίας των διαφορετικών επεξεργαστών προς την επίλυση ενός κοινού προβλήματος. Οι βασικοί παράγοντες που χρησιμοποιούνται για την κατηγοριοποίηση τέτοιων συστημάτων είναι (i) οι ίδιες οι μονάδες επεξεργασίες και (ii) το μέσο διασύνδεσης που τις συνδέει. Οι μονάδες επεξεργασίας έχουν τη δυνατότητα να επικοινωνούν και να αλληλεπιδρούν είτε με χρήση διαμοιραζόμενης μνήμης, είτε με χρήση μεθόδων ανταλλαγής μηνυμάτων.

Ένας σημαντικός παράγοντας που έχει οδηγήσει σε αυτού του μεγέθους την αποδοχή της παράλληλης επεξεργασίας είναι η βελτίωση της ταχύτητας και γενικά των δυνατοτήτων των σύγχρονων δικτύων υπολογιστών. Η μείωση του χρόνου που απαιτείται για την επικοινωνία μεταξύ δύο υπολογιστών έχει καταστήσει αποδοτική τη χρήση κατανεμημένων συστημάτων επεξεργασίας για την εκτέλεση παράλληλων εφαρμογών. Ένα κατανεμημένο σύστημα υπολογισμού είναι ένα σύνολο από υπολογιστικές οντότητες που συνδέονται μέσω δικτύου και οι οποίες επικοινωνούν μεταξύ τους μέσω μηνυμάτων για την επίτευξη κάποιου κοινού στόχου (π.χ. επίλυση ενός προβλήματος). Ένα τέτοιο σύστημα μπορεί να αποτελείται από συνηθισμένους υπολογιστές και είναι μια αποδοτική και σχετικά φτηνή λύση για την επίλυση υπολογιστικά απαιτητικών προβλημάτων. Επιπλέον, με την ανάπτυξη και καθολική επικράτηση του διαδικτύου και με την ευρεία χρήση προγραμματιστικών εργαλείων ανεξάρτητων από την υποκείμενη υπολογιστική αρχιτεκτονική παρέχεται η δυνατότητα ανάπτυξης εφαρμογών, οι οποίες δεν απαιτείται να βασίζονται σε εξειδικευμένα συστήματα παράλληλης επεξεργασίας και μπορούν να εκτελέσουν κατανεμημένα οικογένειες παρεμφερών προβλημάτων πάνω από τα υποστρώματα δικτύου TCP/IP, κάνοντας χρήση της υπάρχουσας υποδομής και ευρέως χρησιμοποιούμενων βιβλιοθηκών δικτυακού λογισμικού.

Ο στόχος και ταυτόχρονα το βασικό πλεονέκτημα της παράλληλης επεξεργασίας είναι ο υποπολλαπλασιασμός του χρόνου που απαιτείται για την εκτέλεση των προγραμμάτων. Παρόλα αυτά, για την επίτευξη του στόχου αυτού υπεισέρχονται ένα σύνολο από παράμετροι που αυξάνουν σημαντικά την πολυπλοκότητα του συστήματος και οι οποίοι μπορούν να επιφέρουν σημαντικές χρονικές καθυστερήσεις στην εκτέλεση, ανάλογα με το είδος του προβλήματος και τη σχεδίαση του αλγόριθμου παραλληλοποίησης. Οι σημαντικότεροι παράγοντες που μπορεί να επηρεάσουν την απόδοση μιας παράλληλης εφαρμογής είναι ο συγχρονισμός μεταξύ των παράλληλων οντοτήτων και οι διαδικασίες επικοινωνίας και ανταλλαγής μηνυμάτων μεταξύ τους.

Στο παρόν κεφάλαιο γίνεται μια εκτενής μελέτη των σημαντικότερων τεχνολογιών παράλληλης και κατανεμημένης επεξεργασίας, ενώ περιγράφονται και κάποια από τα σημαντικότερα θέματα που απαιτείται να λαμβάνονται υπόψη κατά τη σχεδίαση και υλοποίηση παράλληλων εφαρμογών.

2.1 Αρχιτεκτονικές Παράλληλων Συστημάτων

Κατά καιρούς έχουν προταθεί διάφορες ταξινομήσεις για τις αρχιτεκτονικές των παράλληλων και κατανεμημένων συστημάτων, ανάλογα με τον τρόπο λειτουργίας τους. Στις επόμενες παραγράφους επιχειρείται μια αναφορά σε αυτές τις κατηγοριοποιήσεις με βάση διαφορετικά κριτήρια.

2.1.1 Κατηγοριοποίηση κατά Flynn

Η πιο δημοφιλής κατηγοριοποίηση των παράλληλων αρχιτεκτονικών προτάθηκε από τον M. Flynn ([2], [3], [4]). Η κατηγοριοποίηση αυτή βασίζεται στην έννοια της ροής πληροφορίας. Δύο είναι οι τύποι πληροφορίας που εισρέουν σε έναν επεξεργαστή: εντολές και δεδομένα. Η ροή εντολών (instruction stream) ορίζεται ως η ακολουθία εντολών που εκτελείται από τη μονάδα επεξεργασίας, ενώ η ροή δεδομένων (data stream) ορίζεται ως η κυκλοφορία δεδομένων που ανταλλάσσεται μεταξύ της μνήμης και της μονάδας επεξεργασίας. Με βάση την κατηγοριοποίηση του Flynn, κάθε μία από τις ροές εντολών ή δεδομένων μπορεί να είναι μοναδική ή πολλαπλή. Με βάση αυτό το διαχωρισμό, η αρχιτεκτονική ενός υπολογιστικού συστήματος μπορεί να ταξινομηθεί στις εξής τέσσερις κατηγορίες (Σχήμα 1):

- single - instruction, single - data streams (SISD)
- single - instruction, multiple - data streams (SIMD)
- multiple - instruction, single - data streams (MISD)
- multiple - instruction, multiple - data streams (MIMD)

Οι συμβατικοί υπολογιστές ενός επεξεργαστή ανήκουν στα συστήματα SISD. Οι παράλληλοι υπολογιστές ανήκουν είτε στην κατηγορία SIMD, είτε στην κατηγορία MIMD. Στην περίπτωση που υπάρχει μία μοναδική μονάδα ελέγχου και όλοι οι επεξεργαστές εκτελούν την ίδια εντολή συγχρονισμένα, το παράλληλο σύστημα ανήκει στην κατηγορία SIMD. Στην περίπτωση ενός MIMD συστήματος, κάθε επεξεργαστής διαθέτει τη δική του μονάδα ελέγχου και έχει τη δυνατότητα να εκτελεί διαφορετικές εντολές σε διαφορετικά δεδομένα. Στην MISD κατηγορία, η ίδια ροή δεδομένων εισέρχεται σε μια συστοιχία

επεξεργαστών που εκτελούν διαφορετικές ροές εντολών. Τα συστήματα που ανήκουν στην κατηγορία MISD συναντούνται στην πράξη πολύ περιορισμένα (π.χ. συστήματα pipeline και συστήματα που εφαρμόζουν task replication) και για αυτό το λόγο δεν θα γίνει περαιτέρω επέκταση σε αυτά.

	Single Data	Multiple Data
Single Instruction	SISD	**SIMD**
Multiple Instruction	MISD	**MIMD**

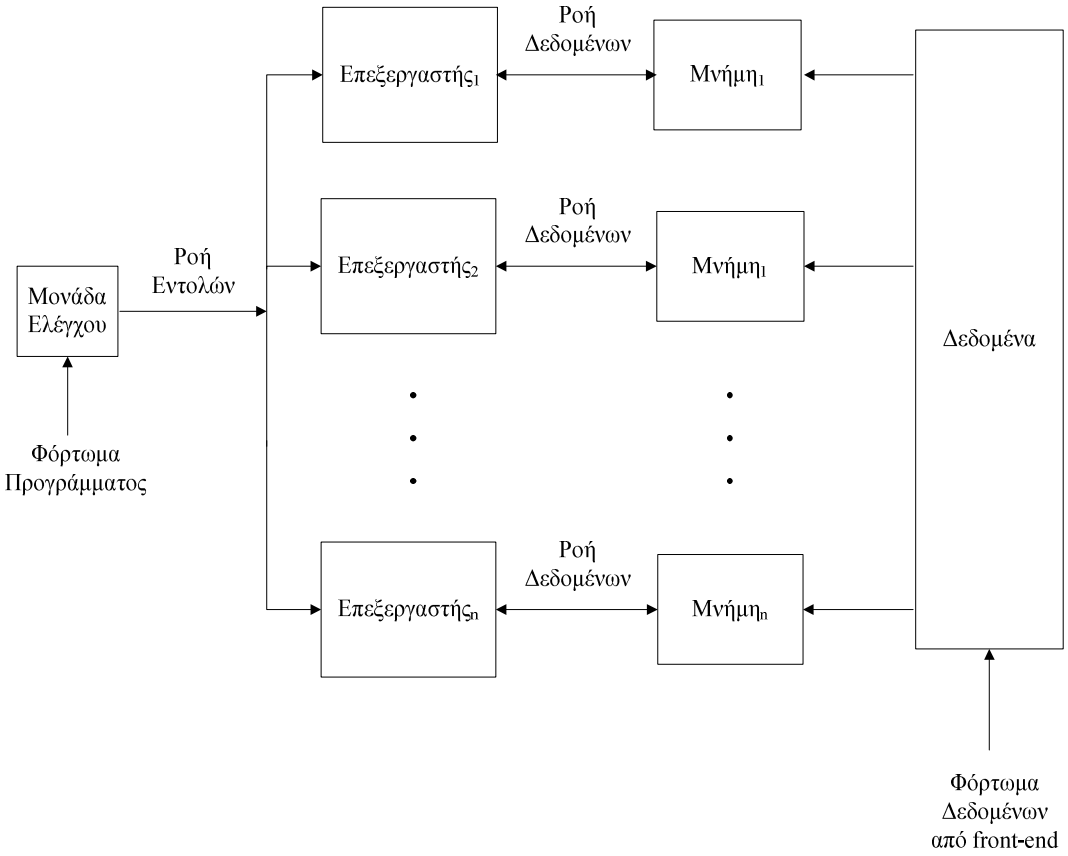
Σχήμα 1. Κατηγοριοποίηση κατά Flynn

2.1.1.1 Αρχιτεκτονική SIMD

Το μοντέλο της SIMD αρχιτεκτονικής αποτελείται από ένα τυπικό front-end υπολογιστή και μια συστοιχία επεξεργαστών, όπως φαίνεται στο Σχήμα 2.

Η συστοιχία επεξεργαστών είναι ένα σύνολο πανομοιότυπων συγχρονισμένων υπολογιστικών στοιχείων που έχουν τη δυνατότητα να εκτελούν ταυτόχρονα τις ίδιες λειτουργίες πάνω σε διαφορετικά δεδομένα. Κάθε επεξεργαστής στη συστοιχία έχει μία μικρή τοπική μνήμη όπου βρίσκονται τα δεδομένα καθώς πραγματοποιείται η παράλληλη επεξεργασία τους. Η συστοιχία επεξεργαστών συνδέεται με το δίαυλο μνήμης του front-end υπολογιστή, έτσι ώστε αυτός να έχει πρόσβαση στα δεδομένα που επεξεργάζονται οι επεξεργαστές. Με αυτόν τον τρόπο, ο front-end υπολογιστής μπορεί να εκτελεί ειδικές εντολές που έχουν ως αποτέλεσμα την ταυτόχρονη επεξεργασία τμημάτων της μνήμης ή τη μεταφορά δεδομένων στη μνήμη. Ένα πρόγραμμα μπορεί να αναπτυχθεί και να εκτελεστεί στον front-end υπολογιστή με τον συνήθη τρόπο, ενώ στο background αποστέλλονται εντολές στη συστοιχία επεξεργαστών για την εκτέλεση λειτουργιών SIMD παράλληλα. Ένα από τα σημαντικότερα πλεονεκτήματα της συγκεκριμένης αρχιτεκτονικής είναι η ομοιότητα μεταξύ του σειριακού προγραμματισμού και του προγραμματισμού που στηρίζεται στην παραλληλία δεδομένων. Συγχρονισμός των επεξεργαστών δεν απαιτείται, καθώς οι

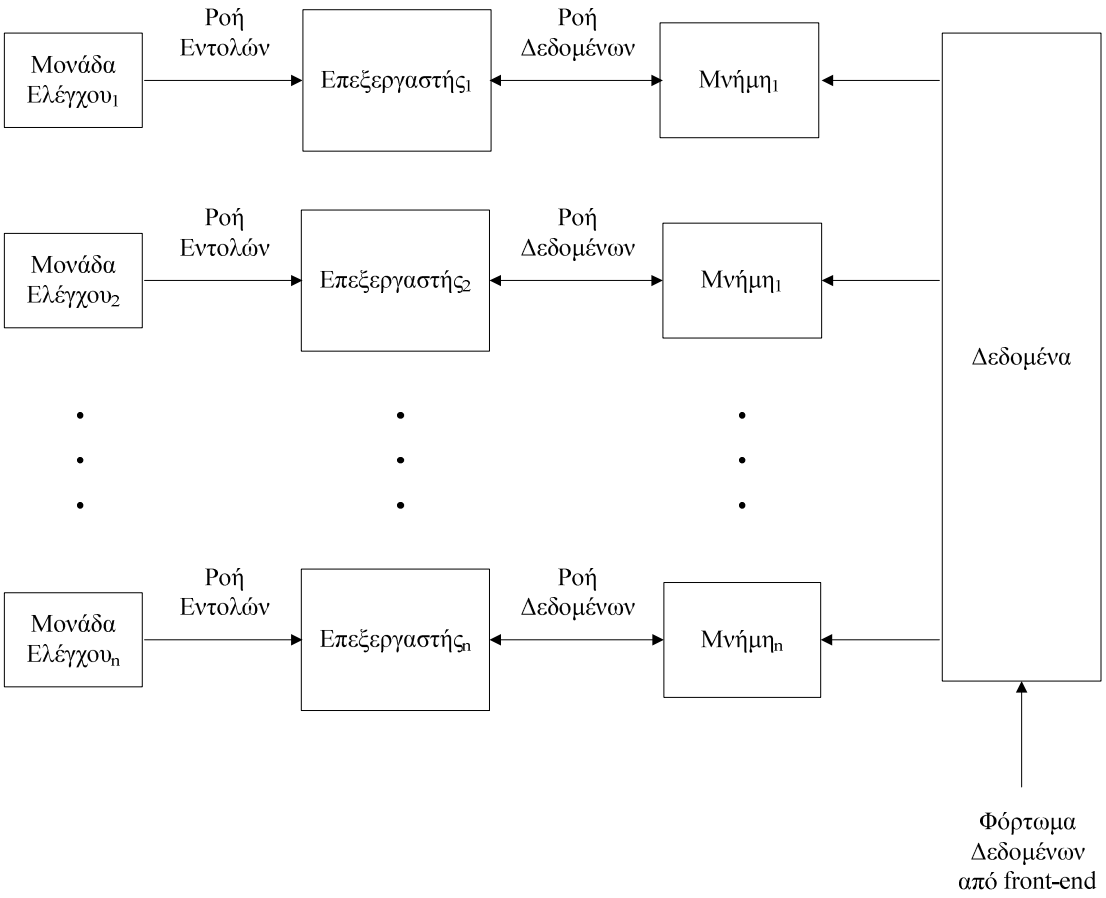
επεξεργαστές είτε είναι ανενεργοί, είτε εκτελούν ακριβώς τις ίδιες εντολές την ίδια χρονική στιγμή (λειτουργία lock-step συγχρονισμού).



Σχήμα 2. Αρχιτεκτονική SIMD

2.1.1.2 Αρχιτεκτονική MIMD

Οι παράλληλες αρχιτεκτονικές MIMD αποτελούνται από πολλαπλούς επεξεργαστές και πολλαπλές οντότητες μνήμης, η σύνδεση μεταξύ των οποίων επιτυγχάνεται με τη βοήθεια κάποιου μέσου διασύνδεσης. Η γενική μορφή της MIMD αρχιτεκτονικής φαίνεται στο Σχήμα 3. Τα συστήματα MIMD χωρίζονται σε δύο μεγάλες κατηγορίες: τα συστήματα διαμοιραζόμενης μνήμης (shared memory) και τα συστήματα προώθησης μηνυμάτων (message passing).



Σχήμα 3. Αρχιτεκτονική MIMD

Στην περίπτωση των συστημάτων διαμοιραζόμενης μνήμης οι επεξεργαστές ανταλλάσσουν πληροφορία για την επίτευξη του απαραίτητου συντονισμού μέσω μιας καθολικής, κοινής μνήμης, στην οποία έχουν πρόσβαση όλοι οι επεξεργαστές, ενώ στην περίπτωση των συστημάτων προώθησης μηνυμάτων η ανταλλαγή πληροφορίας πραγματοποιείται μέσω ενός δικτύου διασύνδεσης. Τα δύο αυτά διαφορετικά μοντέλα επικοινωνίας παρουσιάζονται αναλυτικότερα στην επόμενη παράγραφο.

2.1.1.3 Κατηγορίες Υψηλότερου Επιπέδου

Σε αυτό το σημείο κρίνεται σκόπιμο να σημειωθούν δύο ακόμη υποκατηγορίες που δεν ανήκουν στην αρχική κατηγοριοποίηση κατά Flynn και μπορούν να προέρχονται από οποιαδήποτε από τις αρχιτεκτονικές που περιγράφτηκαν προηγουμένως, οι οποίες παρέχουν

μια υψηλότερου επιπέδου θεώρηση των παράλληλων συστημάτων. Οι κατηγορίες αυτές είναι οι ακόλουθες:

- single-process, multiple data (SPMD): το ίδιο πρόγραμμα εκτελείται ταυτόχρονα από όλους του επεξεργαστές, αλλά οποιαδήποτε χρονική στιγμή οι επεξεργαστές μπορεί να εκτελούν διαφορετικές εντολές μέσα στο πρόγραμμα. Τα SPMD προγράμματα συνήθως έχουν ενσωματωμένη κάποια λογική, έτσι ώστε να διακλαδίζονται κατάλληλα και υπό συνθήκη να εκτελούν μόνο συγκεκριμένα κομμάτια του προγράμματος. Οι διαφορετικοί επεξεργαστές μπορούν να λειτουργούν πάνω σε διαφορετικά δεδομένα.
- multiple-process, multiple data (MPMD): οι MPMD εφαρμογές αποτελούνται τυπικά από πολλαπλά εκτελέσιμα προγράμματα. Σε μια τέτοια εφαρμογή, οι επεξεργαστές εκτελούν ταυτόχρονα δύο ή περισσότερα διαφορετικά και ανεξάρτητα μεταξύ τους προγράμματα, τα οποία μπορούν να λειτουργούν πάνω σε διαφορετικά σύνολα δεδομένων.

2.1.2 Κατηγοριοποίηση με βάση την Αρχιτεκτονική Επικοινωνίας

Όπως αναφέρθηκε και προηγουμένως, η επικοινωνία μεταξύ των διαφορετικών μονάδων επεξεργασίας που απαρτίζουν ένα σύστημα παράλληλης επεξεργασίας μπορεί να στηρίζεται σε δύο διαφορετικές προσεγγίσεις [5]:

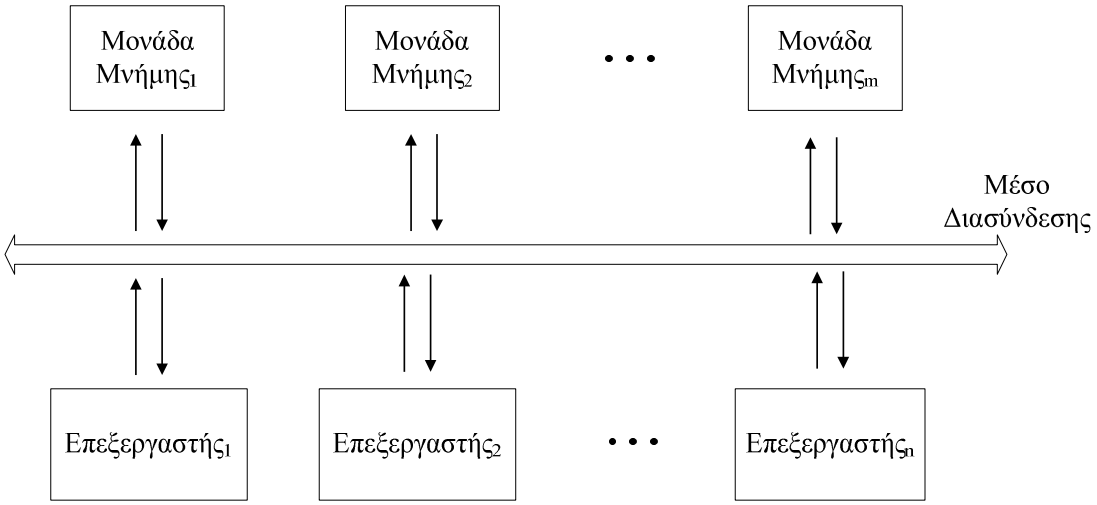
- επικοινωνία μέσω διαμοιραζόμενης μνήμης: σε αυτήν την περίπτωση υπάρχει ένας κοινός, διαμοιραζόμενος χώρος μνήμης στον οποίο έχουν πρόσβαση όλες οι μονάδες επεξεργασίας
- επικοινωνία μέσω δικτύου διασύνδεσης: σε αυτή την περίπτωση η συνολική μνήμη του συστήματος αποτελείται από επιμέρους αυτόνομα τμήματα, τα οποία συνδέονται με κόμβους ενός ή περισσότερων μονάδων επεξεργασίας, ενώ ο συντονισμός μεταξύ των επεξεργαστών επιτυγχάνεται με επικοινωνία μέσω κάποιου δικτύου διασύνδεσης

Στη συνέχεια περιγράφονται αναλυτικότερα οι δύο παραπάνω κατηγορίες:

2.1.2.1 Συστήματα Διαμοιραζόμενης Μνήμης

Η γενική οργάνωση ενός συστήματος διαμοιραζόμενης μνήμης φαίνεται στο Σχήμα 4.

Όπως φαίνεται στο σχήμα αυτό, ένα σύστημα διαμοιραζόμενης μνήμης αποτελείται από ένα σύνολο ανεξάρτητων επεξεργαστών, ένα σύνολο μονάδων μνήμης και κάποιο μέσο διασύνδεσης.



Σχήμα 4. Σύστημα Διαμοιραζόμενης Μνήμης

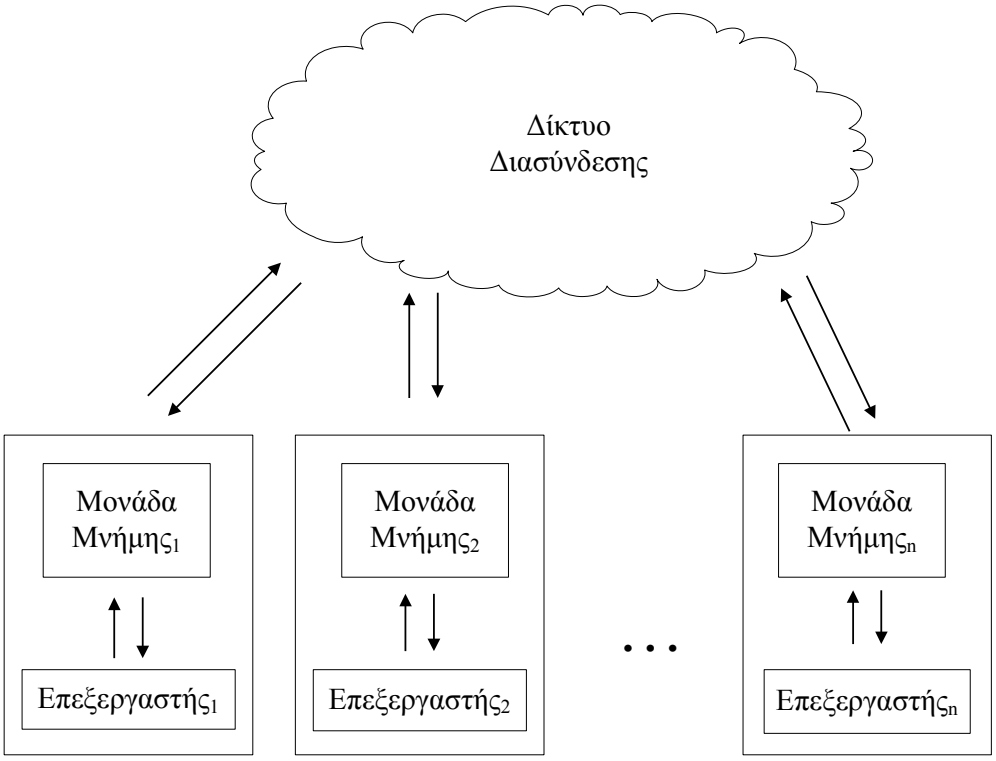
Σε αυτά τα συστήματα, η επικοινωνία μεταξύ των εργασιών που εκτελούνται σε διαφορετικούς επεξεργαστές πραγματοποιείται με γράψιμο και διάβασμα στην καθολική μνήμη. Επίσης, ο συντονισμός και ο συγχρονισμός μεταξύ των μονάδων επεξεργασίας επιτυγχάνεται μέσω της καθολικής μνήμης. Τα συστήματα διαμοιραζόμενης μνήμης διαχωρίζονται σε επιπλέον υποκατηγορίες, η πιο σημαντική από τις οποίες είναι τα συστήματα Ομοιόμορφης Πρόσβασης στη Μνήμη (Uniform Memory Access – UMA). Στα συστήματα UMA, όλοι οι επεξεργαστές είναι όμοιοι, ενώ η διαμοιραζόμενη μνήμη είναι προσβάσιμη από όλους τους επεξεργαστές μέσω του μέσου διασύνδεσης με τον ίδιο τρόπο που ένας επεξεργαστής έχει πρόσβαση στην ιδιωτική μνήμη του. Επιπλέον, όλοι οι επεξεργαστές έχουν ίσους χρόνους πρόσβασης σε κάθε περιοχή μνήμης. Λόγω της ισορροπημένης πρόσβασης στη μνήμη, τα συγκεκριμένα συστήματα ονομάζονται και συμμετρικοί πολύ-επεξεργαστές (Symmetric Multi-Processor – SMP).

Τα πλεονεκτήματα των συστημάτων διαμοιραζόμενης μνήμης είναι ότι η ύπαρξη του καθολικού χώρου μνήμης επιτρέπει την σχετικά εύκολη υλοποίηση παράλληλων εφαρμογών, ενώ λόγω του ότι οι επεξεργαστής βρίσκονται κοντά στη μνήμη, η πρόσβαση σε αυτήν είναι πολύ γρήγορη. Τα μειονεκτήματα αυτών των συστημάτων είναι (i) ότι παρουσιάζουν περιορισμούς στην ικανότητα κλιμάκωσης καθώς αυξάνεται ο αριθμός των

επεξεργαστών, λόγω αυξανόμενης κίνησης στο μέσο διασύνδεσης και (ii) ότι ο προγραμματιστής αναλαμβάνει την ευθύνη διαχείρισης του συγχρονισμού πρόσβασης στην κοινή μνήμη.

2.1.2.2 Συστήματα Επικοινωνίας μέσω Δικτύου Διασύνδεσης

Οι περιορισμοί στην ικανότητα κλιμάκωσης των συστημάτων διαμοιραζόμενης μνήμης οδήγησαν στην ανάπτυξη των αρχιτεκτονικών συστημάτων κατανεμημένης μνήμης, στα οποία η επικοινωνία μεταξύ των επεξεργαστών πραγματοποιείται μέσω κάποιου δικτύου διασύνδεσης. Η γενική αρχιτεκτονική αυτών των συστημάτων φαίνεται στο Σχήμα 5.



Σχήμα 5. Σύστημα Επικοινωνίας μέσω Δικτύου Διασύνδεσης

Όπως φαίνεται και στο παραπάνω σχήμα, η καθολική διαμοιραζόμενη μνήμη έχει αντικατασταθεί από μικρότερες τοπικές μονάδες μνήμης, που συνδέονται απευθείας με τους επεξεργαστές. Τα περισσότερα συστήματα υποστηρίζουν το μοντέλο επικοινωνίας που βασίζεται στην ανταλλαγή μηνυμάτων, σύμφωνα με το οποίο όταν ένας επεξεργαστής χρειάζεται δεδομένα που βρίσκονται στη μνήμη απομακρυσμένου κόμβου, θα πρέπει να τα λάβει σε μορφή μηνύματος και στη συνέχεια να τα επεξεργαστεί κατάλληλα.

Τα πλεονεκτήματα των συστημάτων αυτών είναι (i) παρουσιάζουν μεγάλες δυνατότητες κλιμάκωσης, (ii) κάθε επεξεργαστής μπορεί να έχει πρόσβαση στα τοπικά δεδομένα με μεγάλη ταχύτητα και (iii) μπορούν να κατασκευαστούν από κοινά συστατικά (π.χ. κοινούς desktop υπολογιστές συνδεδεμένους με ένα κοινό δίκτυο), μειώνοντας το κόστος. Το σημαντικότερο μειονέκτημά τους είναι ότι ο προγραμματιστής είναι υπεύθυνος για την υλοποίηση της επικοινωνίας των επεξεργαστών, λαμβάνοντας υπόψη τα χαρακτηριστικά του δικτύου διασύνδεσης (π.χ. latency) κατά τη βελτιστοποίηση της απόδοσης της παράλληλης εφαρμογής.

2.2 Μοντέλα Παράλληλου Προγραμματισμού

Τα μοντέλα παράλληλου προγραμματισμού λειτουργούν ως ένα αφαιρετικό επίπεδο πάνω από τις τεχνολογίες και τις αρχιτεκτονικές υλικού που παρουσιάστηκαν στην προηγούμενη παράγραφο. Τα μοντέλα αυτά σε γενικές γραμμές δεν είναι σχεδιασμένα μόνο για ένα συγκεκριμένο τύπο αρχιτεκτονικής, αλλά θεωρητικά μπορούν να υλοποιηθούν σε όλες τις κατηγορίες του υποκείμενου υλικού. Σε αυτήν την παράγραφο γίνεται μια αναφορά στις σημαντικότερες τεχνολογίες προγραμματισμού που υπάρχουν διαθέσιμες για την ανάπτυξη παράλληλων προγραμμάτων, καθώς και μια σχετικά σύντομη ανάλυση των σημαντικότερων στοιχείων τους.

2.2.1 Πολυνηματικός Προγραμματισμός

Σε ένα παράλληλο σύστημα διαμοιραζόμενης μνήμης, πολλαπλά νήματα ελέγχου λειτουργούν σε μια μοναδική περιοχή μνήμης. Η ανταλλαγή πληροφορίας μεταξύ των διαφορετικών νημάτων γίνεται με απλή απευθείας εγγραφή και ανάγνωση σε μεταβλητές της μνήμης, χρησιμοποιώντας συνήθεις λειτουργίες ανάθεσης. Αν και ο μηχανισμός επικοινωνίας είναι απλός, ο πολυνηματικός προγραμματισμός εισάγει το πρόβλημα του ελέγχου πρόσβασης σε μια περιοχή διαμοιραζόμενης μνήμης τη στιγμή που ανανεώνεται η πληροφορία που αυτή περιέχει. Χωρίς την παρέμβαση κάποιας ρητής λειτουργίας συγχρονισμού, η αλληλεπικαλυπτόμενη εκτέλεση πολλαπλών νημάτων ενδέχεται να οδηγήσει σε εσφαλμένα αποτελέσματα. Για παράδειγμα, δομές δεδομένων που βρίσκονται υπό τροποποίηση δεν πρέπει να χρησιμοποιούνται πριν ολοκληρωθεί η τροποποίηση και σε περίπτωση που κάποιο νήμα επιθυμεί να διαβάσει τη δομή, αυτό θα πρέπει να καθυστερήσει μέχρι το νήμα που γράφει στη δομή να έχει ολοκληρώσει την ενημέρωση

όλων των πεδίων αυτής. Παρόμοια, η ταυτόχρονη εγγραφή μιας δομής δεδομένων από δύο διαφορετικά νήματα μπορεί να οδηγήσει σε σφάλμα, με τα τελικά αποτελέσματα να είναι ένας τυχαίος συνδυασμός των πεδίων που έγραψαν τα δύο συναγωνιζόμενα νήματα. Στη συνέχεια περιγράφονται δύο από τις σημαντικότερες μεθόδους παράλληλου πολυνηματικού προγραμματισμού: τα POSIX threads [6] και το OpenMP [7], [8].

2.2.1.1 POSIX threads

Οι προδιαγραφές νημάτων POSIX (POSIX threads – p-threads) παρέχουν μια ιδιαίτερα χαμηλού επιπέδου και ευρέως διαδεδομένη υλοποίηση του πολυνηματικού μοντέλου προγραμματισμού. Η βιβλιοθήκη p-threads παρέχει συναρτήσεις για τη δημιουργία και τον τερματισμό νημάτων και για το συντονισμό των δραστηριοτήτων τους, μέσω μηχανισμών που παρέχουν αποκλειστική μέθοδο σε συγκεκριμένες περιοχές μνήμης. Τα χαρακτηριστικά που περιέχει η βιβλιοθήκη είναι γενικότερου ενδιαφέροντος και βρίσκουν εφαρμογή και σε εφαρμογές εκτός του παράλληλου προγραμματισμού. Στη συνέχεια παρουσιάζονται συνοπτικά τα σημαντικότερα χαρακτηριστικά των p-threads που σχετίζονται με την ανάπτυξη παράλληλων προγραμμάτων.

Ένα πρόγραμμα p-thread ξεκινάει την εκτέλεσή του ως μοναδικό νήμα ελέγχου, όπως κάθε σειριακό πρόγραμμα. Νέα νήματα ελέγχου μπορούν να δημιουργηθούν ρητά με κλήση της συνάρτησης *pthread_create()* και καθορίζοντας τη συνάρτηση που θα εκτελέσει το νέο νήμα. Περιορισμός στον αριθμό των νημάτων που μπορεί να δημιουργήσει ένα πρόγραμμα δεν υπάρχει. Ο αριθμός αυτός μάλιστα μπορεί να είναι μεγαλύτερος από τον αριθμό των επεξεργαστών στο σύστημα. Ένα νήμα τερματίζει την εκτέλεσή του είτε όταν τερματίσει η συνάρτηση που εκτελείται από αυτό, είτε με ρητή κλήση της συνάρτησης τερματισμού *pthread_kill()*.

Στο μοντέλο POSIX, ολόκληρη η μνήμη που σχετίζεται με ένα νήμα είναι διαμοιραζόμενη, συμπεριλαμβανομένων της στοίβας εκτέλεσης του νήματος, της δυναμικά δεσμευμένης μνήμης σωρού και των καθολικών μεταβλητών, πράγμα που θέτει συγκεκριμένες απαιτήσεις συγχρονισμού. Η βιβλιοθήκη p-threads παρέχει δύο τύπους συγχρονισμού: συγχρονισμό που σχετίζεται με τον έλεγχο εκτέλεσης, και συγχρονισμό πρόσβασης σε δομές δεδομένων. Η πρώτη μορφή συγχρονισμού καθιστά εφικτή την παύση εκτέλεσης (blocking) κάποιου νήματος, αναμένοντας την ολοκλήρωση κάποιου άλλου νήματος. Οι μηχανισμοί συγχρονισμού δεδομένων στηρίζονται στην έννοια μιας δομής *mutex* (mutual exclusion) αμοιβαίου αποκλεισμού, την οποία πρέπει να έχει "κλειδώσει" (lock) ένα νήμα

για να προχωρήσει την εκτέλεσή του και η οποία απελευθερώνεται από το νήμα όταν αυτό ολοκληρώσει την εκτέλεση του τμήματος κώδικα που επεξεργάζεται διαμοιραζόμενα δεδομένα.

Εκτός από τους παραπάνω locking μηχανισμούς, παρέχονται λειτουργίες συγχρονισμού που στηρίζονται σε μεταβλητές κατάστασης (*condition variables*), οι οποίες επιτρέπουν σε ένα νήμα να περιμένει μέχρι μία ή περισσότερες διαμοιραζόμενες θέσεις μνήμης γίνουν αληθείς. Κάθε μεταβλητή κατάστασης σχετίζεται με ένα mutex. Προτού το νήμα ελέγξει την κατάσταση της μεταβλητής, πρέπει να κλειδώσει το σχετικό mutex και στη συνέχεια να κάνει τον έλεγχο. Σε περίπτωση που η μεταβλητή είναι ψευδής, το νήμα μπορεί να διακόψει την εκτέλεσή του, ενημερώνοντας το δρομολογητή ότι αναμένει τη δεδομένη μεταβλητή κατάστασης και απελευθερώνοντας το mutex (κλήση της εντολής *pthread_cond_wait()*). Όταν ένα νήμα αλλάξει την τιμή μια διαμοιραζόμενης μεταβλητής που σχετίζεται με την μεταβλητή κατάστασης, ενημερώνει όσα νήματα αναμένουν (καλώντας την εντολή *pthread_cond_signal()*).

Οι παραπάνω μηχανισμοί μπορούν να συνδυαστούν για την υλοποίηση διάφορων δομών συγχρονισμού.

2.2.1.2 OpenMP

Λόγω του γενικότερου προσανατολισμού του μοντέλου των p-threads, η βιβλιοθήκη POSIX παρουσιάζει ορισμένους συμβιβασμούς που επηρεάζουν τόσο την ευκολία προγραμματισμού, όσο και την απόδοση των προγραμμάτων. Μία εναλλακτική διεπαφή πολυνηματικού προγραμματισμού που είναι ειδικά σχεδιασμένη για την υποστήριξη παράλληλων προγραμμάτων υψηλής απόδοσης είναι το OpenMP. Το OpenMP είναι υλοποιημένο ως συνδυασμός κλήσεων σε βιβλιοθήκες και οδηγιών προς τον μεταγλωττιστή. Με βάση τις οδηγίες αυτές ο μεταγλωττιστής δημιουργεί νήματα, εκτελεί λειτουργίες συγχρονισμού και διαχειρίζεται την διαμοιραζόμενη μνήμη. Το OpenMP δεν ορίζει μια νέα γλώσσα προγραμματισμού, αλλά αποτελείται από ένα σύνολο οδηγιών που ερμηνεύονται από έναν ειδικό OpenMP-enabled μεταγλωττιστή.

Λόγω του ότι το OpenMP σχεδιάστηκε ειδικά για παράλληλες εφαρμογές, η χρήση νημάτων είναι εξαιρετικά δομημένη, ακολουθώντας το μοντέλο fork/join: σε κάθε φάση εκτέλεσης, ένα μοναδικό νήμα ελέγχου διαχωρίζεται σε έναν αριθμό ανεξάρτητων νημάτων και το τέλος της φάσης έρχεται όταν όλα τα νήματα έχουν ολοκληρώσει την εκτέλεση των εργασιών που τους έχουν ανατεθεί στη συγκεκριμένη φάση. Στο OpenMP, μια φάση

εκτέλεσης fork/join αντιστοιχεί σε μια παράλληλη περιοχή, η οποία οριοθετείται από τις οδηγίες PARALLEL και END PARALLEL. Ο αριθμός των νημάτων σε μια περιοχή καθορίζεται από το χρήστη είτε καθολικά, είτε για τη συγκεκριμένη περιοχή και κάθε νήμα εκτελεί τον κώδικα που περικλείεται μεταξύ των παραπάνω οδηγιών.

Οι μηχανισμοί συγχρονισμού του OpenMP περιλαμβάνουν τα ακόλουθα:

- Κρίσιμα τμήματα (critical sections), τα οποία διασφαλίζουν ότι μόνο ένα νήμα εκτελείται κάθε φορά στον κώδικα που αυτά περικλείουν (κατά αναλογία με την έννοια του mutex στα p-threads)
- Ατομικές ενημερώσεις (atomic updates), οι οποίες συμπεριφέρονται όπως τα κρίσιμα τμήματα, αλλά παρέχουν την επιπλέον δυνατότητα να βελτιστοποιούνται σε κάποιες πλατφόρμες υλικού
- Φράγματα (barriers), τα οποία συγχρονίζουν όλα τα νήματα σε μια παράλληλη περιοχή
- Επιλογή αφέντη (master selection), η οποία διασφαλίζει ότι το τμήμα κώδικα που περικλείει εκτελείται μόνο από ένα νήμα, ακόμη και αν αυτό ανήκει σε κάποια παράλληλη περιοχή

Μια ιδιαιτερότητα του OpenMP σε σχέση με τα p-threads είναι στον τρόπο που διαχειρίζεται τη διαμοιραζόμενη μνήμη. Το OpenMP δεν επιτρέπει το διαμοιρασμό μεταβλητών που βρίσκονται στη στοίβα, πράγμα που επιτρέπει τη δημιουργία αποδοτικότερου κώδικα.

2.2.2 Παράλληλη Εικονική Μηχανή (Parallel Virtual Machine – PVM)

Το σύστημα PVM επιτρέπει την ανάπτυξη εφαρμογών σε ένα σύνολο ετερογενών υπολογιστών που συνδέονται μέσω δικτύου και οι οποίοι εμφανίζονται στους χρήστες ως ένας μοναδικός παράλληλος υπολογιστής (“εικονική μηχανή”) [9]. Το PVM παρέχει ένα σύνολο από λειτουργίες ελέγχου εργασιών και δυναμικής διαχείρισης πόρων. Παρέχει στους προγραμματιστές μια βιβλιοθήκη ρουτινών για την εκκίνηση και τον τερματισμό εργασιών, για συγχρονισμό και για την τροποποίηση της διαμόρφωσης της εικονικής μηχανής.

Ένα βασικό πλεονέκτημα του PVM είναι η διαλειτουργικότητα μεταξύ ετερογενών υπολογιστών. Προγράμματα που έχουν αναπτυχθεί για μια δεδομένη αρχιτεκτονική μπορούν να αντιγραφούν σε κάποια άλλη αρχιτεκτονική, να μεταγλωττιστούν και να εκτελεστούν χωρίς τροποποιήσεις. Επιπλέον, αυτά τα εκτελέσιμα προγράμματα μπορούν να

επικοινωνούν μεταξύ τους. Μια εφαρμογή PVM αποτελείται από έναν αριθμό εργασιών που συνεργάζονται έτσι ώστε να παρέχουν λύση σε ένα πρόβλημα. Μια εργασία μπορεί να εναλλάσσει τις φάσεις υπολογισμού/επικοινωνίας με άλλες εργασίες. Το προγραμματιστικό μοντέλο του PVM καταλήγει σε ένα δίκτυο ακολουθιακών εργασιών που επικοινωνούν μεταξύ τους μέσω ανταλλαγής μηνυμάτων, ενώ κάθε εργασία έχει το δικό της έλεγχο εκτέλεσης.

Το υπολογιστικό περιβάλλον του PVM είναι η εικονική μηχανή, ένα δυναμικό σύνολο ετερογενών συστημάτων που συνδέονται μέσω δικτύου και τα οποία παρουσιάζονται ως ενιαίος παράλληλος υπολογιστής. Οι υπολογιστικοί κόμβοι στο δίκτυο μπορεί να είναι συστήματα ενός επεξεργαστή, συστήματα πολλαπλών επεξεργαστών ή ακόμη και συστοιχίες υπολογιστών που εκτελούν το λογισμικό PVM. Τα συστατικά του PVM είναι (i) η βιβλιοθήκη των ρουτινών εκτέλεσης και μια οντότητα-δαίμονας (daemon) που εκτελείται σε όλους τους κόμβους που απαρτίζουν την εικονική μηχανή. Πριν την εκτέλεση μιας εφαρμογής PVM, ο χρήστης πρέπει να εκκινήσει το περιβάλλον εκτέλεσης του PVM και να διαμορφώσει κατάλληλα την εικονική μηχανή. Κατά την εκτέλεση, παρέχεται στο χρήστη μια κονσόλα, μέσω της οποίας μπορεί διαδραστικά να εκκινήσει και να τροποποιήσει την εικονική μηχανή.

Μια εφαρμογή PVM αποτελείται από έναν αριθμό ακολουθιακών προγραμμάτων, κάθε ένα από τα οποία αντιστοιχεί σε μία ή περισσότερες εργασίες σε ένα παράλληλο πρόγραμμα. Αυτά τα προγράμματα μεταγλωττίζονται ξεχωριστά για κάθε κόμβο στην εικονική μηχανή. Τα εκτελέσιμα αρχεία τοποθετούνται σε τοποθεσίες που είναι προσβάσιμες από τους άλλους κόμβους. Ένα από αυτά τα ακολουθιακά προγράμματα, το οποίο ονομάζεται εργασία εκκίνησης, πρέπει να ξεκινήσει χειροκίνητα από το χρήστη σε έναν κόμβο. Οι υπόλοιπες εργασίες στους άλλους κόμβους δημιουργούνται αυτόματα από την εργασία εκκίνησης.

Τυπικά, οι εργασίες που αποτελούν μια εφαρμογή PVM είναι πανομοιότυπες, αλλά επεξεργάζονται διαφορετικά δεδομένα. Οι εφαρμογές PVM ακολουθούν δηλαδή το μοντέλο SPMD που περιγράφτηκε σε προηγούμενη παράγραφο. Παρόλα αυτά, υπάρχει η δυνατότητα οι εργασίες να εκτελούν διαφορετικές λειτουργίες.

2.2.3 Διεπαφή Προώθησης Μηνυμάτων (Message Passing Interface - MPI)

Το MPI είναι μια προδιαγραφή ενός συνόλου συναρτήσεων για τη διαχείριση της μετακίνησης δεδομένων μεταξύ επικοινωνούντων διεργασιών [10]. Βασικός στόχος του

είναι η παροχή μιας πρότυπης βιβλιοθήκης ρουτινών για τη συγγραφή φορητών και αποδοτικών προγραμμάτων ανταλλαγής μηνυμάτων. Συγκεκριμένες υλοποιήσεις έχουν αναπτυχθεί για την C, την Fortran, την C++, καθώς και άλλες γλώσσες προγραμματισμού. Το MPI ορίζει συναρτήσεις για επικοινωνία σημείο-προς-σημείο (point-to-point) μεταξύ δύο διεργασιών, για συλλογική επικοινωνία, για συγχρονισμό, για παράλληλο I/O και για τη διαχείριση διεργασιών. Οι μηχανισμοί επικοινωνίας καθορίζουν τους τύπους και τη διάταξη των δεδομένων που μεταφέρονται μεταξύ των διεργασιών, επιτρέποντας στις διάφορες MPI υλοποιήσεις να επιτυγχάνουν τη βελτιστοποίηση διαχείρισης μη-συνεχόμενων δεδομένων στη μνήμη και υποστηρίζοντας ετερογενή συστήματα. Τα προγράμματα MPI συνήθως υλοποιούνται με βάση το μοντέλο SPMD, όπου όλες οι διεργασίες εκτελούν το ίδιο εκτελέσιμο και ο διαχωρισμός των λειτουργιών που εκτελεί η κάθε διεργασία γίνεται με βάση κάποια μεταβλητή κατάστασης. Παρόλα αυτά, μια MPI εφαρμογή μπορεί να υλοποιηθεί και με βάση το MPMD μοντέλο, όπου οι παράλληλοι επεξεργαστές μπορούν να εκτελούν δύο ή περισσότερα διαφορετικά εκτελέσιμα, ανάλογα με τη διαμόρφωση που έχει κάνει ο χρήστης.

Μια εφαρμογή MPI είναι ουσιαστικά ένα σύνολο από εργασίες που εκτελούνται ταυτόχρονα (σε έναν ή περισσότερους επεξεργαστές) και οι οποίες μπορούν να επικοινωνούν μεταξύ τους. Ένα πρόγραμμα αποτελείται από τον κώδικα που ανέπτυξε ο προγραμματιστής, ο οποίος συνδέεται με τη βιβλιοθήκη συναρτήσεων του MPI που παρέχονται από κάποια συγκεκριμένη υλοποίηση του προτύπου. Σε κάθε εργασία αποδίδεται μια μοναδική τάξη (rank), η οποία αποτελεί ένα αναγνωριστικό κατά την εκτέλεση της MPI εφαρμογής. Η τάξη της εργασίας είναι ένας ακέραιος αριθμός μεταξύ του 0 και του n-1 για μια εφαρμογή που αποτελείται από n εργασίες. Οι τάξεις αυτές χρησιμοποιούνται από τις εργασίες για την μεταξύ τους αναγνώριση κατά την αποστολή και λήψη μηνυμάτων, για την εκτέλεση συλλογικών λειτουργιών και γενικά για την επίτευξη κάθε είδους συνεργασίας.

Το MPI αποτελεί το πλέον διαδεδομένο μοντέλο για την ανάπτυξη παράλληλων εφαρμογών με δυνατότητα κλιμάκωσης και αυτή ακριβώς η αποδοχή του το καθιστά την τεχνολογία με την μεγαλύτερη φορητότητα. Παρόλα αυτά, ο προγραμματισμός σε MPI θεωρείται αρκετά χρονοβόρος και απαιτητικός, για το λόγο ότι ο προγραμματιστής πρέπει να υλοποιήσει με ρητό τρόπο τη λογική ανταλλαγής μηνυμάτων μεταξύ των διεργασιών και τον όλο συντονισμό που απαιτείται για την επίλυση ενός προβλήματος.

Στη συνέχεια περιγράφονται τα βασικότερα στοιχεία του προτύπου MPI.

```
#include <mpi.h>
//εισαγωγή άλλων απαιτούμενων βιβλιοθηκών

int main(int argc,  char** argv)
{
        int myrank;
        int noOfProcesses;

        MPI_Init(&argc, &argv);
        MPI_Comm_rank(MPI_COMM_WORLD,  &myrank);
        MPI_Comm_size(MPI_COMM_WORLD,  &noOfProcesses);

        // υλοποίηση αλγορίθμου
        //ανταλλαγή μηνυμάτων

        MPI_Finalize();
        return 0;
}
```

Listing 1. Βασικός Σκελετός Προγράμματος MPI

2.2.3.1 Ομάδες Επικοινωνίας (Communicators)

Μια βασική απαίτηση σε όλα τα συστήματα ανταλλαγής μηνυμάτων είναι η διασφάλιση ενός ασφαλούς χώρου επικοινωνίας, όπου τα μη συσχετιζόμενα μηνύματα διαχωρίζονται μεταξύ τους. Για παράδειγμα, τα μηνύματα που προέρχονται από λειτουργίες της βιβλιοθήκης πρέπει να μπορούν να λαμβάνονται και να αποστέλλονται χωρίς παρεμβολές από άλλα μηνύματα που δημιουργούνται στο σύστημα. Στο MPI, όπου δεν υπάρχει η έννοια της εικονικής μηχανής, η χρήση μόνο μια ετικέτας (tag) μηνύματος δεν είναι αρκετή για τον ασφαλή διαχωρισμό μεταξύ των μηνυμάτων βιβλιοθήκης και των μηνυμάτων χρήστη. Για την επίτευξη της ασφαλούς επικοινωνίας, στο MPI εισάγεται η έννοια του communicator, το οποίο μπορεί να θεωρηθεί ως μια αντιστοίχηση ενός πλαισίου επικοινωνίας σε ένα σύνολο εργασιών.

Ο βασικός σκελετός ενός προγράμματος MPI φαίνεται στο Listing 1. Όταν ξεκινάει μια εφαρμογή MPI, όλες οι εργασίες συσχετίζονται με τον εξ ορισμού communicator MPI_COMM_WORLD. Μετά την κλήση της συνάρτησης MPI_Init(), αυτός ο εξ ορισμού communicator ορίζει ένα πλαίσιο επικοινωνίας που περιλαμβάνει το σύνολο των MPI εργασιών που υπάρχουν διαθέσιμες για την εκτέλεση. Στις εργασίες που σχετίζονται με έναν δεδομένο communicator αναθέτονται ως αναγνωριστικά διαδοχικοί ακέραιοι αριθμοί

μεταξύ του 0 και του μεγέθους του communicator μείον ένα. Αυτά τα αναγνωριστικά, τα οποία ονομάζονται τάξεις των εργασιών, χρησιμοποιούνται για το διαχωρισμό αυτών μέσα σε μια συγκεκριμένη ομάδα επικοινωνίας. Για παράδειγμα, ο προγραμματιστής μπορεί με βάση την τάξη να αναθέσει διαφορετικούς υπολογισμούς σε κάθε εργασία. Η ανάκτηση της τάξης μιας εργασίας γίνεται με κλήση της συνάρτησης MPI_Comm_rank, όπως φαίνεται και στο Listing 1.

Όταν απαιτείται κάποιο νέο πλαίσιο επικοινωνίας, το MPI παρέχει έναν αριθμό συναρτήσεων για τη δημιουργία νέων ομάδων επικοινωνίας από κάποιες υπάρχουσες.

2.2.3.2 Επικοινωνία Εργασιών

Η επικοινωνία μεταξύ των εργασιών μιας εφαρμογής MPI βασίζεται στο μοντέλο ανταλλαγής μηνυμάτων. Το πρότυπο MPI παρέχει ένα σύνολο από συναρτήσεις για αποστολή και λήψη μηνυμάτων. Η επικοινωνία μεταξύ δύο εργασιών περιλαμβάνει τα ακόλουθα συστατικά:

(i). αποστολέας, ο οποίος συνήθως αναγνωρίζεται από την τάξη του

(ii). δέκτης, ο οποίος επίσης αναγνωρίζεται από την τάξη του

(iii). δεδομένα μηνύματος

(iv). ετικέτα μηνύματος, η οποία επιτρέπει τον κατάλληλο χειρισμό πολλαπλών μηνυμάτων μεταξύ δύο εργασιών

(v). ομάδα επικοινωνίας, η οποία παρέχει το ίδιο το πλαίσιο της επικοινωνίας

Ο βασικός τρόπος αποστολής και λήψης μηνυμάτων είναι με μπλοκάρισμα (blocking send – blocking receive). Επίσης, παρέχεται ένας αριθμός από παραλλαγές του βασικού αυτού τρόπου. Επιπλέον, το MPI υποστηρίζει και μηχανισμούς ασύγχρονης επικοινωνίας, με τη βοήθεια non-blocking συναρτήσεων αποστολής και λήψης.

2.2.3.3 Συγχρονισμός

Ο συγχρονισμός μεταξύ των παράλληλων εργασιών μιας εφαρμογής MPI μπορεί να επιτευχθεί είτε μέσω της ανταλλαγής μηνυμάτων είτε χρησιμοποιώντας ρητές λειτουργίες φράγματος (barrier synchronization):

- συγχρονισμός μέσω επικοινωνίας: με χρήση κατάλληλης μορφής συναρτήσεων αποστολής και λήψης μηνυμάτων (π.χ. blocking receive), μπορεί να δημιουργηθούν συγκεκριμένα σημεία συγχρονισμού σε ένα πρόγραμμα. Για παράδειγμα, μια εργασία που εκτελεί blocking receive πρέπει να περιμένει μέχρι

να ληφθεί ένα μήνυμα. Έτσι, ο αποστολέας του μηνύματος μπορεί να καθυστερεί την αποστολή του για όσο χρονικό διάστημα χρειάζεται να τον περιμένει ο παραλήπτης.

- barrier synchronization: η συγκεκριμένη μέθοδος συγχρονισμού επιτυγχάνεται όταν όλες οι εργασίες σε μια ομάδα επικοινωνίας καλούν μια συνάρτηση φράγματος (*MPI_Barrier()*). Μια εργασία περιμένει μέχρι όλες οι εργασίες που περιλαμβάνονται στην ομάδα επικοινωνίας να φτάσουν στο φράγμα. Η κλήση στη συνάρτηση φράγματος επιστρέφει όταν όλες αυτές οι εργασίες έχουν εκτελέσει την κλήση σε αυτήν.

2.2.3.4 Συλλογικές Λειτουργίες (Collective Operations)

Οι συλλογικές λειτουργίες στο MPI αναφέρονται σε εκείνες τις λειτουργίες που εφαρμόζονται σε όλες τις εργασίες μιας ομάδας επικοινωνίας. Έτσι, μια συλλογική λειτουργία τυπικά αντιστοιχεί σε μια τέτοια ομάδα. Η λειτουργία εκτελείται όταν όλες οι εργασίες της ομάδας καλέσουν την αντίστοιχη ρουτίνα. Υπάρχουν τρεις κατηγορίες συλλογικών λειτουργιών: ελέγχου εργασιών, καθολικών υπολογισμών και μεταφοράς δεδομένων. Παράδειγμα της πρώτης κατηγορίας είναι η συνάρτηση *MPI_Barrier()* που αναφέρθηκε στην προηγούμενη παράγραφο. Η κατηγορία καθολικών υπολογισμών περιλαμβάνει μεταξύ άλλων λειτουργίες για υπολογισμό αθροίσματος, εύρεση ελαχίστου ή μεγίστου, κτλ. Δύο χαρακτηριστικές συναρτήσεις για το σκοπό αυτό είναι οι *MPI_Reduce()* και *MPI_Allreduce()*. Η λειτουργία που θα εκτελέσουν δίνεται και στις δύο σαν όρισμα και διαφέρουν στο ότι η πρώτη επιστρέφει το αποτέλεσμα μόνο σε μία εργασία, ενώ η δεύτερη σε όσες εργασίες την έχουν καλέσει. Οι κύριες λειτουργίες που ανήκουν στην κατηγορία συλλογικών λειτουργιών μεταφοράς δεδομένων είναι οι λειτουργίες πολυεκπομπής (broadcast), διασποράς (scatter) και συγκέντρωσης (gather). Στη λειτουργία broadcast, μια διεργασία στέλνει ένα μήνυμα σε κάθε άλλο μέλος μιας ομάδας επικοινωνίας, ενώ η λειτουργία scatter επιτρέπει σε μια διεργασία να στείλει διαφορετικό μήνυμα σε κάθε μέλος. Η συμμετρική λειτουργία της scatter είναι η gather, με την οποία μια διεργασία λαμβάνει ένα μήνυμα από κάθε άλλο μέλος της ομάδας επικοινωνίας. Αυτές οι βασικές λειτουργίες μπορούν να συνδυαστούν για τη δημιουργία επιπλέον πολυπλοκότερων μηχανισμών συλλογικής επικοινωνίας.

2.3 Πλατφόρμες και Τεχνολογίες Κατανεμημένων Αντικειμένων

Ο όρος κατανεμημένα αντικείμενα αναφέρεται σε ανεξάρτητα αντικείμενα λογισμικού που είναι σχεδιασμένα να συνεργάζονται για την επίτευξη κάποιου κοινού σκοπού και τα οποία τυπικά εκτελούνται σε διαφορετικούς υπολογιστές που συνδέονται μεταξύ τους μέσω κάποιου δικτύου. Στις σύγχρονες εφαρμογές, πρακτικά ολόκληρη η στοίβα των πρωτοκόλλων επικοινωνίας βασίζεται στο Πρωτόκολλο Διαδικτύου (Internet Protocol – IP). Σε αυτήν την παράγραφο γίνεται αναφορά στις τεχνολογίες κατανεμημένων αντικειμένων που έχουν ξεχωρίσει τα τελευταία χρόνια.

2.3.1 Απομακρυσμένη Κλήση Μεθόδου (Remote Method Invocation – RMI)

Το RMI [11] είναι μια τεχνολογία που στηρίζεται στην Java και επιτρέπει στην Εικονική Μηχανή Java (Java Virtual Machine – JVM) ενός υπολογιστή να καλεί μεθόδους αντικειμένων οι οποίες εκτελούνται σε JVM άλλων απομακρυσμένων υπολογιστών, επιτυγχάνοντας το διαμοιρασμό πόρων και την κατανομή υπολογιστικού φορτίου μεταξύ διαφορετικών συστημάτων. Σε αντίθεση με άλλα συστήματα απομακρυσμένης εκτέλεσης που απαιτούν ότι μόνο απλοί τύποι δεδομένων ή προκαθορισμένες δομές ανταλλάσσονται μεταξύ των μεθόδων, το RMI επιτρέπει τη χρήση οποιουδήποτε αντικειμένου Java, ακόμη και αν ο πελάτης ή ο εξυπηρετητής δεν έχουν καμία εκ των προτέρων γνώση για αυτό.. Αυτό επιτυγχάνεται με το δυναμικό φόρτωμα των απαιτούμενων νέων κλάσεων.

Οι εφαρμογές RMI συνήθως αποτελούνται από δύο ξεχωριστά προγράμματα, έναν εξυπηρετητή και έναν πελάτη. Ένας τυπικός εξυπηρετητής δημιουργεί κάποια απομακρυσμένα αντικείμενα, καθιστά δυνατή την πρόσβαση στις αναφορές τους και αναμένει την κλήση μεθόδων σε αυτά από πελάτες. Ένας τυπικός πελάτης αποκτά μια απομακρυσμένη αναφορά σε ένα ή περισσότερα απομακρυσμένα αντικείμενα ενός εξυπηρετητή και καλεί κάποιες μεθόδους σε αυτά. Το RMI παρέχει το μηχανισμό με τον οποίο ο εξυπηρετητής και ο πελάτης επικοινωνούν και ανταλλάσσουν πληροφορία.

Οι βασικές λειτουργίες που επιτελούνται σε μια εφαρμογή RMI είναι οι εξής:

- Εντοπισμός απομακρυσμένων αντικειμένων: οι εφαρμογές χρησιμοποιούν διάφορους τρόπους για την απόκτηση αναφορών σε απομακρυσμένα αντικείμενα. Για παράδειγμα, μια εφαρμογή μπορεί να καταχωρήσει τα απομακρυσμένα αντικείμενά της στο μητρώο RMI (RMI registry), μια απλή υπηρεσία διαχείρισης ονοματολογίας. Εναλλακτικά, οι αναφορές στα απομακρυσμένα αντικείμενα μπορούν να μεταφέρονται ως τμήματα άλλων απομακρυσμένων κλήσεων.

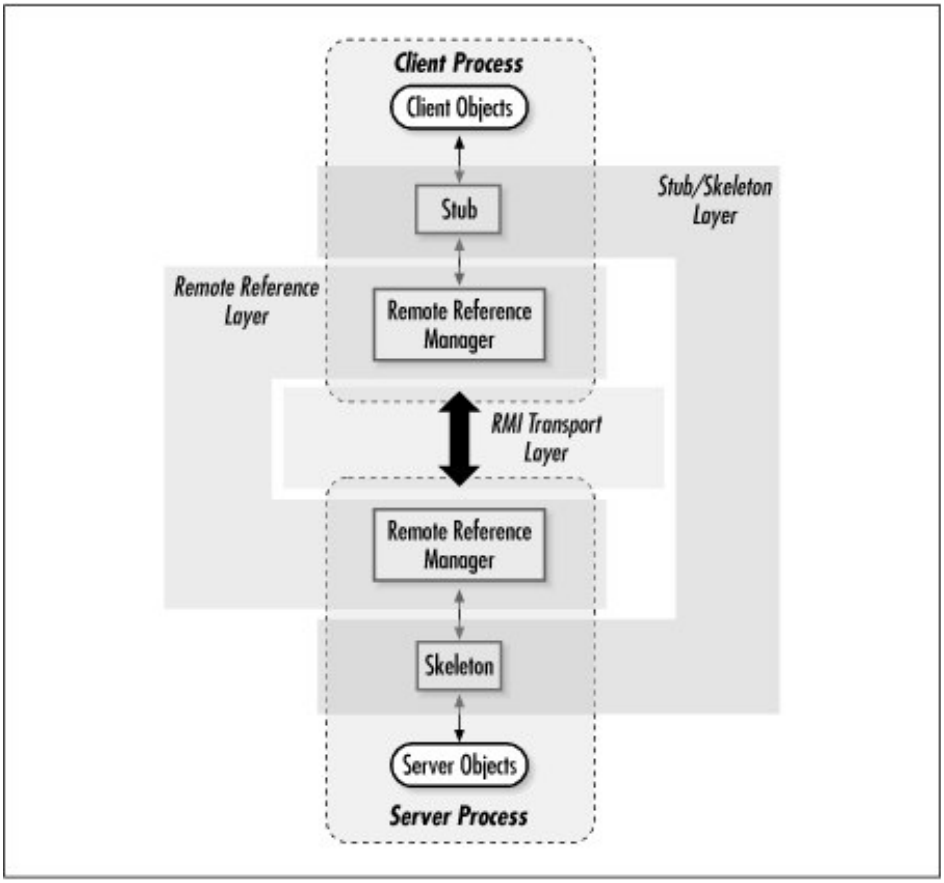
- Επικοινωνία με απομακρυσμένα αντικείμενα: οι λεπτομέρειες της επικοινωνίας μεταξύ των απομακρυσμένων αντικειμένων αναλαμβάνονται από το RMI. Από την πλευρά του προγραμματιστή, οι απομακρυσμένες κλήσεις μοιάζουν με συνήθεις κλήσεις αντικειμένων Java.
- Δυναμικό φόρτωμα ορισμού κλάσεων για μεταφερόμενα αντικείμενα: λόγω της απαίτησης για μεταφορά αντικειμένων κατά την επικοινωνία, το RMI παρέχει μηχανισμούς για φόρτωμα των ορισμών των κλάσεων των αντικειμένων, καθώς και για τη μετάδοση των δεδομένων τους.

Η αρχιτεκτονική ενός συστήματος RMI φαίνεται στο Σχήμα 6. Αποτελείται από τρία ανεξάρτητα στρώματα:

- το στρώμα στελέχους/σκελετού (stub/skeleton layer): δημιουργεί τις διεπαφές λογισμικού που χειρίζονται οι προγραμματιστές προκειμένου να διεκπεραιώσουν κατανεμημένες κλήσεις
- το στρώμα απομακρυσμένης διεπαφής (remote interface layer): διαχειρίζεται το σύνολο των δεικτών (αναφορών) σε απομακρυσμένα αντικείμενα, φροντίζοντας το συγχρονισμό των αντιγράφων ανεξάρτητα από τα στελέχη και τους σκελετούς
- και το στρώμα μεταφοράς (transport layer): υλοποιεί τη μετακίνηση δεδομένων, δημιουργώντας συνδέσεις και μεταφράζοντας τις αναφορές σε διευθύνσεις

2.3.2 Jini

Το Jini [12] είναι μια τεχνολογία για την ανάπτυξη εφαρμογών που αποτελούνται από υπηρεσίες. Ορίζει ένα προγραμματιστικό μοντέλο που βασίζεται στην τεχνολογία Java, την οποία επεκτείνει, επιτρέποντας την κατασκευή ασφαλών κατανεμημένων συστημάτων που αποτελούνται από ομάδες υπηρεσιών δικτύου και πελατών με καλά ορισμένη συμπεριφορά. Βασικός άξονας του συστήματος Jini είναι η ομαδοποίηση των χρηστών και των πόρων που απαιτούνται από αυτούς, με σκοπό τη μετατροπή του δικτύου σε ένα ευέλικτο και εύκολα διαχειρίσιμο εργαλείο, στο οποίο οι πόροι μπορούν να ανευρεθούν από κάθε πελάτη (άνθρωπο ή μηχανή). Οι πόροι μπορεί να είναι υλικό, λογισμικό ή και συνδυασμός των δύο. Το επίκεντρο του συστήματος είναι να κάνει το δίκτυο μια δυναμική οντότητα που έχει τη δυνατότητα να προσαρμόζεται στη δυναμική φύση μιας ομάδας χρηστών, επιτρέποντας την ευέλικτη δημιουργία και καταστροφή υπηρεσιών.



Σχήμα 6. Αρχιτεκτονική Συστήματος RMI

Ένα σύστημα Jini αποτελείται από τα ακόλουθα στοιχεία:

- ένα σύνολο συστατικών που παρέχουν μια υποδομή για την ομαδοποίηση υπηρεσιών σε ένα κατανεμημένο σύστημα
- ένα προγραμματιστικό μοντέλο που υποστηρίζει την παραγωγή αξιόπιστων κατανεμημένων υπηρεσιών
- ένα σύνολο υπηρεσιών που μπορούν να γίνουν τμήμα μιας ομάδας χρηστών, παρέχοντας κάποια λειτουργικότητα στα μέλη αυτής

2.3.3 Common Object Request Broker Architecture (CORBA)

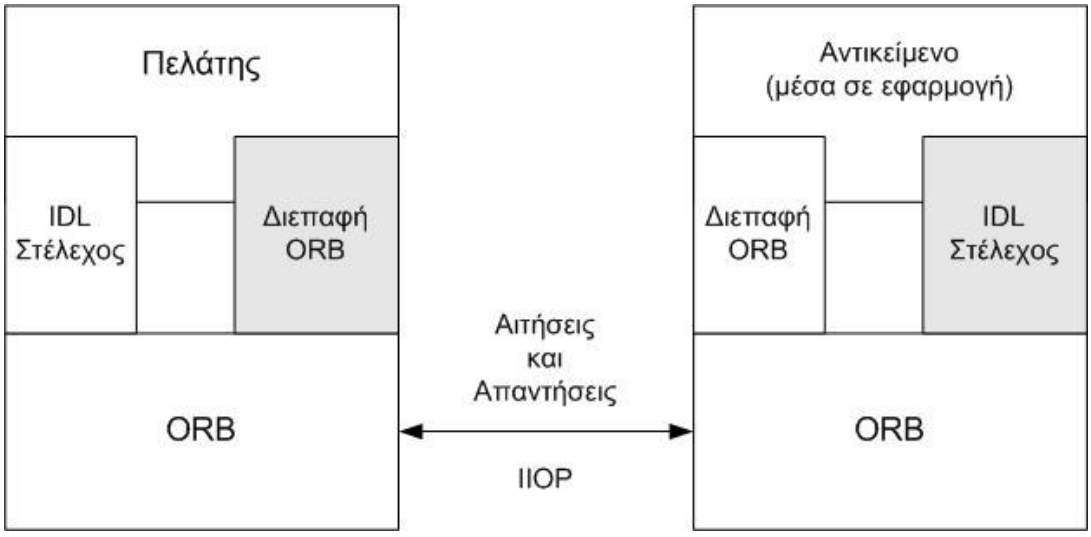
Η αρχιτεκτονική CORBA [13] είναι μια ομάδα προτύπων που αναπτύχθηκαν από τον οργανισμό Object Management Group (OMG) και που έχει ως σκοπό τη διευκόλυνση ανάπτυξης κατανεμημένων εφαρμογών. Σημειώνεται ότι η CORBA είναι απλώς ένα σύνολο προδιαγραφών για τη δημιουργία και τη χρήση κατανεμημένων αντικειμένων και όχι κάποια γλώσσα προγραμματισμού. Η αρχιτεκτονική είναι βασισμένη στο μοντέλο του

αντικειμένου, όπως αυτό ορίζεται από τον OMG στο Object Management Architecture Guide, το οποίο είναι αφαιρετική οντότητα και δεν υλοποιείται απευθείας από κάποια συγκεκριμένη τεχνολογία. Αυτή η προσέγγιση επιτρέπει την ανάπτυξη εφαρμογών με πρότυπες μεθοδολογίες, χρησιμοποιώντας το μοντέλο του αντικειμένου ως βασικό δομικό στοιχείο. Ένα σύστημα CORBA είναι ένα σύνολο από αντικείμενα, το οποίο απομονώνει τους πελάτες από τους εξυπηρετητές μέσω μιας καλά καθορισμένης διεπαφής ενθυλάκωσης. Σε αυτό το σημείο πρέπει να σημειωθεί ότι τα αντικείμενα στην αρχιτεκτονική CORBA διαφέρουν από τα τυπικά αντικείμενα των αντικειμενοστραφών γλωσσών προγραμματισμού στα εξής:

- μπορούν να εκτελούνται σε οποιαδήποτε πλατφόρμα
- μπορούν να βρίσκονται σε οποιαδήποτε τοποθεσία σε ένα δίκτυο
- μπορούν να υλοποιηθούν με βάση οποιαδήποτε γλώσσα παρέχει υλοποίηση των διεπαφών ενθυλάκωσης της CORBA

Ένα από τα δύο κύρια στοιχεία της CORBA είναι η γλώσσα περιγραφής διεπαφών (Interface Description Language - IDL). Η IDL ορίζει τους τύπους των αντικειμένων σε μια κατανεμημένη εφαρμογή, ορίζοντας τις διεπαφές τους. Μια διεπαφή αποτελείται από ένα σύνολο ονομαζόμενων λειτουργιών και από τις παραμέτρους αυτών των λειτουργιών. Σημειώνεται ότι η IDL χρησιμοποιείται για την περιγραφή των διεπαφών μόνο και όχι των συγκεκριμένων υλοποιήσεων. Μέσω της IDL, μια συγκεκριμένη υλοποίηση ενός αντικειμένου δίνει στον πελάτη πληροφορία σχετικά με το ποιες λειτουργίες είναι διαθέσιμες και τον τρόπο που αυτές μπορούν να κληθούν. Αφού οριστούν οι διεπαφές τους, τα αντικείμενα CORBA μπορούν να υλοποιηθούν σε διαφορετικές γλώσσες προγραμματισμού που παρέχουν κάποια αντιστοίχιση για την IDL (για παράδειγμα C, C++, Java κλπ). Για να χρησιμοποιηθεί ένα τέτοιο αντικείμενο, μπορεί να χρησιμοποιηθεί οποιαδήποτε γλώσσα προγραμματισμού για την υλοποίηση απομακρυσμένων αιτήσεων σε αυτό.

Το δεύτερο βασικό στοιχείο της CORBA είναι η διεπαφή του Μεσίτη Αίτησης Αντικειμένου (Object Request Broker – ORB). Ο ORB είναι μια λογική οντότητα που μπορεί να υλοποιηθεί με διάφορους τρόπους (π.χ. μία ή περισσότερες διεργασίες ή ένα σύνολο από βιβλιοθήκες). Για το διαχωρισμό των εφαρμογών από τις λεπτομέρειες υλοποίησης, οι προδιαγραφές CORBA ορίζουν μια αφηρημένη διεπαφή για τον ORB, η οποία παρέχει διάφορες βοηθητικές λειτουργίες (π.χ. μετατροπή αναφορών σε αντικείμενα σε αλφαριθμητικά και αντίστροφα).



Σχήμα 7. Αρχιτεκτονική Συστήματος CORBA

Κατά την ανάπτυξη ενός συστήματος CORBA απαιτείται προφανώς η υλοποίηση του ORB. Η υλοποίηση αυτή είναι υπεύθυνη για τις ακόλουθες εργασίες:

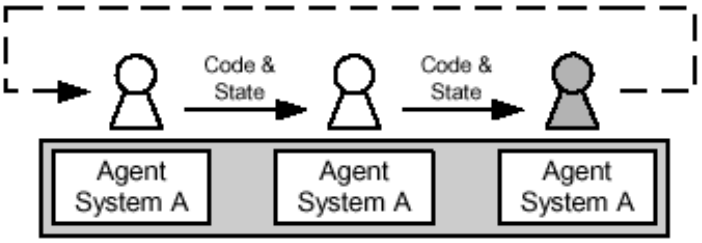
- εύρεση της υλοποίησης ενός αντικειμένου που σχετίζεται με μια αίτηση
- προετοιμασία της υλοποίησης του αντικειμένου για λήψη της αίτησης
- πραγματοποίηση της επικοινωνίας των δεδομένων που απαρτίζουν την αίτηση

Για την επικοινωνία μεταξύ των ORBs έχουν αναπτυχθεί από τον OMG οι προδιαγραφές ενός αφηρημένου πρωτοκόλλου επικοινωνίας, του General Inter-Orb Protocol (GIOP). Η σημαντικότερη ίσως υλοποίηση του GIOP είναι το Internet Inter-ORB Protocol (IIOP), το οποίο λειτουργεί με βάση το πρωτόκολλο TCP/IP.

Ο τρόπος λειτουργίας ενός συστήματος CORBA φαίνεται σχηματικά στο Σχήμα 7.

2.3.4 Κινητοί Πράκτορες

Με τον γενικό όρο "πράκτορας λογισμικού" [14] αναφερόμαστε σε μια αυτόνομη οντότητα λογισμικού, η οποία αυτοματοποιεί κάποιες δραστηριότητες που ένας άνθρωπος ή οποιαδήποτε διεργασία λογισμικού μπορεί να έχει μεταβιβάσει σε αυτήν. Κάθε πράκτορας λογισμικού έχει τη δική του πληροφορία κατάστασης, τη δική του συμπεριφορά καθώς και το δικό του νήμα ελέγχου και μπορεί να αλληλεπιδρά ελεύθερα με άλλες οντότητες με σκοπό την επίλυση κάποιου συγκεκριμένου προβλήματος.



Σχήμα 8. Το υπολογιστικό παράδειγμα του κινητού αντιπρόσωπου

Οι *κινητοί* πράκτορες έχουν την ιδιότητα της *κινητικότητας*: μπορούν να μεταναστεύουν από έναν υπολογιστή σε έναν άλλο και να συνεχίζουν εκεί την εκτέλεσή τους από το σημείο στο οποίο σταμάτησαν. Το υπολογιστικό παράδειγμα του κινητού πράκτορα φαίνεται στο Σχήμα 8. Κατά τη μετακίνηση αυτή, εκτός από τον κώδικα, ο πράκτορας παίρνει μαζί του και την πληροφορία κατάστασης (τιμές τοπικών μεταβλητών, ορίσματα και τιμές τοπικών μεταβλητών ρουτινών που έχουν κληθεί στο ίδιο νήμα εκτέλεσης και δεν έχουν τερματίσει, στοίβα εκτέλεσης κτλ). Ένα από τα πλέον βασικά χαρακτηριστικά τους είναι η αυτονομία, αφού από τη στιγμή που θα ξεκινήσει η εκτέλεσή τους μπορούν αυτόνομα να αποφασίσουν σε ποιες τοποθεσίες θα μεταναστεύσουν και ποιον κώδικα θα εκτελέσουν. Οι κινητοί πράκτορες αποτελούν μορφή κινητού κώδικα. Σε αντίθεση με τα κλασσικά παραδείγματα της απομακρυσμένης αξιολόγησης (remote evaluation) και του κατ' απαίτηση κώδικα (code on demand) [15], οι κινητοί πράκτορες είναι ενεργοί υπό την έννοια ότι μπορούν να μεταναστεύσουν μεταξύ υπολογιστών οποιαδήποτε στιγμή κατά την εκτέλεσή τους. Αυτή η ιδιότητα τους καθιστά ένα ιδιαίτερα ισχυρό εργαλείο στην ανάπτυξη κατανεμημένων εφαρμογών πάνω σε ένα δίκτυο υπολογιστών.

2.3.4.1 Μορφές Μετανάστευσης

Όπως προαναφέρθηκε, η μετανάστευση ενός κινητού πράκτορα συμπεριλαμβάνει την μετακίνηση του κώδικα του πράκτορα, καθώς και της πληροφορίας κατάστασής του. Η κατάσταση του πράκτορα αποτελείται από την κατάσταση των δεδομένων (όπως για παράδειγμα οι καθολικές μεταβλητές και στατικές μεταβλητές κλάσεων) και την κατάσταση εκτέλεσης (όπως για παράδειγμα τοπικές μεταβλητές και παράμετροι καθώς και εκτελούμενα νήματα). Ανάλογα με το ποια πληροφορία μεταφέρεται κατά την μετανάστευση ενός πράκτορα, μπορούν να διακριθούν δύο μορφές αυτής [17]:

- Ισχυρή μετανάστευση:

Σε αυτό το μοντέλο, το υποκείμενο σύστημα συλλαμβάνει ολόκληρη την κατάσταση του πράκτορα (αποτελούμενη από δεδομένα και κατάσταση εκτέλεσης) και την μεταφέρει μαζί με τον κώδικα στην επόμενη τοποθεσία προορισμού. Όταν ο πράκτορας φτάσει στον προορισμό, η κατάστασή του επαναφέρεται αυτόματα. Από την πλευρά του προγραμματιστή το μοντέλο αυτό είναι πολύ ελκυστικό, καθώς η όλη διαδικασία της παγίδευσης της κατάστασης, της μεταφοράς και της επαναφοράς γίνεται με τελείως διαφανή τρόπο από το υποκείμενο σύστημα. Το πρόβλημα που προκύπτει είναι ότι για την παροχή αυτού του βαθμού διαφάνειας σε ετερογενή περιβάλλοντα, απαιτείται τουλάχιστον ένα καθολικό μοντέλο για την κατάσταση ενός πράκτορα, καθώς επίσης και ένα συντακτικό μεταφοράς για το κάθε είδος πληροφορίας. Επιπλέον, ένα σύστημα πρακτόρων πρέπει να παρέχει λειτουργίες για την ανάκτηση και τη μεταφορά της κατάστασης ενός πράκτορα. Τέλος, η πλήρης κατάσταση ενός πράκτορα (δεδομένα και κατάσταση εκτέλεσης) μπορεί να είναι αρκετά μεγάλη, όπως για παράδειγμα στην περίπτωση των πολυνηματικών πρακτόρων, οπότε η ισχυρή μετανάστευση μπορεί να γίνει μια πολύ χρονοβόρα και ακριβή διαδικασία.

- Ασθενής μετανάστευση:

Οι δυσκολίες που παρουσιάζει η προηγούμενη μορφή μετανάστευσης οδήγησε στο μοντέλο της ασθενούς μετανάστευσης, στο οποίο μεταφέρεται μόνο η πληροφορία της κατάστασης δεδομένων και όχι και της κατάστασης εκτέλεσης ενός πράκτορα. Το μέγεθος της πληροφορίας της μεταφερόμενης κατάστασης μπορεί να περιοριστεί ακόμη περισσότερο αφήνοντας τον προγραμματιστή να επιλέξει τις μεταβλητές που αποτελούν την κατάσταση του πράκτορα. Επιπλέον, ο προγραμματιστής πρέπει να παρέχει μια μέθοδο εκκίνησης που θα αποφασίζει με βάση την κωδικοποιημένη πληροφορία κατάστασης από ποιο σημείο να συνεχίσει την εκτέλεση μετά από κάθε μετανάστευση. Αυτό το μοντέλο μειώνει δραστικά το μέγεθος της πληροφορίας που πρέπει να μεταφερθεί, αλλά βάζει μεγαλύτερο φόρτο στον προγραμματιστή και επιπλέον καθιστά τα προγράμματα-πράκτορες πολυπλοκότερα.

Οι περισσότερες πλατφόρμες πολλαπλών πρακτόρων που έχουν παρουσιαστεί τα τελευταία χρόνια (π.χ. Grasshopper [18], JADE [19]) στηρίζονται στη Java. Η Java παρέχει ένα σύνολο από μηχανισμούς και χαρακτηριστικά (π.χ. ανεξαρτησία από πλατφόρμα, δυναμικό φόρτωμα κλάσεων, πολυνηματικός προγραμματισμός, σειριοποίηση αντικειμένων) που την καθιστούν ιδιαίτερα ελκυστική για την ανάπτυξη τέτοιων συστημάτων. Παρόλα αυτά, η Java δεν επιτρέπει εγγενώς την ανάκτηση της κατάστασης

εκτέλεσης ενός νήματος, πράγμα που καθιστά την ισχυρή μετανάστευση αδύνατη. Όλες οι Java-based πλατφόρμες κινητών πρακτόρων υποστηρίζουν το μοντέλο της ασθενούς μετανάστευσης, παρέχοντας μηχανισμούς για μεταφορά του κώδικα και των δεδομένων που περιέχει ένα αντικείμενο πράκτορα, ενώ η υλοποίηση του κύκλου ζωής του πράκτορα επιβαρύνει τον προγραμματιστή.

2.3.4.2 Διαλειτουργικότητα – Πρότυπα

Μια βασική απαίτηση για τα συστήματα πολλαπλών πρακτόρων είναι η δυνατότητα διαλειτουργικότητας μεταξύ διαφορετικών και ετερογενών συστημάτων. Για το σκοπό αυτό έχουν προταθεί κατά καιρούς κάποια πρότυπα, τα σημαντικότερα από τα οποία έχουν συσταθεί από την οργάνωση FIPA (Foundation for Intelligent Physical Agents) [20]. Οι προδιαγραφές της FIPA περιγράφουν πρωτόκολλα, τα οποία στοχεύουν στην ομαλή συνδεσιμότητα μεταξύ πρακτόρων οποιασδήποτε προέλευσης και υλοποίησης, καθώς και μεταξύ πρακτόρων και οντοτήτων που ανήκουν σε άλλα συστήματα. Οι προδιαγραφές αυτές μπορούν να διαχωριστούν σε τέσσερις γενικές κατηγορίες:

- επικοινωνία πρακτόρων
- μεταφορά πρακτόρων
- διαχείριση πρακτόρων
- αρχιτεκτονική συστήματος

Από τις παραπάνω κατηγορίες σημαντικότερα είναι τα πρωτόκολλα που σχετίζονται με την επικοινωνία των πρακτόρων στο μοντέλο FIPA. Η επικοινωνία βασίζεται στην Γλώσσα Επικοινωνίας Πρακτόρων (Agent Communication Language – ACL), η οποία ορίζει ένα αφηρημένο μοντέλο μηνύματος και συγκεκριμένες παραμέτρους για αυτό. Η ACL έχει αναπτυχθεί με γνώμονα την περιγραφή οντολογιών και μπορεί να επεκταθεί για να ικανοποιεί τις απαιτήσεις οποιασδήποτε εφαρμογής πολλαπλών πρακτόρων. Προφανώς, τα πρότυπα ACL μπορούν να εφαρμοστούν σε υλοποιήσεις με διαφορετικές τεχνολογίες. Πρακτικά, τα πρωτόκολλα που χρησιμοποιούνται είναι τα εξής: sockets, RMI, IIOP και SOAP/XML.

2.4 Απόδοση Παράλληλων Συστημάτων – Δείκτες Απόδοσης

Ο στόχος κατά την ανάπτυξη παράλληλων συστημάτων είναι η μείωση του απαιτούμενου χρόνου εκτέλεσης ενός προγράμματος. Ο όρος παράλληλο σύστημα εδώ

αναφέρεται στο συνδυασμό του παράλληλου αλγόριθμου και της παράλληλης αρχιτεκτονικής υλικού. Το βασικό στοιχείο που χαρακτηρίζει μία υλοποίηση ενός παράλληλου συστήματος είναι η *κλιμάκωση* (*scalability*). Η κλιμάκωση ορίζεται ως ένα μέτρο της ικανότητας που παρουσιάζει ένα παράλληλο σύστημα για την αποδοτική χρησιμοποίηση ενός αυξανόμενου αριθμού επεξεργαστών. Η ανάλυση της ικανότητας κλιμάκωσης μπορεί να χρησιμοποιηθεί για την επιλογή του καλύτερου συνδυασμού αλγορίθμου-αρχιτεκτονικής για ένα πρόβλημα κάτω από διαφορετικούς περιορισμούς στο ρυθμό αύξησης του μεγέθους του προβλήματος και του αριθμού των διαθέσιμων επεξεργαστών. Επίσης, μπορεί να χρησιμοποιηθεί για την πρόβλεψη της απόδοσης ενός παράλληλου αλγόριθμου σε μια συγκεκριμένη αρχιτεκτονική για μεγάλο αριθμό επεξεργαστών με βάση την μετρούμενη απόδοση σε μικρότερο αριθμό επεξεργαστών. Τέλος, για ένα συγκεκριμένο μέγεθος προβλήματος, η ανάλυση της κλιμάκωσης μπορεί να υποδείξει το βέλτιστο αριθμό επεξεργαστών που θα χρησιμοποιηθούν και τη μέγιστη απόδοση που μπορεί να επιτευχθεί.

2.4.1 Βασικοί Δείκτες Απόδοσης

Το βασικό μέγεθος αξιολόγησης ενός παράλληλου συστήματος είναι η επιτάχυνση (speedup) *S*, η οποία ορίζεται ως ο λόγος του χρόνου εκτέλεσης T_s του σειριακού αλγορίθμου προς τον χρόνο εκτέλεσης T_p του παράλληλου αλγορίθμου (χρόνος που απαιτείται για να ολοκληρώσει την εκτέλεση και ο τελευταίος επεξεργαστής):

$$S = \frac{T_s}{T_p}$$

Ένας παράλληλος αλγόριθμος εμπεριέχει κάποιες επιβαρύνσεις στο χρόνο εκτέλεσης σε σχέση με τον αντίστοιχο σειριακό, οι οποίες οφείλονται σε παράγοντες όπως η επικοινωνία μεταξύ των διεργασιών, ο συγχρονισμός ή οι επιπλέον υπολογισμοί που περιλαμβάνει ο παράλληλος αλγόριθμος (π.χ. δημιουργία και κατανομή εργασιών). Αν T_o είναι η επιβάρυνση που προκαλεί ο παράλληλος αλγόριθμος, τότε προκύπτει η εξής σχέση:

$$nT_p = T_s + T_o$$

ή

$$T_p = \frac{T_s + T_o}{n}$$

Οπότε τελικά η επιτάχυνση δίνεται από τη σχέση:

$$S_n = \frac{nT_s}{T_s + T_o}$$

Τυπικά, η τιμή του *S* είναι μικρότερη ή ίση του αριθμού *n* των επεξεργαστών. Στην περίπτωση που το *S* παραμένει πολύ κοντά στο *n* καθώς το *n* αυξάνεται (γραμμική επιτάχυνση), το δεδομένο παράλληλο σύστημα θεωρείται καλά κλιμακούμενο (scalable). Στη βιβλιογραφία έχουν αναφερθεί περιπτώσεις που έχει παρατηρηθεί υπερ-γραμμική επιτάχυνση, πράγμα που οφείλεται συνήθως σε φαινόμενα που σχετίζονται με τη μνήμη. Όταν ένα συγκεκριμένο πρόγραμμα εκτελείται παράλληλα, η μνήμη που απαιτείται για την κάθε εργασία είναι υποπολλαπλάσια της μνήμης που απαιτεί η σειριακή εκτέλεση. Αυτό οδηγεί σε αποδοτικότερη χρησιμοποίηση της cache μνήμης και της εικονικής μνήμης, η οποία τελικά μειώνει το χρόνο εκτέλεσης σε σχέση με την αντίστοιχη σειριακή εκτέλεση.

Ως αποδοτικότητα (efficiency) *E* ενός παράλληλου συστήματος ορίζεται ο λόγος της επιτάχυνσης *S* προς τον αριθμό *n* των διαθέσιμων επεξεργαστών:

$$E_n = \frac{S_n}{n} = \frac{T_s}{T_s + T_o} = \frac{1}{1 + \frac{T_o}{T_s}}$$

Η τιμή του *E* τυπικά βρίσκεται στο διάστημα (0,1] και μπορεί να ξεπεράσει τη μονάδα σε περιπτώσεις που παρατηρείται υπερ-γραμμική επιτάχυνση.

Ένα ακόμη μέγεθος που έχει προταθεί για την αξιολόγηση ενός παράλληλου συστήματος είναι το ποσοστό επέκτασης υπολογιστικού φορτίου (workload expansion ratio) *W*, το οποίο ορίζεται ως:

$W = \frac{\sum_{i=1}^{n} t_i}{T_s}$, όπου t_i ο χρόνος ολοκλήρωσης εκτέλεσης για τον επεξεργαστή i.

Το μέγεθος αυτό δείχνει το ποσοστό αύξησης του υπολογιστικού φορτίο του παράλληλου αλγόριθμου σε σχέση με τον αντίστοιχο σειριακό λόγω των επιβαρύνσεων της παράλληλης επεξεργασίας.

Τέλος, ένα ακόμη μέγεθος είναι ο βαθμός αξιοποίησης πόρων (resource utilization) *U*, ο οποίος ορίζεται ως:

$U = \frac{T_{avg}}{T_p} = \frac{\sum_{i=1}^{n} t_i}{nT_p}$, όπου T_{avg} ο μέσος χρόνος εκτέλεσης του αλγορίθμου σε όλους τους επεξεργαστές του παράλληλου συστήματος.

2.4.2 Παράγοντες που Επηρεάζουν την Κλιμάκωση

Η κλιμάκωση ενός παράλληλου συστήματος περιορίζεται κυρίως από τους εξής παράγοντες:

(α) Σειριακό – μη παραλληλοποιήσιμο κομμάτι της εφαρμογής

Οι περισσότερες παράλληλες εφαρμογές αποτελούνται από ένα ή περισσότερα τμήματα κώδικα τα οποία δεν μπορούν να παραλληλοποιηθούν και τα οποία προφανώς απαιτείται να εκτελεστούν με σειριακό τρόπο σε έναν ή περισσότερους επεξεργαστές. Οι επιπτώσεις αυτής της διαπίστωσης στην κλιμάκωση μιας παράλληλης εφαρμογής εκφράστηκε για πρώτη φορά από τον Amdahl [21], από όπου προέκυψε και ο αντίστοιχος θεμελιώδης νόμος της παράλληλης επεξεργασίας.

Αν θεωρηθεί ότι r_s είναι το ποσοστό του χρόνου εκτέλεσης της εφαρμογής που αντιστοιχεί στο σειριακό τμήμα και r_p το αντίστοιχο για το παράλληλο τμήμα (προφανώς ισχύει ότι $r_s + r_p = 1$), τότε η επιτάχυνση μπορεί να προκύψει ως εξής:

$$S_n = \frac{T_s}{T_p} = \frac{T_s}{r_s T_s + \frac{r_p T_s}{n}} = \frac{1}{r_s + \frac{1 - r_s}{n}} = \frac{n}{1 + (n-1) r_s}$$

Η παραπάνω σχέση υποδεικνύει την εξάρτηση της επιτάχυνσης από το ποσοστό του σειριακού τμήματος της εφαρμογής. Μια σημαντική παρατήρηση προκύπτει αν θεωρήσουμε ότι έχουμε διαθέσιμο ένα μεγάλο αριθμό επεξεργαστών. Παίρνοντας το όριο της παραπάνω συνάρτησης όταν το n τείνει στο άπειρο, προκύπτει ότι:

$$\lim_{n \to \infty} S_n = \frac{1}{r_s}$$

Η τελευταία σχέση δείχνει τη μέγιστη επιτάχυνση που μπορεί να επιτευχθεί σε μια εφαρμογή που περιλαμβάνει ένα μη παραλληλοποιήσιμο τμήμα. Για παράδειγμα, ένας αλγόριθμος που περιέχει ένα σειριακό κομμάτι του οποίου η εκτέλεση διαρκεί 10% του

συνολικού χρόνου εκτέλεσης, τότε η μέγιστη επιτάχυνση που μπορεί να επιτευχθεί είναι 10, ανεξάρτητα από τον αριθμό των επεξεργαστών που θα χρησιμοποιηθούν.

Ο νόμος του Amdahl που περιγράφτηκε παραπάνω παρέχει ένα μοντέλο για τη σχέση μεταξύ της αναμενόμενης επιτάχυνσης των παράλληλων υλοποιήσεων ενός αλγορίθμου σε σχέση με το σειριακό αλγόριθμο, με την προϋπόθεση ότι το μέγεθος του προβλήματος παραμένει σταθερό. Αυτό σημαίνει ότι το r_s παραμένει επίσης σταθερό και ότι ένα πρόβλημα συνήθως δεν μπορεί να παρουσιάσει ικανοποιητική κλιμάκωση όταν ο αριθμός n των διαθέσιμων επεξεργαστών αυξάνεται, διαπίστωση η οποία εκ πρώτης όψεως περιορίζει την αξία του παράλληλου προγραμματισμού.

Παρόλα αυτά, στην πράξη έχει παρατηρηθεί ότι παράλληλα συστήματα με πάρα πολύ μεγάλο αριθμό επεξεργαστών μπορούν να επιτύχουν σημαντικές επιδόσεις. Ο νόμος του Gustafson [22] περιγράφει αυτό ακριβώς το γεγονός, διαπιστώνοντας ότι κάθε ικανοποιητικά μεγάλο πρόβλημα μπορεί να εκτελεστεί παράλληλα με αποδοτικό τρόπο. Βασικός άξονας στο νόμο του Gustafson είναι η άρση της υπόθεσης του σταθερού μεγέθους ενός δεδομένου προβλήματος. Αντίθετα, προτείνει ότι ο χρόνος εκτέλεσης είναι αυτό που πρέπει να θεωρείται σταθερό, καθώς αυξάνεται ο αριθμός n των επεξεργαστών. Με άλλα λόγια, το μέγεθος του προβλήματος αυξάνεται καθώς αυξάνεται το n. Μια πρώτη προσέγγιση είναι ότι το παράλληλο τμήμα, και όχι το σειριακό, αυξάνεται με την αύξηση του μεγέθους του προβλήματος. Με βάση τις παραπάνω υποθέσεις και αγνοώντας τις όποιες επιβαρύνσεις προκαλεί η παράλληλη επεξεργασία, προκύπτει το μέγεθος της *σχετικής επιτάχυνσης* (*scaled speedup*) *SS* ως εξής:

$$SS_n = \frac{r_s(n) + n \times r_p(n)}{r_s(n) + r_p(n)} = r_s(n) + n \times r_p(n) =$$

$$= r_s(n) + n \times (1 - r_s(n)) = n + r_s(n)(1 - n)$$

Αν θεωρηθεί ότι το $r_s(n)$ μειώνεται καθώς αυξάνεται ο αριθμός του προβλήματος, τότε από την παραπάνω σχέση φαίνεται ότι η επιτάχυνση τείνει στο n, όπως απαιτείται για ένα καλά κλιμακούμενο σύστημα. Με βάση τον παραπάνω ορισμό, μια εφαρμογή θεωρείται καλά κλιμακούμενη, όταν αυξάνοντας τον αριθμό των επεξεργαστών και του μεγέθους του προβλήματος κατά τον ίδιο παράγοντα, ο χρόνος εκτέλεσης παραμένει ο ίδιος.

(β) επιβάρυνση επικοινωνίας και συγχρονισμού

Ένας δεύτερος παράγοντας που περιορίζει την ικανότητα κλιμάκωσης των παράλληλων συστημάτων είναι οι επιβαρύνσεις που εμπεριέχει η παράλληλη επεξεργασία. Οι επιβαρύνσεις αυτές οφείλονται στην επικοινωνία που απαιτείται μεταξύ των εργασιών για συγχρονισμό και για μεταφορά δεδομένων γενικότερα. Καταρχάς, για την επίτευξη της επικοινωνίας χρησιμοποιούνται κύκλοι μηχανής και άλλοι πόροι που θα μπορούσαν να χρησιμοποιηθούν για ωφέλιμους υπολογισμούς. Επιπλέον, συχνά εμφανίζεται το φαινόμενο να παραμένουν κάποιοι επεξεργαστές ανενεργοί, αναμένοντας απαραίτητα δεδομένα για τους υπολογισμούς τους ή σήματα συγχρονισμού. Για παράδειγμα, όταν για να συνεχίσει μια διεργασία τους υπολογισμούς της απαιτείται να λάβει μεγάλο όγκο δεδομένων από μια άλλη διεργασία, ο χρόνος μεταφοράς των δεδομένων ενδέχεται να είναι αρκετά μεγάλος, προκαλώντας σημαντική αύξηση του συνολικού χρόνου εκτέλεσης. Μια τεχνική που περιορίζει την επιβάρυνση σε αυτή την περίπτωση είναι η αλληλοεπικάλυψη υπολογισμών και επικοινωνίας (π.χ. με χρήση μηχανισμών ασύγχρονης επικοινωνίας).

(γ) ανισοκατανομή υπολογιστικού φορτίου

Ένας ακόμη περιοριστικός παράγοντας για την ικανότητα κλιμάκωσης είναι η ανεπαρκής επίτευξη ισορροπίας μεταξύ των υπολογιστικών φορτίων που ανατίθενται στους επεξεργαστές. Αυτό συνεπάγεται ότι κάποιοι επεξεργαστές θα παραμένουν ανενεργοί, περιμένοντας να ολοκληρώσουν την εκτέλεσή τους οι επεξεργαστές στους οποίους έχει δοθεί μεγαλύτερο υπολογιστικό φορτίο. Για αυτό λόγο, ένα βασικό μέλημα κατά την ανάπτυξη παράλληλων εφαρμογών είναι η επίτευξη ισορροπίας στην κατανομή του υπολογιστικού φορτίου (load balancing). Η ισοκατανομή του υπολογιστικού φορτίου σε περιπτώσεις που υπάρχει εκ των προτέρων πληροφορία για το χρόνο εκτέλεσης των παράλληλων εργασιών στους διάφορους επεξεργαστές μπορεί να επιτευχθεί σχετικά εύκολα με διάφορες ευρετικές (heuristic) μεθόδους. Η πιο συνήθης περίπτωση όμως είναι να υπάρχει μεταβλητότητα στο υπολογιστικό φορτίο των διαφόρων εργασιών και να μην είναι δυνατή η εκ των προτέρων γνώση για τους απαιτούμενους χρόνους εκτέλεσης αυτών. Σε αυτές τις περιπτώσεις ενδείκνυται η χρήση του μοντέλου "δεξαμενής εργασιών" (work-pool), σύμφωνα με το οποίο γίνεται κατανομή των εργασιών με βάση το μοντέλο της αίτησης-ανάθεσης.

3 Παράλληλος Αλγόριθμος Υπολογισμού Πρόβλεψης Διάδοσης σε Ραδιοκανάλι με βάση την Τεχνική Ανίχνευσης Ακτίνων

Η ανίχνευση ακτίνων (ray tracing) είναι μια τεχνική που βασίζεται στην αριθμητική προσομοίωση των αρχών της γεωμετρικής οπτικής (Geometrical Optics – GO), μιας διαδεδομένης προσεγγιστικής μεθόδου που εφαρμόζεται για τον υπολογισμό υψίσυχνου ηλεκτρομαγνητικού πεδίου και η οποία στηρίζεται στην κυματική θεωρία διάδοσης [23] και στην Γεωμετρική Θεωρία Περίθλασης (Geometrical Theory of Diffraction – GTD) [24] και την επέκτασή της, την Ομοιόμορφη Γεωμετρική Θεωρία Περίθλασης (Uniform Geometrical Theory of Diffraction – UTD) [25], [26], οι οποίες συμπληρώνουν τη θεωρία της GO. Τα μοντέλα πρόβλεψης διάδοσης που βασίζονται στο ray tracing έχουν σημαντικό ρόλο στη σχεδίαση δικτύων ασύρματων επικοινωνιών, καθώς λαμβάνουν υπόψη ποικίλα φυσικά φαινόμενα, όπως ανάκλαση, περίθλαση και διάχυση. Το βασικότερο πρακτικό μειονέκτημα αυτών είναι ότι μπορούν να γίνουν εξαιρετικά υπολογιστικά σε υπολογιστικούς χρόνους, καθώς αυξάνονται τα απαιτούμενα επίπεδα ακρίβειας και η εκάστοτε υπό μελέτη περιοχή. Στο παρόν κεφάλαιο παρουσιάζεται μια παράλληλη εκδοχή ενός ray tracing αλγορίθμου πρόβλεψης διάδοσης σε ραδιοκανάλι, ο οποίος βασίζεται στην ηλεκτρομαγνητική θεωρία των ειδώλων. Η υλοποίηση του παράλληλου αλγορίθμου βασίζεται στη διεπαφή προώθησης μηνυμάτων. Η αποσύνθεση του προβλήματος στηρίζεται στον σταδιακό υπολογισμό του δέντρου των ειδώλων που εμπλέκεται στον αλγόριθμο, ενώ για την παράλληλη εκτέλεση χρησιμοποιούνται τεχνικές δυναμικής κατανομής φορτίου, οι οποίες στηρίζονται στα υπολογιστικά παραδείγματα του "αφέντη-εργάτη" (master-worker) και της "δεξαμενής εργασιών" (work-pool). Στα πλαίσια της διατριβής πραγματοποιήθηκε πειραματική μελέτη και ανάλυση της απόδοσης του αλγορίθμου για διαφορετικά μεγέθη

προβλημάτων, η οποία δείχνει ότι η παράλληλη υλοποίηση λειτουργεί με αποδοτικό τρόπο. Επιπλέον, γίνεται μελέτη της επίδρασης διαφόρων τεχνικών κατανομής υπολογιστικού φορτίου που έχουν προταθεί στη βιβλιογραφία.

3.1 Εισαγωγή

Μια από τις βασικότερες απαιτήσεις κατά τη σχεδίαση των σημερινών συστημάτων ασύρματων επικοινωνιών είναι η ύπαρξη ακριβούς πρόβλεψης για τη διάδοση των ραδιοκυμάτων. Οι βασικοί περιορισμοί σε κάθε σύστημα ασύρματων επικοινωνιών οφείλονται στα χαρακτηριστικά του ραδιοκαναλιού. Τα μοντέλα διάδοσης σε ραδιοκανάλι είναι απαραίτητα για την ανάλυση και προσομοίωση ασύρματων συστημάτων, ενώ η γνώση των χαρακτηριστικών του ραδιοκαναλιού είναι απαραίτητη για την ανάπτυξη και εγκατάσταση κάθε ασύρματου συστήματος. Τα μοντέλα ραδιοκαναλιού χρησιμοποιούνται για υποστήριξη κατά την εύρεση μεθόδων που αντιμετωπίζουν τις ατέλειες του ραδιοκαναλιού, όπως για παράδειγμα κατά το σχεδιασμό equalizer ή κατά την επιλογή συστήματος μονού ή πολλαπλού φέροντος. Τα απλά μοντέλα διάδοσης έχουν ήδη σημαντικό ρόλο στην ανάπτυξη, το σχεδιασμό και την εγκατάσταση συστημάτων κινητών επικοινωνιών, όπου η κάλυψη είναι η βασική επιδίωξη. Στις μέρες μας, όπου η κινητή τηλεφωνία χρησιμοποιείται από ένα μεγάλο ποσοστό του πληθυσμού, στις περισσότερες αστικές περιοχές η χωρητικότητα αντικαθιστά την κάλυψη ως τον σημαντικότερο παράγοντα και οι απαιτήσεις για ακριβή πρόβλεψη διάδοσης γίνονται μεγαλύτερες. Επιπλέον, με την ανάπτυξη διαφόρων προηγμένων συστημάτων επικοινωνίας, όπως για παράδειγμα τα συστήματα πολλαπλής εισόδου πολλαπλής εξόδου (multiple input, multiple output – MIMO), η σε βάθος μελέτη των χαρακτηριστικών του ραδιοκαναλιού σε συγκεκριμένες περιοχές γίνεται ολοένα και πιο σημαντική.

Οι τεχνικές ray tracing παράγουν ντετερμινιστικά μοντέλα καναλιών, η λειτουργία των οποίων στηρίζεται στην επεξεργασία συγκεκριμένων περιβαλλόντων που ορίζονται από το χρήστη. Το βασικό μειονέκτημα των ντετερμινιστικών μοντέλων του ραδιοκαναλιού είναι οι εξαιρετικά αυξημένες υπολογιστικές απαιτήσεις σε σχέση με τα αντίστοιχα στατιστικά ([27], [28]). Για την αντιμετώπιση των απαιτήσεων αυτών, διάφορες προσεγγίσεις έχουν προταθεί στη βιβλιογραφία. Τυπικά χρησιμοποιούμενες τεχνικές είναι η προ-επεξεργασία της βάσης δεδομένων, ο έλεγχος ορατότητας, ο προ-υπολογισμός του δέντρου ειδώλων, καθώς και διάφορες άλλες προσεγγιστικές τεχνικές που επιτυγχάνουν τη μείωση του απαιτούμενου χρόνου εκτέλεσης σε βάρος της ακρίβειας της προσέγγισης. Μια διαφορετική

προσέγγιση που μπορεί να παρέχει αποδοτική λύση στο παραπάνω πρόβλημα είναι η παράλληλη επεξεργασία. Ένα πλεονέκτημα της συγκεκριμένης προσέγγισης είναι ότι δεν λειτουργεί σε βάρος της ακρίβειας των υπολογισμών. Το βασικότερο πλεονέκτημα είναι ότι μια παράλληλη υλοποίηση έχει τη δυνατότητα κλιμάκωσης καθώς αυξάνονται οι διαθέσιμοι υπολογιστικοί πόροι, παρέχοντας με αυτόν τον τρόπο μια αποδοτική λύση για προβλήματα με ολοένα αυξανόμενα μεγέθη. Το κύριο μειονέκτημα της προσέγγισης είναι ότι η ανάπτυξη μιας παράλληλης υλοποίησης είναι τυπικά πολύπλοκη, καθώς ένα πλήθος αλληλοσχετιζόμενων παραγόντων που επηρεάζουν τη συνολική απόδοση πρέπει να ληφθούν υπόψη.

Στο παρόν κεφάλαιο της διατριβής παρουσιάζεται η παραλληλοποίηση ενός ray tracing μοντέλου για την πρόβλεψη διάδοσης σε ραδιοκανάλι με βάση την ηλεκτρομαγνητική θεωρία των ειδώλων ([29], [30], [31]). Το μοντέλο αυτό βασίζεται στη δημιουργία ενός δέντρου ειδώλων του σταθμού βάσης, θεωρώντας όλους τους τοίχους και τα εμπόδια σε μια περιοχή ως πιθανούς ανακλαστές. Έμφαση δίνεται στη μοντελοποίηση περιβαλλόντων για συστήματα σταθερής ασύρματης πρόσβασης (fixed wireless access – FWA) και, ως εκ τούτου, λαμβάνεται υπόψη η περίθλαση από κορυφές κτιρίων ως ένας από τους κυρίαρχους μηχανισμούς διάδοσης.

Το βασικό μέλημα κατά τη σχεδίαση και ανάπτυξη μιας παράλληλης υλοποίησης είναι η κλιμάκωση και η αποδοτικότητα. Για την επίτευξη αυτών των στόχων, ένας αριθμός από ζητήματα λήφθηκε υπόψη, όπως η διάσπαση των εργασιών, η μείωση του κόστους επικοινωνίας, η κατανομή των εργασιών και η ισοκατανομή του υπολογιστικού φορτίου. Η φύση του προβλήματος πρόβλεψης θέτει συγκεκριμένες δυσκολίες στην ισοκατανομή του φορτίου, καθώς τυπικά μια περιοχή μπορεί να αποτελείται από μία ή περισσότερες υπο-περιοχές με αυξημένη πυκνότητα εμποδίων (hot spots), οι οποίες απαιτούν πολύ περισσότερους υπολογισμούς σε σχέση με άλλες. Για την αντιμετώπιση αυτού του προβλήματος, η προσέγγιση που υιοθετήθηκε για την παραλληλοποίηση του προβλήματος στηρίζεται στο διαμερισμό της κατασκευής του δέντρου των ειδώλων, οδηγώντας σε μικρού μεγέθους επιμέρους εργασίες, οι οποίες δημιουργούνται δυναμικά. Η κατανομή των εργασιών βασίζεται στα υπολογιστικά παραδείγματα master-worker και work-pool, ενώ εξετάζεται η επίδραση διαφόρων σχημάτων κατανομής που έχουν προταθεί στη βιβλιογραφία. Η υλοποίηση βασίζεται στις προδιαγραφές MPI, έτσι ώστε να είναι δυνατή η εύκολη μεταφορά της υλοποίησης σε διαφορετικές αρχιτεκτονικές. Διάφορα πειράματα πραγματοποιήθηκαν για προβλήματα πρόβλεψης διάδοσης διαφορετικών μεγεθών. Για την αποτίμηση της απόδοσης της παράλληλης υλοποίησης χρησιμοποιήθηκαν μεγέθη όπως η

επιτάχυνση, ο βαθμός αξιοποίησης πόρων και το ποσοστό επέκτασης υπολογιστικού φόρτου. Τα αποτελέσματα που προέκυψαν δείχνουν ότι η προτεινόμενη υλοποίηση παρουσιάζει καλή κλιμάκωση για διαφορετικά μεγέθη προβλημάτων.

Στη συνέχεια περιγράφεται συνοπτικά ο σειριακός ray tracing αλγόριθμος στον οποίο στηρίζεται η παράλληλη υλοποίηση που παρουσιάζεται στο παρόν κεφάλαιο και περιγράφεται αναλυτικά η παράλληλη υλοποίηση. Τέλος, παρουσιάζονται πειραματικά αποτελέσματα από μια σειρά πειραμάτων που πραγματοποιήθηκαν για διαφορετικές περιοχές και διαφορετικές τάξεις ανακλάσεων και γίνεται ανάλυση της απόδοσης του παράλληλου αλγορίθμου.

3.2 Σχετικές Εργασίες

Γενικά, έχει αναγνωρισθεί ότι τα μοντέλα πρόβλεψης διάδοσης σε ραδιοκανάλι που βασίζονται σε τεχνικές ray tracing είναι εξαιρετικά απαιτητικά σε υπολογιστικούς χρόνους σε σχέση με τα αντίστοιχα στατιστικά μοντέλα. Για να μειωθεί ο απαιτούμενος υπολογιστικός χρόνος, διάφορες προσεγγίσεις έχουν προταθεί στη βιβλιογραφία. Ανάμεσα σε αυτές τις προσεγγίσεις αναφέρονται τεχνικές για απλοποίηση του ίχνους των κτιρίων στη βάση δεδομένων [32], καθώς και διάφορες άλλες προσεγγιστικές μέθοδοι. Για παράδειγμα, στο [33] η πολύοδη διάδοση δεν λαμβάνεται υπόψη όταν η απόσταση μεταξύ του κινητού και του σταθμού βάσης ξεπερνάει μια συγκεκριμένη τιμή κατωφλίου. Στο [34] προτείνονται προοδευτικές προσεγγιστικές τεχνικές, σύμφωνα με τις οποίες τα ενδιάμεσα αποτελέσματα πρόβλεψης διάδοσης συνεχώς ανατροφοδοτούνται στο χρήστη, ενώ το υπολογιστικό φορτίο ρυθμίζεται δυναμικά κατά τη διαδικασία πρόβλεψης. Ένα κοινό μειονέκτημα των παραπάνω προσεγγίσεων είναι ότι η μείωση του απαιτούμενου χρόνου εκτέλεσης επιτυγχάνεται σε βάρος της ακρίβειας του μοντέλου πρόβλεψης. Διάφορες μέθοδοι που δεν επηρεάζουν την ακρίβεια έχουν προταθεί, όπως προ-επεξεργασία της βάσης δεδομένων με ομαδοποίηση αντικειμένων, έλεγχος ορατότητας (visibility determination) και τεχνικές προ-υπολογισμού του δέντρου περίθλασης [28]. Οι μέθοδοι αυτές επιτυγχάνουν μεν τη μείωση του χρόνου εκτέλεσης, παρόλα αυτά δεν παρέχουν επαρκή λύση για αυξανόμενο μέγεθος προβλημάτων.

Η παράλληλη επεξεργασία έχει αναγνωριστεί ως μια λύση που έχει τη δυνατότητα να αντιμετωπίσει τις απαιτήσεις των ray tracing μοντέλων πρόβλεψης διάδοσης, καθώς η ικανότητα κλιμάκωσης με την αύξηση των διαθέσιμων υπολογιστικών πόρων αποδεικνύεται αποτελεσματική για αυξανόμενο μέγεθος προβλημάτων. Διάφορες

προσεγγίσεις έχουν προταθεί για παράλληλη εκτέλεση ray tracing μοντέλων για πρόβλεψη διάδοσης. Στα [35] και [36] προτείνεται ένα παράλληλο μοντέλο σε outdoor μικροκυψέλες για τον υπερ-υπολογιστή Cray T3E, το οποίο στηρίζεται στο MPI. Αν και ο υπολογιστικός χρόνος μειώνεται σε σχέση με τη σειριακή εκτέλεση, το μοντέλο δεν επιτυγχάνει καλή κλιμάκωση. Στην εργασία που παρουσιάζεται στα [37], [38] και [39], αναπτύχθηκε ένα παράλληλο τρισδιάστατο μοντέλο διάδοσης που εκτελείται σε μια Beowulf συστοιχία υπολογιστών με 200 κόμβους. Το μοντέλο αυτό έχει υλοποιηθεί επίσης με βάση το MPI και χρησιμοποιήθηκε για την εύρεση των καθολικά βέλτιστων θέσεων τοποθέτησης εκπομπών σε indoor συστήματα ασύρματων επικοινωνιών. Κάθε επεξεργαστής έχει τοπικά ένα πλήρες αντίγραφο του κτιρίου και λαμβάνει υπόψη μόνο τις ανακλάσεις και τις μεταδόσεις. Μια άλλη προσέγγιση για την παράλληλη εκτέλεση μοντέλου πρόβλεψης διάδοσης σε ένα δίκτυο σταθμών εργασίας έχει προταθεί στο [40]. Η προσέγγιση αυτή στηρίζεται σε μια υποδομή με βάση τα υπολογιστικά μοντέλα του αφέντη-εργάτη και των παράλληλων φάσεων (phase-parallel). Η συνολική επεξεργασία διαχωρίζεται σε ένα σύνολο μικρότερων εργασιών είτε σε επίπεδο ακτίνων είτε σε επίπεδο ακτινοβολούντων σημείων, οι οποίες κατανέμονται στους κόμβους με χρήση δυναμικών σχημάτων. Η συγκεκριμένη παράλληλη υλοποίηση επιτυγχάνει σχεδόν γραμμική επιτάχυνση για διαφορετικά μεγέθη περιοχών και για αυξανόμενο αριθμό χρησιμοποιούμενων κόμβων. Τέλος, ένα ακόμη παράλληλο τρισδιάστατο μοντέλο παρουσιάζεται στο [41]. Το συνολικό υπολογιστικό φορτίο διαμοιράζεται ανάλογα με τον αριθμό των ακτίνων που εκτοξεύονται και οι εργασίες που σχηματίζονται κατανέμονται στατικά στους διαθέσιμους κόμβους με τυχαίο τρόπο. Όλες οι παράλληλες προσεγγίσεις που προαναφέρθηκαν σχετίζονται με αλγόριθμους ray tracing που βασίζονται στη μέθοδο Εκτόξευσης και Αναπήδησης Ακτίνων (Shoot and Bouncing Ray – SBR) [42], [43]. Με βάση αυτή τη μέθοδο, οι ακτίνες εκτοξεύονται ομοιοκατευθυντικά από την πηγή με μια συγκεκριμένη, αρκετά μικρή γωνία διαχωρισμού, έτσι ώστε να μην παραλείπονται όψεις κτιρίων σε μεγάλη απόσταση από τον πομπό. Η τεχνική SBR είναι εγγενώς παραλληλοποιήσιμη, καθώς οι ακτίνες που εκτοξεύονται από τον πομπό είναι ανεξάρτητες μεταξύ τους, επιτρέποντας το μοντέλο του παράλληλου προγραμματισμού να είναι άμεσα εφαρμόσιμο στον κώδικα SBR.

Το παράλληλο ray tracing μοντέλο που αναπτύχθηκε στα πλαίσια της διατριβής βασίζεται στην ηλεκτρομαγνητική θεωρία των ειδώλων. Το μοντέλο λαμβάνει υπόψη όλους τους τοίχους και τα εμπόδια ως πιθανούς ανακλαστές και υπολογίζει τις θέσεις των ειδώλων του σταθμού βάσης. Η συγκεκριμένη τεχνική λειτουργεί δημιουργώντας ένα δέντρο ειδώλων για κάθε θέση του σταθμού βάσης, το οποίο στη συνέχεια χρησιμοποιείται για τον

υπολογισμό των ισχύων των ακτίνων που λαμβάνονται σε κάθε σημείο της περιοχής. Το δέντρο των ειδώλων δημιουργείται με αλυσιδωτό τρόπο, πράγμα που δυσχεραίνει το διαχωρισμό των εργασιών και την ισοκατανομή του υπολογιστικού φορτίου ανάμεσα στους επεξεργαστές. Μια προσέγγιση για την παραλληλοποίηση αυτού του μοντέλου παρουσιάζεται στο [28], η οποία βασίζεται στο γεγονός ότι το ίδιο δέντρο ειδώλων μπορεί να χρησιμοποιηθεί για τον υπολογισμό των ακτίνων που καταφθάνουν σε οποιοδήποτε σημείο μιας δεδομένης περιοχής. Έτσι, το υπολογιστικό φορτίο διαμοιράζεται τεμαχίζοντας τον υπολογισμό των ακτίνων σε διαφορετικές εργασίες, κάθε μια από τις οποίες αναλαμβάνει ένα συγκεκριμένο αριθμό σημείων-δεκτών. Το δέντρο ειδώλων δημιουργείται στην αρχή του αλγορίθμου με σειριακό τρόπο, πράγμα που περιορίζει την κλιμάκωση και την αποδοτικότητα της παράλληλης επεξεργασίας. Το παράλληλο μοντέλο που αναπτύχθηκε στα πλαίσια της παρούσας διατριβής στηρίζεται στο διαμερισμό της κατασκευής του δέντρου ειδώλων, πράγμα που παρουσιάζει το διπλό πλεονέκτημα ελαχιστοποίησης των σειριακών τμημάτων του αλγορίθμου και επίτευξης ισοκατανομής του υπολογιστικού φορτίου.

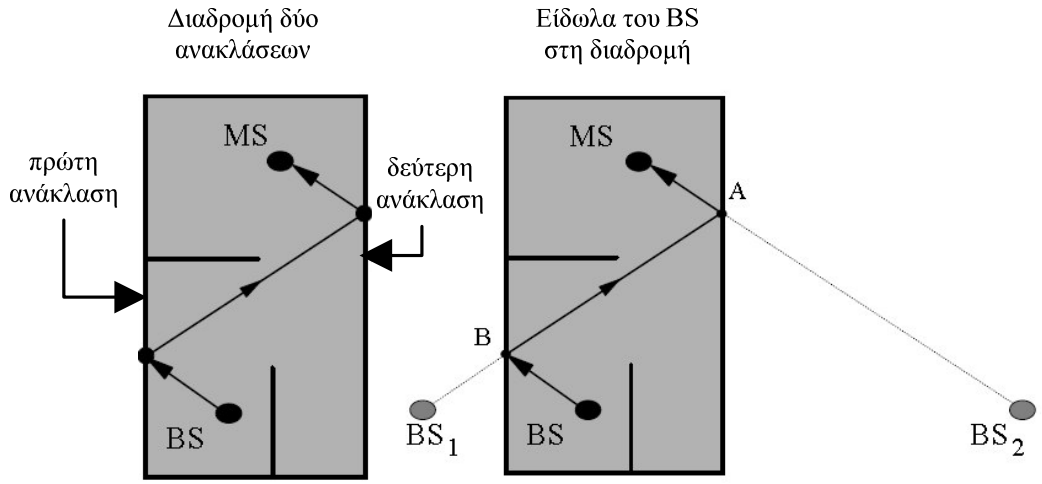
3.3 Σειριακό Μοντέλο Ray Tracing

Το σειριακό μοντέλο ray tracing, στο οποίο στηρίζεται η παράλληλη υλοποίηση που παρουσιάζεται στο παρόν κεφάλαιο της διατριβής, δίνει έμφαση στη μοντελοποίηση περιβαλλόντων για συστήματα FWA και λαμβάνει πλήρως υπόψη την περίθλαση από οροφές κτιρίων, τόσο πριν όσο και μετά τις ανακλάσεις, καθώς πρόκειται για τον κυρίαρχο μηχανισμό διάδοσης όταν οι κεραίες τοποθετούνται στις οροφές κτιρίων ή σε μεγάλο ύψος σε εξωτερικούς τοίχους. Το μοντέλο λειτουργεί παίρνοντας ως είσοδο βάσεις δεδομένων περιοχών που περιέχουν πληροφορία για τρισδιάστατα κτίρια και φυλλώματα και λαμβάνει υπόψη τους εξής μηχανισμούς διάδοσης [44], [45]:

- διάδοση κενού χώρου
- διάδοση μέσα από τοίχους
- ανακλάσεις από τους τοίχους των κτιρίων
- περίθλαση από τις οροφές των κτιρίων
- εξασθένιση που προκαλείται σε κάθε ακτίνα όταν αυτή διέρχεται μέσα από φυλλώματα

3.3.1 Βασική Λειτουργία

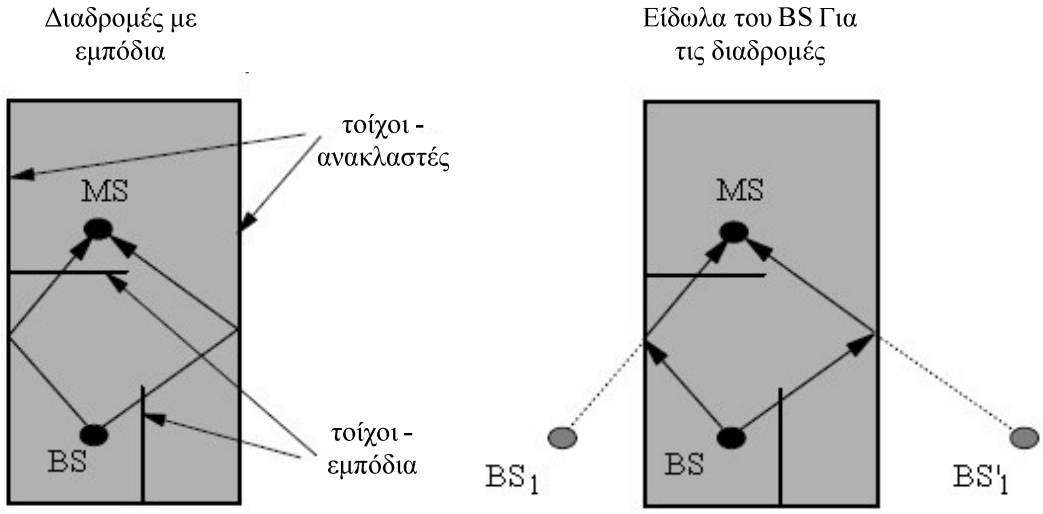
Η βασική ιδέα της εν λόγω τεχνικής φαίνεται στο Σχήμα 9 [46].



Σχήμα 9. Αναπαράσταση διαδρομής λόγω ανακλάσεων στον αλγόριθμο

Στην αριστερή πλευρά του σχήματος φαίνεται μια διαδρομή δύο ανακλάσεων σε ένα απλό περιβάλλον ενός δωματίου. Στην δεξιά πλευρά φαίνεται η ίδια διαδρομή, όπως αυτή αναπαρίσταται από τον αλγόριθμο. Ο σταθμός βάσης (base station – BS) αρχικά ανακλάται από τον πρώτο τοίχο, δημιουργώντας το είδωλο BS_1. Στη συνέχεια το είδωλο αυτό ανακλάται από τον δεύτερο τοίχο, δημιουργώντας το νέο είδωλο BS_2. Η διαδρομή δημιουργείται με την αντίστροφη σειρά. Αρχικά σχηματίζεται μια γραμμή μεταξύ της θέσης του κινητού τερματικού (mobile station – MS) και το BS_2, βρίσκοντας το σημείο επαφής (σημείο Α). Στη συνέχεια, δημιουργείται μια ευθεία από το σημείο Α στο BS_1 και βρίσκεται το δεύτερο σημείο ανάκλασης (σημείο Β). Από το σημείο Β με την ίδια διαδικασία φτάνουμε στον BS, οπότε σε αυτήν την περίπτωση υπάρχει μια διαδρομή που δημιουργείται αποκλειστικά μέσω ανάκλασης. Σε κάθε περίπτωση, ο αλγόριθμος ελέγχει αν ένα είδωλο οδηγεί σε έγκυρη διαδρομή (Σχήμα 10) και στη συνέχεια ελέγχει για τοίχους ή άλλα εμπόδια που μπορεί να παρεμβάλλονται στη διαδρομή της ακτίνας. Σε αυτήν την περίπτωση μια απώλεια διάδοσης προστίθεται στην ακτίνα, ανάλογα με το συντελεστή διάδοσης του εμποδίου. Το Σχήμα 10 δείχνει δύο διαδρομές, στη μία από τις οποίες υπάρχει διάδοση μέσω τοίχου πριν το MS, ενώ στην άλλη υπάρχει διάδοση μέσω τοίχου πριν μια ανάκλαση. Όταν ο αριθμός των διαδόσεων μέσω τοίχου υπερβαίνει έναν προκαθορισμένο αριθμό, ο

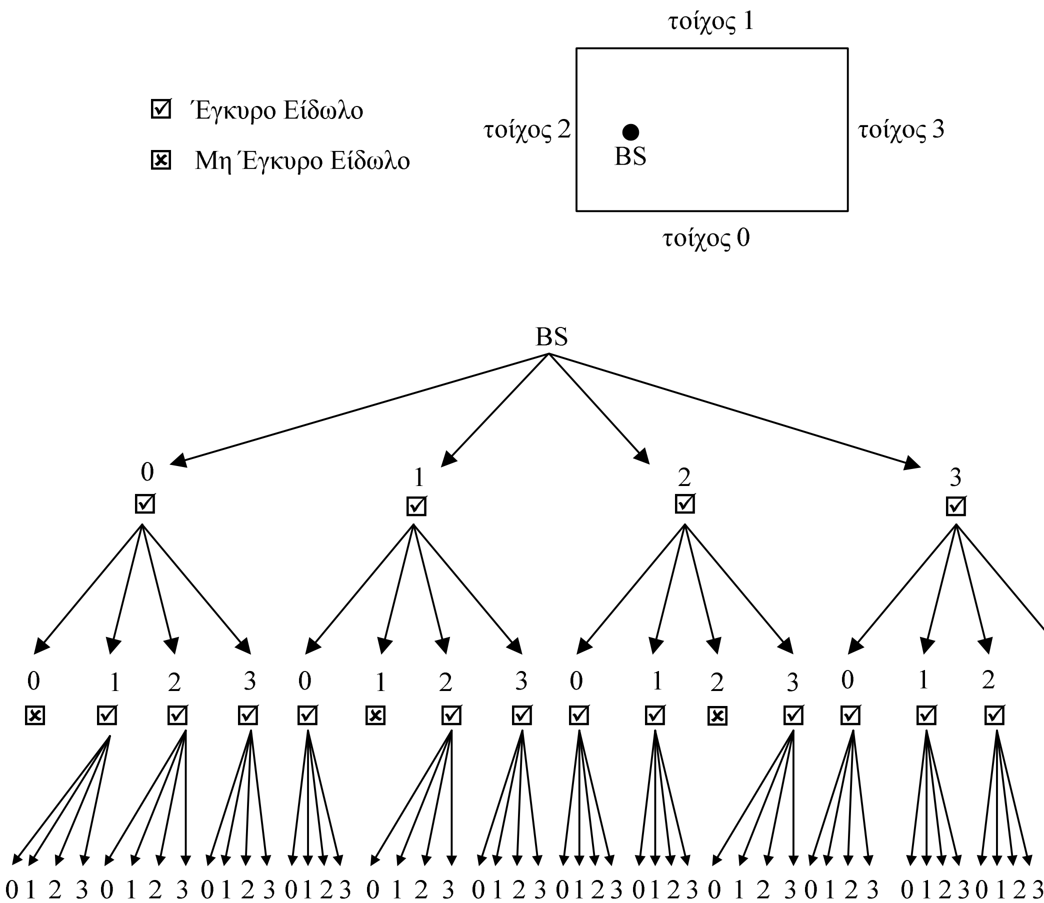
αλγόριθμος σταματάει την ανίχνευση της ακτίνας, ενώ αν βρίσκεται σε στάδιο υπολογισμού ειδώλων, δεν χρησιμοποιεί το συγκεκριμένο είδωλο για τον υπολογισμό ειδώλων υψηλότερης τάξης.



Σχήμα 10. Αναπαράσταση διαδρομών με εμπόδια στον αλγόριθμο

3.3.2 Δέντρο Ειδώλων

Αρχικά τα είδωλα του BS υπολογίζονται για όλους τους τοίχους-ανακλαστές που είναι απευθείας ορατοί από το σταθμό βάσης και για όσους δεν ξεπερνούν το ανώτατο όριο διαδόσεων μέσω τοίχων. Στη συνέχεια αυτά τα είδωλα δημιουργούν με τη σειρά τους τα δικά τους είδωλα με παρόμοιο τρόπο με αυτόν του BS. Η διαδικασία αυτή συνεχίζεται και νέα είδωλα δημιουργούνται με ιεραρχικό τρόπο, μέχρι τον μέγιστο αριθμό ανακλάσεων που ορίζεται από το χρήστη. Σε αυτό το σημείο σημειώνεται ότι το δέντρο ειδώλων που σχηματίζεται με αυτόν τον τρόπο είναι ανεξάρτητο από τις θέσεις των MS και δημιουργώντας το μία μόνο φορά, αυτό μπορεί να χρησιμοποιηθεί για όλες τις θέσεις των MS στην δεδομένη περιοχή. Στο Σχήμα 11 φαίνεται ένα απλό παράδειγμα σχηματισμού δέντρου ειδώλων για ένα ορθογώνιο δωμάτιο.



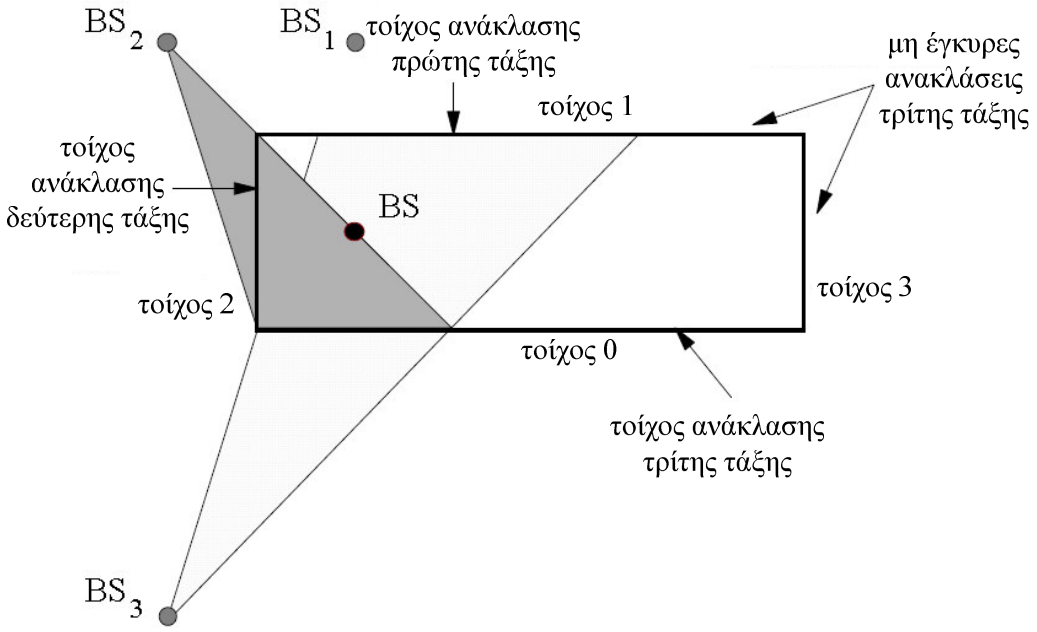
Σχήμα 11. Απλό παράδειγμα δέντρου ειδώλων

3.3.3 Ζώνες Φωτισμού Ειδώλων

Κάθε είδωλο συσχετίζεται με μια συγκεκριμένη “ζώνη φωτισμού” (illumination zone), την περιοχή για την οποία το είδωλο αυτό μπορεί να δώσει έγκυρη διαδρομή. Εκτός από τη θέση του ειδώλου και τον τοίχο στον οποίο αυτό αντιστοιχεί, υπολογίζεται και αποθηκεύεται επιπλέον και η ζώνη φωτισμού αυτού. Μόνο τοίχοι και εμπόδια που βρίσκονται εντός της ζώνης φωτισμού του ειδώλου μπορούν να χρησιμοποιηθούν για τη δημιουργία νέων ειδώλων μεγαλύτερης τάξης, ενώ επιπλέον τα νέα είδωλα δεν αντιστοιχούν σε ολόκληρο τον τοίχο, αλλά μόνο στο τμήμα αυτού που φωτίζεται από το είδωλο-γονέα. Οι ζώνες φωτισμού των ειδώλων μεγαλύτερης τάξης έχουν την τάση να γίνονται στενότερες, πράγμα που δεν επιτρέπει στον αριθμό των ειδώλων να αυξάνονται εκθετικά καθώς αυξάνεται η τάξη ανάκλασης. Επιπλέον, με βάση την τεχνική αυτή

μπορούν να υπολογιστούν ευκολότερα τα είδωλα που φωτίζουν μια συγκεκριμένη θέση: αντί να γίνεται ανίχνευση κάθε πιθανής ακτίνας πίσω προς τον BS, βρίσκονται τα σημεία τομής με τους αντίστοιχους τοίχους και για να εξεταστεί η εγκυρότητά τους, αρκεί να ελεγχθεί το αν η θέση του MS βρίσκεται μέσα στη ζώνη φωτισμού του συγκεκριμένου ειδώλου.

Η βασική αρχή λειτουργίας της εν λόγω τεχνικής φαίνεται στο Σχήμα 12.



Σχήμα 12. Τεχνική Ζωνών Φωτισμού

3.3.4 Σειριακός Αλγόριθμος

Τα κύρια βήματα του αλγορίθμου είναι τα ακόλουθα:

- Φόρτωμα και επεξεργασία της βάσης δεδομένων της υπό μελέτη περιοχής: η βάση δεδομένων που περιέχει όλα τα δεδομένα για το έδαφος, τα φυλλώματα και τα κτίρια φορτώνεται στην κεντρική μνήμη. Όταν γίνεται το φόρτωμα, πραγματοποιείται και προ-επεξεργασία της βάσης ώστε να γίνει πιο αποδοτική η χρήση της από το μοντέλο.
- Δημιουργία του δέντρου ειδώλων του σταθμού βάσης: για μια δεδομένη θέση του BS, ακολουθείται η διαδικασία που περιγράφτηκε στις προηγούμενες παραγράφους για τον υπολογισμό των ειδώλων μέχρι και την τάξη ανάκλασης

που έχει οριστεί από το χρήστη. Κατά τη διαδικασία αυτή λαμβάνονται υπόψη τα ύψη των τοίχων-ανακλαστών και του BS, καθώς και η πραγματική θέση, ο προσανατολισμός και το μέγεθος κάθε τοίχου, έτσι ώστε να παραχθούν μόνο τα είδωλα που δίνουν έγκυρες διαδρομές.

- Ανίχνευση Διαδρομών: μετά τη δημιουργία του δέντρου ειδώλων του BS, για κάθε σημείο MS υπολογίζονται τα ύψη των σημείων της περιοχής που βρίσκονται στη διαδρομή μεταξύ της κεραίας του BS και της κεραίας του MS. Με αυτόν τον τρόπο δημιουργείται ένα κατακόρυφο προφίλ κατά μήκος της διαδρομής, με βάση το οποίο ελέγχεται αν υπάρχει διαδρομή line-of-sight (LOS) μεταξύ των δύο κεραιών. Αν υπάρχουν εμπόδια στη διαδρομή, υπολογίζονται οι ακριβείς συντεταγμένες των σημείων περίθλασης στα κτίρια και στο έδαφος. Στην περίπτωση που η απευθείας διαδρομή περνάει μέσα από φυλλώματα, υπολογίζονται τα σημεία τομής της ακτίνας με τα φυλλώματα και στη συνέχεια το μήκος της διαδρομής μέσα στα φυλλώματα. Επίσης, ανιχνεύονται οι ακτίνες που διαδίδονται πάνω από φυλλώματα, με περίθλαση στις κορυφές των δέντρων. Με παρόμοιο τρόπο ανιχνεύονται οι ακτίνες που προέρχονται από ανάκλαση.
- Υπολογισμός πεδίων λαμβάνοντας υπόψη πιθανά εμπόδια στη ζώνη Fresnel: αφού βρεθεί η ακριβής γεωμετρία μιας ακτίνας, πραγματοποιείται υπολογισμός της ισχύος του πεδίου αυτής. Υπολογίζονται οι γωνίες άφιξης για τις κεραίες στον BS και στο MS και τα αντίστοιχα κέρδη των κεραιών θεωρούνται κατά τον υπολογισμό των πεδίων. Το μοντέλο υπολογίζει τους συντελεστές ανάκλασης και περίθλασης κατά μήκος κάθε ακτίνας σαν συνάρτηση των γωνιών άφιξης και αναχώρησης. Η εξασθένιση λόγω διάδοσης μέσα από φυλλώματα λαμβάνεται υπόψη ως συνάρτηση του μήκους της διαδρομής μέσα στο φύλλωμα. Οι υπολογισμοί πεδίου πραγματοποιούνται με βάση τις αρχές της UTD και της GO, ενώ μεταξύ των διαφόρων φαινομένων που συμβαίνουν στις ακτίνες, θεωρούνται απώλειες ελευθέρου χώρου. Το μοντέλο λαμβάνει υπόψη και την περίπτωση που υπάρχουν εμπόδια κοντά στη διαδρομή LOS, τα οποία μπορούν να επηρεάσουν τη λήψη προκαλώντας μια επιπλέον εξασθένιση σε αυτήν του ελευθέρου χώρου, ακόμη αν και αν η διαδρομή LOS δεν εμποδίζεται. Το μοντέλο προσθέτει μια επιπλέον εξασθένιση όταν εμποδίζεται η πρώτη ζώνη Fresnel. Κάθε εξασθένιση λόγω εμποδίου στην πρώτη ζώνη Fresnel θεωρείται ανεξάρτητα από τα υπόλοιπα εμπόδια. Συνολικά, η απώλεια διάδοσης μιας ακτίνας εξαρτάται από την εξασθένιση λόγω της συνολικής απόστασης που διένυσε η ακτίνα και από τις

ανακλάσεις, περιθλάσεις, τις μερικές παρεμποδίσεις των ζωνών Fresnel και από την εξασθένιση λόγω διάδοσης μέσω φυλλωμάτων.

- Χαρακτηρισμός καναλιού: αφού ολοκληρωθεί η επεξεργασία όλων των ειδώλων και ανιχνευθούν οι έγκυρες διαδρομές για μια συγκεκριμένη θέση, υπολογίζονται οι λαμβανόμενες ισχύες και οι χρονικές καθυστερήσεις και δημιουργείται το προφίλ του ραδιοκαναλιού, με βάση το οποίο υπολογίζεται η συνολική λαμβανόμενη ισχύς για κάθε σημείο.

3.4 Παράλληλο Μοντέλο Ray Tracing

Από την περιγραφή που προηγήθηκε στην προηγούμενη παράγραφο προκύπτει ότι το μοντέλο ray tracing που βασίζεται στην ηλεκτρομαγνητική θεωρία των ειδώλων δεν είναι εγγενώς παραλληλοποιήσιμο. Το κύριο εμπόδιο είναι ότι αυτό στηρίζεται στη δημιουργία ενός δέντρου ειδώλων, η παραλληλοποίηση της οποίας δεν είναι προφανής.

Μια προσέγγιση στην παραλληλοποίηση του αλγορίθμου είναι η σειριακή δημιουργία του δέντρου και στη συνέχεια ο παράλληλος υπολογισμός των ακτίνων στα σημεία-δέκτες της περιοχής, όπως προτείνεται στο [28]. Αν α είναι το ποσοστό της επεξεργασίας που αντιστοιχεί στη δημιουργία του δέντρου, τότε, όπως υποδεικνύεται από το νόμο του Amdahl [21], η μέγιστη επιτάχυνση S_n που μπορεί να επιτευχθεί χρησιμοποιώντας n επεξεργαστές είναι $S_n = n/[1+(n-1)a]$. Αν θεωρήσουμε έναν πολύ μεγάλο αριθμό επεξεργαστών, η μέγιστη επιτάχυνση που μπορεί να επιτευχθεί είναι $\lim_{n\to\infty}(S_n)=1/\alpha$. Στην προκειμένη περίπτωση, το α ενδέχεται να γίνει αρκετά μεγάλο (π.χ. στο [28] η τιμή του α είναι 13%, επομένως η μέγιστη επιτάχυνση που μπορεί να επιτευχθεί είναι μικρότερη από 8), πράγμα που περιορίζει σε μεγάλο βαθμό την κλιμάκωση της συγκεκριμένης προσέγγισης.

Μια εναλλακτική προσέγγιση θα ήταν η παραλληλοποίηση με βάση τεχνικές αποσύνθεσης δεδομένων (data decomposition). Η περιοχή μπορεί να διασπαστεί σε μικρότερες υπο-περιοχές και σε κάθε επεξεργαστή να ανατίθεται μία από αυτές, επομένως κάθε επεξεργαστής δημιουργεί το δέντρο ειδώλων για τα εμπόδια που υπάρχουν στην υπο-περιοχή που έχει ανατεθεί σε αυτόν. Παρόλα αυτά, η μέθοδος αυτή δεν θεωρείται αποδοτική, καθώς τα δεδομένα των ειδώλων που έχουν υπολογιστεί από έναν συγκεκριμένο επεξεργαστή χρειάζεται να γίνουν διαθέσιμα σε όλους τους υπόλοιπους για τον υπολογισμό

των ειδώλων μεγαλύτερης τάξης, πράγμα που έχει ως αποτέλεσμα τη μεταφορά μεγάλου όγκου δεδομένων και συνεπακόλουθα μεγάλο κόστος επικοινωνίας.

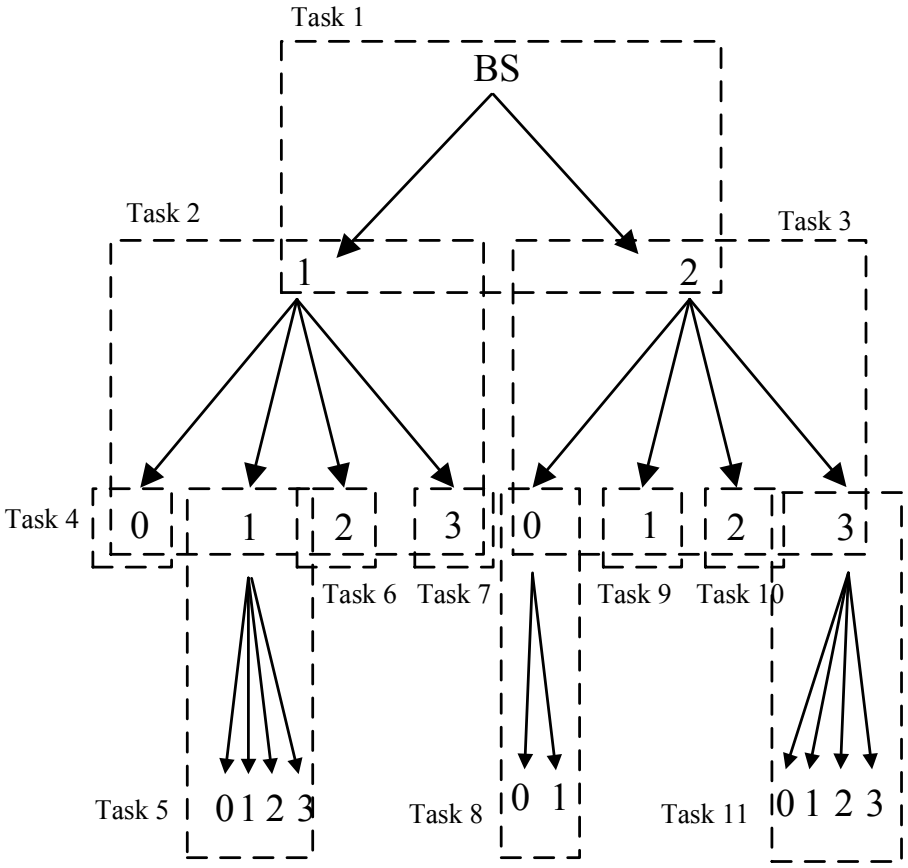
Η προσέγγιση που υιοθετήθηκε στο μοντέλο που περιγράφεται στο παρόν κεφάλαιο συνίσταται σε έναν εναλλακτικό διαχωρισμό της διαδικασίας σύνθεσης του δέντρου των ειδώλων. Το δέντρο αυτό κατασκευάζεται διαδοχικά, χρησιμοποιώντας τους κόμβους που υπολογίστηκαν σε προηγούμενα βήματα ως είσοδο και υπολογίζοντας τα είδωλα στην επόμενη τάξη ανάκλασης για ολόκληρη την περιοχή, ενώ οι υπολογισμοί των ακτίνων στα σημεία-δέκτες υπολογίζονται στην ίδια εργασία για ολόκληρη την περιοχή και μόνο για τα είδωλα που υπολογίστηκαν προηγουμένως. Κύριος στόχος είναι η ανάπτυξη μιας κλιμακούμενης και αποδοτικής παράλληλης υλοποίησης για αυξανόμενο αριθμό χρησιμοποιούμενων επεξεργαστών. Για την επίτευξη αυτού του σκοπού χρειάζεται να ληφθούν υπόψη ένα σύνολο ζητημάτων, όπως η διάσπαση του συνολικού υπολογιστικού φορτίου σε μικρότερες εργασίες, η μείωση της επιβάρυνσης λόγω επικοινωνίας και η κατανομή των εργασιών. Τα ζητήματα αυτά αναλύονται στις επόμενες παραγράφους.

3.4.1 Αποσύνθεση Υπολογιστικού Φορτίου - Σχηματισμός Επιμέρους Εργασιών

Όπως προαναφέρθηκε, το κύριο εμπόδιο για την παραλληλοποίηση ενός μοντέλου ray tracing που βασίζεται στη θεωρία των ειδώλων είναι το γεγονός ότι το μοντέλο εμπεριέχει τη δημιουργία ενός δέντρου ειδώλων, πράγμα που καθιστά τους υπολογισμούς αλληλοεξαρτώμενους και επομένους την όλη διαδικασία μη εύκολα παραλληλοποιήσιμη. Παρόλα αυτά, είναι δυνατή η αποδοτική παράλληλη εκτέλεσή της, κατασκευάζοντας το δέντρο των ειδώλων με σταδιακό τρόπο, χρησιμοποιώντας τις πληροφορίες από είδωλα που υπολογίστηκαν σε μια συγκεκριμένη τάξη ανάκλασης για τον υπολογισμό των ειδώλων στην αμέσως επόμενη τάξη.

Κάθε είδωλο που έχει υπολογισθεί (και το οποίο αντιστοιχεί σε έναν κόμβο του δέντρου), λειτουργεί ως είδωλο-γονέας και όλα τα εμπόδια σε ολόκληρη την περιοχή λαμβάνονται υπόψη για τον υπολογισμό όλων των ειδώλων του ειδώλου-γονέα (με βάση τον αλγόριθμο που περιγράφτηκε στην παράγραφο 3.3) σε μία εργασία. Ο υπολογισμός των ακτίνων στα σημεία-δέκτες ολόκληρης της περιοχής γίνεται μέσα στην ίδια εργασία και μόνο για τα είδωλα που υπολογίστηκαν σε αυτήν. Αν ο αριθμός των σημείων-δεκτών είναι πολύ μεγάλος (λόγω μεγάλου μεγέθους της περιοχής ή πυκνού πλέγματος σημείων), είναι θεωρητικά πιθανό να εμφανιστεί μεγάλη ετερογένεια στις υπολογιστικές απαιτήσεις των επιμέρους εργασιών, πράγμα που μπορεί να οδηγήσει σε προβλήματα ανισοκατανομής του

υπολογιστικού φορτίου. Μια λύση σε αυτό το πιθανό πρόβλημα είναι η επιπλέον διάσπαση της διαδικασίας υπολογισμού των ακτίνων, κατατέμνοντας τα σημεία-δέκτες σε υποσύνολα, πράγμα που οδηγεί σε εργασίες μικρότερου μεγέθους. Βέβαια σε αυτήν την περίπτωση τα διάφορα υπο-δέντρα χρειάζεται να γίνονται διαθέσιμα στους υπολογιστικούς κόμβους μαζί με την πληροφορία για τα σημεία-δέκτες, πράγμα που ενδέχεται να οδηγήσει σε επιβαρύνσεις επικοινωνίας. Παρόλα αυτά, όπως φαίνεται από τα πειραματικά αποτελέσματα που παρουσιάζονται σε επόμενη παράγραφο, η προσέγγιση υπολογισμού των ακτίνων για ένα υπο-δέντρο και για όλα τα σημεία της περιοχής λειτουργεί αποδοτικά, ακόμη και για σχετικά μεγάλες περιοχές και πυκνό πλέγμα σημείων. Η διαδικασία παραλληλοποίησης που περιγράφτηκε παραπάνω φαίνεται σχηματικά στο Σχήμα 13 για την περίπτωση ενός απλού δέντρου ειδώλων.



Σχήμα 13. Αποσύνθεση Υπολογισμών

Σύμφωνα με το παραπάνω σχήμα παραλληλοποίησης, τα είδωλα στην πρώτη τάξη ανάκλασης χρειάζεται να υπολογιστούν με σειριακό τρόπο, θεωρώντας τον αρχικό κόμβο του δέντρου που αντιστοιχεί στο σταθμό βάσης ως το είδωλο-γονέα. Το σειριακό αυτό τμήμα περιορίζει την κλιμάκωση του παράλληλου μοντέλου, όπως περιγράφτηκε προηγουμένως. Για την αντιμετώπιση αυτού του περιορισμού, χρησιμοποιείται μια data decomposition μέθοδος για τον υπολογισμό των ειδώλων στην πρώτη τάξη ανάκλασης: η συνολική υπό μελέτη περιοχή χωρίζεται σε υπο-περιοχές, οι οποίες κατανέμονται στους διαθέσιμους επεξεργαστές για τον παράλληλο υπολογισμό των ειδώλων πρώτης τάξης. Λόγω του ότι δεν υπάρχει εκ των προτέρων γνώση για υπο-περιοχές υψηλής πυκνότητας σε εμπόδια, η βάση δεδομένων χωρίζεται ομοιόμορφα σε έναν αριθμό υπο-περιοχών. Αυτό μπορεί βέβαια να οδηγήσει σε ανισοκατανομή φορτίου μεταξύ των επεξεργαστών, ιδίως όταν το μέγεθος των υπο-περιοχών αυξάνεται. Παρόλα αυτά, η ενδεχόμενη ανισοκατανομή φορτίου παρουσιάζεται μόνο στη διαδικασία υπολογισμού των ειδώλων πρώτης τάξης και αντισταθμίζεται κατά τους υπολογισμούς ανώτερων τάξεων ανάκλασης.

3.4.2 Κατανομή Εργασιών και Ισοκατανομή Υπολογιστικού Φορτίου

Είναι προφανές ότι το σχήμα παραλληλοποίησης που περιγράφτηκε στην προηγούμενη παράγραφο συνεπάγεται τη δυναμική δημιουργία εργασιών κατά τη διάρκεια της εκτέλεσης. Για κάθε είδωλο που υπολογίζεται, μια νέα εργασία δημιουργείται για τον υπολογισμό των ειδώλων στην επόμενη τάξη ανάκλασης, μέχρι να υπολογιστεί η μέγιστη τάξη ανάκλασης που ορίζεται κατά την εκκίνηση. Για το χειρισμό των δυναμικά δημιουργούμενων εργασιών, χρησιμοποιήθηκαν τα υπολογιστικά παραδείγματα master-worker και work-pool. Σε έναν κόμβο επεξεργασίας ανατίθεται ο ρόλος του αφέντη, ενώ όλοι οι υπόλοιποι αναλαμβάνουν το ρόλο των εργατών. Η διεργασία-αφέντης είναι υπεύθυνη για την επεξεργασία των δεδομένων εισόδου, την κατανομή των επιμέρους εργασιών, τη σύνθεση των αποτελεσμάτων και το συντονισμό μεταξύ των εργατών. Κάθε διεργασία-εργάτης επανειλημμένα ζητά νέες εργασίες από τη διεργασία-αφέντη, εκτελεί τους αντίστοιχους υπολογισμούς και τελικά επιστρέφει τα αποτελέσματα στη διεργασία-αφέντη. Προφανώς δεν υπάρχει απευθείας επικοινωνία μεταξύ των εργατών. Οι εναπομένουσες εργασίες τοποθετούνται σε ένα work-pool, το οποίο διαχειρίζεται η διεργασία-αφέντης. Κάθε φορά που η διεργασία-αφέντης λαμβάνει ένα αποτέλεσμα από έναν εργάτη, προσθέτει τα είδωλα που υπολογίστηκαν στο work-pool, μέχρι να ολοκληρωθούν όλες οι προκαθορισμένες τάξεις ανάκλασης. Κάθε ένα από αυτά τα είδωλα

θα χρησιμοποιηθεί ως είδωλο-γονέας για τον υπολογισμό των ειδώλων μεγαλύτερης τάξης. Η διαδικασία ολοκληρώνεται όταν το work-pool αδειάσει και ταυτόχρονα δεν υπάρχουν σε εκκρεμότητα εργασίες που εκτελούνται στις διεργασίες-εργάτες.

Θεωρητικά, οι δυναμικά δημιουργούμενες εργασίες ενδέχεται να οδηγήσουν σε κατάσταση απραξίας, όταν ένας ή περισσότεροι κόμβοι επεξεργασίας περιμένουν την ανάθεση κάποιας εργασίας ενώ κάποιος άλλος κόμβος εκτελεί κάποιον υπολογισμό που θα δημιουργήσει επιπλέον εργασίες. Το φαινόμενο αυτό μπορεί να αμβλυνθεί δίνοντας προτεραιότητα σε εργασίες που αντιστοιχούν σε μικρότερες τάξεις ανάκλασης (εργασίες για επεξεργασία ειδώλων πρώτης τάξης εκτελούνται πριν τις εργασίες που αντιστοιχούν σε είδωλα δεύτερης τάξης, κτλ), καθώς και με χρήση σχετικά μικρών μεγεθών εργασιών, έτσι ώστε οι εργασίες που αντιστοιχούν σε τάξη ανάκλασης n να εκτελούνται πριν εξαντληθούν οι εργασίες της τάξης $n+1$. Όπως προέκυψε από τα πειραματικά αποτελέσματα, η προσέγγιση αυτή αποδεικνύεται αποδοτική για διάφορα μεγέθη προβλημάτων, εξαλείφοντας τις πιθανές καταστάσεις απραξίας που περιγράφτηκαν προηγουμένως.

Για την κατανομή των εργασιών στους κόμβους επεξεργασίας χρησιμοποιήθηκαν τρία διαφορετικά σχήματα που έχουν προταθεί στη βιβλιογραφία. Η αποστολή πολλών μικρών μηνυμάτων ενδέχεται να έχει ως αποτέλεσμα να κυριαρχεί η καθυστέρηση του δικτύου στην επιβάρυνση επικοινωνίας. Συχνά είναι πιο αποδοτική η συνένωση μικρότερων μηνυμάτων σε ένα μεγαλύτερο, αυξάνοντας με αυτόν τον τρόπο το πραγματικό εύρος ζώνης επικοινωνίας. Τα διαφορετικά σχήματα που χρησιμοποιήθηκαν είναι τα εξής: σταθερό μέγεθος εργασίας (fixed-task-size), μειούμενο μέγεθος εργασίας (reducing-task-size) και μεταβλητό μέγεθος εργασίας (variable-task-size) [40], [47], [48]. Στην περίπτωση του fixed-task-size σχήματος, η διεργασία-αφέντης αναθέτει πάντα τον ίδιο αριθμό F ειδώλων-γονέων σε κάθε αίτηση μιας διεργασίας-εργάτη. Αν ο αριθμός των εναπομενουσών μονάδων προς επεξεργασία είναι T_{rem}, τότε ο αριθμός των μονάδων που ανατίθενται σε μια διεργασία-εργάτη είναι $A = \min(T_{rem}, F)$. Στο reducing-task-size σχήμα, αν ο n αριθμός των διαθέσιμων κόμβων επεξεργασίας, η διεργασία-αφέντης ξεκινά αναθέτοντας $A = F$ μονάδες επεξεργασίας σε κάθε διεργασία-εργάτη. Όταν η τιμή του T_{rem} γίνεται μικρότερη από nF, η διεργασία-αφέντης αναθέτει μία μόνο μονάδα επεξεργασίας σε κάθε αίτηση, έτσι ώστε να ελαχιστοποιείται ο τελικός άεργος χρόνος των επεξεργαστών και να επιτυγχάνεται καλύτερη ισοκατανομή του υπολογιστικού φορτίου. Στο variable-task-size σχήμα, η διεργασία-αφέντης αναθέτει $A = \lceil T_{rem} R / n \rceil$ μονάδες σε κάθε διεργασία-εργάτη, όπου $0 < R < 1$ είναι ένας παράγοντας ρύθμισης. Σε κάθε

περίπτωση, ο αριθμός των μονάδων επεξεργασίας που απομένουν μετά από κάθε ανάθεση είναι $T_{rem} = T_{rem} - A$. Οι διαφορετικές επιπτώσεις των διαφορετικών σχημάτων ανάθεσης στην απόδοση της παράλληλης υλοποίησης παρουσιάζονται στη συνέχεια στην παράγραφο 3.5.2.

3.4.3 Αλληλοεπικάλυψη Επικοινωνίας- Επεξεργασίας

Ένα από τα σημαντικότερα ζητήματα για την επίτευξη κλιμάκωσης και αποδοτικότητας μιας παράλληλης υλοποίησης είναι η μείωση του κόστους επικοινωνίας. Μια ευρέως χρησιμοποιούμενη και αποδοτική τεχνική είναι η αλληλοεπικάλυψη των λειτουργιών επεξεργασίας και επικοινωνίας (computation-communication overlapping). Για την επίτευξη της αλληλοεπικάλυψης, ακολουθείται η χρήση δύο νημάτων σε κάθε διεργασία: ένα νήμα επεξεργασίας και ένα νήμα επικοινωνίας. Το νήμα επεξεργασίας είναι υπεύθυνο για την εκτέλεση των λειτουργιών που απασχολούν τη CPU, όπως για παράδειγμα υπολογισμός δέντρου ειδώλων και ακτίνων στα σημεία-δέκτες, ενώ το νήμα επικοινωνίας αναλαμβάνει τις λειτουργίες μεταφοράς δεδομένων από και προς τους υπόλοιπους υπολογιστικούς κόμβους. Η επικοινωνία μπορεί να υλοποιηθεί με κλήσεις στις αντίστοιχες blocking ή non-blocking συναρτήσεις της βιβλιοθήκης MPI. Στην υλοποίηση που έγινε στα πλαίσια της διατριβής χρησιμοποιήθηκαν κλήσεις στις non-blocking συναρτήσεις που παρέχει το MPI (MPI_Isend(), MPI_Irecv() και MPI_Waitall()). Χρησιμοποιώντας δύο νήματα στη διεργασία-αφέντη, προκύπτει ότι εκτός από το συντονισμό της επεξεργασίας ανάμεσα στις διεργασίες-εργάτες, η διεργασία-αφέντης αναλαμβάνει και χρήσιμους υπολογισμούς (υπολογισμούς ειδώλων και ακτίνων), αξιοποιώντας με αυτόν τον τρόπο πόρους που διαφορετικά θα παρέμεναν άεργοι.

3.4.4 Υλοποίηση

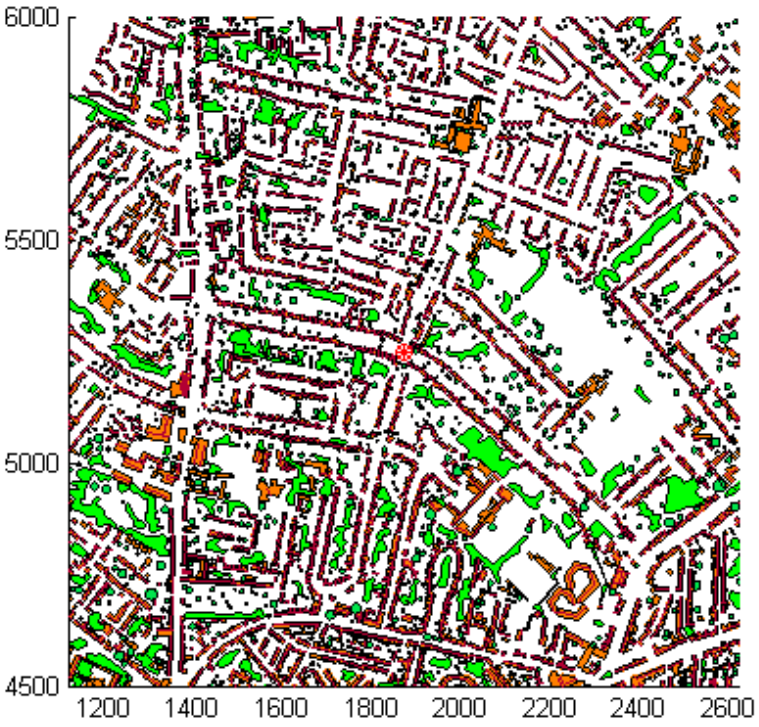
Η υλοποίηση του αρχικού σειριακού μοντέλου πρόβλεψης διάδοσης στηρίζεται στην C++. Για την υλοποίηση του παράλληλου μοντέλου έγινε χρήση της βιβλιοθήκης MPICH [49], ενώ η υλοποίηση της πολυ-νηματικής λειτουργικότητας που περιγράφτηκε στην προηγούμενη παράγραφο βασίστηκε στο POSIX API. Τα διαφορετικά στάδια που απαρτίζουν το παράλληλο μοντέλο πρόβλεψης διάδοσης συνοψίζονται στον Πίνακα 1.

Στάδιο	Περιγραφή
Αρχικοποίηση	όλες οι διεργασίες φορτώνουν τα set-up δεδομένα από ένα αρχείο εισόδου, όλες οι διεργασίες φορτώνουν τη βάση δεδομένων της περιοχής στην κεντρική μνήμη
Υπολογισμοί Δέντρου Ειδώλων	σταδιακή κατασκευή του δέντρου ειδώλων, τα είδωλα της πρώτης τάξης υπολογίζονται εφαρμόζοντας data decomposition τεχνικές και χωρίζοντας την περιοχή σε υπο-περιοχές, μοντέλα master-worker και work-pool για την κατανομή των εργασιών, αλληλοεπικάλυψη επεξεργασίας-επικοινωνίας με χρήση ενός νήματος επικοινωνίας και ενός νήματος επεξεργασίας σε κάθε διεργασία, επικοινωνία μεταξύ υπολογιστικών κόμβων μέσω του MPI, οι εργασίες δημιουργούνται από τη διεργασία-αφέντη, οι οποίες περιέχουν ένα ή περισσότερα είδωλα-γονείς, ανάλογα με το σχήμα κατανομής εργασιών που έχει επιλεγεί, οι εργασίες ανατίθενται στις διεργασίες-εργάτες με βάση ένα σχήμα αίτησης-ανάθεσης, τα είδωλα που έχουν υπολογισθεί και που περιέχονται σε ένα μήνυμα που λαμβάνει η διεργασία-αφέντης τοποθετούνται στο work-pool για τον υπολογισμό των ειδώλων στην επόμενη τάξη ανάκλασης
Υπολογισμοί Ακτίνων στα σημεία-δέκτες	η επεξεργασία των ακτίνων πραγματοποιείται μόνο για το συγκεκριμένο υπο-δέντρο που υπολογίσθηκε στη συγκεκριμένη εργασία, κάθε διεργασία-εργάτης πραγματοποιεί τον υπολογισμό των ακτίνων για ολόκληρη την περιοχή και για το συγκεκριμένο υπο-δέντρο – σε περίπτωση ανισοκατανομής φορτίου, η περιοχή διαχωρίζεται σε υπο-περιοχές και σε κάθε διεργασία-εργάτη ανατίθεται ένα υποσύνολο των σημείων-δεκτών, τα αποτελέσματα από τους υπολογισμούς των ακτίνων αποστέλλονται στη διεργασία-αφέντη μαζί με τα αποτελέσματα του υπο-δέντρου που υπολογίσθηκε προηγουμένως σε ενιαίο μήνυμα
Συγκέντρωση Αποτελεσμάτων	το ολοκληρωμένο δέντρο ειδώλων συγκεντρώνεται στο σύστημα αρχείων της διεργασίας-αφέντη, ένα αρχείο δημιουργείται για κάθε σημείο-δέκτη στο οποίο καταφθάνουν μία ή περισσότερες ακτίνες, το οποίο περιέχει δεδομένα όπως: ισχύς με και χωρίς Fresnel zone blocking, χρόνος άφιξης, γωνίες άφιξης και αναχώρησης θ και φ και στις δύο κεραίες, απώλειες περίθλασης στις κορυφές των δέντρων

Πίνακας 1. Στάδια του Παράλληλου Μοντέλου Πρόβλεψης Διάδοσης

3.5 Πειραματικά Αποτελέσματα - Απόδοση Παράλληλης Υλοποίησης

Ένας αριθμός από πειράματα εκτελέστηκαν για την αποτίμηση του παράλληλου μοντέλου πρόβλεψης διάδοσης. Τα πειράματα πραγματοποιήθηκαν σε τρεις SMP κόμβους, ο καθένας από τους οποίους αποτελείται από 8 επεξεργαστές Intel Xeon στα 2GHz και 8GB RAM, οι οποίοι συνδέονται με δίκτυο 1Gb Ethernet και στους οποίους είναι εγκατεστημένο το λειτουργικό σύστημα Debian Linux. Για να διασφαλιστεί ένα όσο το δυνατόν πιο ομοιογενές περιβάλλον επικοινωνίας, ένας από τους κόμβους χρησιμοποιείται μόνο για την εκτέλεση της διεργασίας-αφέντη (πράγμα που σημαίνει ότι μόνο ένας από τους οκτώ επεξεργαστές χρησιμοποιείται), ενώ στους διαθέσιμους υπολογιστές των δύο άλλων κόμβων εκτελούνται οι διεργασίες-εργάτες. Συνεπώς, τα πειράματα πραγματοποιήθηκαν χρησιμοποιώντας μέχρι και 17 επεξεργαστές.



Σχήμα 14. Τοποθεσίες κτιρίων, φυλλωμάτων, και θέσης BS στο τμήμα της περιοχής του Cambridge που θεωρήθηκε

Επιπλέον, για να διασφαλιστεί ότι κάθε διεργασία εκτελείται συνεχώς και αποκλειστικά στον ίδιο επεξεργαστή, χρησιμοποιείται η δυνατότητα που παρέχει το λειτουργικό σύστημα Linux για δρομολόγηση εργασιών με βάση μια cpu-affinity-mask. Στο Listing 2 φαίνεται

ένα απόσπασμα κώδικα που αναλαμβάνει την ανάθεση των διεργασιών ενός MPI προγράμματος στους επεξεργαστές ενός κόμβου SMP.

```
#include <mpi.h>
#include <sched.h>
//…

int main(int argc, char **argv)
{
        MPI_Init(&argc, &argv);
        int myrank;
        MPI_Comm_rank(MPI_COMM_WORLD, &myrank);

        //...

        int CPU_ID = 0;
        cpu_set_t mask;                 // μάσκα για τον ορισμό των επεξεργαστών στους
                                        // οποίους  επιτρέπεται να εκτελεστεί η διεργασία

        int noOfCores = sysconf(_SC_NPROCESSORS_CONF); // εύρεση αριθμού
                                                       // επεξεργαστών
                                                       // στον κόμβο
        if(noOfCores != 1)
        {
                if(myrank==0)
                        CPU_ID = 0;                    // η διεργασία-αφέντης εκτελείται
                                                       // στον επεξεργαστή 0 του πρώτου
                                                       //κόμβου
                else
                        CPU_ID = myrank%noOfCores;     // οι διεργασίες-εργάτες
                                                       //μοιράζονται κυκλικά στους
                                                       // επεξεργαστές των
                                                       // υπόλοιπων κόμβων

                CPU_ZERO(&mask);                       // μηδενισμός μάσκας
                CPU_SET(CPU_ID,&mask);                 // ορισμός μάσκας
                if (sched_setaffinity(0, sizeof(mask), &mask) == -1)
                {
                        printf("WARNING: could not set CPU affinity for process %d!
                                                                Exiting...\n",myrank);
                        return -1;
                }
        }

        //...

}
```

Listing 2. Δρομολόγηση εργασιών με χρήση της cpu affinity mask

Τρεις περιοχές διαφορετικού μεγέθους χρησιμοποιήθηκαν στα πειράματα, οι οποίες ανήκουν στην ευρύτερη περιοχή του Cambridge (Σχήμα 14). Η πρώτη περιοχή (περιοχή A) έχει μέγεθος 500mx500m και εκτείνεται στο χώρο που ορίζεται από τα διαγώνια σημεία (1620,5000) και (2120,5500). Η περιοχή αυτή περιέχει 276 κτίρια με 1642 τοίχους και 223 δέντρα. Η δεύτερη περιοχή (περιοχή B) έχει μέγεθος 1000mx1000m και εκτείνεται στο χώρο που ορίζουν τα διαγώνια σημεία (1370,4750) και (2370,5750). Περιέχει 955 κτίρια, 5846 τοίχους και 701 δέντρα. Τέλος, η τρίτη περιοχή (περιοχή C) έχει μέγεθος 1500mx1500m και ορίζεται από τα διαγώνια σημεία (1120,4500) και (2620,6000). Περιέχει 2163 κτίρια και 14512 τοίχους, ενώ ο αριθμός των δέντρων που υπάρχουν σε αυτήν είναι 1378. Στα πειράματα που εκτελέστηκαν και για τις τρεις περιοχές θεωρήθηκε μέγιστη τάξη ανακλάσεων ίση με 3. Επίσης, εκτελέστηκαν πειράματα για την περιοχή B και για 4 και 5 τάξεις ανάκλασης.

Το set-up που χρησιμοποιήθηκε σε όλες τις περιπτώσεις φαίνεται στον Πίνακα 2.

Θέση BS	(1870,5250)
Ύψος BS	15m
Συχνότητα	2,5GHz
Ύψος κεραίας MS από το έδαφος	1,5m
Πόλωση κεραίας BS	κατακόρυφη
Πόλωση κεραίας MS	κατακόρυφη
Εκπεμπόμενη Ισχύς	30dBm
Ανάλυση πλέγματος σημείων-δεκτών	15m

Πίνακας 2. Set-up πειραματικών εκτελέσεων

Για την ανάλυση της απόδοσης του παράλληλου μοντέλου πρόβλεψης διάδοσης χρησιμοποιήθηκαν τα μεγέθη της επιτάχυνσης, της αποδοτικότητας, του ποσοστού

επέκτασης υπολογιστικού φορτίου και του βαθμού αξιοποίησης πόρων, όπως αυτά ορίστηκαν στο προηγούμενο κεφάλαιο.

#CPUs	pid	#υπο-περιοχών - χρόνος (sec)	#ειδώλων πρώτης τάξης - χρόνος (sec)	#ειδώλων δεύτερης τάξης - χρόνος (sec)	συνολικός χρόνος (sec)	ιδανική επιτάχυνση	επιτάχυνση
1	0	98 - 3,01	833 - 18,94	3046 - 66,11	89,07	1	1
2	0	47 - 1,56	280 - 9,57	1665 - 33,48	45,66	1,98	1,95
	1	51 - 1,47	553 - 9,60	1381 - 33,48	45,66		
3	0	10 - 1,04	194 - 6,43	1117 - 22,37	30,88	2,93	2,88
	1	37 - 0,98	292 - 6,41	1020 - 22,39	30,88		
	2	51 - 1,02	347 - 6,37	909 - 22,40	30,88		
4	0	8 - 0,82	172 - 4,81	810 - 16,84	23,49	3,86	3,79
	1	29 - 0,73	239 - 4,84	727 - 16,82	23,48		
	2	25 - 0,74	186 - 4,81	779 - 16,83	23,47		
	3	36 - 0,74	236 - 4,81	730 - 16,86	23,49		
5	0	6 - 0,66	125 - 3,86	657 - 13,50	19,03	4,76	4,68
	1	19 - 0,59	211 - 3,85	576 - 13,51	19,03		
	2	21 - 0,61	147 - 3,86	631 - 13,48	19,01		
	3	29 - 0,61	183 - 3,84	583 - 13,48	19,01		
	4	23 - 0,59	167 - 3,88	599 - 13,49	19,02		
6	0	5 - 0,55	111 - 3,25	537 - 11,23	16,07	5,65	5,54
	1	17 - 0,49	126 - 3,21	517 - 11,25	16,06		
	2	21 - 0,50	140 - 3,20	508 - 11,26	16,07		
	3	19 - 0,50	132 - 3,21	508 - 11,24	16,07		
	4	13 - 0,52	170 - 3,20	489 - 11,24	16,06		
	5	23 - 0,49	154 - 3,21	487 - 11,25	16,06		

Πίνακας 3. Στατιστικά για την Περιοχή Α (500mx500m)

3.5.1 Ανάλυση Απόδοσης Παράλληλης Υλοποίησης

Στους Πίνακες 2, 3 και 4 παρουσιάζονται αναλυτικά αποτελέσματα εκτέλεσης για τις περιοχές Α, Β και C αντίστοιχα, χρησιμοποιώντας το fixed-task-size σχήμα κατανομής με $F = 1$. Οι στήλες *"# υπο-περιοχών - χρόνος (sec)"*, *"# ειδώλων πρώτης τάξης - χρόνος (sec)"* και *"# ειδώλων δεύτερης τάξης - χρόνος (sec)"* αναφέρονται στον αριθμό των εργασιών που

αναλαμβάνονται από την κάθε διεργασία-εργάτη σε κάθε φάση και το χρόνο που μεσολαβεί μεταξύ της εκκίνησης της πρώτης εργασίας και τον τερματισμό της τελευταίας εργασίας σε κάθε επεξεργαστή στη δεδομένη φάση.

#CPUs	pid	#υπο-περιοχών - χρόνος (sec)	#ειδώλων πρώτης τάξης - χρόνος (sec)	#ειδώλων δεύτερης τάξης - χρόνος (sec)	συνολικός χρόνος (sec)	επιτάχυνση
1	0	116 - 25,20	2930 - 315,73	20328 - 1980,375	2322,36	1
2	0	54 - 12,61	1489 - 158,44	10207 - 998,38	1170,61	1,98
	1	62 - 12,59	1441 - 158,92	10121 - 997,85	1170,50	
3	0	28 - 8,25	960 - 105,38	6840 - 667,84	782,73	2,97
	1	45 - 8,09	985 - 105,64	6771 - 667,73	782,64	
	2	43 - 8,86	985 - 107,14	6717 - 665,50	782,61	
4	0	20 - 5,92	734 - 79,30	5132 - 501,61	588,22	3,95
	1	31 - 5,97	734 - 79,50	5055 - 501,37	588,07	
	2	33 - 6,52	715 - 81,16	5071 - 499,24	588,10	
	3	32 - 6,81	747 - 78,44	5070 - 501,62	588,00	
5	0	11 - 5,17	591 - 63,18	4079 - 401,39	471,13	4,93
	1	28 - 4,72	605 - 63,62	4055 - 401,38	470,99	
	2	26 - 4,77	584 - 65,58	4067 - 399,42	471,00	
	3	28 - 4,82	575 - 63,55	4069 - 401,32	470,88	
	4	23 - 5,75	575 - 62,57	4058 - 401,42	470,87	
6	0	6 - 3,98	516 - 52,48	3424 - 334,87	392,87	5,91
	1	19 - 4,52	482 - 53,02	3377 - 333,88	392,79	
	2	21 - 3,95	482 - 55,48	3358 - 331,91	392,67	
	3	22 - 3,89	483 - 52,72	3412 - 334,76	392,65	
	4	26 - 4,02	482 - 52,99	3385 - 334,37	392,60	
	5	22 - 4,88	485 - 51,66	3372 - 334,82	392,54	

Πίνακας 4. Στατιστικά για την Περιοχή Β (1000mx1000m)

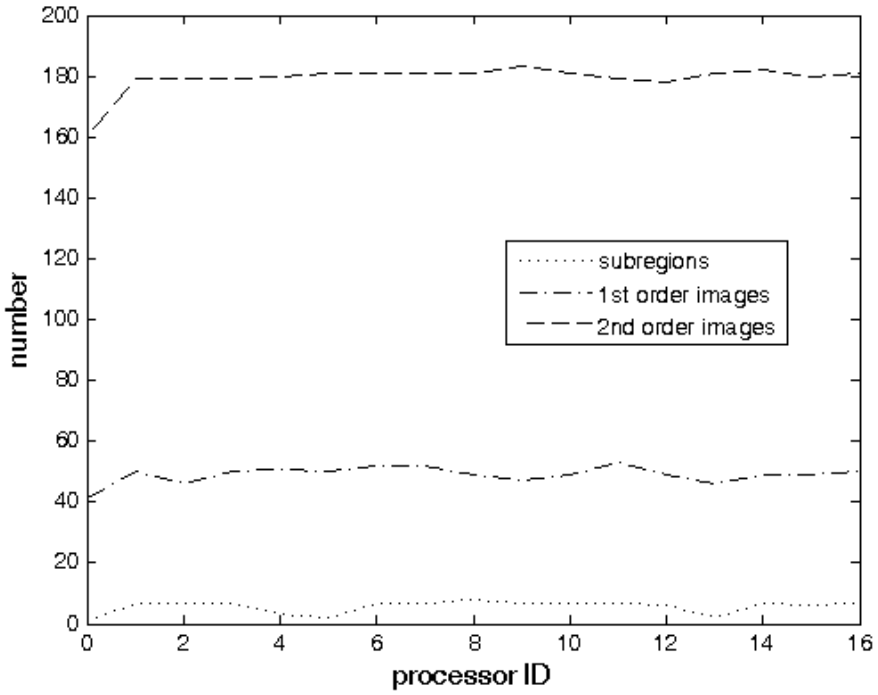
Στην περίπτωση της περιοχής Α, το σειριακό τμήμα που αντιστοιχεί στο φόρτωμα της βάσης δεδομένων στην κεντρική μνήμη (περίπου 1 sec) είναι συγκρίσιμο με τους υπολογισμούς για το ray tracing και επομένως η επιτάχυνση που επιτυγχάνεται περιορίζεται από αυτό. Η στήλη *"ιδανική επιτάχυνση"* στον Πίνακα 3 αναφέρεται στη μέγιστη δυνατή επιτάχυνση λαμβάνοντας υπόψη αυτό το σειριακό τμήμα. Στις περιπτώσεις των περιοχών Β

και C, το αντίστοιχο σειριακό τμήμα είναι αμελητέο και για αυτό το λόγο δεν λαμβάνεται υπόψη. Για συντομία, παρουσιάζονται αποτελέσματα για χρήση από 1 έως 6 επεξεργαστών.

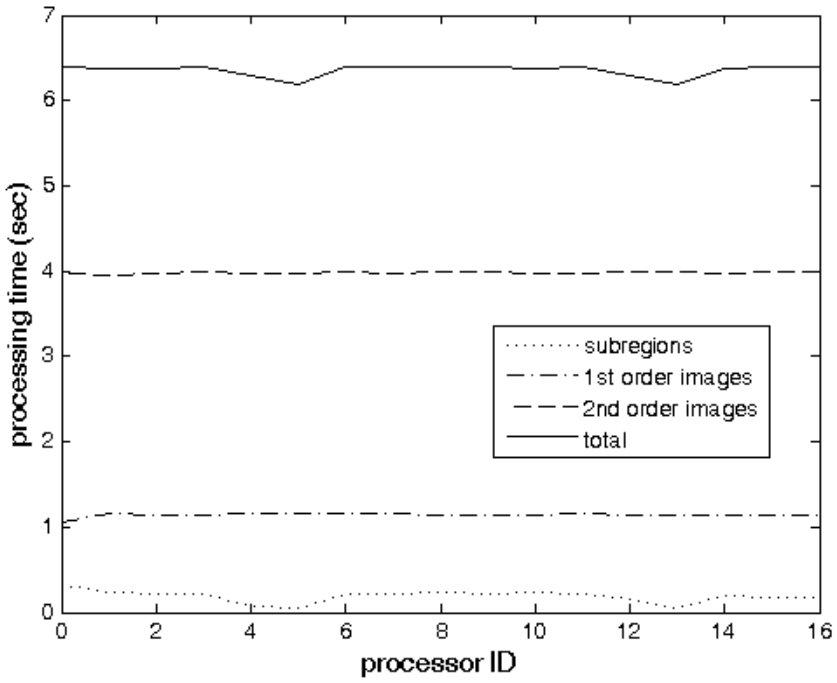
#CPUs	pid	#υπο-περιοχών - χρόνος (sec)	#ειδώλων πρώτης τάξης - χρόνος (sec)	#ειδώλων δεύτερης τάξης - χρόνος (sec)	συνολικός χρόνος (sec)	επιτάχυνση
1	0	255 - 193,12	7175 - 3323,81	88730 - 38145,61	41663,68	1
2	0	125 - 96,38	3633 - 1669,04	44631 - 19182,46	20949,66	1,99
	1	130 - 96,52	3542 - 1669,29	44099 - 19182,48	20949,49	
3	0	91 - 63,95	2429 - 1114,68	29816 - 12811,88	13992,08	2,98
	1	79 - 64,48	2348 - 1114,53	29454 - 12811,75	13992,03	
	2	85 - 64,48	2407 - 1114,36	29460 - 12811,75	13991,80	
4	0	59 - 48,00	1825 - 836,59	22389 - 9619,10	10505,29	3,97
	1	72 - 48,23	1739 - 836,81	22117 - 9618,56	10504,91	
	2	59 - 48,33	1798 - 836,38	22136 - 9619,16	10505,13	
	3	65 - 48,21	1759 - 836,64	22088 - 9618,91	10504,96	
5	0	45 - 38,32	1415 - 669,25	17918 - 7695,39	8406,95	4,96
	1	47 - 38,68	1446 - 669,51	17727 - 7694,87	8404,57	
	2	45 - 38,71	1435 - 669,13	17702 - 7695,23	8404,52	
	3	61 - 38,36	1450 - 669,52	17695 - 7695,24	8404,49	
	4	57 - 38,75	1429 - 669,37	17688 - 7694,73	8404,06	
6	0	35 - 31,79	1208 - 558,08	14969 - 6417,20	7009,18	5,94
	1	45 - 32,30	1201 - 557,95	14740 - 6417,30	7008,93	
	2	47 - 32,26	1224 - 557,74	14780 - 6417,46	7008,80	
	3	43 - 32,12	1209 - 558,05	14729 - 6417,21	7008,67	
	4	48 - 32,42	1164 - 557,87	14765 - 6417,32	7008,86	
	5	37 - 31,86	1169 - 558,50	14747 - 6417,27	7008,84	

Πίνακας 5. Στατιστικά για την Περιοχή C (1500mx1500m)

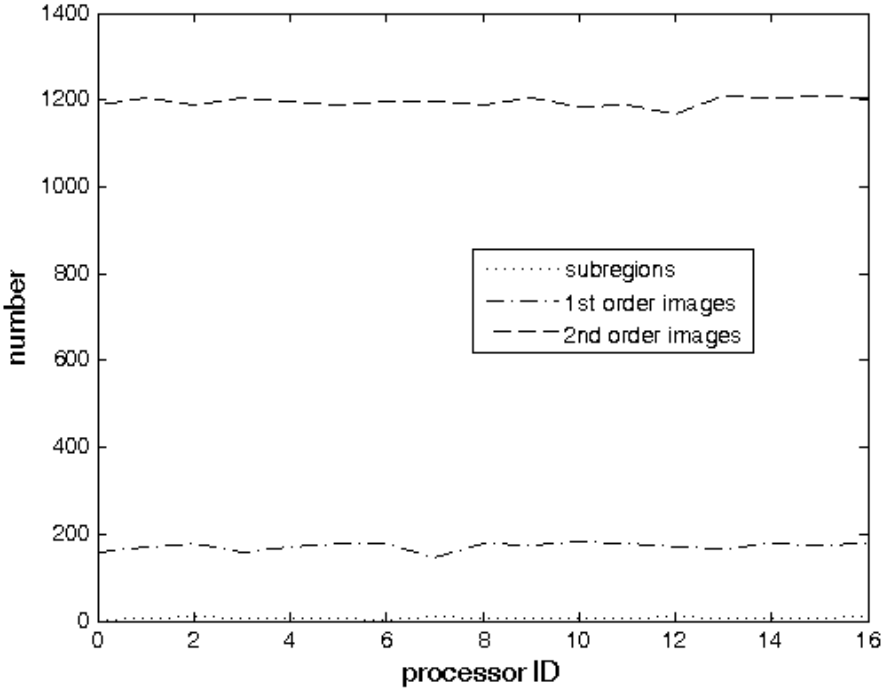
Για την περίπτωση που χρησιμοποιούνται 17 επεξεργαστές, τα αντίστοιχα αποτελέσματα παρουσιάζονται στις γραφικές παραστάσεις των Σχημάτων 15-16, 17-18 και 19-20 για τις περιοχές A, B και C, αντίστοιχα. Ο άξονας x αναπαριστά τα αναγνωριστικά των επεξεργαστών που συμμετέχουν στην παράλληλη επεξεργασία (0-16) και οι άξονες y αναπαριστούν των αριθμό των εργασιών που αναλαμβάνει ο κάθε επεξεργαστής στην κάθε φάση και τους αντίστοιχους χρόνους επεξεργασίας, όπως και στους Πίνακες 3-5.



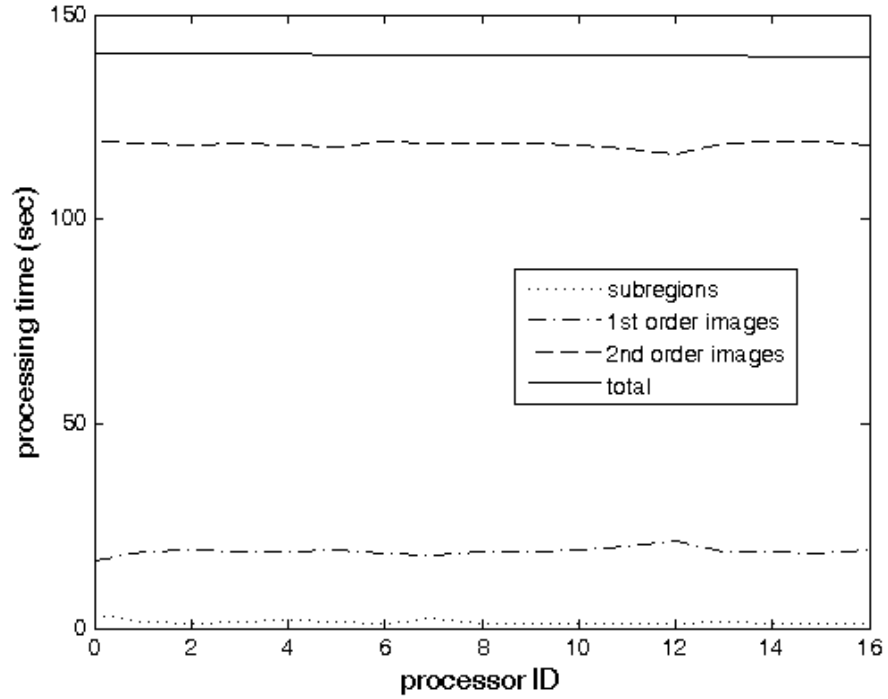
Σχήμα 15. Αριθμός υπο-περιοχών, ειδώλων πρώτης τάξης και ειδώλων δεύτερης τάξης που υπολογίστηκαν από τον κάθε επεξεργαστή για 17 επεξεργαστές και για την Περιοχή Α



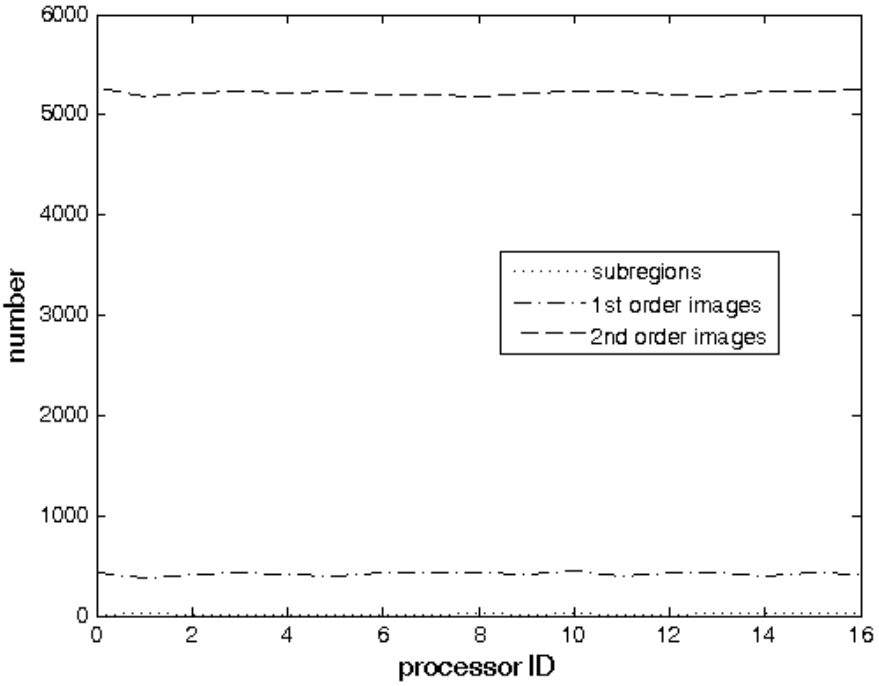
Σχήμα 16. Χρόνοι επεξεργασίας υπο-περιοχών, ειδώλων πρώτης τάξης και ειδώλων δεύτερης τάξης που υπολογίστηκαν από τον κάθε επεξεργαστή για 17 επεξεργαστές και για την Περιοχή Α



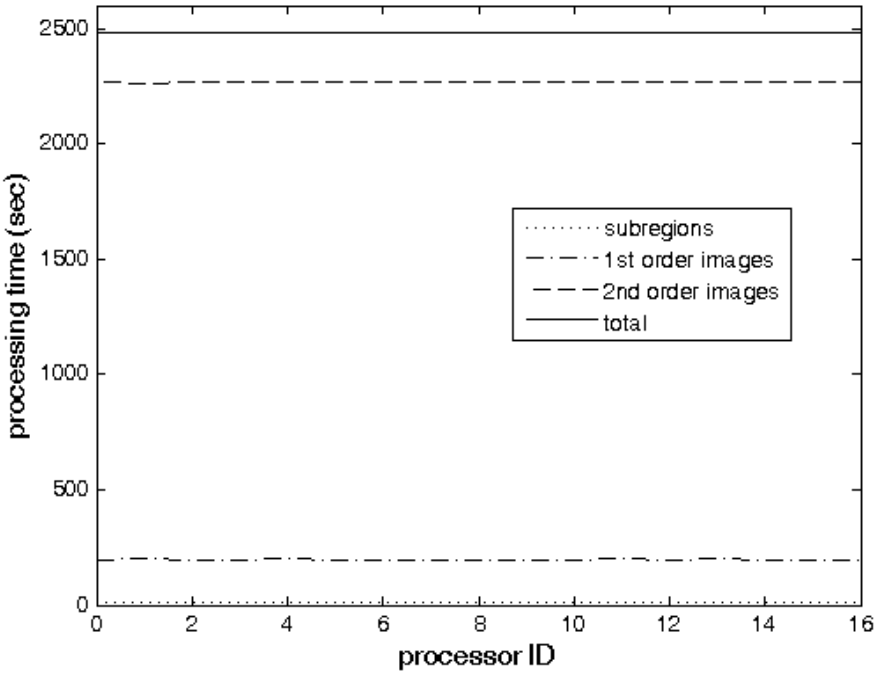
Σχήμα 17. Αριθμός υπο-περιοχών, ειδώλων πρώτης τάξης και ειδώλων δεύτερης τάξης που υπολογίστηκαν από τον κάθε επεξεργαστή για 17 επεξεργαστές και για την Περιοχή Β



Σχήμα 18. Χρόνοι επεξεργασίας υπο-περιοχών, ειδώλων πρώτης τάξης και ειδώλων δεύτερης τάξης που υπολογίστηκαν από τον κάθε επεξεργαστή για 17 επεξεργαστές και για την Περιοχή Β



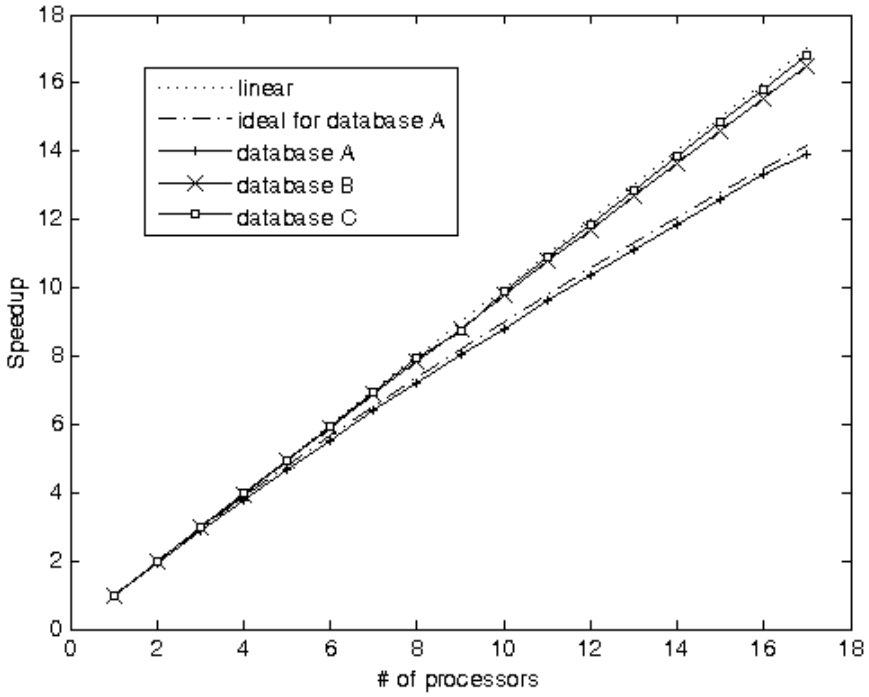
Σχήμα 19. Αριθμός υπο-περιοχών, ειδώλων πρώτης τάξης και ειδώλων δεύτερης τάξης που υπολογίστηκαν από τον κάθε επεξεργαστή για 17 επεξεργαστές και για την Περιοχή C



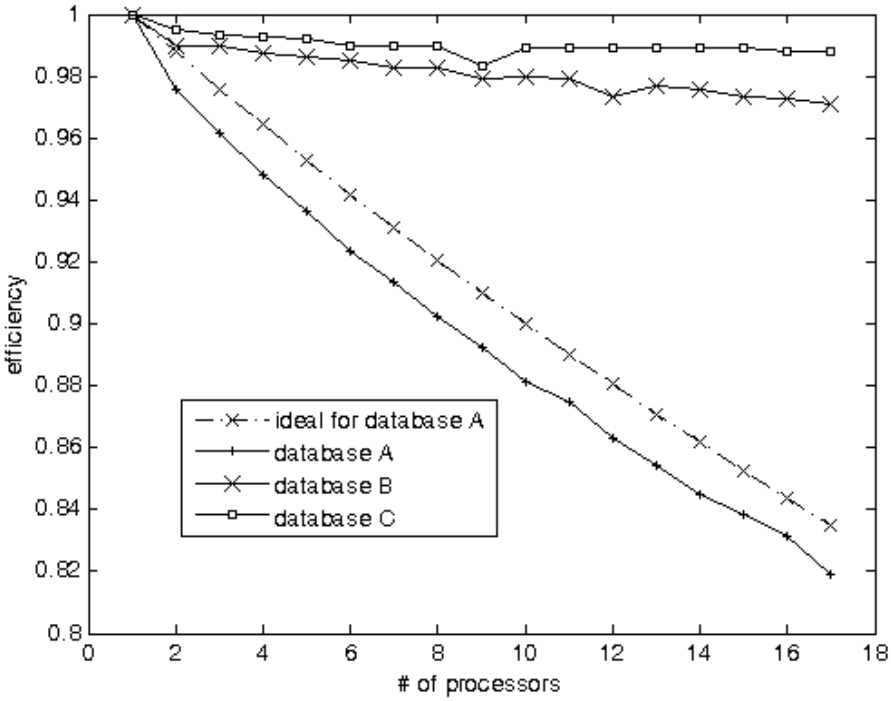
Σχήμα 20. Χρόνοι επεξεργασίας υπο-περιοχών, ειδώλων πρώτης τάξης και ειδώλων δεύτερης τάξης που υπολογίστηκαν από τον κάθε επεξεργαστή για 17 επεξεργαστές και για την Περιοχή C

Μια πρώτη παρατήρηση είναι ότι οι χρόνοι επεξεργασίας για διαφορετικούς υπολογισμούς ακόμη και στην ίδια τάξη ανάκλασης είναι αρκετά διαφορετικοί για όλες τις περιοχές. Για παράδειγμα, στην περιοχή C, ο μέγιστος χρόνος επεξεργασίας υπο-περιοχής είναι 9,66 sec, ο ελάχιστος 0.38 sec και ο μέσος χρόνος 0.76 sec. Οι αντίστοιχοι χρόνοι για την επεξεργασία των ειδώλων πρώτης τάξης είναι 29,17 sec, 0,37 sec και 0,46 sec, ενώ για τα είδωλα δεύτερης τάξης είναι 10,61 sec, 0,37 sec και 0,43 sec.

Παρόλα αυτά, η δυναμική κατανομή των εργασιών από τη διεργασία-αφέντη επιτυγχάνει την ισοκατανομή του υπολογιστικού φορτίου σε όλες τις φάσεις της επεξεργασίας, καθώς οι διαφορές στους χρόνους εκτέλεσης κάθε φάσης είναι πολύ μικρές ανάμεσα στους επεξεργαστές. Το πιο σημαντικό είναι ότι οι άεργοι χρόνοι των επεξεργαστών είναι πολύ μικροί σε όλες τις περιπτώσεις, πράγμα που υποδηλώνει ότι το συνολικό υπολογιστικό φορτίο έχει κατανεμηθεί ομοιόμορφα μεταξύ των επεξεργαστών. Για παράδειγμα, για την περιοχή C και 6 επεξεργαστές, ο μέγιστος άεργος χρόνος είναι 0,50 sec, ενώ για 17 επεξεργαστές ο αντίστοιχος χρόνος είναι 1,20 sec. Αν και η κάθε εργασία που ανατίθεται σε μια διεργασία-εργάτη περιλαμβάνει τον υπολογισμό των ακτίνων σε ολόκληρη την περιοχή, οι εργασίες είναι αρκετά μικρές ώστε να επιτυγχάνεται σε μεγάλο βαθμό η ισοκατανομή του φορτίου. Παρόμοια συμπεράσματα προκύπτουν και για τις περιοχές A και B.

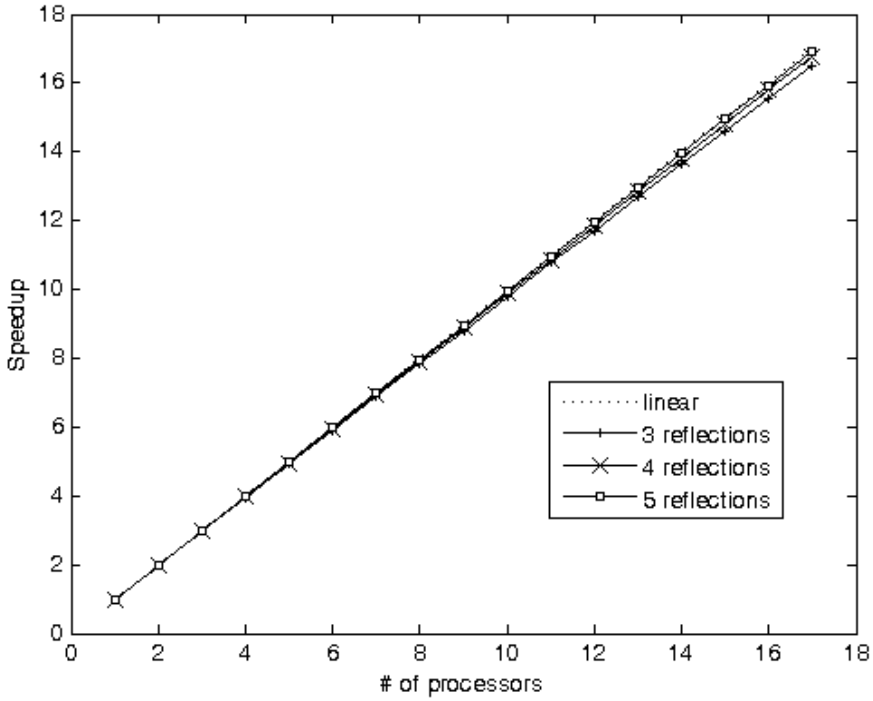


Σχήμα 21. Επιτάχυνση για τις περιοχές A,B και C και για 3 τάξεις ανακλάσεων

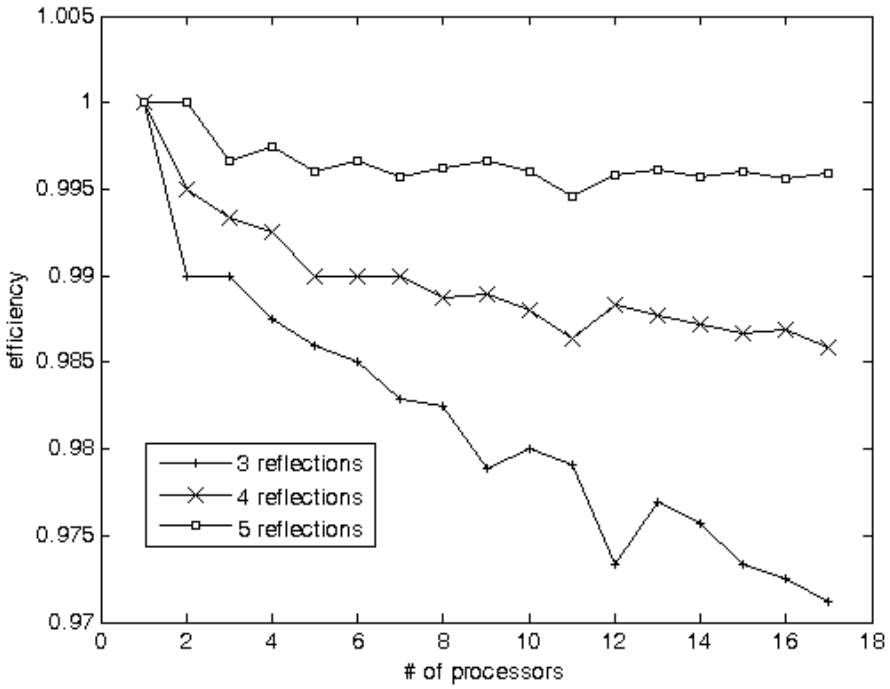


Σχήμα 22. Αποδοτικότητα για τις περιοχές A, B και C και 3 τάξεις ανακλάσεων

Στη γραφική παράσταση του Σχήματος 21 φαίνεται η επιτάχυνση που επιτυγχάνεται για τις τρεις περιοχές και για 3 τάξεις ανάκλασης, για διαφορετικό αριθμό επεξεργαστών. Για τις περιοχές B και C, η επιτάχυνση είναι σχεδόν γραμμική με τον αριθμό των χρησιμοποιούμενων επεξεργαστών. Στην περίπτωση της περιοχής A, η επιτάχυνση που επιτυγχάνεται είναι πολύ κοντά στο θεωρητικό μέγιστο που υποδεικνύεται από το νόμο του Amdahl. Οι επιταχύνσεις για την περιοχή B και διαφορετικό αριθμό ανακλάσεων (3, 4 και 5) φαίνονται στο Σχήμα 23. Και σε αυτήν την περίπτωση η επιτάχυνση που επιτυγχάνεται είναι σχεδόν γραμμική και γίνεται οριακά καλύτερη καθώς η μέγιστη τάξη ανάκλασης (και κατ' επέκταση και το συνολικό υπολογιστικό φορτίο) αυξάνεται. Στα Σχήματα 22 και 24 φαίνονται οι γραφικές παραστάσεις της αποδοτικότητας του παράλληλου μοντέλου για τις τρεις διαφορετικές περιοχές και για διαφορετικό αριθμό ανακλάσεων και την περιοχή B, αντίστοιχα. Σε όλες τις περιπτώσεις, εκτός της περιοχής A, οι τιμές της αποδοτικότητας είναι πολύ κοντά στη μονάδα, ενώ και στην περίπτωση της περιοχής A, οι λαμβανόμενες τιμές είναι πολύ κοντά στις θεωρητικά ιδανικές.

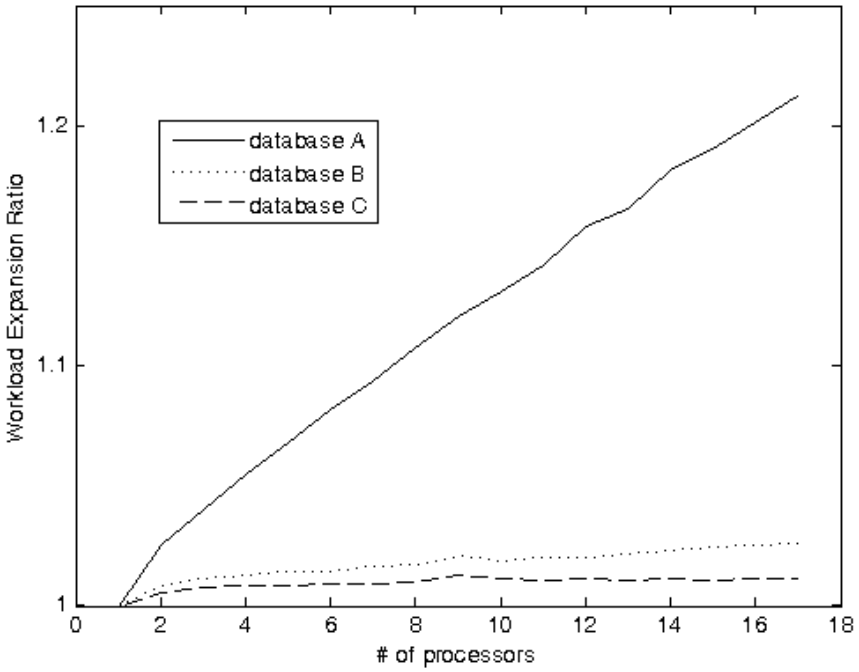


Σχήμα 23. Επιτάχυνση για 3, 4 και 5 τάξεις ανακλάσεων και για την περιοχή B



Σχήμα 24. Αποδοτικότητα για 3, 4 και 5 τάξεις ανακλάσεων και για την περιοχή B

Το ποσοστό επέκτασης υπολογιστικού φορτίου και ο βαθμός αξιοποίησης πόρων για τις τρεις περιοχές και για 3 τάξεις ανακλάσεων φαίνονται στα Σχήματα 25 και 26 αντίστοιχα. Το ποσοστό επέκτασης υπολογιστικού φορτίου είναι ένα μέγεθος που δείχνει την ποσότητα του επιπλέον υπολογιστικού χρόνου που προσθέτει η παράλληλη υλοποίηση. Για τις περιοχές B και C, οι τιμές του είναι κοντά στη μονάδα για οποιονδήποτε αριθμό χρησιμοποιούμενων επεξεργαστών. Στην περίπτωση της περιοχής A όμως, παρατηρείται μια αξιοσημείωτη αύξηση της τιμής του, καθώς ο αριθμός των επεξεργαστών αυξάνεται. Αυτό οφείλεται κατά κύριο λόγο στο γεγονός ότι ο χρόνος για το φόρτωμα της βάσης δεδομένων της περιοχής στην κεντρική μνήμη είναι συγκρίσιμος με τον συνολικό απαιτούμενο χρόνο εκτέλεσης και επομένως τα επιπλέον φορτώματα που πραγματοποιούνται από τον κάθε επεξεργαστή προκαλούν μια μη-αμελητέα επιβάρυνση. Καθώς ο παράγοντας επέκτασης υπολογιστικού φορτίου συνδέεται άμεσα με την πολυπλοκότητα της δεδομένης επεξεργασίας, αναμένεται ότι οι περισσότερο πολύπλοκοι υπολογισμοί αντιστοιχούν σε μικρότερες τιμές αυτού.



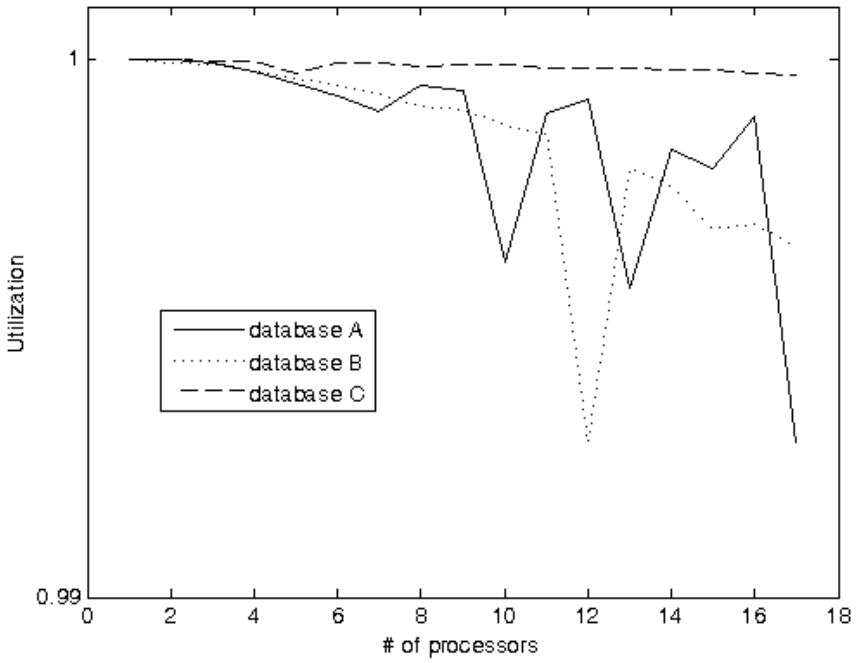
Σχήμα 25. Παράγοντας Επέκτασης Υπολογιστικού Φορτίου για τις Περιοχές A, B και C και για 3 τάξεις ανακλάσεων

Ο βαθμός αξιοποίησης σε όλες τις περιπτώσεις είναι πολύ κοντά στη μονάδα, πράγμα που υποδεικνύει ότι όλοι οι επεξεργαστές ξοδεύουν πολύ λίγο χρόνο σε άεργη κατάσταση.

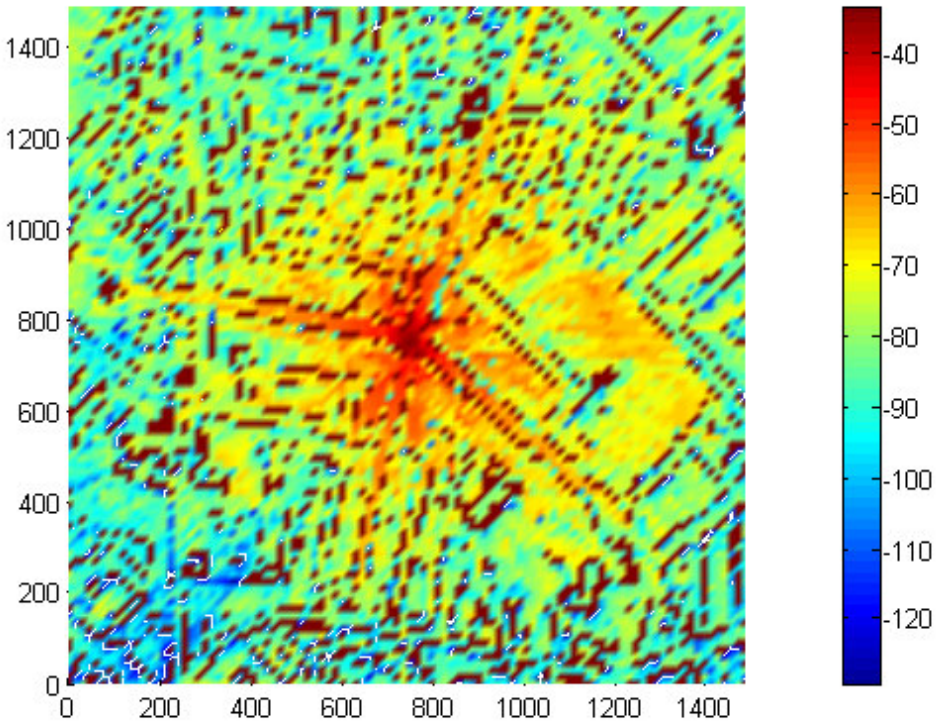
Ενδεικτικά αποτελέσματα διάδοσης από την εκτέλεση του αλγορίθμου φαίνονται στο Σχήμα 27. Στο σχήμα αυτό φαίνονται οι ισχύες που λαμβάνονται στα σημεία της περιοχής C, θεωρώντας τρεις τάξεις ανακλάσεων.

3.5.2 Σύγκριση Σχημάτων Ανάθεσης Εργασιών

Τα πειράματα που περιγράφτηκαν στην προηγούμενη παράγραφο πραγματοποιήθηκαν και για τα τρία σχήματα κατανομής εργασιών που περιγράφτηκαν στην παράγραφο 3.4.2. Συγκεκριμένα, οι τιμές του F για τα fixed-task-size και reducing-task-size σχήματα θεωρήθηκαν από 1 έως 5, ενώ οι τιμές του R για το variable-task-size σχήμα θεωρήθηκαν στο εύρος 0,05 έως 0,3 με βήμα 0,05. Τα αποτελέσματα για τα τρία διαφορετικά σχήματα διαφέρουν ανάλογα με τον αριθμό των χρησιμοποιούμενων επεξεργαστών. Στους Πίνακες 6-8 φαίνονται τα καλύτερα και χειρότερα σχήματα ανάθεσης εργασιών για κάθε περιοχή και για 17, 10 και 3 επεξεργαστές, αντίστοιχα.



Σχήμα 26. Βαθμός Αξιοποίησης Πόρων για τις Περιοχές A, B και C και για 3 τάξεις ανακλάσεων



Σχήμα 27. Λαμβανόμενη ισχύς για την περιοχή C και για 3 τάξεις ανακλάσεων

Μια πρώτη παρατήρηση είναι ότι οι διαφορές μεταξύ των καλύτερων και χειρότερων σχημάτων ανάθεσης δεν είναι πολύ έντονες, ειδικά για μικρό αριθμό επεξεργαστών. Ακόμα και στην περίπτωση των 17 επεξεργαστών, η μέγιστη βελτίωση στην επιτάχυνση μεταξύ του καλύτερου και χειρότερου σχήματος είναι μόλις 0,12 για την περιοχή B. Παρόλα αυτά, οι διαφορές αυτές παρουσιάζουν μια τάση να γίνονται πιο ορατές καθώς αυξάνεται ο αριθμός των επεξεργαστών και ενδέχεται να γίνουν αξιοσημείωτες για μεγαλύτερο αριθμό. Μια ακόμη παρατήρηση είναι ότι κάποια σχήματα δείχνουν να είναι πιο αποδοτικά για συγκεκριμένα μεγέθη του παράλληλου συστήματος. Όπως φαίνεται στον Πίνακα 8, για μικρό αριθμό επεξεργαστών το variable-task-size σχήμα με σχετικά μεγάλες τιμές για το R παρουσιάζει μικρότερη απόδοση σε σχέση με τα υπόλοιπα σχήματα. Από την άλλη πλευρά, το variable-task-size σχήμα έχει καλύτερη απόδοση για μεγαλύτερο αριθμό επεξεργαστών, σε αντίθεση με fixed-task-size σχήματα, όπως φαίνεται στον Πίνακα 6.

περιοχή	καλύτερο	T_{max}	T_{sum}	speedup	χειρότερο	T_{max}	T_{sum}	speedup
A	V0.3	6,36	107,39	14,01	F2	6,57	110,06	13,85
B	V0.3	140,33	2375,97	16,55	F4	141,35	2381,23	16,43
C	V0.1	2479,61	42129,4	16,80	F4	2490,25	42158,33	16,73

Πίνακας 6. Καλύτερο και χειρότερο σχήμα ανάθεσης για 17 επεξεργαστές

περιοχή	καλύτερο	T_{max}	T_{sum}	speedup	χειρότερο	T_{max}	T_{sum}	speedup
A	V0.25	10,02	100,11	8,89	F3	10,16	100,79	8,77
B	R5	236,61	2363,03	9,82	R3	236,92	2364,26	9,8
C	R2	4209,80	42094,39	9,90	V0.3	4224,95	42242,94	9,86

Πίνακας 7. Καλύτερο και χειρότερο σχήμα ανάθεσης για 10 επεξεργαστές

περιοχή	καλύτερο	T_{max}	T_{sum}	speedup	χειρότερο	T_{max}	T_{sum}	speedup
A	V0.15	30,71	92,12	2,90	F1	30,88	92,64	2,88
B	R4	781,68	2344,81	2,97	V0.3	784,22	2352,48	2,96
C	F4	13983,25	41948,65	2,98	V0.3	14098,45	42294,89	2,96

Πίνακας 8. Καλύτερο και χειρότερο σχήμα ανάθεσης για 3 επεξεργαστές

4 Σύστημα Πολλαπλών Πρακτόρων για την Κατανεμημένη Εκτέλεση του Αλγορίθμου Βελτιστοποίησης Σμήνους Σωματιδίων

Στο παρόν κεφάλαιο παρουσιάζεται ένα κατανεμημένο σύστημα που αναπτύχθηκε στα πλαίσια της διατριβής για την εκτέλεση του αλγορίθμου Βελτιστοποίησης Σμήνους Σωματιδίων, με βάση το υπολογιστικό παράδειγμα των κινητών πρακτόρων. Ο αλγόριθμος αυτός είναι ένας στοχαστικός αλγόριθμος βελτιστοποίησης που στηρίζεται σε έναν μεγάλο αριθμό εκτιμήσεων μιας συνάρτησης κόστους, ο οποίος μπορεί να γίνει εξαιρετικά απαιτητικός σε υπολογιστικό χρόνο καθώς το πρόβλημα βελτιστοποίησης γίνεται πολυπλοκότερο. Παρόλα αυτά, οι διαδοχικοί επαναληπτικοί υπολογισμοί της συνάρτησης κόστους είναι ανεξάρτητοι μεταξύ τους και μπορούν να εκτελεστούν παράλληλα. Το σύστημα πολλαπλών πρακτόρων που αναπτύχθηκε επιτυγχάνει τη μείωση του χρόνου εκτέλεσης του αλγορίθμου, επιτρέποντας την εκτέλεσή του σε ένα σύνολο ετερογενών υπολογιστών που συνδέονται μέσω δικτύου και παρέχοντας ευκολία και ευελιξία στην εγκατάσταση, καθώς και τη δυνατότητα για εύκολη επέκταση. Παρουσιάζονται λεπτομέρειες υλοποίησης, καθώς και αποτελέσματα σχετικά με την απόδοση του συστήματος για την επίλυση προβλήματος βελτιστοποίησης διαγράμματος ακτινοβολίας στοιχειοκεραίας.

4.1 Εισαγωγή

Ένα από τα βασικότερα ζητούμενα κατά τη σχεδίαση ενός συστήματος είναι η βελτιστοποίηση της συμπεριφοράς αυτού ως προς κάποια συγκεκριμένα χαρακτηριστικά

και προδιαγραφές. Τα πρακτικά προβλήματα σε συστήματα ασύρματων επικοινωνιών σε πολλές περιπτώσεις σχετίζονται με την εύρεση κατάλληλων παραμέτρων, διαστάσεων ή γενικότερα δεδομένων εισόδου, τέτοιων ώστε η τελική λύση να ικανοποιεί τις αρχικές προδιαγραφές. Τυπικές περιπτώσεις στις οποίες απαιτείται η βελτιστοποίηση σε ένα τέτοιο σύστημα είναι η εύρεση της βέλτιστης θέσης των σταθμών βάσης, ώστε να επιτυγχάνεται μέγιστη χωρητικότητα συστήματος και η εύρεση των χαρακτηριστικών μιας στοιχειοκεραίας, ώστε να επιτυγχάνεται ένα συγκεκριμένο σύνθετο διάγραμμα ακτινοβολίας.

Η μαθηματική περιγραφή ενός συστήματος περιέχει μεταξύ άλλων και τις παραμέτρους βελτιστοποίησης ενός προβλήματος. Οι παράμετροι αυτοί εμπλέκονται συνήθως σε σύνθετες συναρτήσεις πολλών μεταβλητών, ενώ το ζητούμενο βέλτιστο αναζητείται ανάμεσα στα στάσιμα σημεία των συναρτήσεων αυτών. Στις περιπτώσεις που είναι γνωστές οι μορφές της περιγραφής αυτής, καθώς και των παραγώγων της, είναι δυνατή η χρήση μεθόδων που στηρίζονται στην εύρεση και τον έλεγχο των στάσιμων σημείων αυτών για τη βελτιστοποίηση. Χαρακτηριστικό παράδειγμα είναι η ομάδα μεθόδων κατάβασης κλίσεων (gradient descent methods), οι οποίες στηρίζονται στην πράξη παραγώγισης ως προς τις μεταβλητές - παραμέτρους βελτιστοποίησης. Ένα από τα πλεονεκτήματα αυτών των μεθόδων είναι ο ταχύς χρόνος εκτέλεσης.

Παρόλα αυτά, η ύπαρξη ή η εκ των προτέρων γνώση μίας καλώς ορισμένης, παραγωγίσιμης συνάρτησης, η οποία θα εξαρτάται από τις παραμέτρους βελτιστοποίησης δεν είναι εγγυημένη σε ένα μεγάλο αριθμό προβλημάτων. Επίσης, σε πληθώρα περιπτώσεων, έστω κι αν υπάρχει αυτή η συνάρτηση, παρουσιάζει πυκνές διακυμάνσεις σε τέτοιο βαθμό, ώστε η εξάρτηση από το αρχικό σημείο να είναι καθοριστική, κι επομένως η ποιότητα της προσφερόμενης λύσης τοπικού ακρότατου εξαιρετικά αμφίβολη. Για την εύρεση του ολικού ακρότατου υπό αυτές τις συνθήκες είναι απαραίτητο να χρησιμοποιηθούν στοχαστικές μέθοδοι αναζήτησης, οι οποίες βασίζονται στον τυχαίο καθορισμό των παραμέτρων βελτιστοποίησης και στην αξιολόγηση ποιότητας της λύσεως που παρέχεται. Οι στοχαστικοί αλγόριθμοι δεν χρησιμοποιούν αναλυτική επεξεργασία του προβλήματος (όπως παραγώγιση), αλλά υπολογισμούς των τελικών τιμών. Ένας από τους σημαντικότερους περιορισμούς αυτών των μεθόδων είναι ότι είναι εξαιρετικά απαιτητικές σε υπολογιστικό χρόνο, ο οποίος για την περίπτωση των στοχαστικών μεθόδων είναι πολλές τάξεις μεγέθους μεγαλύτερος από οποιονδήποτε ντετερμινιστικό αλγόριθμο κατάβασης κλίσεων. Η αιτία των απαιτήσεων αυτών είναι οι επαναλαμβανόμενες εκτελέσεις του βασικού προγράμματος επίλυσης για την αποτίμηση των αποτελεσμάτων.

Μια διαδεδομένη μέθοδος στοχαστικής βελτιστοποίησης είναι η Βελτιστοποίηση Σμήνους Σωματιδίων (Particle Swarm Optimization – PSO). Ο αλγόριθμος PSO χρησιμοποιεί ένα πληθυσμό σωματιδίων, τα οποία κινούνται στο χώρο λύσεων με στόχο την εύρεση της ολικά βέλτιστης θέσης-λύσης και συνίσταται στο διαδοχικό υπολογισμό μιας αντικειμενικής συνάρτησης και στην κατάλληλη ανανέωση των θέσεων των σωματιδίων. Όπως όλες οι στοχαστικές μέθοδοι βελτιστοποίησης, έτσι και ο PSO έχει μεγάλες απαιτήσεις σε υπολογιστικό χρόνο. Παρόλα αυτά, πρόκειται για έναν φύσει επαναληπτικό αλγόριθμο. Οι υπολογισμοί της αντικειμενικής συνάρτησης είναι δυνατόν να εκτελεστούν παράλληλα, μειώνοντας έτσι σημαντικά τον απαιτούμενο χρόνο εκτέλεσης για υπολογιστικά απαιτητικά προβλήματα βελτιστοποίησης.

Στα πλαίσια της διατριβής αναπτύχθηκε ένα σύστημα πολλαπλών πρακτόρων για την κατανεμημένη εκτέλεση του αλγορίθμου PSO. Το σύστημα υλοποιήθηκε με βάση την Java και το API που παρέχει η πλατφόρμα JADE [27], [51]. Τα βασικά πλεονεκτήματα της συγκεκριμένης προσέγγισης είναι η ευκολία που παρέχεται για την εγκατάσταση του συστήματος, η δυνατότητα εκτέλεσης σε ετερογενή υπολογιστικά περιβάλλοντα, καθώς και η ευκολία επέκτασης.

4.2 Βελτιστοποίηση Σμήνους Σωματιδίων

Ο PSO [52],[53] είναι ένας στοχαστικός αλγόριθμος βελτιστοποίησης, ο οποίος στηρίζεται στις αρχές κίνησης και ευφυΐας που χαρακτηρίζουν ένα φυσικό σμήνος (π.χ. σμήνος μελισσών ή κοπάδι ψαριών). Για παράδειγμα, σε ένα σμήνος μελισσών, σκοπός των μελισσών είναι η εύρεση της τοποθεσίας με την μέγιστη συγκέντρωση λουλουδιών σε μια συγκεκριμένη περιοχή. Χωρίς εκ των προτέρων γνώση της περιοχής, οι μέλισσες ξεκινούν σε τυχαίες θέσεις και με τυχαίες ταχύτητες την αναζήτηση λουλουδιών. Κάθε μέλισσα έχει μνήμη των τοποθεσιών που η ίδια βρήκε τα περισσότερα λουλούδια, καθώς και επίγνωση των θέσεων που οι υπόλοιπες μέλισσες ανακάλυψαν μεγάλη πυκνότητα λουλουδιών. Με βάση τις παραπάνω γνώσεις, μια μέλισσα βρίσκεται στο δίλημμα να επιστρέψει στη θέση όπου η ίδια βρήκε τα περισσότερα λουλούδια, ή να ανακαλύψει την περιοχή που οι υπόλοιπες μέλισσες έχουν αναφέρει ότι βρήκαν μέγιστη πυκνότητα λουλουδιών. Έτσι, επιταχύνει και στις δύο αυτές κατευθύνσεις μεταβάλλοντας την τροχιά της έτσι ώστε να μετακινηθεί σε κάποια ενδιάμεση θέση, ανάλογα με το βάρος που δίνει στην προσωπική εμπειρία και την κοινωνική επιρροή. Στην πορεία, η μέλισσα ενδέχεται να ανακαλύψει

κάποια τοποθεσία με μεγαλύτερη συγκέντρωση λουλουδιών από αυτήν που είχε βρει προηγουμένως. Σε αυτήν την περίπτωση, θα έλκεται στη συνέχεια από αυτή τη θέση, καθώς και από τη θέση με τη μεγαλύτερη συγκέντρωση για όλο το σμήνος. Φυσικά, υπάρχει και η περίπτωση η μέλισσα να βρεθεί σε μια θέση με τη μεγαλύτερη συγκέντρωση σε ολόκληρο το σμήνος, οπότε ολόκληρο το σμήνος θα έλκεται προς αυτή τη θέση, καθώς και στη θέση όπου η κάθε μέλισσα έχει βρει το προσωπικό της μέγιστο. Με αυτόν τον τρόπο, οι μέλισσες εξερευνούν ολόκληρη την περιοχή. Κάθε φορά που περνούν από μια συγκεκριμένη θέση, την ελέγχουν και τη συγκρίνουν με τις περιοχές μέγιστης συγκέντρωσης που έχουν βρει προηγουμένως, αναζητώντας τη θέση με την καθολικά μέγιστη συγκέντρωση λουλουδιών. Τελικά, όλες οι μέλισσες οδηγούνται σε αυτή τη θέση και δημιουργούν ένα σμήνος γύρω από αυτήν, καθώς δεν μπορούν να βρουν σημεία μεγαλύτερης συγκέντρωσης και έλκονται συνεχώς προς αυτήν.

4.2.1 Βασική Ορολογία

Οι σημαντικότερες έννοιες του βασικού αλγορίθμου PSO είναι οι ακόλουθες:

- *Σωματίδιο (Particle)*: Κάθε οντότητα στο σμήνος (π.χ. μια μέλισσα στη φυσική αναλογία που περιγράφτηκε στην προηγούμενη παράγραφο) αναφέρεται ως σωματίδιο. Όλα τα σωματίδια στο σμήνος λειτουργούν αυτόνομα με βάση την ίδια αρχή: επιταχύνουν προς την προσωπική βέλτιστη θέση και προς τη βέλτιστη θέση ολόκληρου του σμήνους, ενώ συνεχώς ελέγχουν την αξία της θέσης στην οποία βρίσκονται τη δεδομένη στιγμή.
- *Θέση (Position)*: Στην παραπάνω αναλογία η θέση αναφέρεται στην τοποθεσία μιας μέλισσας στην περιοχή, η οποία αναπαριστάται από συντεταγμένες στον τρισδιάστατο χώρο. Γενικά, μπορεί να γίνει επέκταση σε χώρο N διαστάσεων, ανάλογα με το δεδομένο πρόβλημα. Ο χώρος αυτός είναι ο χώρος λύσεων για το υπό βελτιστοποίηση πρόβλημα, ενώ κάθε σύνολο συντεταγμένων αναπαριστά μια πιθανή λύση του προβλήματος. Στην παραπάνω φυσική αναλογία, η λύση είναι μια φυσική τοποθεσία στον τρισδιάστατο χώρο, αλλά με παρόμοιο τρόπο ο χώρος λύσεων μπορεί να αναπαριστά το πλάτος και τη φάση της διέγερσης των στοιχείων μιας στοιχειοκεραίας.
- *Αντικειμενική Συνάρτηση (Fitness Function)*: Η αντικειμενική συνάρτηση παρέχει ένα μέτρο για την εκτίμηση της ποιότητας μιας συγκεκριμένης θέσης. Η συνάρτηση αυτή παίρνει ως είσοδο τις συντεταγμένες της θέσης και επιστρέφει μια τιμή που

αναπαριστά την ποιότητα της θέσης αυτής. Στην παραπάνω φυσική αναλογία η αντικειμενική συνάρτηση είναι η συγκέντρωση των λουλουδιών (όσο υψηλότερη είναι η τιμή της συγκέντρωσης, τόσο καλύτερη θεωρείται η αντίστοιχη θέση). Σε προβλήματα ασύρματων επικοινωνιών, η αντικειμενική συνάρτηση θα μπορούσε να είναι το κέρδος μιας κεραίας, το βάρος, κτλ. Σε γενικές γραμμές, η αντικειμενική συνάρτηση παρέχει μια διεπαφή μεταξύ του φυσικού προβλήματος και του αλγορίθμου βελτιστοποίησης.

- *Προσωπικό Βέλτιστο (Personal Best – pbest)*: Στην παραπάνω αναλογία κάθε μέλισσα έχει μνήμη της θέσης που προσωπικά βρήκε τα περισσότερα λουλούδια. Αυτή η θέση με την βέλτιστη τιμή της αντικειμενική συνάρτησης που προσωπικά ανακάλυψε ένα σωματίδιο ονομάζεται pbest. Κάθε σωματίδιο έχει ένα pbest που καθορίζεται από τη διαδρομή που έχει ακολουθήσει στο χώρο λύσεων. Σε κάθε σημείο της μετέπειτα διαδρομής, το σωματίδιο συγκρίνει την τιμή της αντικειμενικής συνάρτησης για το σημείο αυτό με το pbest και αν η τρέχουσα θέση αντιστοιχεί σε καλύτερη τιμή της αντικειμενική συνάρτησης, το pbest αντικαθιστάται από τη θέση αυτή.
- *Καθολικό Βέλτιστο (Global Best – gbest)*: Κάθε μέλισσα έχει επιπλέον γνώση της θέσης μέγιστης συγκέντρωσης λουλουδιών που έχει ανακαλυφθεί από ολόκληρο το σμήνος. Αυτή η θέση της βέλτιστης τιμής της αντικειμενική συνάρτησης ονομάζεται gbest. Προφανώς υπάρχει μόνο ένα gbest για ολόκληρο το σμήνος, από το οποίο έλκεται το κάθε σωματίδιο. Σε κάθε σημείο της διαδρομής του, κάθε σωματίδιο συγκρίνει την τιμή της αντικειμενική συνάρτησης της τρέχουσας θέσης με την αντίστοιχη τιμή του gbest και αν αυτή η τιμή βρεθεί καλύτερη, τότε το gbest αντικαθίσταται από την τρέχουσα θέση.

4.2.2 Αλγόριθμος

Ο αλγόριθμος PSO συνίσταται στα ακόλουθα βήματα:

1. Τυχαία αρχικοποίηση των θέσεων και των ταχυτήτων των σωματιδίων του σμήνους: Κάθε σωματίδιο ξεκινάει σε μια τυχαία θέση και με τυχαία ταχύτητα, τόσο ως προς το μέτρο, όσο και ως προς την κατεύθυνση. Καθώς η αρχική αυτή θέση είναι η μοναδική θέση που έχει επισκεφθεί το σωματίδιο στην αρχή του αλγορίθμου, η θέση αυτή γίνεται αυτόματα και το pbest του σωματιδίου, ενώ το gbest επιλέγεται ανάμεσα στις αρχικές αυτές θέσεις.

2. Συστηματική μετακίνηση των σωματιδίων μέσα στο χώρο λύσεων: Κάθε σωματίδιο μέσα στο σμήνος μετακινείται ένα προς ένα κατά μια μικρή απόσταση. Τα ακόλουθα βήματα εφαρμόζονται σε κάθε σωματίδιο ξεχωριστά:
    a. Υπολογισμός του fitness κάθε σωματιδίου και σύγκριση με τα pbest και gbest: η αντικειμενική συνάρτηση, χρησιμοποιώντας τις τρέχουσες συντεταγμένες του σωματιδίου στο χώρο λύσης, επιστρέφει μια τιμή για την τρέχουσα θέση. Αν η τιμή αυτή είναι καλύτερη από την τιμή που αντιστοιχεί στο pbest ή/και στο gbest, τότε η αντίστοιχη θέση αντικαθίσταται από την τρέχουσα.
    b. Ανανέωση της ταχύτητας του σωματιδίου: ο χειρισμός της ταχύτητας ενός σωματιδίου είναι το βασικότερο λειτουργικό στοιχείο της διαδικασίας βελτιστοποίησης. Η ταχύτητα μεταβάλλεται ανάλογα με τις σχετικές θέσεις των pbest και gbest και επιταχύνει προς τις κατευθύνσεις αυτών των βέλτιστων θέσεων με βάση την ακόλουθη εξίσωση:

$$u_n = w \cdot u_n + c_1 \cdot rand() \cdot (pbest_n - x_n) + c_2 \cdot rand() \cdot (gbest_n - x_n),$$

όπου u_n είναι η ταχύτητα του σωματιδίου στη διάσταση n και x_n η θέση του σωματιδίου στη διάσταση n. Η ανανέωση αυτή γίνεται για κάθε μία από τις διαστάσεις του N-διάστατου χώρου βελτιστοποίησης. Όπως προκύπτει από την παραπάνω εξίσωση, η νέα ταχύτητα είναι ίση με το άθροισμα της προηγούμενης ταχύτητας πολλαπλασιασμένης επί έναν συντελεστή w και δύο παραγόντων που την αυξάνουν προς τα gbest και pbest στη συγκεκριμένη διάσταση. Τα c_1 και c_2 είναι συντελεστές που καθορίζουν τη σχετική έλξη προς τα pbest και gbest και οι οποίοι συχνά αναφέρονται αντίστοιχα ως "γνωστικός" (cognitive) και "κοινωνικός" (social) συντελεστής. Ο συντελεστής c_1 καθορίζει το βαθμό που ένα σωματίδιο επηρεάζεται από τη μνήμη της προσωπικής βέλτιστης θέσης, ενώ ο c_2 καθορίζει το βαθμό που αυτό επηρεάζεται από το υπόλοιπο σμήνος. Μεγάλες τιμές για το c_1 οδηγούν στην καλύτερη εξερεύνηση του χώρου λύσεων (exploration), ενώ μεγάλες τιμές για το c_2 οδηγούν σε καλύτερη εκμετάλλευση του υποτιθέμενου ολικού μεγίστου (exploitation). Η συνάρτηση rand() επιστρέφει έναν αριθμό στο διάστημα (0,1) και προσομοιώνει την μερικώς απρόβλεπτη συμπεριφορά που παρατηρείται σε ένα φυσικό σμήνος. Το w αναπαριστά ένα βάρος αδράνειας

(inertial weight) του σωματιδίου και παίρνει τιμές μεταξύ του 0 και του 1. Το w καθορίζει σε ποιο βαθμό το σωματίδιο παραμένει στην αρχική του πορεία, ανεπηρέαστο από την έλξη των gbest και pbest και είναι ένας παράγοντας που επίσης μπορεί να χρησιμοποιηθεί για τη στάθμιση του exploration και του exploitation. Από την παραπάνω σχέση προκύπτει ακόμη ότι τα σωματίδια που βρίσκονται περισσότερο απομακρυσμένα από τα pbest και gbest δέχονται τη μεγαλύτερη έλξη από αυτές τις θέσεις και κινούνται προς αυτές με μεγαλύτερες ταχύτητες σε σχέση με τα σωματίδια που βρίσκονται πλησιέστερα. Ένα σωματίδιο συνεχίζει να αυξάνει την ταχύτητά του προς την κατεύθυνση των pbest και gbest μέχρι να τις προσπεράσουν, οπότε έλκονται προς την αντίθετη κατεύθυνση. Για να γίνεται πιο αποδοτική η εξερεύνηση του χώρου, καθώς και για να μην υπερβαίνουν εύκολα τα σωματίδια τα όρια του χώρου λύσεων, συνήθως ορίζεται μια μέγιστη τιμή u_{max} για την ταχύτητα.

c. Μετακίνηση του σωματιδίου: Αφού υπολογιστεί η νέα ταχύτητα, το σωματίδιο μετακινείται στη νέα του θέση. Η νέα ταχύτητα εφαρμόζεται για ένα συγκεκριμένο χρονικό διάστημα Δt (το οποίο συνήθως επιλέγεται ίσο με ένα) και η νέα συντεταγμένη x_n υπολογίζεται για κάθε διάσταση του N-διάστατου χώρου με βάση την ακόλουθη σχέση:

$$x_n = x_n + u_n \cdot \Delta t$$

Σε περίπτωση που η νέα θέση του σωματιδίου είναι εκτός των ορίων μιας συγκεκριμένης μεταβλητής, γίνεται εφαρμογή κατάλληλων οριακών συνθηκών (π.χ. μηδενισμός της συνιστώσας της ταχύτητας στη συγκεκριμένη διεύθυνση).

3. Επανάληψη: Αφού οι διαδικασίες που περιγράφτηκαν παραπάνω ολοκληρωθούν, η ίδια διαδικασία επαναλαμβάνεται από το Βήμα 2. Τα σωματίδια μετακινούνται για διακριτά χρονικά διαστήματα και στη συνέχεια αποτιμάται η νέα θέση τους. Με αυτόν τον τρόπο λαμβάνεται ένα στιγμιότυπο ολόκληρου του σμήνους σε συγκεκριμένες χρονικές στιγμές. Οι θέσεις όλων των σωματιδίων στη συνέχεια αποτιμούνται και γίνονται διορθώσεις/ανανεώσεις στις θέσεις των pbest και gbest, πριν τα σωματίδια αφεθούν να μετακινηθούν μέχρι την λήψη του επόμενου στιγμιότυπου. Ο κύκλος αυτός επαναλαμβάνεται μέχρι να ικανοποιηθεί κάποιο κριτήριο τερματισμού. Το κριτήριο αυτό μπορεί να είναι ένας μέγιστος αριθμός επαναλήψεων, η λήψη μιας ικανοποιητικής τιμής για τη αντικειμενική συνάρτηση, ή

η εμφάνιση μιας ελάχιστης τυπικής απόκλισης μεταξύ των επαναλήψεων (σε αυτήν την περίπτωση θεωρείται ότι έχει επιτευχθεί σύγκλιση του αλγορίθμου στο ολικό βέλτιστο της περιοχής).

4.3 Παράλληλος Αλγόριθμος

Όπως προκύπτει από την περιγραφή της προηγούμενης ενότητας, ο PSO είναι φύσει επαναληπτικός αλγόριθμος. Εξετάζοντας τη σχέση και τις αλληλεξαρτήσεις μεταξύ των διαφορετικών βημάτων του PSO, προκύπτει ότι μόνο το στάδιο της ανανέωσης των ταχυτήτων και των θέσεων των σωματιδίων εξαρτάται από προηγούμενα βήματα. Αντίθετα, οι υπολογισμοί των τιμών της αντικειμενική συνάρτησης για τα σωματίδια του σμήνους στο αντίστοιχο βήμα είναι εντελώς ανεξάρτητοι μεταξύ τους, πράγμα που επιτρέπει την παράλληλη εκτέλεσή τους. Με βάση αυτή τη διαπίστωση, ο αλγόριθμος μπορεί να καταταχθεί στο σχήμα διαχωριζόμενων εξαρτήσεων (separable dependencies pattern [54]), σύμφωνα με το οποίο ένα τμήμα του αλγορίθμου μπορεί να διαχωριστεί περαιτέρω σε ένα σύνολο ανεξάρτητων εργασιών, ενώ ένα άλλο τμήμα του απαιτεί την ικανοποίηση ενός αριθμού αλληλεξαρτήσεων.

```
repeat
        parallel and distributed fitness evaluation
        for i ← 1 to number of individuals do
                f_i ← G(p⃗_i)
        end for

        synchronization barrier
        …
        find best in neighbourhood
        …
        velocity and position update
until criterion
```

Listing 3. Ψευδοκώδικας παράλληλου σύγχρονου αλγορίθμου PSO

Πράγματι, ο PSO μπορεί να αποσυντεθεί σε ένα εντελώς παράλληλο (embarrassingly parallel) τμήμα για τον υπολογισμό της αντικειμενικής συνάρτησης και ένα τμήμα συγχρονισμένης ανανέωσης των ταχυτήτων και θέσεων των σωματιδίων, το οποίο εκτελείται μία μόνο φορά σε κάθε επανάληψη. Ο ψευδοκώδικας για την παράλληλη

εκτέλεση της σύγχρονης μορφής του PSO (όπου η ανανέωση των ταχυτήτων και θέσεων πραγματοποιείται ταυτόχρονα για όλα τα σωματίδια, χωρίς να απαιτείται ανταλλαγή πληροφορίας σχετικά με τα pbest και gbest) φαίνεται στο Listing 3. Σημειώνεται ότι οι υπολογισμοί για την ανανέωση των θέσεων και των ταχυτήτων των σωματιδίων είναι τυπικά υπολογιστικά πολύ απλούστερες σε σύγκριση με τους υπολογισμούς των τιμών της αντικειμενικής συνάρτησης και για αυτό το λόγο η παράλληλη υλοποίηση του αλγορίθμου αναμένεται να παρουσιάζει ικανοποιητική απόδοση.

4.4 Υλοποίηση Συστήματος

Η υλοποίηση του κατανεμημένου συστήματος PSO βασίζεται στο υπολογιστικό παράδειγμα των κινητών πρακτόρων. Οι ιδιότητες που αποδίδονται στους κινητούς πράκτορες καθιστούν τη συγκεκριμένη τεχνολογία ιδιαίτερα ελκυστική για την παροχή δικτυακών υπηρεσιών κατανεμημένης επεξεργασίας. Συγκεκριμένα, κάποια από τα χαρακτηριστικά που μπορούν να διευκολύνουν την ανάπτυξη και χρήση τέτοιων υπηρεσιών είναι και τα ακόλουθα:

- ελάχιστες απαιτήσεις δικτυακής υποδομής: ένα σύστημα κινητών πρακτόρων μπορεί να αρκεστεί σε μια απλή σύνδεση TCP/IP μεταξύ των κόμβων που ανήκουν σε αυτό για τη μετακίνηση των πρακτόρων και την ανταλλαγή πληροφοριών μεταξύ τους
- ανεξαρτησία μεταξύ υλοποίησης συστήματος κινητών πρακτόρων και της υποδομής υλικού στην οποία αυτό θα εκτελεστεί: όπως αναφέρθηκε, τα περισσότερα συστήματα πρακτόρων υλοποιούνται σε Java, η οποία είναι εγγενώς ανεξάρτητη από την υποκείμενη πλατφόρμα. Έτσι, ο προγραμματιστής μπορεί να αναπτύξει ένα σύστημα πρακτόρων που χωρίς περαιτέρω τροποποιήσεις θα μπορεί να εκτελείται σε ετερογενή περιβάλλοντα.
- διαφανής επικοινωνία μεταξύ των πρακτόρων: τα συστήματα πρακτόρων επιτρέπουν την ομοιόμορφη επικοινωνία μεταξύ των πρακτόρων, ανεξάρτητα από την τοποθεσία του δικτύου που αυτοί βρίσκονται. Τις λεπτομέρειες ανταλλαγής πληροφορίας αναλαμβάνει το μεσισμικό του συστήματος, αποκρύπτοντας από τον προγραμματιστή και το χρήστη τις ιδιαιτερότητες των διαφόρων δικτυακών υποδομών.

- ενσωμάτωση του αλγορίθμου κατανομής, συγχρονισμού και διαχείρισης των παράλληλων εργασιών στις λειτουργίες των πρακτόρων: η οργάνωση του πλάνου κατανομής με βάση τους πόρους που παρέχονται κατά την εκτέλεση γίνεται από τους ίδιους τους πράκτορες, με βάση την ευφυΐα που έχει δοθεί σε αυτούς κατά την ανάπτυξη του συστήματος. Αυτό απαλλάσσει το χρήστη από την απαίτηση ιδιαίτερων γνώσεων παράλληλης επεξεργασίας.

Για την υλοποίηση του συστήματος χρησιμοποιήθηκε το μοντέλο αφέντη-εργάτη. Σύμφωνα με αυτό το μοντέλο, την εκτέλεση του αλγορίθμου χειρίζονται δύο είδη οντοτήτων: ο αφέντης και ο εργάτης, οι οποίες αντιστοιχούν σε συγκεκριμένες διεργασίες. Υπάρχει μόνο μία διεργασία-αφέντης, η οποία είναι υπεύθυνη για τη δημιουργία των εργατών, καθώς και για τον έλεγχο της ροής εκτέλεσης του αλγορίθμου εφαρμόζοντας συγκεκριμένες απαιτήσεις συγχρονισμού. Κάθε διεργασία-εργάτης υπολογίζει επαναληπτικά την τιμή της αντικειμενικής συνάρτησης για συγκεκριμένες θέσεις του χώρου λύσεων και επιστρέφει το αποτέλεσμα στην διεργασία-αφέντη. Η διεργασία-αφέντης είναι ένας στάσιμος πράκτορας (CoordinatorAgent), ενώ κάθε διεργασία-εργάτης είναι ένας κινητός πράκτορας (ParticleAgent) που έχει την ικανότητα να μεταναστεύει σε απομακρυσμένους κόμβους του δικτύου και να εκτελεί τοπικά τις εργασίες που του έχουν ανατεθεί. Ο υπολογισμός των τιμών της αντικειμενικής συνάρτησης ανατίθεται στους ParticleAgents, ενώ η ανανέωση θέσεων και ταχυτήτων των σωματιδίων γίνεται από τον CoordinatorAgent στο τέλος κάθε επανάληψης του αλγορίθμου.

4.4.1 Η Πλατφόρμα JADE

Το σύστημα JADE (Java Agent Development Framework) είναι μια πλατφόρμα λογισμικού που παρέχει βασικές λειτουργίες μεσισμικού ανεξάρτητες από συγκεκριμένες εφαρμογές, οι οποίες διευκολύνουν την υλοποίηση κατανεμημένων εφαρμογών σύμφωνα με το υπολογιστικό παράδειγμα των κινητών πρακτόρων. Απλοποιεί την υλοποίηση συστημάτων πολλαπλών πρακτόρων μέσω ενός μεσισμικού συμβατού με τις προδιαγραφές FIPA και ενός συνόλου εργαλείων για τις διαδικασίες αποσφαλμάτωσης και εγκατάστασης. Η πλατφόρμα των πρακτόρων μπορεί να κατανέμεται σε διαφορετικά μηχανήματα (για τα οποία δεν υπάρχει απαίτηση να έχουν το ίδιο λειτουργικό σύστημα), ενώ ο έλεγχος της λειτουργίας της μπορεί να γίνεται μέσω μιας απομακρυσμένης γραφικής διεπαφής χρήστη. Το σύστημα είναι υλοποιημένο αποκλειστικά με τη γλώσσα προγραμματισμού Java και

μοναδική απαίτηση για την εκτέλεσή του είναι η ύπαρξη του περιβάλλοντος εκτέλεσης Java Runtime Environment 1.4 και άνω.

Το JADE περιλαμβάνει τα εξής:

- Ένα περιβάλλον εκτέλεσης, το οποίο παρέχει τους απαραίτητους μηχανισμούς για την εκτέλεση των πρακτόρων και των λειτουργιών που επιτελούνται στον κύκλο ζωής τους (π.χ. επικοινωνία, μετανάστευση).
- Μια βιβλιοθήκη κλάσεων που μπορούν να χρησιμοποιηθούν ή/και να επεκταθούν για την ανάπτυξη συγκεκριμένων υλοποιήσεων πρακτόρων.
- Ένα σύνολο από γραφικά εργαλεία για τη διαχείριση και την επίβλεψη των δραστηριοτήτων των ενεργών πρακτόρων.

Στις επόμενες παραγράφους περιγράφονται τα βασικότερα χαρακτηριστικά της πλατφόρμας.

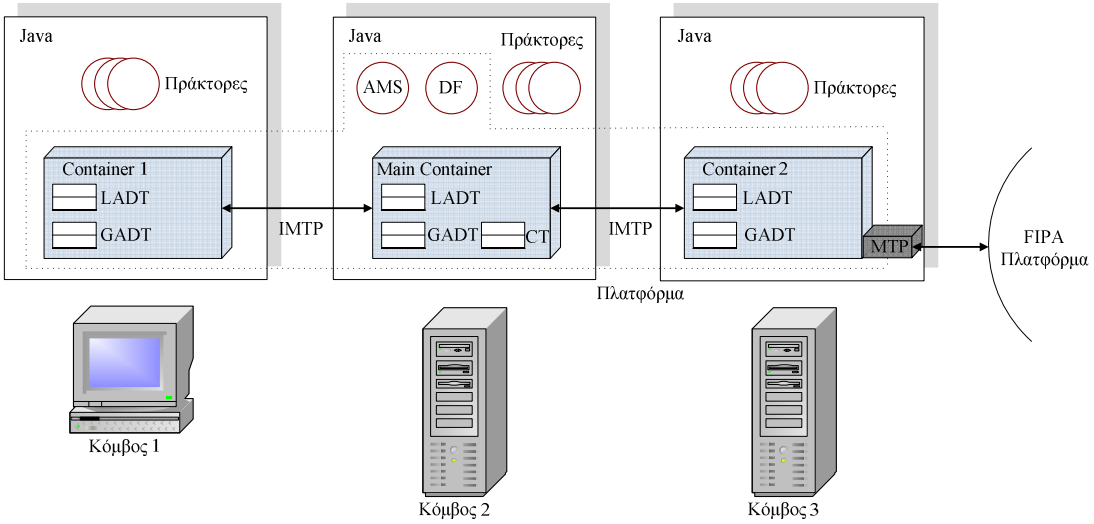
4.4.1.1 Περιβάλλον Εκτέλεσης

Το περιβάλλον εκτέλεσης μιας πλατφόρμας JADE απεικονίζεται στο Σχήμα 28. Μια πλατφόρμα JADE αποτελείται από *agent containers*, τα οποία μπορούν να βρίσκονται κατανεμημένα πάνω σε ένα δίκτυο. Οι πράκτορες ζουν και εργάζονται μέσα στα containers, τα οποία είναι διεργασίες Java που παρέχουν το περιβάλλον εκτέλεσης και όλες τις υπηρεσίες που απαιτούνται για την φιλοξενία και εκτέλεση των πρακτόρων. Υποχρεωτική μέσα σε κάθε πλατφόρμα είναι η ύπαρξη ενός και μοναδικού main container, που αποτελεί το σημείο έναρξης της πλατφόρμας. Το main container είναι η πρώτη οντότητα της πλατφόρμας που εκτελείται, ενώ όλα τα υπόλοιπα containers που θα εκτελεστούν στην συνέχεια πρέπει να συνδεθούν με το main container, καταχωρώντας τον εαυτό τους σε αυτό. Οι αρμοδιότητες του main container είναι οι ακόλουθες:

- διαχείριση του πίνακα των containers (container table – CT), ο οποίος είναι ένα μητρώο αναφορών και διευθύνσεων όλων των containers που απαρτίζουν την πλατφόρμα
- διαχείριση του καθολικού πίνακα περιγραφής των πρακτόρων (global agent descriptor table – GADT), ο οποίος είναι ένα μητρώο όλων των πρακτόρων που είναι ενεργοί στην πλατφόρμα και που περιέχει πληροφορία για την τρέχουσα κατάσταση και την τοποθεσία που αυτοί βρίσκονται.

- φιλοξενία των πρακτόρων AMS (Agent Management System) και DF (Directory Facilitator), οι οποίοι παρέχουν υπηρεσίες διαχείρισης των πρακτόρων, λευκών σελίδων (white pages), και κίτρινων σελίδων (yellow pages) αντίστοιχα.

Για να μην καθίσταται το main container σημείο συμφόρησης, κάθε container διαχειρίζεται τοπικά ένα αντίγραφο του GADT. Σε γενικές γραμμές, οι λειτουργίες της πλατφόρμας δεν εμπλέκουν το main container, αλλά μονάχα τα τοπικά αντίγραφα του GADT και τα δύο containers που φιλοξενούν τους πράκτορες που αποτελούν το υποκείμενο και το αντικείμενο μιας λειτουργίας (π.χ. αποστολέα και δέκτη ενός μηνύματος). Όταν ένα container χρειάζεται να ανακαλύψει πού βρίσκεται ο αποδέκτης ενός μηνύματος, πρώτα γίνεται αναζήτηση σε έναν τοπικό πίνακα περιγραφής πρακτόρων (local agent description table – LADT), και μόνο αν η αναζήτηση αποτύχει επιχειρείται επικοινωνία με το main container, το αποτέλεσμα από την οποία διατηρείται στο LADT για μελλοντική χρήση.



Σχήμα 28. Περιβάλλον Εκτέλεσης του JADE

Η ταυτότητα των πρακτόρων περιέχεται σε ένα Αναγνωριστικό Πράκτορα (Agent Identifier – AID), το οποίο αποτελείται από έναν συγκεκριμένο αριθμό πεδίων, σύμφωνα με τα πρότυπα FIPA. Τα βασικότερα πεδία ενός αντικειμένου AID είναι το όνομα του πράκτορα και οι διευθύνσεις του. Το όνομα είναι ένα καθολικά μοναδικό χαρακτηριστικό που κατασκευάζεται από τη συνένωση ενός ονόματος που δίνεται από το χρήστη στον πράκτορα (το οποίο χρησιμοποιείται για την επικοινωνία εντός της πλατφόρμας) και του ονόματος της πλατφόρμας. Οι διευθύνσεις του πράκτορα κληρονομούνται από την πλατφόρμα και κάθε μία από αυτές αντιστοιχεί σε ένα τερματικό σημείο του Πρωτοκόλλου

Μεταφοράς Μηνύματος (Message Transport Protocol – MTP), μέσω του οποίου μπορούν να αποσταλούν και να ληφθούν μηνύματα συμβατά με τα πρότυπα FIPA. Το MTP είναι μια υπηρεσία που παρέχεται από την πλατφόρμα για τη μεταφορά των μηνυμάτων μεταξύ πρακτόρων μέσα σε μια πλατφόρμα ή μεταξύ πρακτόρων που βρίσκονται σε διαφορετικές πλατφόρμες. Τα μηνύματα παρέχουν ένα "φάκελο" μεταφοράς που αποτελείται από ένα σύνολο παραμέτρων, όπως για παράδειγμα τον αποδέκτη του μηνύματος. Όπως φαίνεται και στο Σχήμα 28, το JADE παρέχει μια συγκεκριμένη υλοποίηση του MTP, το IMTP (Internal Message Transport Protocol), το οποίο χρησιμοποιείται για την επικοινωνία των πρακτόρων που ανήκουν στην ίδια πλατφόρμα και το οποίο στηρίζεται στην τεχνολογία του RMI.

Με την εκκίνηση του main container, αυτόματα δημιουργούνται οι εξής δύο ειδικοί πράκτορες από το σύστημα του JADE, οι ρόλοι των οποίων καθορίζονται από τα πρότυπα FIPA:

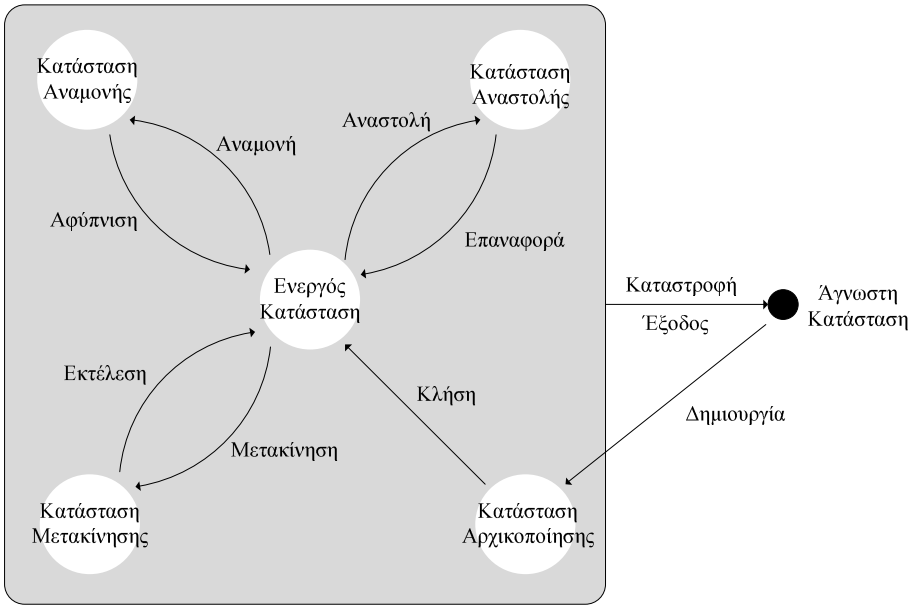
- Ο πράκτορας AMS (Agent Management System), ο οποίος επιβλέπει ολόκληρη την πλατφόρμα. Μέσω του AMS μπορούν οι πράκτορες να αποκτήσουν πρόσβαση στην υπηρεσία λευκών σελίδων της πλατφόρμας ή να διαχειριστούν τον κύκλο ζωής τους. Στο JADE ο AMS αποτελεί την διευθύνουσα αρχή της πλατφόρμας και είναι ο μόνος πράκτορας που μπορεί να εκτελέσει λειτουργίες διαχείρισης της πλατφόρμας, όπως για παράδειγμα δημιουργία και καταστροφή πρακτόρων, καταστροφή containers και τερματισμό ολόκληρης της πλατφόρμας. Κατά την εκκίνηση κάθε πράκτορα, είναι απαραίτητη η καταχώρησή του στον AMS, ώστε να αποκτήσει ένα AID που θα χρησιμοποιηθεί σε οποιαδήποτε μελλοντική λειτουργία αυτός εμπλέκεται.
- Ο πράκτορας DF (Directory Facilitator), ο οποίος υλοποιεί την υπηρεσία κίτρινων σελίδων της πλατφόρμας και χρησιμοποιείται από κάθε πράκτορα που επιθυμεί να καταχωρήσει τις υπηρεσίες που αυτός παρέχει ή να αναζητήσει υπηρεσίες που παρέχονται από άλλους πράκτορες. Επιπλέον, υπάρχει η δυνατότητα εγγραφής στον DF, ώστε να ειδοποιούνται οι πράκτορες όταν καταχωρείται μια νέα υπηρεσία ή όταν γίνεται κάποια αλλαγή σε μια ήδη καταχωρημένη υπηρεσία, με βάση συγκεκριμένα κριτήρια.

4.4.1.2 Κύκλος Ζωής Πρακτόρων - Μοντέλο Εκτέλεσης

Τα βασικότερα στοιχεία που χαρακτηρίζουν έναν πράκτορα λογισμικού στην πλατφόρμα JADE είναι οι ιδιότητές του, η κατάστασή του και η συμπεριφορά του. Οι ιδιότητες του

πράκτορα καθορίζονται από τον προγραμματιστή κατά την υλοποίησή του. Αντίθετα, η κατάσταση μπορεί να πάρει συγκεκριμένες μόνο τιμές και με προκαθορισμένους τρόπους μετάβασης, όπως ορίζεται από τα πρότυπα FIPA. Ο κύκλος ζωής ενός πράκτορα και οι δυνατές καταστάσεις στις οποίες αυτός μπορεί να περιέλθει φαίνονται στο Σχήμα 29.

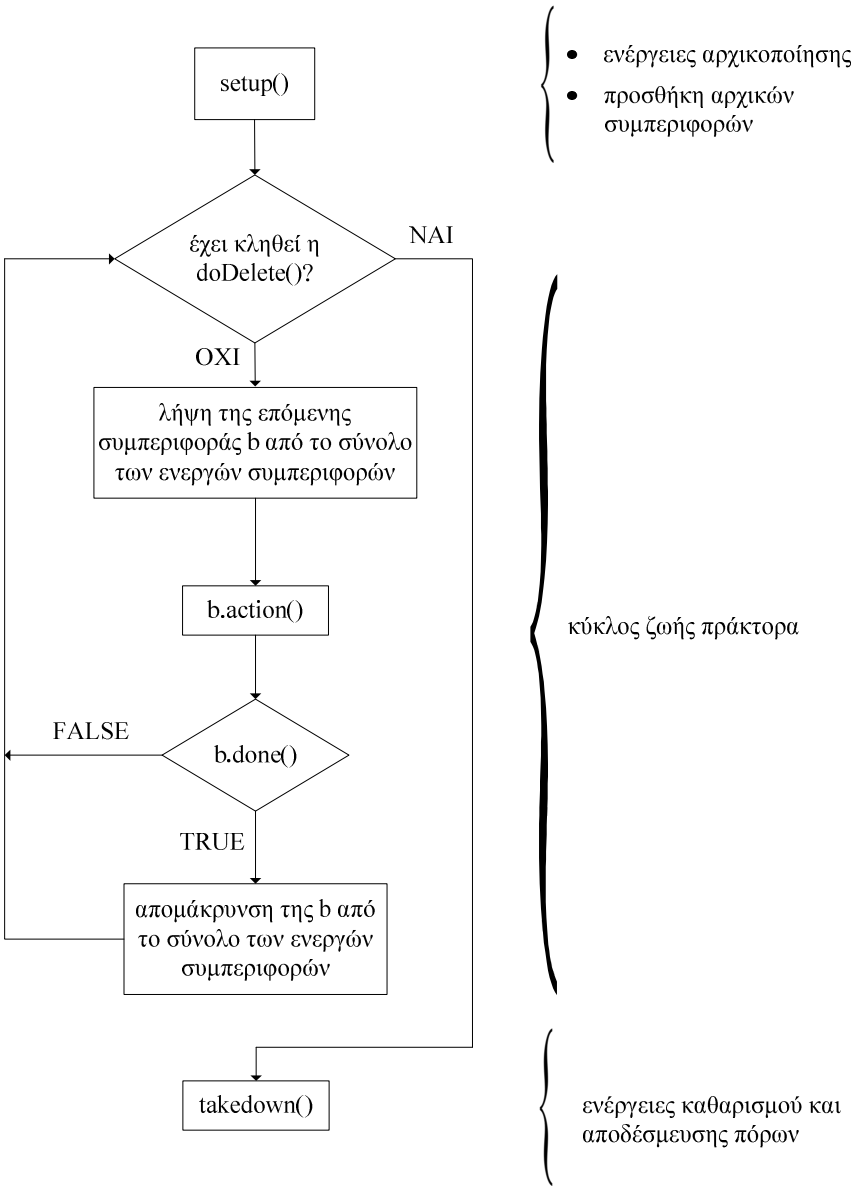
Η κύρια κατάσταση είναι η ενεργός, ενώ οι υπόλοιπες αποτελούν μεταβατικές καταστάσεις, στις οποίες ο πράκτορας μπορεί να περιέλθει λόγω ερεθισμάτων του περιβάλλοντος ή δικών του αποφάσεων (π.χ. η κατάσταση μετακίνησης υποδεικνύει ότι ένας πράκτορας βρίσκεται σε διαδικασία μετανάστευσης σε απομακρυσμένη τοποθεσία - ο πράκτορας εισέρχεται σε αυτήν την κατάσταση μετά την κλήση των αντίστοιχων μεθόδων για μετακίνηση και επιστρέφει στην ενεργό κατάσταση όταν ολοκληρωθεί η διαδικασία μεταφοράς και κληθεί η μέθοδος συνέχισης της εκτέλεσής του).



Σχήμα 29. Κύκλος Ζωής Πρακτόρων κατά FIPA

Το υπολογιστικό μοντέλο εκτέλεσης ενός πράκτορα στο JADE στηρίζεται στην ταυτόχρονη εκτέλεση πολλαπλών εργασιών, οι οποίες στην ορολογία του JADE ονομάζονται “συμπεριφορές” (behaviours). Η εκτέλεση ενός πράκτορα ξεκινάει με κλήση της μεθόδου setup(), στην οποία ο προγραμματιστής κάνει τις απαραίτητες προεργασίες και δημιουργεί το “χαρακτήρα” του νέου πράκτορα μέσω συμπεριφορών. Κάθε λειτουργικότητα/υπηρεσία που παρέχει ένας πράκτορας πρέπει να υλοποιηθεί ως μία ή

περισσότερες συμπεριφορές. Η εκτέλεση των συμπεριφορών αυτών πραγματοποιείται αυτόματα με διάφανο για τον προγραμματιστή τρόπο από έναν εσωτερικό δρομολογητή της βασικής αφαιρετικής κλάσης jade.core.Agent. Μια συμπεριφορά αναπαριστά μια εργασία που μπορεί να εκτελέσει ένας πράκτορας και υλοποιείται ως αντικείμενο μιας κλάσης που επεκτείνει την κλάση jade.core.behaviours.Behaviour.



Σχήμα 30. Διάγραμμα Ροής Εκτέλεσης Πράκτορα

Κάθε πράκτορας διατηρεί ένα σύνολο ενεργών συμπεριφορών και για να εκτελέσει ένας πράκτορας μια συμπεριφορά, χρειάζεται να προστεθεί αυτή σε αυτό το σύνολο με κλήση

της μεθόδου addBehaviour() της κλάσης Agent. Κάθε κλάση συμπεριφοράς απαιτείται να υλοποιεί τη μέθοδο action(), η οποία ορίζει τις λειτουργίες που εκτελούνται όταν εκτελείται η συγκεκριμένη συμπεριφορά, καθώς και τη μέθοδο done(), η οποία επιστρέφει μια boolean τιμή και υποδεικνύει αν η συγκεκριμένη συμπεριφορά έχει ολοκληρωθεί και συνεπώς αν πρέπει να αφαιρεθεί από το σύνολο των ενεργών συμπεριφορών. Το διάγραμμα ροής της εκτέλεσης ενός πράκτορας στο JADE φαίνεται στο Σχήμα 30.

Αφού ολοκληρωθεί η μέθοδος setup(), ο πράκτορας είναι ενεργός, διαθέτει ένα σύνολο συμπεριφορών και έχει πρόσβαση σε μία ουρά μηνυμάτων. Με την ολοκλήρωση των εργασιών του, ο πράκτορας μπορεί να καταστραφεί μέσω κλήσεως της μεθόδου doDelete().

4.4.1.3 Επικοινωνία

Η επικοινωνία μεταξύ των πρακτόρων είναι ένα από τα βασικότερα χαρακτηριστικά του JADE και έχει υλοποιηθεί με βάση τις προδιαγραφές FIPA. Το μοντέλο της επικοινωνίας στηρίζεται στην ασύγχρονη ανταλλαγή μηνυμάτων. Κάθε πράκτορας διαθέτει μια προσωπική ουρά μηνυμάτων, στην οποία το περιβάλλον εκτέλεσης τοποθετεί μηνύματα που έχουν ληφθεί από άλλους πράκτορες. Κάθε φορά που λαμβάνεται νέο μήνυμα, ο πράκτορας-παραλήπτης ειδοποιείται. Ο τρόπος που ο παραλήπτης θα παραλάβει το μήνυμα από την ουρά για επεξεργασία αποτελεί επιλογή του προγραμματιστή κατά την υλοποίηση του πράκτορα.

Η δομή του μηνύματος ορίζεται από την FIPA-ACL προδιαγραφή. Κάθε μήνυμα περιλαμβάνει μια σειρά από πεδία, όπως για παράδειγμα τον αποστολέα, τη λίστα των παραληπτών, το σκοπό της επικοινωνίας (π.χ. REQUEST ή INFORM) και το περιεχόμενο (content), το οποίο περιέχει την πληροφορία της επικοινωνίας. Η πραγματική θέση των παραληπτών και ο τρόπος μεταφοράς είναι τελείως διαφανής στον προγραμματιστή, αφού ο AMS αναλαμβάνει αυτές τις διαδικασίες αυτόματα. Από την άλλη πλευρά, η λήψη του μηνύματος απαιτεί πρόσβαση του πράκτορα στην ιδιωτική ουρά μηνυμάτων του. Ο τρόπος προσπέλασης της ουράς δεν είναι μοναδικός. Πρακτικά χρησιμοποιούνται οι ακόλουθες μέθοδοι:

- Με μπλοκάρισμα (μέθοδος blockingReceive()) ή ασύγχρονα (μέθοδος receive()). Η blocking μέθοδος προκαλεί αναστολή όλων των δραστηριοτήτων και των συμπεριφορών του πράκτορα, μέχρι την εμφάνιση του ζητούμενου μηνύματος. Αντίθετα η non-blocking έκδοση απλά επιστρέφει null αν το μήνυμα δεν υπάρχει στην ουρά.

- Και οι δύο αυτοί μέθοδοι μπορούν να επεκταθούν με δυνατότητες ταιριάσματος προτύπου (pattern matching) προκειμένου να αναζητηθούν μηνύματα με συγκεκριμένο περιεχόμενο.
- Η πρόσβαση με μπλοκάρισμα δύναται να έχει προθεσμία λήξης (timeout). Εισάγοντας ένα χρονικό περιθώριο, εξασφαλίζεται η έξοδος της μεθόδου ακόμη και με αποτυχία, οπότε επιστρέφεται null.

4.4.1.4 Μετανάστευση

Το API του JADE παρέχει επίσης την ιδιαίτερα σημαντική λειτουργία μετακίνησης πράκτορα σε άλλη τοποθεσία. Υποστηρίζεται η ασθενής μορφή μετανάστευσης και η εκτέλεση του πράκτορα ξεκινάει από την αρχή μετά από κάθε μετακίνηση, συνεπώς ένας πράκτορας πρέπει να έχει υλοποιηθεί ως μια μηχανή πεπερασμένων καταστάσεων (finite state machine) για να διατηρεί κάποια πληροφορία κατάστασης.

Η μετακίνηση ενός πράκτορα περιλαμβάνει την αποστολή του κώδικα και της κατάστασής του μέσω ενός καναλιού δικτύου, πράγμα που συνεπάγεται ότι ο προγραμματιστής του κινητού πράκτορα πρέπει να διαχειριστεί τις διαδικασίες σειριοποίησης και αποσειριοποίησης. Κάποιοι από τους πόρους που ανήκουν στον πράκτορα θα μεταφερθούν μαζί του, ενώ κάποιοι άλλοι θα αποσυνδεθούν πριν τη μεταφορά και θα ξανασυνδεθούν στον προορισμό (με τον ίδιο τρόπο που γίνεται διάκριση μεταξύ των μεταβατικών (transient) και μη μεταβατικών πεδίων στο Java Serialization API).

Η διαδικασία μετανάστευσης υλοποιείται όπως η κλωνοποίηση ενός πράκτορα. Οι μέθοδοι beforeMove(), afterMove() (beforeClone(), afterClone() για κλωνοποίηση αντίστοιχα) αναπρογραμματίζονται (override) από τον προγραμματιστή για την εκτέλεση ειδικών λειτουργιών αμέσως πριν και αμέσως μετά την μετανάστευση (κλωνοποίηση), η οποία λαμβάνει χώρα με κλήση της doMove() (doClone()).

4.4.2 Κατανεμημένες Οντότητες

Για την υλοποίηση της κατανεμημένης πλατφόρμας PSO υιοθετήθηκε το μοντέλο αφέντη-εργάτη. Την εκτέλεση του αλγορίθμου χειρίζονται δύο οντότητες: ο αφέντης και ο εργάτης, οι οποίες αντιστοιχούν σε συγκεκριμένες διεργασίες. Υπάρχει μόνο μία διεργασία-αφέντης, η οποία συνοπτικά είναι υπεύθυνη για τη δημιουργία των εργατών, καθώς και για τον έλεγχο της ροής εκτέλεσης του αλγορίθμου εφαρμόζοντας συγκεκριμένες απαιτήσεις συγχρονισμού. Κάθε διεργασία-εργάτης υπολογίζει επαναληπτικά την τιμή της

αντικειμενικής συνάρτησης για συγκεκριμένες θέσεις του χώρου λύσεων και επιστρέφει το αποτέλεσμα στη διεργασία-αφέντη. Η διεργασία-αφέντης είναι ένας στάσιμος πράκτορας (CoordinatorAgent), ενώ κάθε διεργασία-εργάτης είναι ένας κινητός πράκτορας (ParticleAgent) που έχει την ικανότητα να μεταναστεύει σε απομακρυσμένους κόμβους του δικτύου και να εκτελεί τοπικά τις εργασίες που του έχουν ανατεθεί. Ο υπολογισμός των τιμών της αντικειμενικής συνάρτησης ανατίθεται στους ParticleAgents, ενώ η ανανέωση θέσεων και ταχυτήτων των σωματιδίων γίνεται από τον CoordinatorAgent.

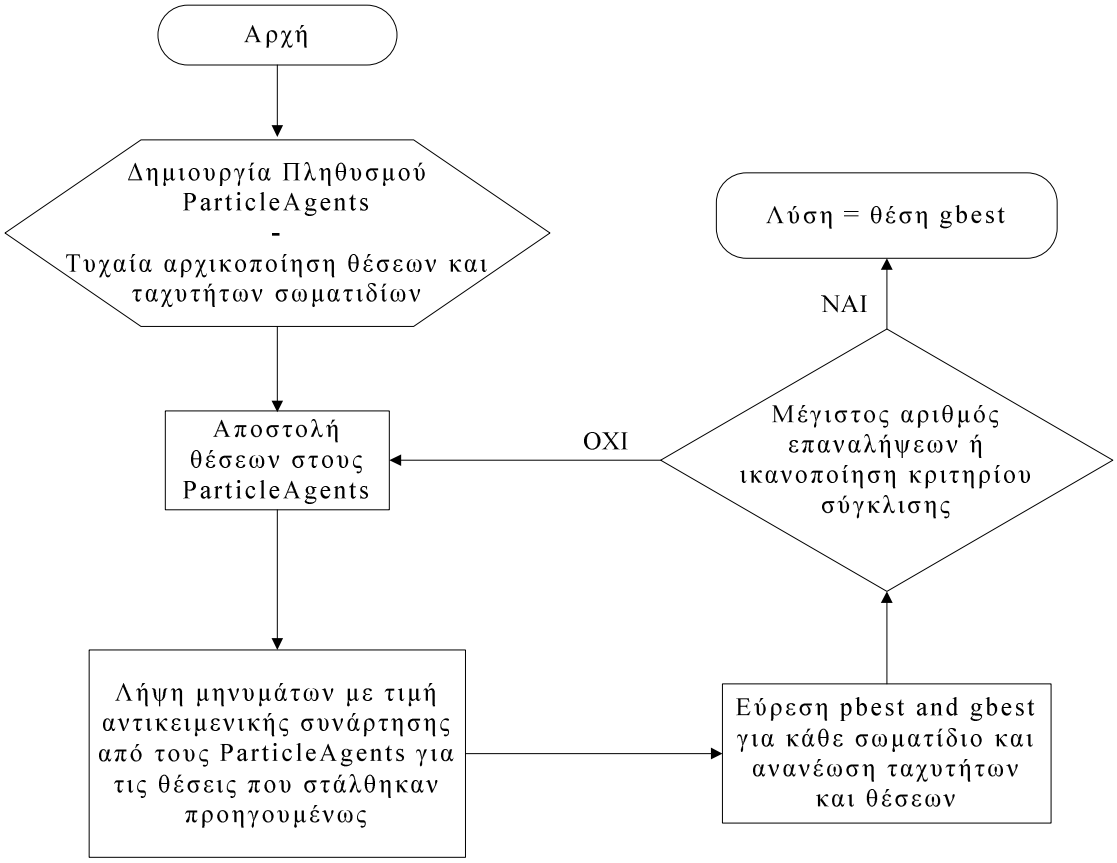
4.4.2.1 CoordinatorAgent

Ο CoordinatorAgent είναι υπεύθυνος για την αρχικοποίηση και τη συνολική διαχείριση της εκτέλεσης του αλγορίθμου. Συγκεκριμένα, οι εργασίες του είναι οι ακόλουθες:

1. Setup του προβλήματος: οι παράμετροι του αλγορίθμου διαβάζονται από αρχεία παραμέτρων και οι θέσεις και οι ταχύτητες των σωματιδίων αρχικοποιούνται τυχαία.
2. Δημιουργία και αποστολή μηνυμάτων θέσης στους ParticleAgents: ο CoordinatorAgent πυροδοτεί τους ParticleAgents στέλνοντας ένα μήνυμα που περιέχει τις συντεταγμένες της θέσης του χώρου λύσης που ο ParticleAgent θα χρησιμοποιήσει για τον υπολογισμό της τιμής της αντικειμενικής συνάρτησης. Σε περίπτωση που έχουμε φτάσει στο μέγιστο αριθμό επαναλήψεων ή το κριτήριο σύγκλισης του αλγορίθμου έχει ικανοποιηθεί, στέλνεται ένα συγκεκριμένο μήνυμα, ώστε να τερματιστεί η εκτέλεση του ParticleAgent.
3. Συγκέντρωση των υπολογισμένων τιμών της αντικειμενικής συνάρτησης από τους ParticleAgents: μετά την αποστολή των μηνυμάτων θέσης, ο CoordinatorAgent περιμένει ένα μήνυμα από όλους τους ParticleAgents, που περιέχει την τιμή της αντικειμενικής συνάρτησης που υπολογίστηκε από αυτούς. Αυτός είναι ο μόνος συγχρονισμός που απαιτείται ώστε να εκτελείται με συνέπεια ο παράλληλος αλγόριθμος.
4. Εύρεση του προσωπικού βέλτιστου σημείου κάθε σωματιδίου και του ολικού βέλτιστου/βέλτιστου γειτονιάς: αφού ληφθούν τα παραπάνω μηνύματα από όλους τους ParticleAgents για μια συγκεκριμένη επανάληψη, ο CoordinatorAgent έχει όλα τα απαραίτητα δεδομένα για την εύρεση των νέων βέλτιστων θέσεων για το κάθε σωματίδιο ξεχωριστά, αλλά και για ολόκληρο το σμήνος ή τις γειτονιές του.
5. Ανανέωση των θέσεων και των ταχυτήτων των σωματιδίων: αφού βρεθούν οι τιμές των pbest και gbest, ο CoordinatorAgent ανανεώνει τις θέσεις και τις ταχύτητες των

σωματιδίων με βάση τις εξισώσεις που αναφέρθηκαν κατά την περιγραφή του αλγορίθμου σε προηγούμενη παράγραφο. Οι ανανεωμένες θέσεις αποστέλλονται στη συνέχεια στους ParticleAgents για τον υπολογισμό των τιμών της αντικειμενικής συνάρτησης στην επόμενη επανάληψη.

Ο κύκλος ζωής του CoordinatorAgent απεικονίζεται στο Σχήμα 31.



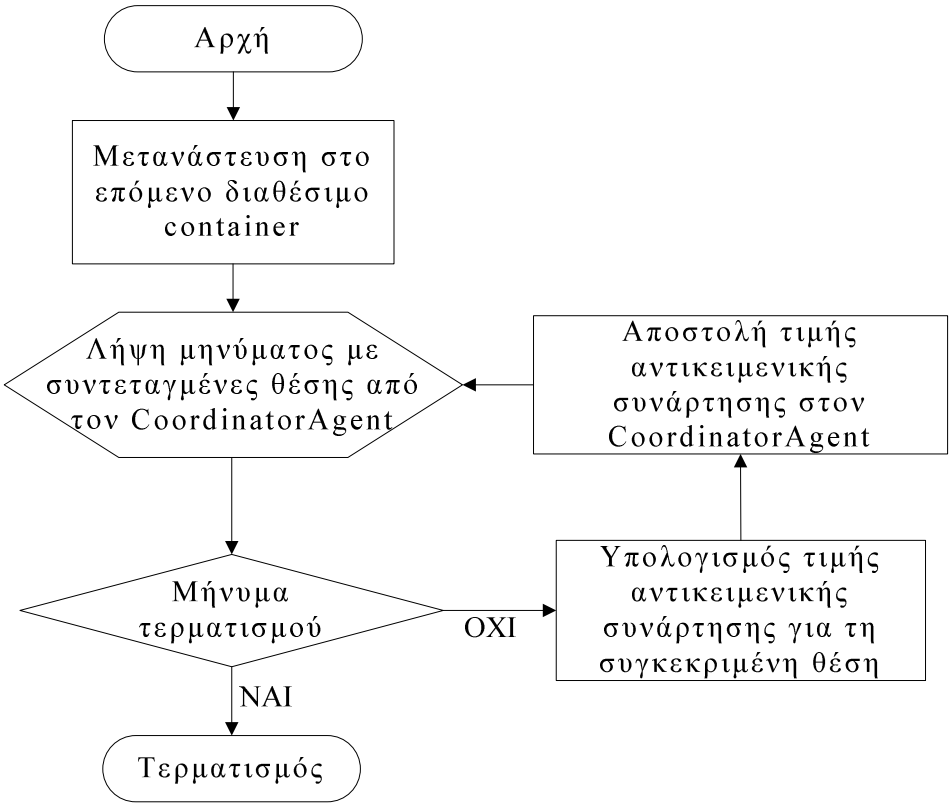
Σχήμα 31. Κύκλος Ζωής του CoordinatorAgent

4.4.2.2 ParticleAgent

Οι εργασίες που πραγματοποιεί ο ParticleAgent είναι οι ακόλουθες:

1. Μετανάστευση στον επόμενο διαθέσιμο κόμβο: τη στιγμή που ένα αντικείμενο ParticleAgent στιγμιοποιείται, δίνεται σε αυτό η διεύθυνση ενός container. Ο ParticleAgent χρησιμοποιεί αυτή τη διεύθυνση για να μεταναστεύσει στον αντίστοιχο κόμβο του δικτύου, όπου εκτελεί τοπικά τις εργασίες που του έχουν ανατεθεί. Η λίστα με τις διευθύνσεις των διαθέσιμων κόμβων αποκτάται στην αρχή

της εκτέλεσης από τον CoordinatorAgent και οι ParticleAgents αποστέλλονται κυκλικά σε αυτούς τους κόμβους.



Σχήμα 32. Κύκλος Ζωής του ParticleAgent

2. Αναμονή μηνύματος θέσης από τον CoordinatorAgent: αφού ο ParticleAgent μεταναστεύσει σε έναν απομακρυσμένο κόμβο, αυτός παραμένει ανενεργός μέχρι να ληφθεί ένα μήνυμα από τον CoordinatorAgent. Αυτό το μήνυμα λειτουργεί ως διεγερτικό για τον ParticleAgent και περιέχει τις συντεταγμένες του σημείου του χώρου λύσης όπου θα υπολογιστεί η αντικειμενική συνάρτηση. Επίσης, υπάρχει και η ειδική περίπτωση που το μήνυμα υποδεικνύει τον τερματισμό του ParticleAgent.
3. Υπολογισμός της τιμής της αντικειμενικής συνάρτησης και αποστολή του αποτελέσματος στον CoordinatorAgent: όταν ένα μήνυμα που περιέχει συντεταγμένες μιας θέσης του χώρου λύσης λαμβάνεται από τον CoordinatorAgent, υπολογίζεται η αντίστοιχη τιμή της αντικειμενικής συνάρτησης και προωθείται στον CoordinatorAgent. Ο ParticleAgent γίνεται και πάλι ανενεργός, μέχρι να ληφθεί άλλο μήνυμα από τον CoordinatorAgent.

Ο κύκλος ζωής του ParticleAgent απεικονίζεται στο Σχήμα 32.

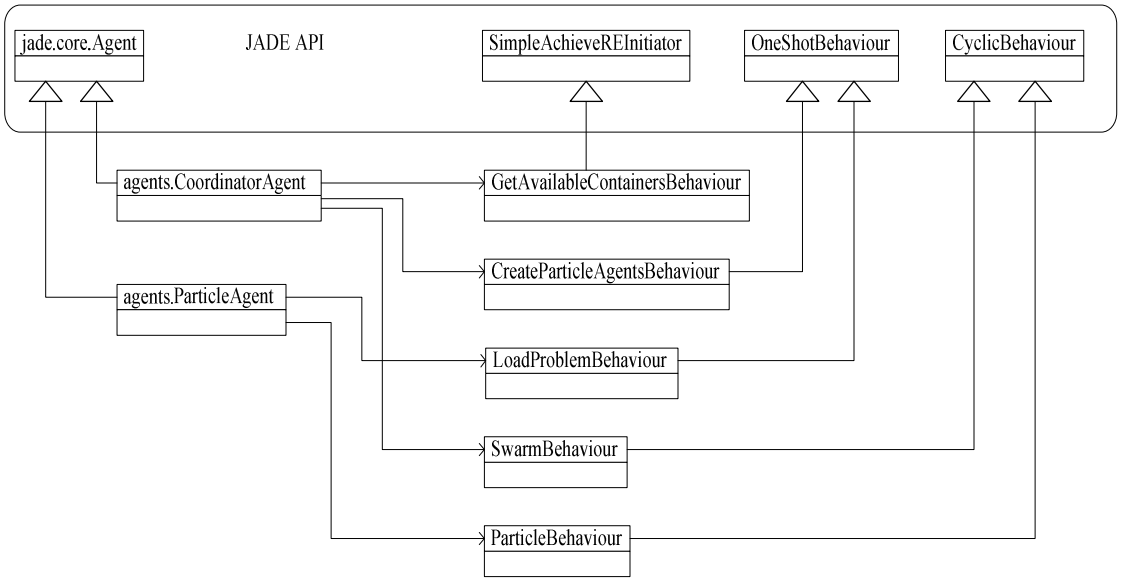
```
public class CoordinatorAgent extends jade.core.Agent
{
   protected void setup()
   {
      read parameters and input data
      add behavior for getting available Agent Containers  (using a library provided
                                                protocol for interacting with the AMS)
      add behavior for creating ParticleAgents (passing to them the address of the Agent
                                                 Container where they will migrate)
      add behavior for performing the parallel PSO steps (assign fitness calculations to
                                              ParticleAgents, receive fitness values,
                                              update positions and velocities and apply
                                              synchronization requirements)
   }
}
```

Listing 4. Λειτουργικότητα του CoordinatorAgent

4.5 Υλοποίηση

Οι ψευδοκώδικες υλοποίησης των παραπάνω κατανεμημένων οντοτήτων με βάση το API του JADE φαίνονται στα Listing 4 και Listing 5. Όλες οι λειτουργίες της πλατφόρμας υλοποιούνται από αυτές τις δύο κλάσεις πρακτόρων, οι οποίες είναι υλοποιήσεις της αφαιρετικής κλάσης jade.core.Agent. Ο κύκλος ζωής τους καθορίζεται από τις συμπεριφορές της πλατφόρμας, οι οποίες αποτελούν επεκτάσεις των βασικών κλάσεων συμπεριφορών που παρέχουν οι βιβλιοθήκες του JADE. Συγκεκριμένα, η συμπεριφορά GetAvailableContainersBehaviour επεκτείνει την jade.proto.SimpleAchieveREInitiator (η οποία υλοποιεί το request πρωτόκολλο αλληλεπίδρασης της FIPA) και χρησιμοποιεί την οντολογία FIPA_SL0 για επικοινωνία με τον AMS. Οι υπόλοιπες συμπεριφορές επεκτείνουν τις κλάσεις jade.core.behaviours.OneShotBehaviour και jade.core.behaviours.CyclicBehaviour.

Το UML διάγραμμα των κλάσεων των πρακτόρων και των συμπεριφορών της πλατφόρμας φαίνεται στο Σχήμα 33.



Σχήμα 33. UML διάγραμμα των κλάσεων πρακτόρων και συμπεριφορών της πλατφόρμας

```
public class ParticleAgent extends jade.core.Agent{
    protected void setup()    {
        migrate to the remote Agent Container indicated by the CoordinatorAgent
    }
    protected void afterMove()    {
        load problem classes (using the Java Reflection API)
        add behavior for performing fitness calculations (receive assignment from
                                                CoordinatorAgent, calculate fitness and
                                                return the computed fitness value)
    }
}
```

Listing 5. Λειτουργικότητα του ParticleAgent

4.6 Εγκατάσταση – Ρυθμίσεις

Η κατανεμημένη πλατφόρμα βελτιστοποίησης μπορεί να εγκατασταθεί σε δικτυακά συνδεδεμένους υπολογιστές, οι οποίοι όμως δεν απαιτείται να ανήκουν στο ίδιο τοπικό δίκτυο. Η μόνη απαίτηση είναι να διαθέτουν στατικές διευθύνσεις IP και, σε περίπτωση που βρίσκονται πίσω από κάποιο τείχος προστασίας, να επιτρέπουν την εισερχόμενη και εξερχόμενη κίνηση στις θύρες που χρησιμοποιεί το JADE και το RMI.

```
public class ArrayProblem extends Problem
{
	public ArrayProblem()
	{
		// setup problem specifications
		setupProblem();
		// define parameter space
		defineOptimizationConstraints();
	}
	public double fitness(double[] solution)
	{
		double penalty = 0.0;
		for (int k=0; k<diagramsNumber; k++)
		{
			// decode array parameters for diagram k
			ArrayParameters parameters = decodeParameters(solution);
			// compute array for new paramers set
			array.compute(parameters)
			// compute N penalty factors
			double[] p = new double[N];
			f[1] = array.penalty1();
			...
			f[N] = array.penaltyN();
			// computing weighted sum
			double f = 0.0;
			for (int i=0; i<N; i++)
				penalty += w[i]*f[i];
		}
		// return normalized fitness value
		return 1.0 / (1.0 + Math.sqrt(penalty));
	}
}
```

Listing 6. Ορισμός Προβλήματος Στοιχειοκεραίας

Επίσης, η μόνη απαίτηση για προεγκατεστημένο λογισμικό είναι το Περιβάλλον Εκτέλεσης της Java (τουλάχιστον έκδοση 1.4), ανεξάρτητα από το υποκείμενο λειτουργικό σύστημα. Εξαίρεση αποτελεί η περίπτωση που απαιτείται η κλήση native βιβλιοθηκών ή εκτελέσιμων αρχείων. Σε αυτήν την περίπτωση πρέπει να παρέχονται αρχεία συμβατά με το λειτουργικό σύστημα του κάθε κόμβου.

Για την κατανεμημένη εκτέλεση ενός προβλήματος βελτιστοποίησης, εκτός από τα βασικά συστατικά της πλατφόρμας (βιβλιοθήκες JADE, κώδικας κατανεμημένης πλατφόρμας) απαιτείται η ύπαρξη μιας βιβλιοθήκης που περιέχει το ίδιο το προς βελτιστοποίηση πρόβλημα. Τα παραπάνω μπορούν να οργανώνονται σε διαφορετικά JAR

αρχεία. Η διαδικασία εγκατάστασης είναι απλή: όλα τα αρχεία βιβλιοθηκών απαιτούνται μόνο στον κόμβο που θα φιλοξενήσει τον CoordinatorAgent, ενώ οι υπόλοιποι κόμβοι χρειάζονται μόνο το JAR αρχείο των βιβλιοθηκών του JADE. Η κατανομή των βιβλιοθηκών πραγματοποιείται με χρήση απλών scripts που αναλαμβάνουν την κατάλληλη προετοιμασία του περιβάλλοντος εκτέλεσης σε κάθε κόμβο.

Η διεπαφή ενός δεδομένου προβλήματος με την κατανεμημένη πλατφόρμα παρέχεται μέσω της abstract κλάσης Problem. Ο προγραμματιστής απαιτείται να επεκτείνει κατάλληλα τη συγκεκριμένη κλάση, υλοποιώντας την abstract μέθοδο *double fitness(double[] solution)*, η οποία χρησιμοποιείται για τον ορισμό της αντικειμενικής συνάρτησης. Για παράδειγμα, στο Listing 6 φαίνεται ο ορισμός του προβλήματος βελτιστοποίησης διαγράμματος στοιχειοκεραίας που χρησιμοποιήθηκε σε διάφορα πειράματα, τα αποτελέσματα από την εκτέλεση των οποίων παρουσιάζονται στην επόμενη παράγραφο.

```
problem = problem.ArrayProblem
parameters = parametersArray.xml
jar = ArrayProblem.jar
generations = 400
```

Listing 7. Αρχείο Ιδιοτήτων για την αρχικοποίηση της κατανεμημένης πλατφόρμας

Οι ρυθμίσεις που απαιτούνται για την αρχικοποίηση της πλατφόρμας περιγράφονται σε αρχεία ιδιοτήτων Java και κατάλληλα XML αρχεία, στα οποία πρέπει να έχει πρόσβαση ο CoordinatorAgent. Κατά την εκκίνηση της πλατφόρμας, διαβάζονται οι παράμετροι που καθορίζονται στο αρχείο ιδιοτήτων, όπως το όνομα του JAR αρχείου που περιέχει την υλοποίηση του προβλήματος, η μέγιστη διάρκεια (επαναλήψεις) της διαδικασίας βελτιστοποίησης και μια αναφορά στο XML αρχείο ρυθμίσεων που περιέχει τις παραμέτρους του PSO (Listing 7).

Στο Listing 8 φαίνεται ένα παράδειγμα XML αρχείου παραμέτρων για την εκτέλεση του PSO.

```
<parameters>
    <pop__size>200</pop__size>
    <c1>2.0</c1>
    <c2>2.0</c2>
    <v_max>0.1</v_max>
    <w>1.0</w>
    <params>
        <entry>
            <string>w2</string>
            <double>0.9</double>
        </entry>
        <entry>
            <string>w1</string>
            <double>0.4</double>
        </entry>
    </params>
</parameters>
```

Listing 8. XML Αρχείο Παραμέτρων του PSO

4.7 Πειράματα - Χρόνοι Εκτέλεσης

Για την αποτίμηση της κατανεμημένης πλατφόρμας, θεωρήθηκε πρόβλημα σύνθεσης επίπεδης στοιχειοκεραίας αποτελούμενης από πέντε δίπολα. Ζητούμενο είναι ο καθορισμός του συνόλου των παραμέτρων της στοιχειοκεραίας, ώστε να παραχθούν τέσσερα διαγράμματα ακτινοβολίας στο επίπεδο x-y με τους αντίστοιχους λοβούς στις κατευθύνσεις 0^o, 90^o, 180^o και 270^o και με μέγιστο επιτρεπόμενο πλάτος λοβού 80^o. Οι παράμετροι προς βελτιστοποίηση είναι οι σχετικές θέσεις των στοιχείων στο επίπεδο και το σύνολο των συντελεστών διέγερσης για κάθε διάγραμμα ακτινοβολίας [55].

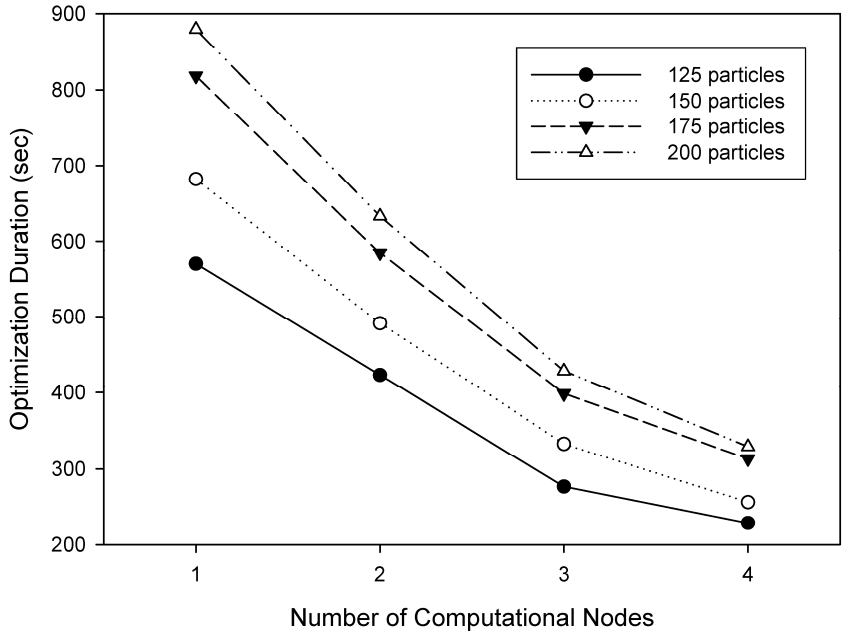
Ένας αριθμός σεναρίων βελτιστοποίησης εκτελέστηκαν για μεταβλητό αριθμό πληθυσμού από 125 έως 200 σωματίδια. Τα συγκεκριμένα μεγέθη πληθυσμών επιτρέπουν την ικανοποιητική εξερεύνηση του πολυ-διάστατου χώρου λύσεων, ενώ διαφορετικός αριθμός σωματιδίων χρησιμοποιήθηκε για την καλύτερη μελέτη της ικανότητας κλιμάκωσης της πλατφόρμας. Οι τιμές των παραμέτρων c_1 και c_2 ορίστηκαν ίσες με 2,0 , η μέγιστη τιμή για το u_{max} ίση με 0,1 , ενώ ο αριθμός των επαναλήψεων σε κάθε σενάριο είναι 400. Για την εκτέλεση των σεναρίων χρησιμοποιήθηκαν τέσσερις υπολογιστικοί κόμβοι Pentium IV 2.8 GHz με 1 GB RAM, συνδεδεμένοι με 100Mbps Ethernet LAN.

Οι συνολικοί χρόνοι εκτέλεσης των διαφορετικών σεναρίων σε διαφορετικό αριθμό κόμβων και οι αντίστοιχες επιταχύνσεις φαίνονται στον Πίνακα 9. Στο Σχήμα 34 φαίνονται και σχηματικά οι χρόνοι αυτοί.

	Χρόνος (sec)				**Επιτάχυνση**			
Πληθυσμός	125	150	175	200	125	150	175	200
Σειριακά	569,97	682,25	818,56	879,84	1,00			
2 κόμβοι	422,87	491,42	584,53	633,15	1,35	1,39	1,40	1,39
3 κόμβοι	276,75	332,77	398,56	428,24	2,06	2,05	2,05	2,06
4 κόμβοι	228,69	256,13	313,58	329,36	2,49	2,66	2,61	2,67

Πίνακας 9. Χρόνοι Εκτέλεσης και επιταχύνσεις για διαφορετικά σενάρια

Το συγκεκριμένο πρόβλημα σύνθεσης κεραίας δεν είναι εξαιρετικά απαιτητικό σε υπολογιστικό χρόνο, καθώς η μέγιστη διάρκεια σειριακής εκτέλεσης για τον μεγαλύτερο πληθυσμό που χρησιμοποιήθηκε είναι περίπου 15 λεπτά. Λόγω του ότι ο μέσος χρόνος εκτέλεσης για τους υπολογισμούς που αντιστοιχούν σε κάθε σωματίδιο είναι μικρός (περίπου 12 msec), η συνολική επιτάχυνση της κατανεμημένης εκτέλεσης επηρεάζεται σε αρκετό βαθμό από την επιβάρυνση της επικοινωνίας κατά την ανταλλαγή μηνυμάτων.



Σχήμα 34. Χρόνοι εκτέλεσης βελτιστοποίησης για διαφορετικό αριθμό κόμβων

Αυτό που προκύπτει από το παραπάνω σχήμα είναι ότι η απόδοση του αλγορίθμου για διαφορετικά μεγέθη πληθυσμού είναι συγκρίσιμη. Η κύρια διαφορά είναι ότι για μεγαλύτερους πληθυσμούς ο ρυθμός μείωσης του χρόνου βελτιστοποίησης είναι επίσης μεγαλύτερος. Αυτό οφείλεται στο ότι το υπολογιστικό φορτίο αυξάνεται σε σύγκριση με την επιβάρυνση της επικοινωνίας, λόγω του ότι το μέγεθος της επεξεργασίας που μεσολαβεί μεταξύ των ανταλλαγών μηνυμάτων για την εύρεση των pbest και gbest είναι μεγαλύτερο. Για μικρούς πληθυσμούς η επιβάρυνση των μηνυμάτων γίνεται μεγαλύτερη συγκριτικά με το υπολογιστικό φορτίο. Επιπλέον, ενώ αυξάνεται ο αριθμός των κόμβων, η μείωση που παρατηρείται στο χρόνο εκτέλεσης μεταξύ διαδοχικών αριθμών κόμβων μειώνεται. Αυτό επίσης οφείλεται στην επιβάρυνση λόγω της ανταλλαγής μηνυμάτων.

5 Τεχνολογία Υπολογιστικού Πλέγματος

Η έννοια του υπολογιστικού πλέγματος εμφανίστηκε για πρώτη φορά στις αρχές της δεκαετίας του 1990 με μεταφορική σημασία, δηλώνοντας μια τεχνολογία που θα επιτρέπει την πρόσβαση σε υπολογιστική ισχύ και άλλες υπολογιστικές υπηρεσίες, με την ίδια ευκολία που έχουμε πρόσβαση σε ηλεκτρική ισχύ μέσω του πλέγματος ηλεκτροδότησης. Σήμερα υπάρχουν διάφοροι ορισμοί για το υπολογιστικό πλέγμα, οι χαρακτηριστικότεροι και επικρατέστεροι των οποίων και οι εξής:

- Ένα υπολογιστικό πλέγμα είναι ένα είδος παράλληλου και κατανεμημένου συστήματος που επιτρέπει τη διανομή, την επιλογή και τη συνάθροιση γεωγραφικά διασκορπισμένων και κατανεμημένων αυτόνομων πόρων, δυναμικά κατά το χρόνο εκτέλεσης και ανάλογα με τη διαθεσιμότητά τους, την απόδοσή τους, το κόστος τους και διάφορες άλλες παραμέτρους ποιότητας υπηρεσίας που μπορεί να καθορίζει ένας χρήστης [56].
- Το υπολογιστικό πλέγμα είναι μια υπηρεσία για διαμοιρασμό υπολογιστικής ισχύος και χώρου αποθήκευσης δεδομένων πάνω από το διαδίκτυο (CERN).
- Η τεχνολογία του υπολογιστικού πλέγματος έχει προκύψει ως ένα σημαντικό νέο πεδίο της κατανεμημένης επεξεργασίας (distributed computing), το οποίο διαφοροποιείται από τις υπόλοιπες κατανεμημένες τεχνολογίες λόγω της εστίασης στον ευρείας κλίμακας διαμοιρασμό δεδομένων, στις καινοτόμες εφαρμογές και σε ορισμένες περιπτώσεις στον υψηλής απόδοσης προσανατολισμό [57].

Βασικό στοιχείο ενός πλέγματος είναι η έννοια της υποδομής (infrastructure), η οποία σχετίζεται με την ευρείας κλίμακας συγκέντρωση πόρων, οι οποίοι μπορεί να είναι υπολογιστικοί κύκλοι, δεδομένα, αισθητήρες ή υπηρεσίες λογισμικού. Αυτού του είδους η

συγκέντρωση απαιτεί μια ιδιαίτερα σημαντική υποδομή υλικού για την επίτευξη των απαραίτητων διασυνδέσεων, καθώς και μια υποδομή λογισμικού για την παρακολούθηση και τον έλεγχο του συνόλου που προκύπτει. Ένα ακόμη θεμελιώδες στοιχείο του πλέγματος είναι η ανάγκη για αξιόπιστες υπηρεσίες (dependable services). Οι χρήστες απαιτούν διαβεβαιώσεις ότι θα λάβουν προβλεπόμενη, συνεχή και υψηλού επιπέδου απόδοση από τα διαφορετικά στοιχεία που αποτελούν το πλέγμα. Ένα δεύτερο σημαντικό μέλημα είναι η συμβατότητα των παρεχόμενων υπηρεσιών (consistency of service). Απαιτούνται προτυποποιημένες υπηρεσίες, οι οποίες είναι προσβάσιμες μέσω προτυποποιημένων διεπαφών και οι οποίες λειτουργούν με συγκεκριμένες προτυποποιημένες παραμέτρους. Τέλος, ένα πλέγμα πρέπει να παρέχει καθολική πρόσβαση (pervasive access), δηλαδή οι προσφερόμενες υπηρεσίες πρέπει να είναι ανά πάσα στιγμή διαθέσιμες, ανεξάρτητα από το περιβάλλον στο οποίο κινείται κανείς.

Σε γενικές γραμμές πάντως, όταν αναφερόμαστε στο πλεγματικό πρόβλημα (grid problem), εννοούμε τον ευέλικτο, ασφαλή και συντονισμένο διαμοιρασμό υπολογιστικών πόρων μεταξύ δυναμικών ομάδων χρηστών, ιδρυμάτων και πόρων [57]. Βασικά επιμέρους ζητήματα του προβλήματος είναι η μοναδική πιστοποίηση (unique authentication), η εξουσιοδότηση (authorization), η πρόσβαση σε πόρους (resource access), καθώς και η ανακάλυψη πόρων (resource discovery).

5.1 Ιστορικά Στοιχεία

Η τεχνολογία του πλέγματος αρχικά εμφανίστηκε ως ένας τρόπος αντιμετώπισης των ολοένα αυξανόμενων απαιτήσεων των υπολογιστικών προβλημάτων. Οι πρώτες προσπάθειες για την ανάπτυξη πλεγμάτων είχαν ως στόχο τη σύνδεση κατανεμημένων σε διαφορετικές περιοχές υπολογιστικών συστημάτων με σκοπό τη δημιουργία ενός υπερ-υπολογιστή για την επίλυση απαιτητικών προβλημάτων. Η προσέγγιση αυτή είναι γνωστή με τον όρο metacomputing [58]. Στα μέσα της δεκαετίας του ’90 άρχισαν να εμφανίζονται τα πρώτα περιβάλλοντα πλέγματος, τα οποία στόχευαν στην δημιουργία μετα-υπολογιστών που θα χρησιμοποιούνταν σε εφαρμογές υψηλής απόδοσης. Δύο από τα σημαντικότερα περιβάλλοντα που προέκυψαν σε αυτή τη φάση ήταν τα FAFNER [59] και I-WAY [60]. Το FAFNER είχε ως στόχο την παραγοντοποίηση πολύ μεγάλων αριθμών, πρόβλημα το οποίο μπορούσε να διασπαστεί σε ένα μεγάλο αριθμό από μικρότερα προβλήματα, τα οποία ήταν δυνατόν να επιλυθούν από συνηθισμένους υπολογιστές. Πολλές από τις τεχνικές που εισήχθησαν από το FAFNER για τη διάσπαση και κατανομή του υπολογιστικού φορτίου

άνοιξαν το δρόμο για τη μετέπειτα δημιουργία διάφορων "cycle scavenging" λογισμικών, όπως για παράδειγμα το seti@home [61]. Το I-WAY στόχευε στη διασύνδεση υπερυπολογιστών μέσω υπαρχόντων υποδομών δικτύων. Μια από τις καινοτομίες του ήταν η εισαγωγή ενός διαμεσολαβητή υπολογιστικών πόρων (computational resource broker), παρόμοιου με αυτούς που χρησιμοποιούνται στις σημερινές υποδομές πλέγματος. Η προσέγγιση που ακολουθήθηκε στην περίπτωση του I-WAY οδήγησε στην δημιουργία του μετέπειτα μεσισμικού Globus [62], το οποίο αποτελεί τη βάση των περισσότερων σύγχρονων υποδομών πλέγματος και Legion [63], μια εναλλακτική προσέγγιση στη δημιουργία κατανεμημένων συστημάτων.

Η τεράστια επιτυχία που γνώρισε ο Παγκόσμιος Ιστός (World Wide Web) οδήγησε στην εξέλιξη των τεχνολογιών πλέγματος στη σημερινή μορφή τους. Το όραμα και η επιδίωξη είναι πλέον η ομοιόμορφη και ελεγχόμενη πρόσβαση σε υπολογιστικούς πόρους. Σκοπός είναι η συγκέντρωση πόρων σε παγκόσμιο επίπεδο και με διάφανο τρόπο, επιτρέποντας τη διάφανη δημιουργία και διαχείριση αυτόνομων, αυτοοργανούμενων υπηρεσιών. Βασικές προϋποθέσεις για την επίτευξη των παραπάνω είναι η δημιουργία πρωτοκόλλων και μηχανισμών για την ασφαλή ανακάλυψη, συγκέντρωση και παροχή πρόσβασης στους υπολογιστικούς πόρους, καθώς και η ανάπτυξη ειδικών εφαρμογών που έχουν τη δυνατότητα να εκμεταλλευτούν ένα τέτοιο περιβάλλον εκτέλεσης. Οι απαιτήσεις αυτές έχουν οδηγήσει στη γενίκευση όλων των υπολογιστικών πόρων σε υπηρεσίες. Βασικός παράγοντας προς αυτή την κατεύθυνση ήταν η ανάδειξη των Υπηρεσιών Ιστού ως η κυρίαρχη τεχνολογία στον τομέα των μεγάλης κλίμακας κατανεμημένων εφαρμογών.

Οι τεχνολογίες πλέγματος στηρίζονται πλέον στην έννοια της υπηρεσίας πλέγματος (grid service) ως την βασική αφαιρετική έννοια, η οποία επιτρέπει στα πλέγματα και τις αντίστοιχες εφαρμογές να αποτελούνται από υπηρεσίες με δυναμικό τρόπο. Για τον ακριβή ορισμό της υπηρεσίας πλέγματος και για την ανάπτυξη τέτοιων υπηρεσιών σε κάθε κλίμακα, απαραίτητη θεωρήθηκε η δημιουργία συγκεκριμένων προτύπων. Η αρχιτεκτονική OGSA (Open Grid Services Architecture) [64] προτάθηκε ως μια επέκταση των υπηρεσιών ιστού (ώστε να υποστηρίζονται λειτουργίες μετάβασης και κατάστασης – transient and stateful behaviors). Συνδυάζοντας την OGSA με συγκεκριμένα πρωτόκολλα πλέγματος ορίστηκε η υποδομή OGSI (Open Grid Services Infrastructure) [65], η οποία παρέχει μια ομοιόμορφη αρχιτεκτονική για τη δημιουργία και διαχείριση πλεγμάτων και εφαρμογών πλέγματος. Αργότερα προτάθηκε ως εναλλακτικό του προτύπου OGSI το πλαίσιο WSRF (Web Services Resource Framework) [66], το οποίο επιτρέπει τη διαχείριση υπηρεσιών ιστού με ή χωρίς κατάσταση (stateless ή stateful) και το οποίο έχει ως στόχο τη

διευκόλυνση της σύγκλισης των αρχιτεκτονικών των υπηρεσιών ιστού και των υπηρεσιών πλέγματος.

5.2 Εικονικοί Οργανισμοί

Ένα περιβάλλον πλέγματος δημιουργείται για την αντιμετώπιση προβλημάτων που σχετίζονται με υπολογιστικούς πόρους (κύκλους CPU, αποθηκευτικός χώρος, δεδομένα, προγράμματα λογισμικού, περιφερειακά κτλ). Η χρήση των πόρων χαρακτηρίζεται συνήθως από το γεγονός ότι αυτοί διατίθενται έξω από τα πλαίσια μιας τοπικής περιοχής διαχείρισης. Αυτή η προσέγγιση προϋποθέτει τη δημιουργία μιας νέας περιοχής διαχείρισης, η οποία αναφέρεται ως Εικονικός Οργανισμός (Virtual Organization – VO) [57].

Όπως προαναφέρθηκε, το πραγματικό και συγκεκριμένο πρόβλημα που σχετίζεται με την έννοια του υπολογιστικού πλέγματος είναι ο συντονισμένος διαμοιρασμός πόρων και η συντονισμένη επίλυση προβλημάτων σε δυναμικούς, πολύ-θεσμικούς VOs. Αυτός ο διαμοιρασμός πόρων δεν περιορίζεται στην ανταλλαγή αρχείων, αλλά επεκτείνεται στην άμεση πρόσβαση σε υπολογιστές, λογισμικό, δεδομένα, καθώς και άλλους πόρους, όπως απαιτείται από πολλές στρατηγικές συλλογικής επίλυσης προβλημάτων και κατανομής δεδομένων, που έχουν εμφανισθεί στη βιομηχανία και τις διάφορες επιστήμες. Επίσης, ο διαμοιρασμός αυτός πρέπει απαραίτητα να είναι εξαιρετικά ελέγξιμος, με τους παρόχους πόρων και τους καταναλωτές να ορίζουν καθαρά και προσεκτικά τί ακριβώς διαμοιράζεται, ποιος επιτρέπεται να έχει πρόσβαση, καθώς και τις συνθήκες κάτω από τις οποίες γίνεται αυτή η κατανομή. Ένα σύνολο από άτομα ή/και ιδρύματα που ορίζονται από τέτοιους κανόνες αποτελεί έναν VO.

Χαρακτηριστικό των VOs είναι ότι διαφέρουν σε μεγάλο βαθμό ως προς το αντικείμενο του σκοπού τους, το μέγεθός τους, τη διάρκειά τους, τη δομή τους και τις κοινότητες από τις οποίες απαρτίζονται. Παρ' όλα αυτά όμως, μια προσεκτική μελέτη των υποκείμενων τεχνολογικών απαιτήσεων οδηγεί στην αναγνώριση κάποιου πλήθους κοινών χαρακτηριστικών και απαιτήσεων για το συνολικό πρόβλημα, τα οποία μπορούν να αποτελέσουν τη βάση για την περαιτέρω ανάπτυξη της συγκεκριμένης τεχνολογίας. Συγκεκριμένα, σε όλες τις περιπτώσεις των VOs υπάρχει απαίτηση για τα ακόλουθα:

- εξαιρετικά ευέλικτες σχέσεις διαμοιρασμού, εκτεινόμενες από σχέσεις πελάτη-εξυπηρετητή μέχρι σχέσεις ομότιμων κόμβων
- εξεζητημένα και ακριβή επίπεδα ελέγχου σχετικά με τον τρόπο χρησιμοποίησης των διαμοιραζόμενων πόρων, συμπεριλαμβανομένου του ελέγχου συμμετοχικής πρόσβασης

(multi-stakeholder access), την αντιπροσώπευση (delegation), και την εφαρμογή τοπικών και καθολικών πολιτικών

▪ υποστήριξη κατανομής διάφορων πόρων, όπως προγράμματα, αρχεία και δεδομένα ή ακόμα και υπολογιστές, αισθητήρων και δικτύων

▪ υποστήριξη εξαιρετικά διαφορετικών τρόπων χρήσης, κυμαινόμενων από μοναδικό χρήστη σε πολλαπλούς χρήστες και από ευαίσθητους ως προς την απόδοση σε ευαίσθητους ως προς το κόστος, πράγμα που σημαίνει ότι πρέπει να υπάρχει πρόβλεψη για θέματα ποιότητας υπηρεσίας (quality of service – QoS), συντονισμού (scheduling), ομοιοκατανομής (co-allocation) και χρέωσης (accounting).

Αναλυτικότερα, ο διαμοιρασμός πόρων πρέπει να γίνεται υπό όρους. Κάθε ιδιοκτήτης πόρων κάνει τους πόρους του διαθέσιμους με βάση περιορισμούς στο πού, πότε και τί μπορεί να γίνει με αυτούς. Η υλοποίηση τέτοιων περιορισμών απαιτεί μηχανισμούς για τη δημιουργία πολιτικών, για την αναγνώριση της ταυτότητας ενός χρήστη ή ενός πόρου (πιστοποίηση) και για τον καθορισμό του κατά πόσο μια λειτουργία είναι συμβατή με τις εφαρμοζόμενες σχέσεις διαμοιρασμού (εξουσιοδότηση). Οι σχέσεις κατανομής μπορούν να μεταβάλλονται δυναμικά στο χρόνο ως προς τους εμπλεκόμενους πόρους, το είδος της πρόσβασης που επιτρέπεται, καθώς και τους συμμετέχοντες στους οποίους επιτρέπεται η πρόσβαση. Αυτές οι σχέσεις μάλιστα δεν εμπλέκουν απαραίτητα ένα συγκεκριμένο σύνολο ατόμων, αλλά μπορούν να εκφράζονται και να ορίζονται έμμεσα από τις πολιτικές που εφαρμόζονται για την πρόσβαση στους πόρους. Επιπλέον, η δυναμική φύση των σχέσεων κατανομής συνεπάγεται τη χρήση μηχανισμών για την ανακάλυψη και το χαρακτηρισμό των σχέσεων σε μια συγκεκριμένη χρονική στιγμή.

Τέλος, ένας συγκεκριμένος πόρος μπορεί να χρησιμοποιείται με διαφορετικούς τρόπους, με βάση τους περιορισμούς που έχουν τεθεί στη κατανομή και τους σκοπούς της. Για παράδειγμα, ένας υπολογιστής μπορεί να χρησιμοποιείται για να εκτελέσει ένα συγκεκριμένο τμήμα λογισμικού σε μια σχέση διαμοιρασμού, ενώ σε κάποια άλλη μπορεί γενικά να παρέχει υπολογιστικούς κύκλους. Λόγω έλλειψης εκ των προτέρων γνώσης για τον τρόπο με τον οποίο μπορεί να χρησιμοποιηθεί ένας πόρος, η κατανομή πρέπει να γίνεται υπό συγκεκριμένες συνθήκες ως προς τα κριτήρια απόδοσης και τους περιορισμούς που μπορεί να επιβάλλονται (π.χ. QoS).

Όλα τα χαρακτηριστικά και οι απαιτήσεις που προαναφέρθηκαν ορίζουν έναν VO. Η έννοια των VOs επιτρέπει σε ανομοιογενείς ομάδες ατόμων ή/και οργανισμών να διαμοιράζονται πόρους με πλήρως ελέγξιμο τρόπο, έτσι ώστε τα μέλη τους να μπορούν να συνεργάζονται για να επιτύχουν έναν κοινό σκοπό.

5.3 Αρχιτεκτονικές Πλέγματος

Η τεχνολογία του υπολογιστικού πλέγματος παρουσιάζεται ως λύση στο πρόβλημα της εγκατάστασης, της διαχείρισης και της εκμετάλλευσης των δυναμικών και πολύπλοκων σχέσεων διανομής μεταξύ VOs, οι οποίοι ενδέχεται να είναι πολύ διαφορετικοί μεταξύ τους. Για την ανάπτυξη και υλοποίηση της συγκεκριμένης τεχνολογίας έχουν προταθεί κατά καιρούς διάφορες πλεγματικές αρχιτεκτονικές, οι οποίες διακρίνουν τα θεμελιώδη συστατικά του συστήματος, διευκρινίζουν το σκοπό και τις λειτουργίες αυτών των συστατικών και αναδεικνύουν τον τρόπο με τον οποίο αυτά τα συστατικά αλληλεπιδρούν μεταξύ τους.

Γενικά, κατά τον ορισμό μιας οποιασδήποτε πλεγματικής αρχιτεκτονικής πρέπει να λαμβάνεται υπόψη ότι η αποτελεσματική λειτουργία των VOs προϋποθέτει τη δυνατότητα εγκαθίδρυσης σχέσεων διαμοιρασμού μεταξύ οποιονδήποτε πιθανών συμμετεχόντων. Αυτό σημαίνει ότι το κεντρικό ζήτημα που πρέπει να αντιμετωπιστεί είναι αυτό της διαλειτουργικότητας (interoperability). Λόγω του ότι το πλεγματικό περιβάλλον είναι ένα κατεξοχήν δικτυακό περιβάλλον, η διαλειτουργικότητα ισοδυναμεί με την ύπαρξη κοινά αποδεκτών και χρησιμοποιούμενων πρωτοκόλλων. Αυτό σημαίνει ότι η πλεγματική αρχιτεκτονική είναι καταρχάς μια αρχιτεκτονική πρωτοκόλλων, όπου με τον όρο πρωτόκολλα ορίζονται οι βασικοί μηχανισμοί μέσω των οποίων οι χρήστες και οι πόροι των VOs διαπραγματεύονται, εγκαθιστούν, καθιερώνουν, διαχειρίζονται και εκμεταλλεύονται τις εκάστοτε σχέσεις διαμοιρασμού. Μια ανοιχτή αρχιτεκτονική βασισμένη σε κοινά πρότυπα προσφέρει επεκτασιμότητα, διαλειτουργικότητα και φορητότητα και διευκολύνει την κατανομή πόρων. Επιπλέον, η υιοθέτηση προτυποποιημένων πρωτοκόλλων επιτρέπει τον ορισμό πρότυπων υπηρεσιών με ενισχυμένες δυνατότητες. Εκτός από τα πρωτόκολλα, γενικά είναι χρήσιμη και η δημιουργία κατάλληλων προγραμματιστικών διεπαφών (APIs) και πακέτων ανάπτυξης λογισμικού (Software Development Kits – SDKs). Το σύνολο των πρωτοκόλλων, των APIs και των SDKs αποτελεί το λεγόμενο πλεγματικό μεσισμικό (grid middleware), το οποίο λειτουργεί ως ένα ενδιάμεσο στρώμα που παρέχει ένα σύνολο υπηρεσιών πάνω από ένα κατανεμημένο δικτυακό περιβάλλον.

Όπως προαναφέρθηκε, θεμελιώδες μέλημα μιας πλεγματικής αρχιτεκτονικής είναι η διαλειτουργικότητα. Το ζήτημα είναι η εξασφάλιση της δυνατότητας να μπορούν οι σχέσεις διανομής να αρχικοποιηθούν ανάμεσα σε αυθαιρέτως διαφορετικά σύνολα, προσαρμόζοντας δυναμικά νέους συμμετέχοντες, πάνω από διαφορετικές πλατφόρμες, γλώσσες και προγραμματιστικά περιβάλλοντα. Μέσα σε αυτά τα πλαίσια, για να είναι

αποδοτικοί οι μηχανισμοί που χρησιμοποιούνται, θα πρέπει να έχουν οριστεί και υλοποιηθεί έτσι ώστε να είναι διαλειτουργικοί πέρα από συγκεκριμένα οργανωτικά όρια, συγκεκριμένες λειτουργικές πολιτικές και συγκεκριμένους τύπους πόρων. Σε περίπτωση που δεν υπάρχει διαλειτουργικότητα, οι συμμετέχοντες σε μια σχέση διανομής ανάμεσα σε VOs πρέπει αναγκαστικά να προχωρήσουν σε διμερείς συμφωνίες διαμοιρασμού, οι οποίες ίσως να μην μπορούν να χρησιμοποιηθούν σε άλλες περιστάσεις από άλλους VOs. Αυτό σημαίνει ότι ο δυναμικός σχηματισμός VOs είναι αδύνατος, πράγμα που περιορίζει σημαντικά τα διαφορετικά είδη VOs που μπορούν να σχηματιστούν.

Βασικά συστατικά στοιχεία λοιπόν των πλεγματικών αρχιτεκτονικών είναι τα πρωτόκολλα. Τα πρωτόκολλα αυτά γενικά προτυποποιούν το διαμοιρασμό πόρων, όπως ακριβώς τα πρωτόκολλα του διαδικτύου (π.χ. HTTP) προτυποποιούν τη διανομή πληροφορίας. Άλλωστε, η τεράστια επιτυχία του διαδικτύου οφείλεται ακριβώς στην ύπαρξη αυτών των καθολικά αποδεκτών πρωτοκόλλων. Ο ορισμός ενός πρωτοκόλλου καθορίζει τον τρόπο με τον οποίο τα στοιχεία ενός κατανεμημένου συστήματος αλληλεπιδρούν μεταξύ τους για να επιτύχουν μια συγκεκριμένη συμπεριφορά, καθώς και τη δομή της πληροφορίας που ανταλλάσσεται κατά την αλληλεπίδραση αυτή. Αυτή η εστίαση στις αλληλεπιδράσεις, παρά στα ενδογενή χαρακτηριστικά των πόρων παρουσιάζει σημαντικά πλεονεκτήματα. Όπως προαναφέρθηκε, οι VOs γενικά είναι ρευστές οντότητες, πράγμα που σημαίνει ότι οι μηχανισμοί για την ανακάλυψη πόρων, την καθιέρωση ταυτότητας, την εξουσιοδότηση και τελικά την έναρξη της κατανομής των πόρων, πρέπει να είναι ευέλικτοι και σχετικά ελαφρείς υπολογιστικά (lightweight), έτσι ώστε οι διακανονισμοί για τη διανομή πόρων να μπορούν να εγκαθίστανται και να αλλάζουν γρήγορα. Επιπλέον, επειδή οι VOs δεν αντικαθιστούν τους υπάρχοντες θεσμούς, αλλά απλά τους συμπληρώνουν και τους επεκτείνουν, οι μηχανισμοί διανομής δεν πρέπει να επιφέρουν δραστικές αλλαγές στις εκάστοτε τοπικές πολιτικές, ενώ πρέπει και να επιτρέπουν στους ανεξάρτητους οργανισμούς να διατηρούν τον έλεγχο πάνω στους δικούς τους πόρους. Επειδή τα πρωτόκολλα σχετίζονται με την αλληλεπίδραση μεταξύ συστατικών και όχι με την υλοποίηση των συστατικών, η προσέγγιση των πρωτοκόλλων επιτρέπει τη διατήρηση του τοπικού ελέγχου.

Ένα από τα βασικά στοιχεία μιας αρχιτεκτονικής πλέγματος είναι η έννοια της υπηρεσίας. Μια υπηρεσία σε ένα δικτυακό περιβάλλον είναι μια οντότητα που παρέχει κάποια συγκεκριμένη λειτουργικότητα, όπως για παράδειγμα μεταφορά αρχείων, δημιουργία διεργασιών, ή την επαλήθευση δικαιωμάτων πρόσβασης. Μια υπηρεσία ορίζεται από το πρωτόκολλο που χρειάζεται να χρησιμοποιήσει κανείς για να

αλληλεπιδράσει με αυτήν και τη συμπεριφορά που αναμένεται ως απόκριση σε διάφορα είδη ανταλλαγών μηνυμάτων (δηλαδή υπηρεσία = πρωτόκολλο + συμπεριφορά). Ο ορισμός αυτός επιτρέπει διάφορους τρόπους υλοποίησης της υπηρεσίας. Μια υπηρεσία μπορεί να έχει ή να μην έχει κάποια χαρακτηριστικά, όπως για παράδειγμα να είναι πάντα διαθέσιμη (persistent), να μπορεί να αναγνωρίζει και να ανακάμπτει από ορισμένα σφάλματα, να επιτρέπει την εκτέλεση με προνόμια, ή ακόμη και να έχει κατανεμημένη υλοποίηση για ενισχυμένη ικανότητα κλιμάκωσης. Άρρηκτα συνδεδεμένη με την έννοια της υπηρεσίας είναι η ύπαρξη μηχανισμών ανακάλυψης (discovery mechanisms), έτσι ώστε να μπορεί ένας πελάτης να καθορίζει τα χαρακτηριστικά κάποιου συγκεκριμένου στιγμιότυπου (instantiation) της υπηρεσίας.

5.3.1 Γενική Προσέγγιση Πλεγματικής Υποδομής

Για την υλοποίηση μιας πλεγματικής υποδομής έχει προταθεί και έχει γίνει γενικά αποδεκτή μια γενικότερη αρχιτεκτονική, που στηρίζεται σε ένα μοντέλο που αποτελείται από τρία στρώματα [64]: το Υπολογιστικό Πλέγμα (Computational Grid), το Πλέγμα Πληροφορίας (Information Grid) και το Πλέγμα Γνώσης (Knowledge Grid), τα οποία περιγράφονται στη συνέχεια.

Το Υπολογιστικό Πλέγμα είναι το κατώτερο στρώμα και σχετίζεται κυρίως με την ευρείας κλίμακας συγκέντρωση υπολογιστικών πόρων και δεδομένων (εναλλακτικά, το στρώμα αυτό μπορεί να ονομάζεται και Πλέγμα Δεδομένων (Data Grid), λόγω του ότι σχετίζεται με το διαμοιρασμό δεδομένων). Αυτού του είδους η συγκέντρωση απαιτεί την ύπαρξη σημαντικής κοινής υποδομής, έτσι ώστε να είναι δυνατή η επίβλεψη και ο έλεγχος τον πόρων στο σύνολο που προκύπτει. Βασικός σκοπός ενός Υπολογιστικού Πλέγματος είναι η δημιουργία ενός τεράστιου υπολογιστικού περιβάλλοντος, χρησιμοποιώντας μια κατανεμημένη συλλογή από αρχεία, βάσεις δεδομένων, υπολογιστές, επιστημονικά όργανα και άλλες συσκευές.

Το Πλέγμα Πληροφορίας αποτελεί το μεσαίο στρώμα, το οποίο επιτρέπει ομοιόμορφη πρόσβαση σε ετερογενείς πηγές πληροφορίας και παρέχει κοινά χρησιμοποιούμενες υπηρεσίες, οι οποίες εκτελούνται σε κατανεμημένους υπολογιστικούς πόρους. Η ομοιόμορφη πρόσβαση στις πηγές πληροφορίας βασίζεται στην περιγραφή της πληροφορίας και στην ενσωμάτωση ετερογενών πηγών μέσω της χρήσης μεταδεδομένων. Η κλιμάκωση των προσφερόμενων υπηρεσιών μπορεί να διαφέρει, ξεκινώντας από απλές κλήσεις μεθόδων και φτάνοντας μέχρι ολοκληρωμένες εφαρμογές. Για παράδειγμα, σε

επιστημονικές εφαρμογές οι υπηρεσίες μπορεί να περιλαμβάνουν την παροχή ειδικών αριθμητικών επιλυτών, όπως επιλυτές πινάκων και επιλυτές μερικών διαφορικών εξισώσεων, οι οποίες μπορούν να χρησιμοποιηθούν σε κώδικες για πρόβλεψη καιρού ή για μελέτη δυναμικής υγρών. Στα πλαίσια των εμπορικών εφαρμογών, οι υπηρεσίες μπορεί να είναι ρουτίνες στατιστικής βασισμένες σε υπάρχουσες βιβλιοθήκες λογισμικού, ή υπηρεσίες πρόβλεψης που παρέχουν κλιμακούμενη λειτουργικότητα. Στα πλαίσια πολυμεσικών εφαρμογών, οι υπηρεσίες μπορεί να υλοποιούν και να παρέχουν αλγόριθμους για ανάλυση πολυμεσικού περιεχομένου. Προφανώς οι υπηρεσίες μπορεί να παρέχονται από μεμονωμένους παρόχους ή από οργανισμούς και να προορίζονται για συγκεκριμένες εφαρμογές ή για γενικό σκοπό.

Το Πλέγμα Γνώσης αποτελεί το υψηλότερο επίπεδο και παρέχει εξειδικευμένες υπηρεσίες, οι οποίες μπορούν να ψάξουν για συγκεκριμένα σχέδια σε υπάρχουσες αποθήκες δεδομένων και να χειριστούν διάφορες υπηρεσίες πληροφοριών. Το πλέγμα γνώσης στοχεύει στη δημιουργία νέων υπηρεσιών, οι οποίες δεν μπορούν να οριστούν ως μεμονωμένες υπηρεσίες. Αυτές απαρτίζονται από ένα σύνολο υπηρεσιών του προηγούμενου στρώματος, οι οποίες μπορεί να είναι τελείως διαφορετικού τύπου, παρέχοντας κάποια συσχέτιση μεταξύ δεδομένων που παράγονται από διαφορετικές υπηρεσίες, ή συνδυάζοντας αποτελέσματα από υπάρχουσες υπηρεσίες με νέους τρόπους.

Όλα τα παραπάνω στρώματα έχουν ως σκοπό την παροχή υπηρεσιών σε διάφορες εφαρμογές, όπως για παράδειγμα εφαρμογές υποστήριξης κινητών συσκευών ή ευρείας κλίμακας εφαρμογές, όπως μοντελοποίηση και προσομοίωση πολύπλοκων προβλημάτων. Παρόλα αυτά, οι περισσότερες ερευνητικές προσπάθειες επικεντρώνονται κυρίως στο Υπολογιστικό Πλέγμα. Στη συνέχεια θα παρουσιαστούν οι διάφορες προσεγγίσεις για την αρχιτεκτονική ενός Υπολογιστικού Πλέγματος.

5.3.2 Γενική Αρχιτεκτονική Υπολογιστικού Πλέγματος

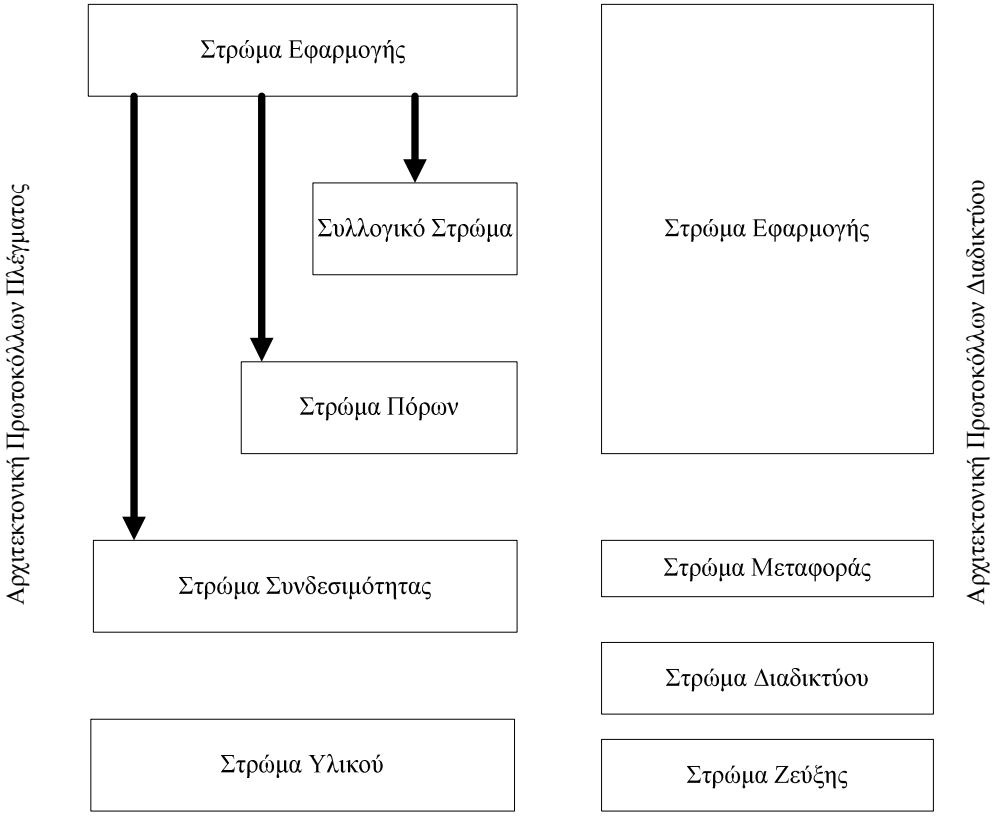
Μια ευρέως διαδεδομένη αρχιτεκτονική για ένα υπολογιστικό πλέγμα έχει προταθεί στο [57]. Στη συγκεκριμένη αρχιτεκτονική δεν έχει γίνει αναλυτικός εντοπισμός και απαρίθμηση όλων των πρωτοκόλλων και υπηρεσιών που απαιτούνται για τη λειτουργία ενός υπολογιστικού πλέγματος, αλλά έχουν αναγνωριστεί γενικότερες απαιτήσεις για γενικές κλάσεις συστατικών. Η οργάνωση των συστατικών γίνεται χωρίζοντας τα συστατικά σε στρώματα, όπως φαίνεται στο Σχήμα 35, όπου φαίνεται και η αναλογία με την αρχιτεκτονική του διαδικτύου. Τα συστατικά σε κάθε στρώμα έχουν κοινά

χαρακτηριστικά και λειτουργικότητα, αλλά μπορούν να χρησιμοποιούν και να βασίζονται σε ικανότητες και συμπεριφορές συστατικών που ανήκουν σε οποιοδήποτε άλλο κατώτερο στρώμα.

Όπως φαίνεται και στο Σχήμα 35, τα στρώματα που απαρτίζουν τη συγκεκριμένη αρχιτεκτονική του υπολογιστικού πλέγματος είναι τα εξής:

- Στρώμα Υλικού (Fabric Layer): παρέχει τους πόρους στους οποίους την κοινή πρόσβαση διαπραγματεύονται τα πρωτόκολλα του υπολογιστικού πλέγματος, όπως για παράδειγμα υπολογιστικοί πόροι, συστήματα αποθήκευσης, κατάλογοι, δικτυακοί πόροι αισθητήρες, κτλ.
- Στρώμα Συνδεσιμότητας (Connectivity Layer): ορίζει βασικά πρωτόκολλα επικοινωνίας και πιστοποίησης, τα οποία απαιτούνται για δικτυακές συναλλαγές πάνω σε μια πλεγματική υποδομή. Τα πρωτόκολλα επικοινωνίας επιτρέπουν την ανταλλαγή δεδομένων μεταξύ πόρων του στρώματος υλικού. Τα πρωτόκολλα πιστοποίησης βασίζονται στις υπηρεσίες επικοινωνίας και παρέχουν κρυπτογραφικά ασφαλείς μηχανισμούς για την επαλήθευση της ταυτότητας χρηστών και πόρων.
- Στρώμα Πόρων (Resource Layer): στηρίζεται πάνω στα πρωτόκολλα επικοινωνίας και πιστοποίησης για τον ορισμό πρωτοκόλλων σχετικών με την ασφαλή διαπραγμάτευση, αρχικοποίηση, επίβλεψη, έλεγχο, χρέωση και πληρωμή λειτουργιών διαμοιρασμού μεμονωμένων πόρων. Οι υλοποιήσεις των πρωτοκόλλων του συγκεκριμένου επιπέδου χρησιμοποιούν λειτουργίες που παρέχονται από το επίπεδο υλικού για την πρόσβαση και τον έλεγχο τοπικών πόρων. Τα πρωτόκολλα του στρώματος πόρων σχετίζονται αποκλειστικά με μεμονωμένους πόρους, πράγμα που σημαίνει ότι αγνοούν θέματα καθολικής κατάστασης και διάφορα άλλα θέματα που σχετίζονται με τη θέση κάποιου πόρου σε ένα κατανεμημένο περιβάλλον.
- Συλλογικό Στρώμα (Collective Layer): περιέχει πρωτόκολλα και υπηρεσίες που δεν σχετίζονται με κανέναν συγκεκριμένο πόρο, αλλά αντίθετα έχουν καθολική φύση και εμπλέκονται με αλληλεπιδράσεις μεταξύ συνόλων από πόρους. Λόγω του ότι τα συστατικά του συλλογικού στρώματος στηρίζονται πάνω στο μικρό αριθμό των πρωτοκόλλων των στρωμάτων συνδεσιμότητας και πόρων, αυτά έχουν τη δυνατότητα υλοποίησης ενός μεγάλου αριθμού διαφορετικών συμπεριφορών διανομής, χωρίς να επιβάλλουν νέες απαιτήσεις στους διανεμόμενους πόρους.
- Στρώμα Εφαρμογής (Application Layer): σε αυτό το στρώμα ανήκουν οι εφαρμογές των χρηστών που λειτουργούν σε ένα περιβάλλον VO. Οι εφαρμογές αυτές δημιουργούνται

καλώντας υπηρεσίες και χρησιμοποιώντας πρωτόκολλα που ορίζονται στα χαμηλότερα στρώματα της αρχιτεκτονικής.



Σχήμα 35. Η διαστρωματωμένη αρχιτεκτονική του υπολογιστικού πλέγματος και η αντιστοιχία της με την αρχιτεκτονική του διαδικτύου.

Βασικό χαρακτηριστικό της παραπάνω δομής είναι ότι ακολουθεί το μοντέλο της κλεψύδρας (hourglass model). Ο στενός λαιμός της κλεψύδρας αντιστοιχεί σε ένα μικρό σύνολο βασικών πρωτοκόλλων. Πάνω σε αυτά τα πρωτόκολλα μπορούν να στηριχτούν ένας μεγάλος αριθμός από άλλα πρωτόκολλα και συμπεριφορές υψηλότερου επιπέδου (κορυφή της κλεψύδρας). Επίσης, τα ίδια τα βασικά πρωτόκολλα στηρίζονται σε ένα μεγάλο αριθμό άλλων πρωτοκόλλων χαμηλότερου επιπέδου (βάση της κλεψύδρας). Στην παραπάνω αρχιτεκτονική, ο λαιμός της κλεψύδρας αντιστοιχεί στα στρώματα πόρων και συνδεσιμότητας, που διαπραγματεύονται τη διανομή μεμονωμένων πόρων. Τα πρωτόκολλα σε αυτά τα στρώματα σχεδιάζονται έτσι ώστε να μπορούν να υλοποιηθούν πάνω σε διάφορα είδη πόρων που ορίζονται στο στρώμα υλικού και μπορούν επίσης να χρησιμοποιηθούν στην κατασκευή μεγάλου εύρους υπηρεσιών και ειδικών συμπεριφορών στο συλλογικό στρώμα. Πρέπει να σημειωθεί ότι η συγκεκριμένη αρχιτεκτονική είναι

υψηλού επιπέδου και θέτει ελάχιστους περιορισμούς στη σχεδίαση και την υλοποίηση των πρωτοκόλλων και των άλλων επιμέρους στοιχείων.

5.3.3 Open Grid Services Architecture (OGSA)

Η πλέον πρόσφατα προτεινόμενη αρχιτεκτονική για ένα πλέγμα είναι η Ανοιχτή Αρχιτεκτονική Υπηρεσιών Πλέγματος (Open Grid Services Architecture – OGSA) [64]. Η αρχιτεκτονική αυτή είναι το αποτέλεσμα της ευθυγράμμισης των ήδη υπαρχόντων πλεγματικών προτύπων με τις διάφορες ανερχόμενες αρχιτεκτονικές που προσανατολίζονται στις υπηρεσίες (Service Oriented Architectures – SOAs), καθώς και με το διαδίκτυο. Η OGSA παρέχει έναν ομοιόμορφο τρόπο περιγραφής υπηρεσιών πλέγματος και καθορίζει ένα κοινό σχέδιο συμπεριφοράς για αυτές. Ορίζει τη συμπεριφορά τους, τους μηχανισμούς περιγραφής υπηρεσίας και την πληροφορία σύνδεσης πρωτοκόλλου, χρησιμοποιώντας την τεχνολογία των υπηρεσιών ιστού.

Στη συγκεκριμένη αρχιτεκτονική το πλέγμα θεωρείται ως ένα επεκτάσιμο σύνολο πλεγματικών υπηρεσιών, οι οποίες μπορεί να συνδυάζονται με διάφορους τρόπους για την επίτευξη των αναγκών των VOs. Οι VOs μάλιστα μπορεί με τη σειρά τους να ορίζονται μερικώς από τις υπηρεσίες που χρησιμοποιούν και μοιράζονται. Αυτό που απαιτείται είναι ο καθορισμός των συμπεριφορών που πρέπει να διαθέτουν αυτές οι υπηρεσίες πλέγματος για να υποστηρίζουν ολοκλήρωση σε κατανεμημένα συστήματα. Αυτό γίνεται με τη χρήση της τεχνολογίας των υπηρεσιών ιστού, από την οποία υιοθετούνται κατάλληλες ιδιότητες, όπως περιγραφή και ανακάλυψη υπηρεσίας (service description and discovery), αυτόματη δημιουργία κώδικα πελάτη και εξυπηρετητή από την περιγραφή της υπηρεσίας, σύνδεση περιγραφών υπηρεσίας με διαλειτουργικά δικτυακά πρωτόκολλα, συμβατότητα με αναδυόμενα πρότυπα, υπηρεσίες και εργαλεία υψηλότερου επιπέδου, καθώς και ευρεία εμπορική υποστήριξη. Η ευθυγράμμιση των δύο αυτών τεχνολογιών ονομάζεται Ανοιχτή Αρχιτεκτονική Υπηρεσιών Πλέγματος, με τον όρο "αρχιτεκτονική" να υποδεικνύει ένα καλά ορισμένο σύνολο βασικών διεπαφών, από το οποίο μπορούν να κατασκευαστούν διάφορα συστήματα, ενώ ο όρος "ανοιχτή" αναφέρεται στην επεκτασιμότητα, την ανεξαρτησία από τον πάροχο και την προσκόλληση σε μια διαδικασία κοινοτικής προτυποποίησης (community standardization process). Η OGSA χρησιμοποιεί τη γλώσσα περιγραφής υπηρεσιών ιστού (Web Services Description Language – WSDL) για τη δημιουργία αυτοπεριγραφόμενων και ανακαλύψιμων υπηρεσιών και διαλειτουργικών

πρωτοκόλλων, με τη δυνατότητα υποστήριξης πολλαπλών συνεργαζόμενων διεπαφών, καθώς και δυνατότητα διαχείρισης αλλαγών.

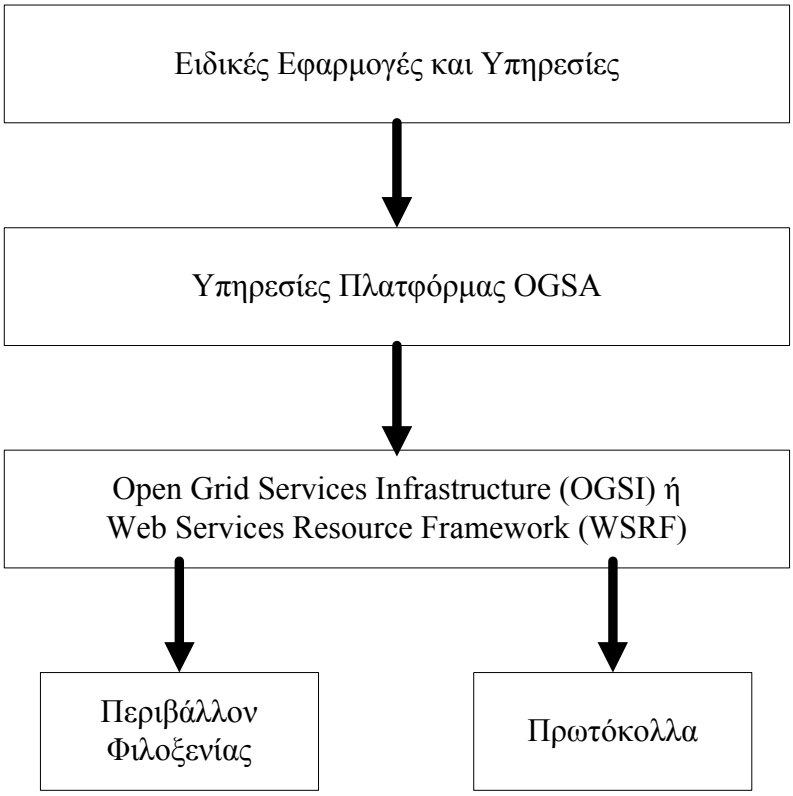
Στην OGSA ορίζονται συμβάσεις και διεπαφές WSDL για τις υπηρεσίες πλέγματος. Βασικά χαρακτηριστικά αυτών των υπηρεσιών είναι ότι εμπεριέχουν την έννοια της κατάστασης (stateful services) και ότι υποστηρίζουν την αξιόπιστη και ασφαλή κλήση τους (reliable and secure invocation), τη διαχείριση διάρκειας ζωής τους (lifetime management), την ανακοίνωση (notification), τη διαχείριση πολιτικών (policy management), τη διαχείριση πιστοποιητικών (credential management), καθώς και την εικονικοποίηση (virtualization). Επιπλέον, η OGSA ορίζει διεπαφές για την ανακάλυψη στιγμιότυπων υπηρεσιών πλέγματος και για τη δημιουργία παροδικών (transient) στιγμιότυπων υπηρεσιών. Το αποτέλεσμα είναι η δημιουργία ενός κατανεμημένου συστήματος υπηρεσιών, που υποστηρίζει τη δημιουργία εξεζητημένων κατανεμημένων υπηρεσιών. Η γενική αρχιτεκτονική του μοντέλου της OGSA φαίνεται στο Σχήμα 36.

5.3.3.1 Service Oriented Architecture (SOA) – Web Services Architecture (WSA)

Μια SOA ορίζεται ως μια χαλαρά συνδεδεμένη (loosely coupled) αρχιτεκτονική που περιέχει ένα σύνολο αφαιρέσεων σχετικά με τα συστατικά της [67]. Τα συστατικά αυτά μπορούν να χρησιμοποιηθούν από πελάτες μέσω κάποιου δικτύου με βάση καλά καθορισμένες πολιτικές. Μια SOA είναι ένας ειδικός τύπος υποδομής για κατανεμημένα συστήματα, ο οποίος διατηρεί κάποιους αντιπρόσωπους που ενεργούν ως υπηρεσίες λογισμικού, εκτελώντας καλά καθορισμένες λειτουργίες. Μια υπηρεσία λειτουργεί κατά κάποιο τρόπο ως ένα συστατικό λογισμικού κάποιας εφαρμογής ή κάποιου πόρου, που εκτίθεται σε κάποιους χρήστες. Αυτό το πρότυπο λειτουργικότητας επιτρέπει στους χρήστες της αντίστοιχης εφαρμογής (ή του αντίστοιχου πόρου) να ενδιαφέρεται μόνο για την περιγραφή της λειτουργίας της υπηρεσίας. Επιπλέον, σύμφωνα με το μοντέλο της SOA όλες οι υπηρεσίες έχουν μία δικτυακά προσβάσιμη διεπαφή και επικοινωνούν μέσω καθορισμένων πρωτοκόλλων και σχημάτων δεδομένων που ονομάζονται μηνύματα.

Ο όρος υπηρεσίες ιστού αναφέρεται σε ένα ανερχόμενο πρότυπο κατανεμημένων συστημάτων που διαφέρει από τις υπόλοιπες προσεγγίσεις (DCE, CORBA, Java RMI, κτλ) λόγω της επικέντρωσής του σε απλά πρωτόκολλα, τα οποία στηρίζονται στην τεχνολογία του διαδικτύου (π.χ. XML) για την αντιμετώπιση του προβλήματος της δημιουργίας ετερογενών κατανεμημένων συστημάτων. Οι υπηρεσίες ιστού ορίζουν μια τεχνική για την περιγραφή συστατικών λογισμικού διαθέσιμα για πρόσβαση, μεθόδων που παρέχουν

πρόσβαση σε αυτά τα συστατικά και μεθόδων ανακάλυψης, που επιτρέπουν τον προσδιορισμό των κατάλληλων σχετικών παρόχων υπηρεσιών. Οι υπηρεσίες ιστού είναι ανεξάρτητες από τη γλώσσα προγραμματισμού, το μοντέλο προγραμματισμού, καθώς και από το χρησιμοποιούμενο λειτουργικό σύστημα.



Σχήμα 36. Η αρχιτεκτονική OGSA

Διάφορα πρότυπα έχουν καθοριστεί για τις υπηρεσίες ιστού, από τα οποία τα SOAP, WSDL και WS-Inspection μπορούν να βρουν εφαρμογή στη δημιουργία υπηρεσιών πλέγματος. Αναλυτικότερα:

- Το SOAP (Simple Object Access Protocol) παρέχει ένα μέσο ανταλλαγής μηνυμάτων μεταξύ του παρόχου μιας υπηρεσίας και του πελάτη της υπηρεσίας. Το SOAP είναι ένα απλό πρωτόκολλο ενθυλάκωσης σε δεδομένα XML, ο οποίος ορίζει κάποιες συμβάσεις για απομακρυσμένη κλήση διεργασιών (remote procedure call – RPC) και κάποιες συμβάσεις για μηνύματα. Το SOAP είναι ανεξάρτητο από το υποκείμενο πρωτόκολλο μεταφοράς.
- Η WSDL (Web Services Description Language) είναι μια γλώσσα που ορίζει ένα έγγραφο XML για την περιγραφή των υπηρεσιών ιστού ως ένα σύνολο από τερματικά

σημεία. Αυτά τα τερματικά σημεία λειτουργούν λαμβάνοντας και στέλνοντας δεδομένα είτε σχετικά με απομακρυσμένη κλήση διεργασίας, είτε με βάση κάποια άλλη μορφή μηνυμάτων βασισμένη σε έγγραφα. Οι διεπαφές των υπηρεσιών ορίζονται αόριστα με βάση τη δομή των μηνυμάτων και την ακολουθία ανταλλαγής αυτών και στη συνέχεια συνδέονται σε ένα συγκεκριμένο πρωτόκολλο δικτύου και ένα συγκεκριμένο σχήμα κωδικοποίησης δεδομένων για τη δημιουργία ενός τερματικού σημείου. Διάφορα συσχετιζόμενα τερματικά σημεία μπορεί να συνενώνονται για τον ορισμό υπηρεσιών. Η WSDL είναι επεκτάσιμη και υποστηρίζει την περιγραφή τερματικών σημείων που μπορεί να χρησιμοποιούν διάφορες μορφές δεδομένων και διάφορα πρωτόκολλα δικτύου.

- Το πρότυπο WS-Inspection αποτελείται από μια απλή XML γλώσσα και κάποιες σχετικές συμβάσεις για τον εντοπισμό περιγραφών υπηρεσιών που δημοσιοποιούνται από έναν πάροχο υπηρεσιών. Ένα έγγραφο γλώσσας WS-Inspection μπορεί να περιέχει ένα σύνολο από περιγραφές υπηρεσίας και συνδέσμους σε άλλες πηγές περιγραφών υπηρεσίας. Η περιγραφή υπηρεσίας είναι συνήθως ένα URL σε ένα έγγραφο WSDL. Επίσης, η περιγραφή υπηρεσίας μπορεί να είναι μια αναφορά σε καταχώρηση σε κάποιο κατάλογο UDDI (Universal Description, Discovery and Integration), ο οποίος είναι ένα άλλο πρότυπο των υπηρεσιών ιστού. Η καταχώρηση αυτή παραπέμπει είτε σε ένα URL για κάποιο άλλο έγγραφο WSDL, είτε σε κάποια άλλη καταχώρηση του UDDI.

Οι υπηρεσίες ιστού έχουν δύο βασικά πλεονεκτήματα [64] για τη δημιουργία πλεγματικών υποδομών:

a. Καταρχήν, η ανάγκη για υποστήριξη δυναμικής ανακάλυψης και δημιουργίας υπηρεσιών σε ετερογενή περιβάλλοντα υπαγορεύει τη χρήση μηχανισμών για την καταχώριση και ανακάλυψη ορισμών διεπαφών και περιγραφών υλοποιήσεων τερματικών σημείων, καθώς και για τη δυναμική δημιουργία πληρεξούσιων (proxies), με βάση συνδέσεις για συγκεκριμένες διεπαφές. Η WSDL παρέχει έναν προτυποποιημένο μηχανισμό για τον ορισμό διεπαφών ανεξάρτητα από την υλοποίησή τους.
b. Η ευρεία διάδοση και υιοθέτηση των μηχανισμών των υπηρεσιών ιστού έχει ως πλεονέκτημα ότι οποιαδήποτε υποδομή βασίζεται στη συγκεκριμένη τεχνολογία μπορεί να εκμεταλλευτεί διάφορα εργαλεία και υπάρχουσες υπηρεσίες, όπως για παράδειγμα τα εργαλεία για την αυτόματη δημιουργία εγγράφων WSDL και περιβαλλόντων φιλοξενίας (hosting environments) για υπηρεσίες ιστού (π.χ. Apache Axis).

5.3.3.2 Πρότυπα OGSI και WSRF

Μια υποδομή πλέγματος που στηρίζεται στην αρχιτεκτονική OGSA μπορεί να αναπτυχθεί με βάση δύο διαφορετικά πρότυπα: είτε το OGSI (Open Grid Services Infrastructure), είτε το WSRF (Web Services Resource Framework).

Με βάση τις προδιαγραφές OGSI [65], ένα στιγμιότυπο μιας υπηρεσίας πλέγματος είναι μια υπηρεσία ιστού που συμμορφώνεται με ένα σύνολο συμβάσεων. Η περιγραφή των διεπαφών και των συμπεριφορών της υπηρεσίας γίνεται μέσω της γλώσσας WSDL. Μια υπηρεσία πλέγματος παρέχει ελεγχόμενη διαχείριση των κατανεμημένων υπηρεσιών με κατάσταση, οι οποίες συχνά έχουν μεγάλη διάρκεια ζωής σε περίπλοκες κατανεμημένες εφαρμογές. Το πλέον αξιοσημείωτο σημείο των προδιαγραφών OGSI είναι η επέκταση της WSDL στην GWSDL (Grid Web Services Description Language), έτσι ώστε να παρέχει επιπλέον μηχανισμούς για την περιγραφή δεδομένων κατάστασης. Επιπλέον, οι συγκεκριμένες προδιαγραφές παρέχουν ένα σύνολο από διεπαφές και συμπεριφορές για τη διαχείριση του κύκλου ζωής των υπηρεσιών. Βασικό χαρακτηριστικό του OGSI είναι και η διαλειτουργικότητα σε επίπεδο μηνυμάτων, η οποία επιτυγχάνεται μέσω της χρήσης της γλώσσας XML για την προτυποποίηση των μηνυμάτων.

Το πρότυπο WSRF αποτελεί μια συλλογή προδιαγραφών για την υποστήριξη υπηρεσιών πλέγματος και άλλων πόρων με κατάσταση, δηλαδή πόρους των οποίων η συμπεριφορά ορίζεται με βάση κάποια υποκείμενη κατάσταση. Η βασική ιδέα πίσω από το συγκεκριμένο πρότυπο είναι η ευθυγράμμιση των υπηρεσιών ιστού και των υπηρεσιών πλέγματος. Σε γενικές γραμμές, το σημαντικότερο πλεονέκτημα από την καθιέρωση του προτύπου WSRF είναι ότι θα είναι δυνατή η χρήση υπαρκτών και δοκιμασμένων προτύπων για την τεχνολογία των υπηρεσιών ιστού στην τεχνολογία των υπηρεσιών πλέγματος [69]. Την παρούσα χρονική περίοδο μάλιστα αυτή η σύγκλιση φαίνεται ότι θα πραγματοποιηθεί, καθώς μεγάλες εταιρίες στο χώρο έχουν ανακοινώσει κοινή πορεία προς την καθολική προτυποποίηση των προδιαγραφών που ορίζει το πρότυπο WSRF [69]. Για την κοινότητα του υπολογιστικού πλέγματος αυτό σημαίνει ότι η αρχιτεκτονική OGSA σε συνδυασμό με το πρότυπο WSRF θα μπορεί να θεωρείται πλέον ως μια λύση η οποία έχει άριστες προοπτικές για καθολική αποδοχή και μεγάλο χρόνο ζωής [70].

Η έννοια του WS-Resource έχει προταθεί ως ένας τρόπος έκφρασης των σχέσεων μεταξύ των πόρων που περιέχουν κάποια κατάσταση και των υπηρεσιών ιστού [71]. Το WSRF είναι ένα σύνολο από προδιαγραφές υπηρεσιών ιστού που ορίζουν τον τρόπο με τον οποίο ένας πόρος με κατάσταση περιγράφεται με βάση κάποιον ορισμό XML, καθώς και τα μέσα

με τα οποία η κατάσταση ενός πόρου μπορεί να ανακτηθεί και να ανανεωθεί με πρότυπο τρόπο. Μέχρι στιγμής, τέσσερις προδιαγραφές έχουν προτυποποιηθεί από τον οργανισμό OASIS [72]:

- WS-ResourceProperties (WSRF-RP) [73]: παρέχει προτυποποίηση για την ορολογία, τις έννοιες και τις λειτουργίες που απαιτούνται για να εκφραστούν οι ιδιότητες των πόρων και η συσχέτισή τους με τη διεπαφή της υπηρεσίας ιστού. Επιπλέον, η προδιαγραφή WSRF-RP προτυποποιεί τις WSDL και XML περιγραφές ενός πόρου, καθώς και τα μηνύματα που ορίζουν τις ικανότητες εύρεσης και ανανέωσης της κατάστασής του και των ιδιοτήτων του.
- WS-ResourceLifetime (WSRF-RL) [74]: παρέχει κάποιους βασικούς μηχανισμούς για τη διαχείριση του κύκλου ζωής ενός πόρου. Η προδιαγραφή αυτή ορίζει μια συγκεκριμένη σειρά ανταλλαγής μηνυμάτων μέσω της οποίας μπορεί μια υπηρεσία να καθιερώσει ή να ανανεώσει έναν προγραμματισμένο χρόνο τερματισμού για ένα WS-Resource και καθορίζει τις συνθήκες κάτω από τις οποίες μια υπηρεσία μπορεί να καθορίσει ότι έχει έρθει η πάροδος αυτού του χρόνου τερματισμού. Το ενδιαφέρον μιας υπηρεσίας για ένα WS-Resource συνήθως διαρκεί κάποιο συγκεκριμένο χρονικό διάστημα και είναι σπάνιο να διαρκεί για πάντα. Σε πολλές περιπτώσεις είναι χρήσιμο για τους πελάτες ενός WS-Resource να προκαλούν την άμεση καταστροφή του. Η άμεση καταστροφή ενός WS-Resource μπορεί να γίνει μέσω της ανταλλαγής μηνυμάτων που ορίζεται στη συγκεκριμένη προδιαγραφή. Επιπλέον, η προδιαγραφή ορίζει τον τρόπο που ένας πόρος μπορεί να καταστραφεί μετά την πάροδο συγκεκριμένου χρονικού διαστήματος.
- WS-ServiceGroup (WSRF-SG) [75]: καθορίζει τον τρόπο με τον οποίο υπηρεσίες ή πόροι μπορούν να ομαδοποιηθούν. Σκοπός της προδιαγραφής WS-ServiceGroup είναι η προτυποποίηση της ορολογίας, των εννοιών και της ανταλλαγής μηνυμάτων, καθώς και των WSDL και XML σχημάτων και εγγράφων που απαιτούνται για την περιγραφή ομαδοποιήσεων υπηρεσιών ιστού και πόρων. Η λειτουργικότητα που παρέχεται από τη συγκεκριμένη προδιαγραφή περιορίζεται στην ουσία στα πολύ βασικά, αλλά μπορεί να αποτελέσει τη βάση για τη δημιουργία περιπλοκότερων μηχανισμών για την ομαδοποίηση διαφορετικών υπηρεσιών και την πρόσβαση σε αυτές μέσω ενός μοναδικού σημείου αναφοράς.
- WS-BaseFaults (WSRF-BF) [76]: παρέχει προτυποποίηση για αναφορά σφαλμάτων, όταν κάποιο σφάλμα συμβεί κατά την κλήση μιας υπηρεσίας. Σκοπός της προδιαγραφής

WS-BaseFaults είναι η προτυποποίηση της ορολογίας, των εννοιών, των XML τύπων, καθώς και της χρήσης της WSDL ως τη βάση για τον ορισμό τύπων σφαλμάτων για διεπαφές υπηρεσιών ιστού. Η συγκεκριμένη προδιαγραφή ορίζει ένα πρότυπο XML τύπο σχήματος που περιέχει πληροφορία για ένα σφάλμα και καθορίζει τον τρόπο με τον οποίο αυτός ο τύπος χρησιμοποιείται μέσα σε διεπαφές WSDL.

5.4 Μεσισμικά Πλέγματος

Ένα από τα βασικότερα συστατικά μιας υποδομής πλέγματος είναι το μεσισμικό. Το μεσισμικό πλέγματος είναι το κομμάτι λογισμικού που βρίσκεται ανάμεσα στον χρήστη και τους πόρους του πλέγματος και παρέχει λειτουργικότητα για ασφάλεια, διαχείριση πόρων, πρόσβαση σε δεδομένα, υποστήριξη πολιτικών, χρέωση και άλλες υπηρεσίες που είναι απαραίτητες ώστε οι εφαρμογές, οι χρήστες και οι πάροχοι πόρων να μπορούν να συνεργάζονται αποδοτικά σε ένα περιβάλλον πλέγματος. Βασικός σκοπός του μεσισμικού σε ένα πλέγμα είναι η απόκρυψη της ετερογενούς φύσης των πόρων και η παροχή στους χρήστες και τις εφαρμογές ενός ομοιογενούς περιβάλλοντος με την παροχή ενός συνόλου τυποποιημένων διεπαφών σε ποικίλες υπηρεσίες. Όπως προαναφέρθηκε, ένα πλεγματικό μεσισμικό αποτελείται από ένα σύνολο πρωτοκόλλων, APIs και SDKs, τα οποία προσφέρονται για τη δημιουργία των διαφόρων υποδομών και εφαρμογών. Μεταφορικά, το μεσισμικό λειτουργεί ως η "κόλλα" που συνδέει όλα τα συστατικά στοιχεία ενός πλέγματος σε μία ενιαία κατανεμημένη οντότητα.

Κατά καιρούς έχουν εμφανιστεί διάφορες αρχιτεκτονικές και υλοποιήσεις πλεγματικών μεσισμικών. Μεταξύ αυτών αναφέρονται τα Condor [77], Legion [78], Nimrod/G [79], Unicore [80], Globus Toolkit [62], gLite [81]. Από αυτές τις υλοποιήσεις η πλέον διαδεδομένη για την τεχνολογία του πλέγματος θεωρείται το Globus Toolkit, του οποίου η τελευταία έκδοση (GT4 [82]) υποστηρίζει πλήρως την δημιουργία μιας υποδομής συμβατής με το μοντέλο των υπηρεσιών πλέγματος και τα πρότυπα WSRF. Επίσης, μεγάλης σημασίας είναι και το gLite, το οποίο αποτελεί τη βάση της πανευρωπαϊκής υποδομής πλέγματος EGEE [84] και το οποίο περιγράφεται λεπτομερέστερα στο επόμενο κεφάλαιο.

5.5 Χρησιμότητα Τεχνολογίας Πλέγματος - Οφέλη

Ο πρωταρχικός λόγος που δόθηκε ενδιαφέρον στην τεχνολογία του πλέγματος ήταν η αντιμετώπιση των ολοένα και αυξανόμενων υπολογιστικών απαιτήσεων των σύγχρονων

προβλημάτων. Ένα πρώτο όφελος από την τεχνολογία του πλέγματος είναι ακριβώς ότι γίνεται δυνατή η σύσταση κατανεμημένων συστημάτων πολύ μεγάλης κλίμακας, τα οποία μπορούν να χρησιμοποιηθούν για τη συντομότερη ή ακριβέστερη επίλυση υπολογιστικών προβλημάτων ή ακόμη και για την επίλυση προβλημάτων που θα ήταν αδύνατο να αντιμετωπιστούν από συμβατικά συστήματα. Επιπλέον, το κόστος υπολογισμού από τη χρήση των τεχνολογιών πλέγματος μειώνεται σημαντικά σε σχέση με άλλες προσεγγίσεις. Χωρίς την τεχνολογία πλέγματος, για να αυξήσει ένας οργανισμός τη διαθέσιμη υπολογιστική ισχύ χρειάζεται τυπικά να επενδύσει σε υποδομές υλικού (είτε αγορά επιπλέον hardware, είτε αναβάθμιση του υπάρχοντος). Στις περισσότερες περιπτώσεις το κόστος αγοράς, συντήρησης και διαχείρισης του νέου εξοπλισμού είναι αρκετά μεγάλο. Χρησιμοποιώντας τις τεχνολογίες πλέγματος, είναι δυνατή η συγκέντρωση περισσευούμενων πόρων που βρίσκονται στο εσωτερικό δίκτυο του οργανισμού, ή ακόμα και πόρων που ανήκουν σε γεωγραφικά απομακρυσμένες περιοχές και σε διαφορετικές περιοχές διαχείρισης. Με αυτόν τον τρόπο θα μπορούσαν για παράδειγμα να χρησιμοποιηθούν οι επιτραπέζιοι ηλεκτρονικοί υπολογιστές που είναι διαθέσιμοι σε μια εταιρία, όταν αυτοί είναι ανενεργοί ή όταν λειτουργούν χαμηλότερα από τις δυνατότητές τους.

Επιπλέον, μια από τις σημαντικότερες χρησιμότητες της τεχνολογίας του πλέγματος είναι ότι επιτρέπει και διευκολύνει την ανάπτυξη εφαρμογών και συστημάτων που σχετίζονται με τη συνεργασία (collaboration) μεταξύ διαφορετικών οργανισμών. Η βασική φιλοσοφία του πλέγματος έγκειται στον διαμοιρασμό πόρων, οι οποίοι πόροι μπορεί μεταξύ άλλων να είναι δεδομένα ή αυτόνομα κομμάτια λογισμικού (π.χ. υπολογιστικοί κώδικες). Λόγω της επικέντρωσης σε πρωτόκολλα και μηχανισμούς για τον ασφαλή και συντονισμένο διαμοιρασμό τέτοιων πόρων, τα τελευταία χρόνια το πλέγμα έχει αποτελέσει τη βάση για την ανάπτυξη συστημάτων και εφαρμογών που επιτρέπουν την ενορχήστρωση πόρων με σκοπό τη δημιουργία πολύπλοκων συνεργασιών μεταξύ διαφορετικών ομάδων. Χαρακτηριστική είναι η εμφάνιση της Ηλεκτρονικής Επιστήμης (e-Science). Η έννοια του e-Science εμπεριέχει τη συνεργασία μεταξύ ερευνητικών ομάδων με διαφορετικά γνωστικά αντικείμενα, με σκοπό την επίτευξη κοινών στόχων, όπως η επίλυση ενός πολύπλοκου προβλήματος που απαιτεί τη χρήση ετερογενών πόρων ή τεχνογνωσίας που δεν είναι δυνατό να κατέχεται από έναν μόνο οργανισμό.

6 Περιβάλλον Προσομοίωσης Συστημάτων Ασύρματων Επικοινωνιών σε Πλέγμα

Στο παρόν κεφάλαιο παρουσιάζεται ένα περιβάλλον για την κατανεμημένη προσομοίωση συστημάτων ασύρματων επικοινωνιών, το οποίο κάνει χρήση των τεχνολογιών πλέγματος και στηρίζεται στην τεχνολογία ιστού για την παροχή πρόσβασης σε αυτό. Μια προσομοίωση συστήματος ασύρματων επικοινωνιών είναι στην ουσία μια Monte Carlo προσομοίωση, η οποία αποτελείται από έναν μεγάλο αριθμό πανομοιότυπων και ανεξάρτητων διεργασιών, οι οποίες είναι δυνατόν να εκτελεστούν κατανεμημένα σε μια υποδομή πλέγματος, με αποδοτικό τρόπο και με μικρό κόστος. Οι απαιτήσεις σε υπολογιστικό χρόνο καθώς αυξάνεται η πολυπλοκότητα της προσομοίωσης (π.χ. αυξημένη τάξη τομεοποίησης, έξυπνες κεραίες κτλ, ή συνδυασμός προσομοιώσεων σε επίπεδα συστήματος και ζεύξης) πολύ σύντομα γίνονται απαγορευτικά μεγάλες. Το περιβάλλον που παρουσιάζεται σε αυτό το κεφάλαιο επιτρέπει την κατανεμημένη εκτέλεση προσομοιώσεων συστημάτων ασύρματων επικοινωνιών στην υποδομή πλέγματος του προγράμματος Enabling Grids for E-sciencE (EGEE) [83], [84]. Το περιβάλλον παρέχει μια ασφαλή δικτυακή πύλη ως διεπαφή στο μεσισμικό πλέγματος (π.χ. για υποβολή και επίβλεψη εργασιών, ανάκτηση αποτελεσμάτων κλπ). Παρουσιάζονται λεπτομέρειες υλοποίησης και στατιστικά αποτελέσματα από την εκτέλεση προσομοιώσεων συστημάτων Wideband Code Division Multiple Access (WCDMA) στο πλέγμα, που σχετίζονται με την απόδοση του περιβάλλοντος.

6.1 Εισαγωγή

Λόγω της στοχαστικής φύσης ενός συστήματος ασύρματων επικοινωνιών, η προσομοίωσή του στηρίζεται τυπικά σε μεθόδους Monte Carlo. Μια προσομοίωση Monte Carlo αποτελείται από έναν μεγάλο αριθμό επιμέρους προσομοιώσεων, τα αποτελέσματα των οποίων συγκεντρώνονται και χρησιμοποιούνται για στατιστικούς υπολογισμούς. Κάθε επιμέρους προσομοίωση είναι μια προσομοίωση του συστήματος και μπορεί να γίνει υπολογιστικά απαιτητική καθώς το μοντέλο του συστήματος γίνεται πολυπλοκότερο (π.χ. λαμβάνοντας υπόψη τα αποτελέσματα αυξημένης τάξης τομεοποίησης, έξυπνες κεραίες, κτλ). Σε ένα σύστημα WCDMA για παράδειγμα, λόγω του ότι ο έλεγχος εισόδου βασίζεται σε μηχανισμούς ήπιας χωρητικότητας (soft capacity), απαιτείται να ληφθεί ένας ικανοποιητικός αριθμός στρωμάτων (tiers) γύρω από την κεντρική κυψέλη για τον υπολογισμό της χωρητικότητας. Οι υψηλές υπολογιστικές απαιτήσεις για μια προσομοίωση, καθώς και ο μεγάλος αριθμός επιμέρους προσομοιώσεων που απαιτούνται από μια μέθοδο Monte Carlo έχουν ως αποτέλεσμα ότι συνολικά η προσομοίωση του συστήματος γίνεται εξαιρετικά χρονοβόρα.

Το παρόν κεφάλαιο πραγματεύεται την ανάπτυξη ενός περιβάλλοντος προσομοίωσης συστημάτων ασύρματων επικοινωνιών σε υποδομή πλέγματος. Επιπλέον, παρουσιάζονται τα στοιχεία λειτουργίας του μέσω της εκτέλεσης προσομοιώσεων μιας σειράς σεναρίων για σύστημα WCDMA. Οι προσομοιώσεις υπολογισμού χωρητικότητας ενός τέτοιου συστήματος αρχικά εκτελούνταν σειριακά σε έναν σταθμό εργασίας [85]. Τυπικά, απαιτούνταν σε κάθε Monte Carlo προσομοίωση περισσότερες από 1000 επιμέρους προσομοιώσεις, πράγμα που καθιστά την προσομοίωση εξαιρετικά υπολογιστικά απαιτητική. Μια λύση για τη μείωση του χρόνου εκτέλεσης των προσομοιώσεων είναι η χρησιμοποίηση μιας συστοιχίας (cluster) υπολογιστών. Αυτή η προσέγγιση χρησιμοποιείται ευρέως για την μείωση του χρόνου εκτέλεσης διαφόρων εφαρμογών. Παρόλα αυτά, υπάρχουν κάποιοι τεχνικοί περιορισμοί που πρέπει να ληφθούν υπόψη. Καταρχήν, το κόστος απόκτησης και συντήρησης μιας τέτοιας υποδομής μπορεί να είναι αρκετά μεγάλο. Επιπλέον, λόγω του ότι μια προσομοίωση WCDMA δεν μπορεί να παραλληλοποιηθεί με αποδοτικό τρόπο λόγω της αλληλεξάρτησης μεταξύ των χρηστών, είναι πολύ δύσκολη η επίτευξη ικανοποιητικής κλιμάκωσης παραλληλοποιώντας την προσομοίωση σε επίπεδο Monte Carlo.

Μια εναλλακτική προσέγγιση είναι η εκτέλεση των προσομοιώσεων σε μια υποδομή πλέγματος. Λόγω του ότι μια προσομοίωση Monte Carlo αποτελείται ουσιαστικά από έναν

μεγάλο αριθμό πανομοιότυπων και τελείως ανεξάρτητων προσομοιώσεων, μπορεί να αποσυντεθεί σε έναν αριθμό διεργασιών που δεν χρειάζεται να επικοινωνούν μεταξύ τους, παρά μόνο κατά την τελική συγκέντρωση αποτελεσμάτων. Αυτό καθιστά τις προσομοιώσεις αυτές κατάλληλες για αποδοτική εκτέλεση σε μια υποδομή πλέγματος. Η μέθοδος της εκτέλεσης Monte Carlo προσομοιώσεων σε πλέγμα έχει ήδη χρησιμοποιηθεί και έχει αποδειχθεί αποδοτική σε διάφορες εφαρμογές (π.χ. εφαρμογές φυσικής σωματιδίων [86] και εφαρμογές διαχείρισης οικονομικού ρίσκου [87]).

Το περιβάλλον προσομοίωσης που παρουσιάζεται σε αυτό το κεφάλαιο χρησιμοποιεί την υποδομή πλέγματος που έχει αναπτυχθεί από το πρόγραμμα EGEE, το οποίο είναι χρηματοδοτούμενο από την Ευρωπαϊκή Ένωση και έχει ως στόχο την εκμετάλλευση των πρόσφατων εξελίξεων στην τεχνολογία πλέγματος για την ανάπτυξη και εγκατάσταση μιας υποδομής πλέγματος που είναι διαθέσιμη στους επιστήμονες 24 ώρες την ημέρα. Η υποδομή EGEE αποτελείται από υπολογιστικούς πόρους κατανεμημένους σε 32 χώρες παγκοσμίως, οι οποίοι ανήκουν σε διαφορετικούς εικονικούς οργανισμούς ανάλογα με το ερευνητικό ενδιαφέρον στο οποίο απευθύνονται ή την γεωγραφική περιοχή.

Γενικά, η αλληλεπίδραση με το μεσισμικό πλέγματος απαιτεί κάποια συγκεκριμένη τεχνογνωσία σχετικά με τα χαρακτηριστικά της υποδομής πλέγματος και τον τρόπο που αυτό λειτουργεί. Για να είναι δυνατή η χρήση του περιβάλλοντος από χρήστες που δεν είναι εξοικειωμένοι με την τεχνολογία πλέγματος, ένας βασικός στόχος ήταν η δημιουργία μιας διεπαφής φιλικής προς το χρήστη. Η διεπαφή αυτή υλοποιήθηκε ως μια δικτυακή πύλη. Η πύλη υλοποιήθηκε με χρήση του πλαισίου λογισμικού GridSphere [88] και αποτελείται από δύο portlets που είναι συμβατά με τα JSR-168 [89] πρότυπα και τα οποία είναι υπεύθυνα για την αλληλεπίδραση με το μεσισμικό πλέγματος.

6.2 Η Υποδομή Πλέγματος

Το περιβάλλον που περιγράφεται σε αυτό το κεφάλαιο χρησιμοποιεί τους υπολογιστικούς πόρους που παρέχονται από το πλέγμα της υποδομής EGEE, το οποίο βασίζεται στο μεσισμικό gLite [81]. Η ανάπτυξή του ξεκίνησε από το CERN για τις ανάγκες του πλέγματος που υποστηρίζει το πείραμα του Large Hadron Collider [90] και στη συνέχεια αποτέλεσε τη βάση για την ανάπτυξη της υποδομής EGEE. Η υποδομή EGEE ξεκίνησε με επίκεντρο τις επιστήμες της φυσικής υψηλών ενεργειών και την βιοπληροφορική, ενώ σήμερα παρέχει υπολογιστικούς πόρους για όλες τις επιστήμες στην

ευρωπαϊκή ερευνητική και ακαδημαϊκή κοινότητα. Η διασύνδεση των παρεχόμενων υπολογιστικών πόρων και η επίτευξη της παρεχόμενης λειτουργικότητας γίνεται με τη βοήθεια του gLite. Το gLite περιλαμβάνει υπηρεσίες για ασφάλεια, διαχείριση εργασιών, διαχείριση δεδομένων, καθώς και για επίβλεψη των πόρων του συστήματος. Η οργάνωση και ο διαχωρισμός των παρεχόμενων πόρων γίνεται με βάση την έννοια των VOs,όπως αυτοί περιγράφτηκαν στο προηγούμενο κεφάλαιο. Η υποδομή πλέγματος του EGEE είναι οργανωμένη σε VOs, ανάλογα με το ερευνητικό αντικείμενο ή τη γεωγραφική περιοχή των χρηστών. Για παράδειγμα, οι VOs Alice, Atlas, Babar, CDF, CMS, Compass, D0, Dteam, LHCb αποτελούν εικονικούς οργανισμούς στους οποίους ανήκουν χρήστες που ασχολούνται με την ανάλυση και επίλυση προβλημάτων φυσικής σωματιδίων. Αντίστοιχα υπάρχουν εικονικοί οργανισμοί για άλλες επιστήμες (π.χ. βιοεπιστήμες και σεισμολογία). Επιπλέον, υπάρχουν εικονικοί οργανισμοί στους οποίους μπορεί να ενταχθεί κάποιος χρήστης ανάλογα με την γεωγραφική περιοχή στην οποία βρίσκεται (π.χ. SEE-VO για την νότιο-ανατολική Ευρώπη και VOCE για την κεντρική Ευρώπη).

Για να αποκτήσει ένας χρήστης πρόσβαση στους υπολογιστικούς πόρους του πλέγματος EGEE, πρέπει καταρχήν να ενταχθεί σε κάποιον εικονικό οργανισμό. Αυτό συνεπάγεται την έκδοση προσωπικών ψηφιακών πιστοποιητικών από την κατάλληλη αρχή πιστοποίησης (Certification Authority – CA), τα οποία χρησιμοποιούνται από το χρήστη κάθε φορά που επιθυμεί να χρησιμοποιήσει τους υπολογιστικούς πόρους του εικονικού οργανισμού στον οποίο ανήκει.

Τη βάση της υποδομής EGEE αποτελεί το gLite , το οποίο παρέχει την απαραίτητη λειτουργικότητα για την χρήση των κατανεμημένων υπολογιστικών πόρων και των κατανεμημένων αποθηκευτικών χώρων. Τα σημαντικότερα συστατικά στοιχεία του gLite είναι τα εξής [91]:

- Διεπαφή Χρήστη (User Interface – UI): αποτελεί το σημείο πρόσβασης στο πλέγμα και μπορεί να είναι οποιοδήποτε μηχάνημα στο οποίο οι χρήστες έχουν προσωπικό λογαριασμό και έχουν εγκαταστήσει τα ψηφιακά πιστοποιητικά τους. Μέσω ενός UI, ένας χρήστης μπορεί να πιστοποιήσει την ταυτότητά του και να αποκτήσει πρόσβαση στους πόρους του πλέγματος. Το UI παρέχει στον χρήστη μια διεπαφή γραμμής εντολών για τη διεξαγωγή βασικών λειτουργιών, όπως υποβολή εργασιών προς εκτέλεση, εύρεση της κατάστασης μιας εργασίας, ανάκτηση των αποτελεσμάτων μιας εργασίας, κτλ.
- Υπολογιστικό Στοιχείο (Computing Element – CE): αποτελεί ένα σύνολο υπολογιστικών πόρων (π.χ. cluster) σε ένα συγκεκριμένο site. Ένα CE περιλαμβάνει

μια πύλη πλέγματος (Grid Gate – GG), η οποία λειτουργεί ως μια γενική διεπαφή για το cluster και ένα τοπικό σύστημα διαχείρισης πόρων (Local Resource Management System – LRMS), το οποίο λειτουργεί ως διαχειριστής των πόρων. Το ίδιο το cluster αποτελείται από ένα σύνολο κόμβων-εργατών (Worker Nodes – WNs), στους οποίους εκτελούνται οι εργασίες.

- Αποθηκευτικό Στοιχείο (Storage Element – SE): παρέχει ομοιόμορφη πρόσβαση στους αποθηκευτικούς πόρους. Μπορεί να ελέγχει απλούς εξυπηρετητές δίσκων, μεγάλες συστοιχίες δίσκων ή ακόμα και συστήματα μαζικής αποθήκευσης (Mass Storage Systems – MSS).
- Υπηρεσία Πληροφορίας (Information Service – IS): παρέχει πληροφορίες για τους πόρους του πλέγματος και για την κατάστασή τους. Η περιγραφή των πληροφοριών γίνεται μέσω του GLUE Schema [92], το οποίο ορίζει ένα δενδρικό μοντέλο δεδομένων. Τα συστήματα πληροφορίας που χρησιμοποιούνται από την IS του gLite είναι τα Globus Monitoring and Discovery System (MDS) [93] και το Relational Grid Monitoring Architecture (R-GMA) [94].
- Σύστημα Διαχείρισης Δεδομένων (Data Management System – DMS): σε ένα περιβάλλον πλέγματος, τα αρχεία μπορούν να έχουν αντίγραφα σε διάφορες τοποθεσίες. Ιδανικά, οι χρήστες δεν χρειάζεται να γνωρίζουν τη φυσική τοποθεσία ενός αρχείου και θα πρέπει να μπορούν να το προσπελάσουν με χρήση λογικών ονομάτων. Ένα αρχείο μπορεί να προσπελαστεί στο πλέγμα με τα εξής διαφορετικά ονόματα: Grid Unique Identifier (GUID), Logical File Name (LFN), Storage URL (SURL) και Transport URL (TURL). Τα δύο πρώτα ονόματα αναγνωρίζουν ένα αρχείο ανεξάρτητα τοποθεσίας, ενώ τα δύο τελευταία περιέχουν αντίστοιχα πληροφορία για την πραγματική τοποθεσία ενός αντιγράφου του αρχείου και για τον τρόπο πρόσβασης σε αυτό.
- Σύστημα Διαχείρισης Υπολογιστικού Φορτίου (Workload Management System – WMS): σκοπός του WMS είναι να δέχεται τις εργασίες των χρηστών, να τις αναθέτει στο καταλληλότερο CE, να παρακολουθεί την κατάστασή τους και να ανακτά τα αποτελέσματα που προκύπτουν από αυτές. Για την περιγραφή των εργασιών που υποβάλλονται στο πλέγμα χρησιμοποιείται μια γλώσσα περιγραφής εργασίας (Job Description Language – JDL), με την οποία μπορούν να δηλωθούν τα χαρακτηριστικά και οι όποιες απαιτήσεις της προς εκτέλεση εργασίας. Η επιλογή του CE στο οποίο αποστέλλεται τελικά η εργασία γίνεται με μια διαδικασία ταιριάσματος (match-making process), η οποία επιλέγει από τα διαθέσιμα CEs αυτά

που ικανοποιούν τις απαιτήσεις που εκφράζονται από το χρήστη στο JDL αρχείο και βρίσκονται κοντά στα απαιτούμενα αρχεία εισόδου. Η πρόσβαση στα αρχεία γίνεται μέσω μιας διεπαφής εντοπισμού δεδομένων (Data Location Interface – DLI), η οποία αποτελεί μια γενική διεπαφή σε έναν κατάλογο αρχείων. Τέλος, η παρακολούθηση της κατάστασης των εργασιών γίνεται από μια υπηρεσία καταγραφής στοιχείων (Logging and Bookkeeping – LB), η οποία συλλέγει γεγονότα από διάφορα στοιχεία του WMS και καταγράφει την κατάσταση καθώς και την ιστορία των εργασιών.

Στη συνέχεια περιγράφονται με περισσότερη λεπτομέρεια τα συστατικά στοιχεία και οι επιμέρους υπηρεσίες του gLite.

6.2.1 Υποδομή Ασφάλειας

Η ασφάλεια στο gLite βασίζεται στην έννοια της κρυπτογράφησης δημοσίου κλειδιού (public key encryption). Κάθε χρήστης (ή οποιαδήποτε άλλη οντότητα – π.χ. ένας εξυπηρετητής) έχει ένα ιδιωτικό κλειδί που δημιουργείται με τυχαίο τρόπο. Αυτό το κλειδί είναι ένας αριθμός που χρησιμοποιείται ως μυστικός κωδικός για απόδειξη της ταυτότητας. Αυτό σημαίνει ότι αν κάποιος κλέψει το ιδιωτικό κλειδί, μπορεί να φέρεται πλήρως ως ο ιδιοκτήτης του κλειδιού και για αυτό το λόγο το ιδιωτικό κλειδί πρέπει να φυλάσσεται προσεκτικά. Κάθε ιδιωτικό κλειδί σχετίζεται μαθηματικά με ένα άλλο κλειδί που ονομάζεται δημόσιο κλειδί και το οποίο μπορεί να είναι γνωστό στον καθένα. Θεωρητικά είναι δυνατόν να υπολογιστεί το ιδιωτικό κλειδί με βάση το δημόσιο κλειδί, όμως πρακτικά αυτό είναι αδύνατο λόγω του χρόνου που απαιτείται για κάτι τέτοιο (ο οποίος αυξάνεται εκθετικά με την αύξηση του μεγέθους του κλειδιού). Αντίθετα, ο υπολογισμός του δημοσίου κλειδιού από το ιδιωτικό κλειδί είναι εύκολη διαδικασία και για αυτό το λόγο αυτά ονομάζονται συχνά ασύμμετρα κλειδιά.

Για να είναι χρήσιμο, το δημόσιο κλειδί πρέπει να συνδέεται με κάποια πληροφορία σχετικά με την οντότητα στην οποία ανήκει το κλειδί (π.χ. χρήστης). Αυτή η πληροφορία υπάρχει σε μια συγκεκριμένη μορφή γνωστή ως X.509 πιστοποιητικό. Το πιο σημαντικό στοιχείο του πιστοποιητικού είναι το Subject Name (SN), το οποίο έχει την εξής μορφή:

/C=GR/O=eScience/OU=CLRC/L=RAL/CN=theodoros athanaileas

Ένα πιστοποιητικό περιέχει επίσης και άλλες πληροφορίες, όπως μια ημερομηνία λήξης μετά το πέρας της οποίας το πιστοποιητικό παύει να ισχύει.

Τα πιστοποιητικά εκδίδονται από μια CA. Για το πλέγμα EGEE υπάρχουν ειδικές CAs που ανήκουν σε ακαδημαϊκούς οργανισμούς και που εξυπηρετούν χρήστες σε συγκεκριμένη

γεωγραφική περιοχή. Η CA ακολουθεί συγκεκριμένες διαδικασίες για την πιστοποίηση της ταυτότητας των χρηστών και για το ότι αυτοί δικαιούνται πιστοποιητικό. Για να επιτρέπεται η επαλήθευση των πληροφοριών του πιστοποιητικού, η CA το υπογράφει με το δικό της ιδιωτικό κλειδί. Κάθε ένας που θέλει να επαληθεύσει την εγκυρότητα ενός πιστοποιητικού χρειάζεται να ξέρει το δημόσιο κλειδί της CA, πράγμα που σημαίνει ότι και η CA έχει ένα δικό της πιστοποιητικό. Αυτό θα μπορούσε να δημιουργήσει πρόβλημα αέναης αναδρομής, αλλά αυτό αποφεύγεται με το να υπογράφει η CA τα δικά της πιστοποιητικά (τα οποία ονομάζονται root certificates). Αυτά τα πιστοποιητικά στη συνέχεια διανέμονται με κάποιο ασφαλή τρόπο.

Για την απευθείας αλληλεπίδραση με μια απομακρυσμένη υπηρεσία, το πιστοποιητικό μπορεί να χρησιμοποιηθεί για την απόδειξη της ταυτότητας. Παρόλα αυτά, σε ένα πλέγμα συχνά απαιτείται μια απομακρυσμένη υπηρεσία να λειτουργήσει εκ μέρους του χρήστη, π.χ. μια εργασία που εκτελείται σε ένα απομακρυσμένο site πρέπει να έχει τη δυνατότητα να συνομιλεί με άλλους κόμβους για μεταφορά αρχείων, και για αυτό το λόγο πρέπει να αποδείξει ότι ανήκει σε κάποιο χρήστη (αυτό ονομάζεται αντιπροσώπευση – delegation). Λόγω του ότι το ιδιωτικό κλειδί είναι πολύ σημαντικό για την ασφάλεια, αυτό δεν πρέπει να στέλνεται σε απομακρυσμένους κόμβους, οι οποίοι ενδέχεται να είναι ανασφαλείς. Η λύση είναι η χρήση πληρεξούσιων πιστοποιητικών (proxy certificates). Αυστηρά μιλώντας, ένα πληρεξούσιο είναι και αυτό ένα πιστοποιητικό και για τη δημιουργία του δημιουργείται ένα ζεύγος δημόσιου/ιδιωτικού κλειδιού, το οποίο υπογράφεται από ένα μακροπρόθεσμο (long-term) ιδιωτικό κλειδί. Τα πληρεξούσια έχουν συνήθως μικρό χρόνο ζωής, τυπικά 12 ώρες.

Όταν γίνεται υποβολή μιας εργασίας, στέλνονται μαζί της το πληρεξούσιο πιστοποιητικό, το ιδιωτικό κλειδί του και το long-term πιστοποιητικό (αλλά όχι το long-term ιδιωτικό κλειδί). Όταν μια εργασία θέλει να αποδείξει την αντιπροσωπευτική της ταυτότητα σε κάποια άλλη υπηρεσία, στέλνει το πληρεξούσιο και το κανονικό πιστοποιητικό, αλλά όχι το ιδιωτικό κλειδί του πληρεξούσιου. Με βάση την αλυσίδα των πιστοποιητικών μπορεί να αποδείξει ότι έχει την εξουσιοδότηση να χρησιμοποιήσει το αντίστοιχο SN. Σε κάποιες περιπτώσεις η εργασία μπορεί να δημιουργήσει δικό της πληρεξούσιο, κάνοντας μεγαλύτερη αυτήν την αλυσίδα. Σε όρους ασφάλειας, το πληρεξούσιο πιστοποιητικό πρόκειται για συμβιβασμό. Από τη στιγμή που το ιδιωτικό κλειδί αποστέλλεται με αυτό, όποιος το κλέψει μπορεί να φέρεται ως ο ιδιοκτήτης του πιστοποιητικού και για αυτό το λόγο χρειάζεται προσοχή στη χρήση τους. Επιπλέον, δεν υπάρχει τρόπος ανάκλησης των πληρεξούσιων. Για αυτό το λόγο τα πληρεξούσια έχουν

περιορισμένη διάρκεια ζωής (τυπικά λίγες ώρες), ώστε να περιορίζεται η ζημιά που μπορεί να γίνει.

Για τη διαχείριση της πληροφορίας σχετικά με τους ρόλους και τα προνόμια των χρηστών μέσα σε έναν VO, στην υποδομή της EGEE γίνεται χρήση του συστήματος VOMS (VO Management Service). Η πληροφορία αυτή παρουσιάζεται στις υπηρεσίες μέσω μιας επέκτασης του πληρεξούσιου. Τη στιγμή που αυτό δημιουργείται, υπάρχει επικοινωνία με έναν ή περισσότερους VOMS εξυπηρετητές, οι οποίοι επιστρέφουν ένα μίνι πιστοποιητικό γνωστό ως Attribute Certificate (AC), το οποίο υπογράφεται από τον VO και περιέχει πληροφορία για ομάδες στις οποίες ανήκει ο χρήστης και οποιουσδήποτε ρόλους έχει ο χρήστης μέσα στον VO. Για τη δημιουργία ενός VOMS proxy, τα ACs ενσωματώνονται σε ένα κανονικό proxy και το συσσωμάτωμα υπογράφεται από το ιδιωτικό κλειδί του πιστοποιητικού του χρήστη. Οι υπηρεσίες μπορούν στη συνέχεια να αποκωδικοποιήσουν την πληροφορία του VOMS και να τη χρησιμοποιήσουν ανάλογα (π.χ. ο χρήστης μπορεί να πραγματοποιήσει μια λειτουργία μόνο αν έχει ένα συγκεκριμένο ρόλο σε έναν συγκεκριμένο VO). Οι ιδιότητες VOMS μπορούν να χρησιμοποιηθούν μόνο σε ένα proxy και δεν μπορούν να ενσωματωθούν σε ένα πιστοποιητικό που εκδίδεται από την CA. Επίσης, κάθε AC έχει δική του διάρκεια ζωής (τυπικά 12 ώρες) και μπορεί να λήξει προτού λήξει το proxy στο οποίο ανήκει.

6.2.2 Σύστημα Διαχείρισης Υπολογιστικού Φορτίου (Workload Management System - WMS)

Το WMS είναι υπεύθυνο για τη διαχείριση των εργασιών που υποβάλλονται από τους χρήστες. Είναι το σύστημα που αντιστοιχεί τις απαιτήσεις μίας υποβαλλόμενης εργασίας με τους διαθέσιμους υπολογιστικούς πόρους και προγραμματίζει την εκτέλεση της εργασίας σε μία κατάλληλη συστοιχία υπολογιστών. Στη συνέχεια καταγράφει την πρόοδο εκτέλεσης της εργασίας και επιτρέπει στο χρήστη να ανακτήσει τα δεδομένα εξόδου όταν η εργασία τελειώσει την εκτέλεση της. Το gLite WMS χρησιμοποιείται στα ακόλουθα μηχανήματα στο πλέγμα:

- *Διεπαφή Χρήστη (UI)*: είναι το μηχάνημα που επιτρέπει στους χρήστες να έχουν πρόσβαση στις λειτουργίες του πλέγματος. Σε αυτό οι χρήστες έχουν προσωπικό λογαριασμό και έχουν εγκαταστημένο το προσωπικό τους πιστοποιητικό. Αποτελεί την πύλη για τις διάφορες υπηρεσίες του πλέγματος. Καταρχάς, μέσω του UI μπορεί να γίνει η πιστοποίηση και εξουσιοδότηση του χρήστη, ώστε να μπορεί αυτός να

χρησιμοποιήσει τους πόρους του πλέγματος. Επίσης, το UI παρέχει στους χρήστες μια διεπαφή γραμμής εντολών για την αλληλεπίδραση με τους πόρους και τις υπηρεσίες του πλέγματος. Οι λειτουργίες που υλοποιεί είναι:

- Εύρεση όλων των υπολογιστικών πόρων που είναι συμβατοί με τις απαιτήσεις μιας υποβαλλόμενης εργασίας
- Υποβολή (submission) εργασιών
- Παρακολούθηση της πορείας εκτέλεσης της εργασίας
- Ακύρωση εργασιών
- Ανάκτηση των πληροφοριών υποβολής εργασιών
- Λήψη της εξόδου των εργασιών που έχουν ολοκληρωθεί
- Ανάκτηση των δεδομένων εξόδου από τις εργασίες που εκτελέστηκαν

Κάθε site που αποτελεί μέρος του gLite πλέγματος έχει ένα ή περισσότερα UIs διαθέσιμα.

- *Διαμεσολαβητής Πόρων (Resource Broker – RB):* είναι το μηχάνημα που λαμβάνει τις εντολές χρηστών για την υποβολή μίας εργασίας και εξετάζει τους καταλόγους πληροφοριών για να βρει τους κατάλληλους υπολογιστικούς πόρους για την εκτέλεση της εργασίας. Στο μηχάνημα αυτό συνήθως εκτελούνται οι ακόλουθες υπηρεσίες:
  - Ο εξυπηρετητής δικτύου (Network Server) παραλαμβάνει τις αιτήσεις από το UI, πιστοποιεί την αυθεντικότητα του χρήστη, αντιγράφει τα αρχεία εισόδου και εξόδου των εργασιών που έχει καθορίσει ο χρήστης μεταξύ του UI και του RB, προαιρετικά καταγράφει το πληρεξούσιο πιστοποιητικό του χρήστη για περιοδική ανανέωση από την υπηρεσία ανανέωσης πληρεξούσιου (Proxy Renewal Service) και προωθεί τις αιτήσεις στον διαχειριστή υπολογιστικού φορτίου (Workload Manager – WM).
  - Όταν μία εργασία υποβληθεί, ο WM καλεί την υπηρεσία ταιριάσματος (Match-making Service) για να βρει με τη βοήθεια του συστήματος πληροφοριών και ενός διαχειριστή εντοπισμού αντιγράφων (Replica Locate Manager) τους υπολογιστικούς πόρους που τηρούν τις προϋποθέσεις για να εκτελέσουν την υποβληθείσα εργασία. Στη συνέχεια καλείται ένας ελεγκτής εργασίας (Job Controller) για να υποβάλλει την εργασία στο τοπικό σύστημα διαχείρισης εργασιών.

- Το τοπικό σύστημα διαχείρισης εργασιών υποβάλλει την εργασία στο CE μαζί με μία επιπλέον εργασία επίβλεψης για κάθε CE και για κάθε χρήστη έτσι ώστε να παρακολουθεί τις εργασίες των χρηστών. Υπεύθυνο για τις ενεργές εργασίες που εκτελούνται είναι ένα σύστημα καταγραφής επίβλεψης (Log Monitor). Εάν μια εργασία δεν εκτελεστεί στη συστοιχία των υπολογιστικών πόρων που έχει ανατεθεί, τότε το Log Monitor ενημερώνει το WM για πιθανή νέα υποβολή της εργασίας σε ένα καινούργιο CE.
- Η κατάσταση εκτέλεσης μιας εργασίας, όπως προκύπτει από τις διάφορες υπηρεσίες του WM, συλλέγεται από την υπηρεσία LB και αποθηκεύεται σε μία βάση δεδομένων. Η συλλογή των πληροφοριών γίνεται από τα LB local-loggers που εκτελούνται στο RB και στο CE. Η αποθήκευση των πληροφοριών αυτών στη βάση δεδομένων γίνεται από τον LB εξυπηρετητή, που συνήθως βρίσκεται στο RB. Η βάση δεδομένων μπορεί να ερωτηθεί από το χρήστη είτε χρησιμοποιώντας το UI είτε από υπηρεσίες που εκτελούνται στο RB.
- Στο RB εκτελούνται και υπηρεσίες όπως MySQL εξυπηρετητής για το Logging and Bookkeeping και GridFTP εξυπηρετητής.

- *Υπολογιστικό Στοιχείο (Computing Element – CE) :* αποτελεί τη διεπαφή πλέγματος "grid interface" σε μία συστοιχία υπολογιστών. Με τον όρο CE αναφερόμαστε και στο μηχάνημα στο οποίο εκτελούνται οι υπηρεσίες του πλέγματος και στις ουρές αναμονής που χρησιμοποιούνται από το τοπικό σύστημα. Το CE διαχειρίζεται μία "φάρμα" ομογενών υπολογιστικών κόμβων τα οποία ονομάζονται κόμβοι-εργάτες (Worker Nodes – WNs). Στα WNs είναι διαθέσιμες όλες οι εντολές και τα APIs για να εκτελούν πράξεις σε πόρους και δεδομένα του πλέγματος. Επίσης στο CE εκτελείται ένας gatekeeper που δέχεται τις αιτήσεις από το τοπικό σύστημα διαχείρισης και δημιουργεί έναν διαχειριστή (Job Manager) για κάθε εργασία, δηλαδή ένα γενικό interface στις βασικές συναρτήσεις των ουρών αναμονής. Οι διαχειριστές εργασιών χρησιμοποιούνται μόνο για να υποβάλλουν ή να ακυρώνουν εργασίες. Η κατάσταση της εκτέλεσης μίας εργασίας παρακολουθείται από ένα grid monitor μέσω μίας διαδικασίας ερωτήσεων από μία διεργασία που τρέχει στο CE για κάθε χρήστη. Κάθε site που αποτελεί μέρος του πλέγματος διαθέτει ένα ή περισσότερα CE και μία φάρμα από WNs που ανήκουν σε αυτά.
- *Κόμβος-Εργάτης (Worker Node – WN):* είναι ο κόμβος όπου εκτελούνται οι εργασίες. Στους κόμβους αυτούς δεν εκτελούνται υπηρεσίες του gLite και απαιτείται μόνο μικρό μέρος του μεσισμικού για να είναι συμβατοί με τις τεχνολογίες πλέγματος, για

παράδειγμα οι εντολές του gLite και οι βιβλιοθήκες που μπορεί μία εργασία να χρειαστεί.

- *Εξυπηρετητής Πληρεξούσιων (Proxy Server):* Πρόκειται για υπηρεσία που εκτελείται αυτόνομα και η οποία σε συνεργασία με την υπηρεσία ανανέωσης πληρεξούσιου που εκτελείται στον RB χρησιμοποιείται για εργασίες που διαρκούν μεγάλο χρονικό διάστημα. Σκοπός της υπηρεσίας είναι η παροχή έγκυρων πληρεξούσιων, ώστε να μην αποτυγχάνουν οι εργασίες λόγω λήξης των πληρεξούσιων του χρήστη.

6.2.3 Γλώσσα Περιγραφής Εργασίας

Για την υποβολή μιας εργασίας στο πλέγμα μέσω του μεσισμικού gLite, είναι απαραίτητο να δοθεί κατά την υποβολή ένα αρχείο περιγραφής της εργασίας με βάση μια γλώσσα περιγραφής. Η Γλώσσα Περιγραφής Εργασίας (Job Description Language – JDL) είναι μια υψηλού επιπέδου γλώσσα που στηρίζεται στην γλώσσα Classified Advertisements (ClassAd) [95] και η οποία χρησιμοποιείται για την περιγραφή μεμονωμένων εργασιών ή εργασιών με οποιαδήποτε μεταξύ τους συσχέτιση. Η JDL χρησιμοποιείται στο gLite για την περιγραφή χαρακτηριστικών και περιορισμών που πρέπει να ληφθούν υπόψη από το WMS για την επιλογή του καλύτερου πόρου για την εκτέλεση μιας εργασίας. Σε αυτό το σημείο περιγράφονται τα βασικά στοιχεία της JDL, ενώ μια πλήρη περιγραφή της σύνταξης της JDL μπορεί να βρεθεί στο [96]. Ένα αρχείο JDL περιέχει γραμμές με την εξής μορφή:

attribute = expression;

Στη συνέχεια φαίνεται ένα παράδειγμα ενός αρχείου JDL:

```
Executable = "test.sh";
InputSandbox = {"test.sh"};
StdOutput = "std.out";
StdError = "std.err";
OutputSandbox = {"std.out", "std.err"};
Requirements = other.GlueCEUniqueID == "lxshare0286.cern.ch:2119/jobmanager-pbs-
short";
```

Listing 9. Παράδειγμα αρχείου περιγραφής εργασίας

Στο Listing 9 περιγράφεται μια απλή εργασία. Ως εκτελέσιμο ορίζεται το script test.sh, και αρχείο εισόδου το ίδιο το script. Επιπλέον, ορίζεται ότι το standard output και το standard error του εκτελέσιμου αρχείου θα καταγραφούν στα αρχεία “std.out” και “std.err” αντίστοιχα, τα οποία θα μεταφερθούν μετά το τέλος της εργασίας στο UI του χρήστη.

Τέλος, ορίζονται κάποιες απαιτήσεις τις οποίες πρέπει να πληροί το CE στο οποίο θα υποβληθεί τελικά η εργασία (στη συγκεκριμένη περίπτωση απαιτείται υποβολή σε CE με το συγκεκριμένο όνομα).

6.2.4 Σύστημα Διαχείρισης Δεδομένων (Data Management System – DMS)

Το DMS επιτρέπει στους χρήστες να μετακινούν αρχεία από και προς το πλέγμα, να τα αντιγράφουν σε διάφορες τοποθεσίες του πλέγματος και να τα εντοπίζουν. Αυτά επιτυγχάνονται με τη μεταφορά δεδομένων με διάφορα πρωτόκολλα, όπως το GridFTP, και αλληλεπιδρώντας με ένα κεντρικό σύστημα καταλόγου πληροφοριών, την υπηρεσία εντοπισμού αντιγράφων (Replica Location Service – RLS). Η αναφορά σε αρχεία στο πλέγμα μπορεί να επιτευχθεί με διάφορους τρόπους, οι οποίοι μπορεί είτε να παρέχουν πληροφορία για τη φυσική τοποθεσία των αντιγράφων των αρχείων, είτε να την αποκρύπτουν (π.χ. με χρήση λογικών ονομάτων). Ρόλος του RLS είναι η αντιστοίχιση μεταξύ των λογικών ονομάτων των αρχείων και των φυσικών θέσεων στις οποίες βρίσκονται αυτά, ώστε να μπορούν να ανακτηθούν. Η υπηρεσία RLS αποτελείται από δύο συνιστώσες: τον κατάλογο τοπικών αντιγράφων (Local Replica Catalog – LRC) και τον κατάλογο μεταδεδομένων αντιγράφων (Replica Metadata Catalog – RMC). Ο LRC περιέχει την αντιστοίχιση μεταξύ των φυσικών ονομάτων των αρχείων και των μοναδικών αναγνωριστικών τους που χρησιμοποιούνται στο πλέγμα, ενώ το RMC περιέχει τις αντιστοιχίες μεταξύ των λογικών ονομάτων των αρχείων και των μοναδικών αναγνωριστικών.

Η οντότητα του πλέγματος που παρέχει πρόσβαση και υπηρεσίες σε αποθηκευτικούς χώρους είναι το στοιχείο αποθήκευσης (SE). Το SE μπορεί να ελέγχει μεγάλους σκληρούς δίσκους ή συστήματα μαζικής αποθήκευσης. Το πρωτόκολλο μεταφοράς δεδομένων που χρησιμοποιείται είναι το GSIFTP [97], το οποίο περιλαμβάνει τις ίδιες λειτουργίες με το FTP πρωτόκολλο και έχει επεκταθεί ώστε να χρησιμοποιεί τους μηχανισμούς ασφάλειας του πλέγματος. Το συγκεκριμένο πρωτόκολλο μεταφοράς δεδομένων είναι υπεύθυνο για την ασφαλή και γρήγορη μεταφορά αρχείων από και προς το SE και χρησιμοποιείται για πρόσβαση σε δεδομένα σε απομακρυσμένα sites.

6.2.5 Σύστημα Πληροφοριών (Information System - IS)

Το IS παρέχει πληροφορίες σχετικά με τους υπολογιστικούς πόρους του πλέγματος και την κατάσταση στην οποία βρίσκονται. Η πληροφορία για την κατάσταση των κόμβων

γίνεται γνωστή από υπηρεσίες που εκτελούνται στους ίδιους τους κόμβους και αποθηκεύεται σε κεντρικές βάσεις δεδομένων. Την πληροφορία αυτή χρησιμοποιεί το WMS για να αντιστοιχήσει τις εργασίες που υποβάλλονται σε αυτό για εκτέλεση με τους κόμβους που ικανοποιούν τις απαιτήσεις τους και να τις δρομολογήσει σε αυτούς. Η πληροφορία αυτή επίσης χρησιμοποιείται από το DMS για να επιλέξει αποθηκευτικούς πόρους, καθώς και από τα συστήματα επίβλεψης.

Το IS του gLite στηρίζεται στο Globus Monitoring and Discovery Service [93]. Η μορφή των πληροφοριών ακολουθεί το GLUE Schema [92], το οποίο στοχεύει στο να ορίσει ένα κοινό θεωρητικό μοντέλο το οποίο θα χρησιμοποιείται για επίβλεψη και ανακάλυψη πόρων του πλέγματος. Αποτελείται από τρία βασικά μέρη τα οποία περιγράφουν τις ιδιότητες και την αξία των CEs και των SEs και ενοποιημένες πληροφορίες για τα δύο αυτά είδη στοιχείων:

- Σε κάθε CE και SE κάθε τοπικού πλέγματος εκτελείται το Grid Resource Information Service (GRIS). Κάθε φορά που γίνεται μια ερώτηση για κάποιο υπολογιστικό πόρο εκτελείται ένα πρόγραμμα παροχής πληροφοριών που επιστρέφει στατικές και δυναμικές πληροφορίες, οι οποίες αποθηκεύονται σε μία βάση δεδομένων και ακολούθως επιστρέφεται το αποτέλεσμα στην αρχική ερώτηση.
- Υψηλότερα στην ιεραρχία βρίσκεται το Grid Information Index Service (GIIS). Σε κάθε site που αποτελεί μέρος του πλέγματος εκτελείται ένα GIIS στο οποίο είναι καταχωρημένα όλα τα τοπικά GRISs. Όταν γίνει μία ερώτηση σε ένα GIIS, αυτό ρωτά όλα τα καταχωρημένα σε αυτό GRISs και επιστρέφει το συνολικό αποτέλεσμα.
- Στο υψηλότερο επίπεδο του Information System βρίσκεται το Berkeley Database Information Index (BDII). Το BDII περιοδικά εκτελεί ερωτήσεις στα GRISs και GIISs που υπάγονται σε αυτό. Κάθε VO έχει διαφορετικό BDII και σε αυτό υπάγονται μόνο τα sites που σχετίζονται με αυτόν.

6.2.6 Ροή Εργασιών

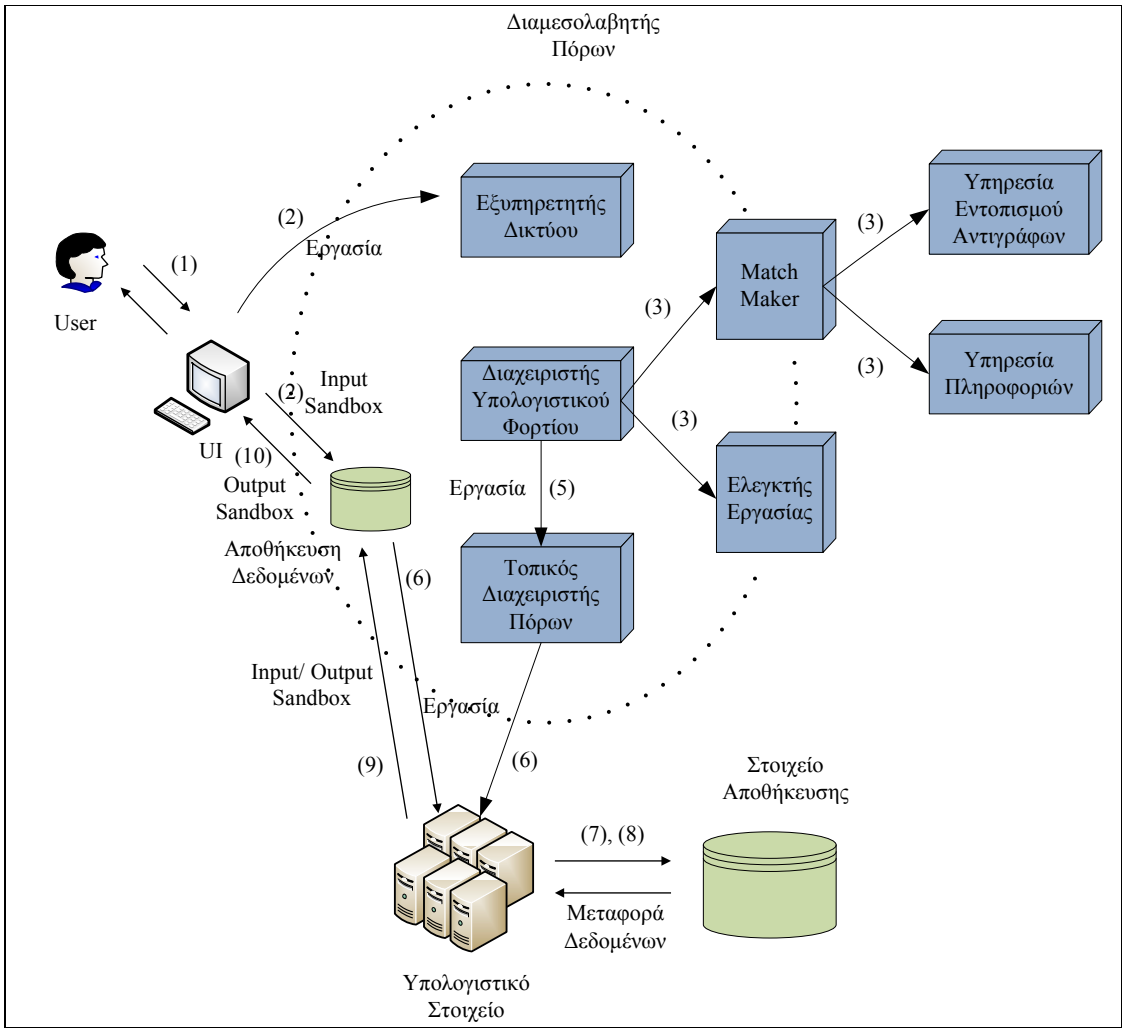
Στο Σχήμα 37 φαίνεται η διαδικασία που ακολουθείται όταν μια εργασία υποβάλλεται στο πλέγμα.

Τα διαδοχικά βήματα που πραγματοποιούνται είναι τα εξής [91]:

1. Αφού αποκτήσει ψηφιακό πιστοποιητικό από μία CA και αφού εγγραφεί σε έναν VO και αποκτήσει έναν λογαριασμό σε ένα UI, ο χρήστης είναι έτοιμος να χρησιμοποιήσει το πλέγμα. Συνδέεται σε ένα μηχάνημα UI και δημιουργεί ένα

πληρεξούσιο πιστοποιητικό, το οποίο τον πιστοποιεί σε κάθε ασφαλή αλληλεπίδραση με το μεσισμικό πλέγματος και έχει περιορισμένο χρόνο ζωής.

2. Ο χρήστης υποβάλλει μια εργασία από το UI σε έναν RB. Στο αρχείο περιγραφής της εργασίας καθορίζονται ένα ή περισσότερα αρχεία για μεταφορά από το UI στο WN, τα οποία αρχικά αντιγράφονται στον RB. Αυτό το σύνολο αρχείων ονομάζεται Input Sandbox. Το γεγονός καταγράφεται στον LB και η κατάσταση της εργασίας είναι SUBMITTED.
3. Το WMS ψάχνει για το καλύτερο διαθέσιμο CE για να εκτελέσει την εργασία. Αυτό γίνεται με ανάκριση του IS και συγκεκριμένα του BDII, έτσι ώστε να καθοριστεί η κατάσταση τον υπολογιστικών και αποθηκευτικών πόρων και του καταλόγου αρχείων, ώστε να βρεθούν οι τοποθεσίες των απαιτούμενων αρχείων εισόδου. Ένα ακόμη γεγονός καταγράφεται στον LB και η κατάσταση της εργασίας γίνεται WAITING.
4. Ο RB ετοιμάζει την εργασία για υποβολή, δημιουργώντας ένα wrapper script που θα μεταφερθεί μαζί με άλλες παραμέτρους στο CE που επιλέχθηκε. Το γεγονός καταγράφεται στον LB και η κατάσταση της εργασίας γίνεται READY.
5. Το CE λαμβάνει την αίτηση και στέλνει την εργασία για εκτέλεση στο τοπικό LRMS. Το γεγονός καταγράφεται στον LB και η κατάσταση της εργασίας γίνεται SCHEDULED.
6. Το LRMS χειρίζεται την εκτέλεση των εργασιών στους τοπικούς WNs. Τα αρχεία του Input Sandbox αντιγράφονται από τον RB στο διαθέσιμο WN όπου εκτελείται η εργασία. Το γεγονός καταγράφεται στο LB και η κατάσταση της εργασίας γίνεται RUNNING.
7. Κατά τη διάρκεια που η εργασία εκτελείται, αρχεία μπορούν να προσπελαστούν είτε απευθείας από ένα SE χρησιμοποιώντας συγκεκριμένα πρωτόκολλα, είτε αντιγράφοντάς τα πρώτα στο τοπικό σύστημα αρχείων με τα εργαλεία διαχείρισης δεδομένων.
8. Η εργασία μπορεί να παράγει νέα αρχεία εξόδου που μπορούν να ανεβούν στο πλέγμα και να γίνουν διαθέσιμα για άλλους χρήστες. Αυτό μπορεί να επιτευχθεί κάνοντας χρήση των εργαλείων διαχείρισης δεδομένων. Το ανέβασμα ενός αρχείου στο πλέγμα ισοδυναμεί με την αντιγραφή του σε ένα SE και την καταχώρησή του σε έναν κατάλογο αρχείων.



Σχήμα 37. Ροή εργασίας κατά την υποβολή της στο πλέγμα

9. Αν η εργασία τερματίσει χωρίς σφάλμα, τα αρχεία εξόδου που καθορίζονται από το χρήστη (Output Sandbox) μεταφέρονται πίσω στον κόμβο του RB. Ένα γεγονός καταχωρείται στο LB και η κατάσταση της εργασίας γίνεται DONE.
10. Σε αυτό το σημείο, ο χρήστης μπορεί να ανακτήσει την έξοδο της εργασίας στο UI. Το γεγονός καταγράφεται στο LB και η κατάσταση της εργασίας γίνεται CLEARED.
11. Αιτήσεις για την ανάκτηση της κατάστασης μιας εργασίας μπορούν να υποβληθούν από το UI στο LB. Επίσης, από το UI ο χρήστης μπορεί να αναζητήσει την κατάσταση των πόρων του πλέγματος μέσω του BDII.
12. Αν το site στο οποίο αποστέλλεται η εργασία δεν έχει τη δυνατότητα να δεχτεί ή να εκτελέσει την εργασία, η εργασία μπορεί αυτόματα να επανα-υποβληθεί σε κάποιο

άλλο CE που ικανοποιεί τις απαιτήσεις του χρήστη. Μετά από ένα μέγιστο αριθμό προσπαθειών, η εργασία χαρακτηρίζεται ως ματαιωμένη (ABORTED). Οι χρήστες μπορούν να ανακτήσουν το ιστορικό μιας εργασίας μέσω του LB.

6.3 Απαιτήσεις για το Περιβάλλον Προσομοίωσης

Το μεσισμικό πλέγματος παρέχει ένα σύνολο από λειτουργίες και μηχανισμούς για ασφάλεια, χρησιμοποίηση πόρων κτλ, ωστόσο, όπως περιγράφτηκε στην προηγούμενη παράγραφο, εισάγει ένα επιπλέον επίπεδο πολυπλοκότητας για το μέσο χρήστη. Για αυτό το λόγο, κατά την ανάπτυξη εφαρμογών πλέγματος θεωρείται απαραίτητη η παροχή διεπαφών χρήστη που αποκρύπτουν την πολυπλοκότητα αλληλεπίδρασης με το μεσισμικό πλέγματος. Μια προσέγγιση που ακολουθείται συχνά είναι η ανάπτυξη πυλών ιστού (web portals). Μια τέτοια πύλη αποτελείται από ένα σύνολο σελίδων HTML, οι οποίες συλλέγουν την πληροφορία που παρέχει ο χρήστης και, στην περίπτωση των εφαρμογών πλέγματος, την προωθούν κατάλληλα στο μεσισμικό. Μέσω των σελίδων της πύλης ο χρήστης μπορεί να ελέγχει την εξέλιξη της εφαρμογής, καθώς και να λαμβάνει τα αποτελέσματα που προκύπτουν μετά το τέλος της. Με αυτόν τον τρόπο, χρησιμοποιείται σαν διεπαφή με το χρήστη το διαδίκτυο, μια τεχνολογία με μεγάλο βαθμό διείσδυσης, η οποία είναι γνώριμη σε μεγάλο αριθμό χρηστών. Ένα σημαντικό πλεονέκτημα της συγκεκριμένης προσέγγισης είναι η δυνατότητα για ανάπτυξη περιβαλλόντων συνεργασίας (collaboration environments). Ένα portal παρέχει ένα ενιαίο σημείο πρόσβασης, μέσω του οποίου μπορούν ένα σύνολο από χρήστες διαφορετικών οργανισμών να έχουν πρόσβαση σε πόρους, καθώς και να αναλύουν και να οπτικοποιούν τα αποτελέσματα ταυτόχρονα και με συνεργατικό τρόπο.

Μια τέτοια πύλη πλέγματος πρέπει να πληροί ορισμένες απαιτήσεις:

- λειτουργικότητα – ευκολία χρήσης: η πύλη θα αποτελέσει τη διεπαφή του χρήστη τόσο με την εφαρμογή όσο και με τους υπόλοιπους χρήστες που θα χρησιμοποιήσουν την εφαρμογή. Το σημαντικότερο χαρακτηριστικό μιας πύλης πλέγματος είναι η παροχή στο χρήστη ενός φιλικού και όσο το δυνατόν απλούστερου περιβάλλοντος αλληλεπίδρασης με το πλέγμα, το οποίο ταυτόχρονα θα μπορεί να παρέχει την απαραίτητη λειτουργικότητα για την επίτευξη των απαιτήσεων που καλείται να καλύψει.

- ασφάλεια: η σχεδίαση της πύλης πρέπει να είναι τέτοια ώστε να διασφαλίζονται τα προσωπικά στοιχεία του χρήστη. Από τη στιγμή μάλιστα που το θέμα της ασφάλειας αποτελεί πρωταρχικό μέλημα σε ένα πλέγμα, είναι εμφανές ότι και η πύλη πρέπει να παρέχει μηχανισμούς για ασφάλεια. Οι μηχανισμοί αυτοί δεν πρέπει να παρακάμπτουν την ήδη υπάρχουσα υποδομή ασφάλειας του πλέγματος, αλλά να χρησιμοποιούν τη σχετική λειτουργικότητα που παρέχει το μεσισμικό.
- ευκολία συντήρησης και επέκτασης: για να έχει η πύλη αυτά τα χαρακτηριστικά, θεωρείται απαραίτητη η σχεδίασή της με βάση επιμέρους δομικά συστατικά, τα οποία θα λειτουργούν ανεξάρτητα μεταξύ τους. Με αυτόν τον τρόπο γίνεται εφικτή η επέκταση ή αλλαγή κάποιου από αυτά χωρίς να απαιτούνται επιπλέον τροποποιήσεις στην υπόλοιπη εφαρμογή. Για παράδειγμα, μια πιθανή αλλαγή στο μεσισμικό του πλέγματος που αφορά την υποβολή εργασιών θα πρέπει να μπορεί να αντιμετωπιστεί ανανεώνοντας μόνο το αντίστοιχο συστατικό της πύλης για την υποβολή εργασιών.

6.4 Χρησιμοποιηθείσες Τεχνολογίες

Με βάση τις απαιτήσεις που παρουσιάστηκαν στην προηγούμενη παράγραφο, έγινε κατάλληλη επιλογή των τεχνολογιών και συστατικών λογισμικού που χρησιμοποιήθηκαν για το σχεδιασμό και την ανάπτυξη της πύλης πλέγματος. Καταρχάς, η σχεδίαση των συστατικών της πύλης που σχετίζονται με την αλληλεπίδραση με το χρήστη έγινε με χρήση των προδιαγραφών των portlets JSR-168 [98]. Για την υλοποίηση των συστατικών αυτών χρησιμοποιήθηκε το πλαίσιο Gridsphere [99], το οποίο παρέχει προκατασκευασμένες δομικές μονάδες λογισμικού και άλλους μηχανισμούς που οι προγραμματιστές μπορούν να χρησιμοποιήσουν για την υλοποίηση πυλών με βάση τις προδιαγραφές των portlets. Τέλος, η παροχή ασφάλειας στην πύλη υποστηρίχθηκε από το σύστημα MyProxy [100]. Στη συνέχεια περιγράφονται αναλυτικότερα οι τεχνολογίες αυτές.

6.4.1 JSR-168 Portlets

Όπως ορίζεται στις προδιαγραφές JSR-168 [98], ένα portlet είναι ένα συστατικό ιστού που στηρίζεται στην Java, το οποίο επεξεργάζεται αιτήσεις και παράγει δυναμικό περιεχόμενο και τη διαχείριση του οποίου αναλαμβάνει ένας portlet container. Τα portlets μπορούν να χρησιμοποιηθούν από τις πύλες ιστού ως pluggable συστατικά διεπαφής

χρήστη, τα οποία παρέχουν το στρώμα παρουσίασης σε μια εφαρμογή. Το περιεχόμενο που παράγει ένα portlet ονομάζεται τεμάχιο (fragment). Ένα τεμάχιο είναι ένα τμήμα markup κώδικα (π.χ. HTML, XHTML, WML), το οποίο τηρεί κάποιους κανόνες και το οποίο μπορεί να συναθροιστεί με άλλα τεμάχια για τη δημιουργία ενός ολοκληρωμένου εγγράφου. Το περιεχόμενο ενός portlet συναθροίζεται με το περιεχόμενο άλλων portlets για τη δημιουργία μιας σελίδας μιας πύλης ιστού.

Οι πελάτες διαδικτύου αλληλεπιδρούν με τα portlets μέσω του υπολογιστικού παραδείγματος της αιτήσεως/αποκρίσεως (request/response), το οποίο υλοποιείται από την πύλη ιστού. Συνήθως, ο χρήστης αλληλεπιδρά με το περιεχόμενο που παράγεται από τα portlets, για παράδειγμα ακολουθώντας συνδέσμους ή υποβάλλοντες φόρμες, πράγμα που έχει ως αποτέλεσμα τη δημιουργία ειδικών ενεργειών που λαμβάνονται από την πύλη και προωθούνται κατάλληλα στα portlets που διαχειρίζονται την αλληλεπίδραση με το χρήστη. Το περιεχόμενο που παράγεται από ένα portlet έχει τη δυνατότητα να διαφέρει μεταξύ διαφορετικών χρηστών με βάση το configuration που έχει κάνει ο χρήστης στο portlet.

Οι προδιαγραφές των portlets έχουν κάποια κοινά σημεία με τις προδιαγραφές των servlets:

- Ένα portlet είναι ένα συστατικό ιστού που στηρίζεται στην Java.
- Ένα portlet παράγει δυναμικό περιεχόμενο.
- Η διαχείριση της εκτέλεσης των portlets, καθώς και του κύκλου ζωής τους γίνεται από έναν container.
- Η αλληλεπίδραση με τον πελάτη γίνεται μέσω του παραδείγματος των αιτήσεων/ αποκρίσεων.

Κάποια από τα σημαντικότερα επιπλέον χαρακτηριστικά που παρέχει ένα portlet και τα οποία δεν συναντούνται στην τεχνολογία των servlets είναι τα ακόλουθα:

- Ένα portlet παράγει μόνο τεμάχια markup και όχι ολόκληρα έγγραφα – η συγκέντρωση των markup τεμαχίων σε μια ολοκληρωμένη σελίδα γίνεται από την πύλη.
- Ένα portlet δεν αντιστοιχεί σε ένα URL.
- Για την αλληλεπίδραση των πελατών με τα portlets απαιτείται η ύπαρξη κάποιου ειδικού συστήματος στην πύλη.
- Τα portlets παρέχουν προκαθορισμένες καταστάσεις που υποδεικνύουν τη λειτουργία που επιτελεί ένα porlet μια δεδομένη χρονική στιγμή και τις λεπτομέρειες εμφάνισης στη διεπαφή χρήστη.

- Τα portlets παρέχουν μηχανισμούς για τη διαχείριση προφίλ χρηστών.

Όπως αναφέρθηκε, η διαχείριση του κύκλου ζωής ενός portlet γίνεται από τον portlet container. Ένας portlet container ουσιαστικά εκτελεί τα portlets, παρέχοντας σε αυτά το απαραίτητο περιβάλλον εκτέλεσης. Επιπλέον, παρέχει μέσα αποθήκευσης για τις προτιμήσεις των χρηστών για τα portlets. Ένας portlet container λαμβάνει αιτήσεις από την πύλη για να εκτελέσει αιτήσεις στα portlets που φιλοξενεί. Η συλλογή του περιεχομένου που παράγουν τα portlets γίνεται από την ίδια την πύλη. Ένας portlet container αποτελεί επέκταση ενός servlet container και μπορεί να υλοποιηθεί είτε πάνω σε έναν προϋπάρχοντα servlet container, είτε ως ανεξάρτητη οντότητα η οποία όμως πρέπει απαραίτητα να υποστηρίζει τις προδιαγραφές των servlets.

Μια από τις σημαντικότερες ιδιότητες των portlets είναι ότι μπορούν να λειτουργούν σαν παράθυρα μέσα σε μια σελίδα. Αυτό παρέχει τη δυνατότητα αφενός στον διαχειριστή του portal να καθορίζει εύκολα την εμφάνιση της πύλης και τις λειτουργίες που αυτή παρέχει σε μια δεδομένη χρονική στιγμή και αφετέρου στο χρήστη να παραμετροποιήσει τον τρόπο εμφάνισης της κάθε σελίδας, να μεγιστοποιήσει ή να ελαχιστοποιήσει το μέγεθος ενός portlet κλπ.

6.4.2 GridSphere

Το GridSphere [99] είναι ένα application framework για την ανάπτυξη διαδικτυακών πυλών πλέγματος με βάση τις προδιαγραφές των portlets, το οποίο αναπτύχθηκε στα πλαίσια του προγράμματος GridLab [101]. Ως application framework νοείται γενικά μία ημι-ολοκληρωμένη εφαρμογή που μπορεί να αποτελέσει τη βάση για την ανάπτυξη ειδικότερων υλοποιήσεων. Ένα framework παρέχει ένα σύνολο προκατασκευασμένων δομικών μονάδων λογισμικού που μπορούν να χρησιμοποιηθούν αυτούσια ή ακόμη και να προσαρμοστούν ή να επεκταθούν για τις ανάγκες μιας συγκεκριμένης υλοποίησης. Εν προκειμένω, το GridSphere προσφέρει μια υλοποίηση των προτύπων JSR 168 και προσφέρει ένα σύνολο portlets πυρήνα που παρέχουν τη βασική λειτουργικότητα που απαιτείται για όλες τις πύλες ιστού. Υποστηρίζει την ανάπτυξη επαναχρησιμοποιήσιμων portlets και portlet services, τα οποία μπορούν να εκτελεστούν και να διαχειριστούν μέσα σε έναν portlet container που το ίδιο παρέχει.

Κάποια από τα σημαντικότερα χαρακτηριστικά που παρέχει η τελευταία έκδοση (3.0.8) του GridSphere είναι τα ακόλουθα:

- 100% συμβατότητα με τις προδιαγραφές JSR 168 – Portlet

- tag library για την ανάπτυξη διεπαφών χρήστη
- εμφάνιση της πύλης μέσω ενός XML descriptor, ο οποίος διευκολύνει την προσαρμογή της εμφάνισης στις ανάγκες της κάθε εφαρμογής
- ένα σύνολο από προκατασκευασμένα portlets που αφορούν τις βασικές λειτουργίες της πύλης (π.χ. portlets για login χρηστών και portlets διαχείρισης)
- πρόκειται για open-source framework

6.4.3 MyProxy

Το MyProxy [100] είναι ένα online σύστημα αποθήκευσης των πιστοποιητικών των χρηστών του πλέγματος, το οποίο επιτρέπει σε πύλες ιστού να αλληλεπιδρούν με τους πόρους του πλέγματος με πρότυπους και ασφαλείς μηχανισμούς. Η κλασική υποδομή ασφάλειας σε ένα πλέγμα παρουσιάζει τους ακόλουθους περιορισμούς όταν βρίσκει εφαρμογή σε εφαρμογές πυλών ιστού:

- Λόγω του ότι τα πιστοποιητικά του χρήστη τυπικά αποθηκεύονται ως αρχεία σε ένα σύστημα αρχείων και το ιδιωτικό κλειδί πρέπει να διατηρείται μυστικό, ο χρήστης πρέπει να διαθέτει ασφαλή πρόσβαση στο σύστημα αρχείων. Αυτό σημαίνει ότι ο χρήστης πρέπει να εργάζεται απευθείας στο μηχάνημα που βρίσκεται το σύστημα αρχείων, ή να είναι logged in σε αυτό με κάποιο ασφαλή τρόπο (π.χ. μέσω σύνδεσης SSH)
- Ένας δεύτερος περιορισμός είναι ότι κάποιες εφαρμογές που προορίζονται για χρήση σε πλέγμα δεν έχουν την ικανότητα να χρησιμοποιήσουν τους μηχανισμούς ασφάλειας του πλέγματος, μεταξύ των οποίων και την εκχώρηση πληρεξούσιων πιστοποιητικών (proxy credential delegation). Για παράδειγμα, ένας περιηγητής ιστού μπορεί να χρησιμοποιήσει πιστοποιητικά πλέγματος για πιστοποίηση, αλλά δεν έχει τη δυνατότητα να πραγματοποιήσει εκχώρηση πιστοποιητικών. Αυτό σημαίνει ότι αν και ο χρήστης μπορεί να πιστοποιηθεί στην πύλη, δεν μπορεί να αναθέσει σε αυτήν να λειτουργήσει εκ μέρους του, πράγμα που περιορίζει τις δυνατότητές της.
- Η τυπική προσέγγιση για πύλες πλέγματος υποδεικνύει ότι για να μπορεί μια πύλη να λειτουργήσει εκ μέρους του χρήστη στο πλέγμα, πρέπει να κατέχει τα μακροπρόθεσμα πιστοποιητικά του. Αυτό σημαίνει ότι ο χρήστης θα πρέπει να έχει αποθηκεύσει σε κάθε πύλη που χρησιμοποιεί τα πιστοποιητικά αυτά, πράγμα που εγκυμονεί κινδύνους για δύο λόγους. Καταρχάς, η ύπαρξη πολλών αντιγράφων των

πιστοποιητικών σημαίνει ότι αυτά είναι περισσότερο εκτεθειμένα και συνεπώς ο κίνδυνος να κλαπούν αυξάνεται. Επιπλέον, από τη στιγμή που η πύλη απαιτείται να διαχειρίζεται με ασφάλεια τα πιστοποιητικά του χρήστη, αυξάνονται και οι απαιτήσεις ασφάλειας για τη λειτουργία της ίδιας της πύλης.

Το σύστημα MyProxy επιτυγχάνει την άρση των παραπάνω περιορισμών, επιτρέποντας στους χρήστες πρόσβαση στα πιστοποιητικά τους ακόμη και από τοποθεσίες που δεν έχουν ασφαλή πρόσβαση στα μακροπρόθεσμα πιστοποιητικά τους. Επιπλέον, απαλείφει την ανάγκη ύπαρξης πιστοποιητικών στην πύλη όταν αυτά δεν είναι απαραίτητα, μειώνοντας τον κίνδυνο κλοπής τους. Τέλος, δίνει στον χρήστη περισσότερο έλεγχο στον τρόπο χρήσης των μακροπρόθεσμων και των πληρεξούσιων πιστοποιητικών.

Ο τρόπος λειτουργίας μιας πύλης που στηρίζεται στο σύστημα MyProxy για την ασφάλεια στο πλέγμα φαίνεται στο Σχήμα 38. Αναλυτικά, πραγματοποιούνται τα ακόλουθα:

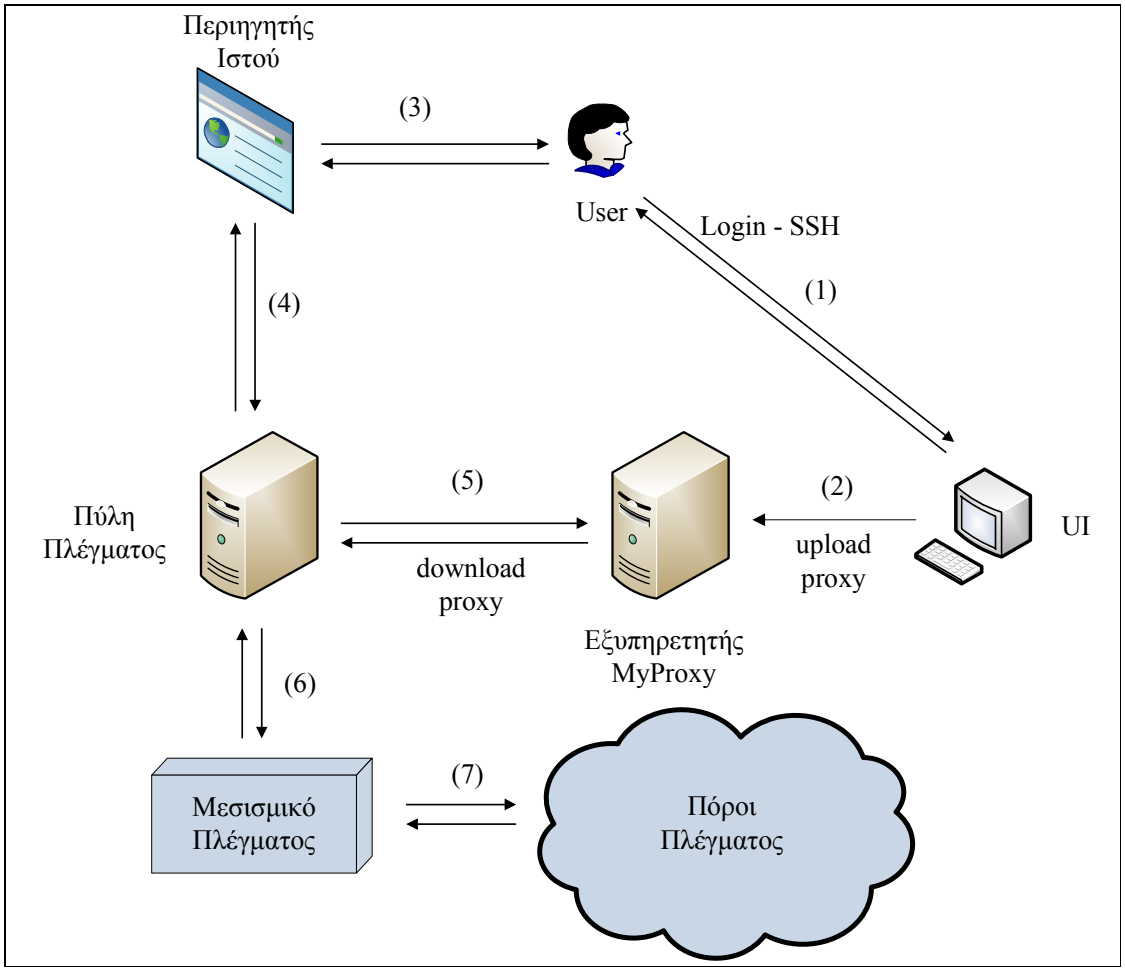
1. Αρχικά ο χρήστης κάνει login με κάποιο ασφαλή τρόπο (π.χ. SSH) στον υπολογιστή που έχει εγκατεστημένη τη διεπαφή χρήστη του μεσισμικού πλέγματος (τυπικά κάποιο μηχάνημα Linux), στον οποίο βρίσκονται τα μακροπρόθεσμα πιστοποιητικά του χρήστη. Χρησιμοποιώντας την ασφαλή σύνδεση, ο χρήστης μπορεί να παρέχει με ασφάλεια τους κωδικούς του και να δημιουργήσει πληρεξούσια πιστοποιητικά.
2. Ο χρήστης έχει τη δυνατότητα να ανεβάσει πληρεξούσια πιστοποιητικά στον εξυπηρετητή MyProxy, χρησιμοποιώντας διαφορετικούς κωδικούς και ορίζοντας κάποια επιπλέον χαρακτηριστικά για αυτά (π.χ. χρόνος ζωής).

3-7. Όταν ο χρήστης επιθυμεί να χρησιμοποιήσει τις υπηρεσίες της πύλης, χρησιμοποιεί κάποιον περιηγητή ιστού και κάνει login στον εξυπηρετητή που φιλοξενεί την πύλη. Χρησιμοποιώντας τους κωδικούς που έδωσε ο χρήστης κατά τη δημιουργία των πληρεξούσιων πιστοποιητικών στον εξυπηρετητή MyProxy, η πύλη μπορεί να κατεβάσει τα πληρεξούσια πιστοποιητικά και στη συνέχεια να τα χρησιμοποιήσει για την όποια αλληλεπίδραση απαιτείται με το μεσισμικό και τους πόρους του πλέγματος.

6.5 Αρχιτεκτονική του Περιβάλλοντος Προσομοίωσης - Υλοποίηση

Η συνολική αρχιτεκτονική του περιβάλλοντος προσομοίωσης φαίνεται στο Σχήμα 39.

Όπως φαίνεται στο σχήμα, το περιβάλλον προσομοίωσης συντίθεται από κάποιες επιμέρους ομάδες συστατικών: (i) τα portlets που αποτελούν τη διεπαφή χρήστη του περιβάλλοντος προσομοίωσης, (ii) τις υπηρεσίες πύλης που υλοποιούν τη λογική της εφαρμογής και (iii) κάποια βοηθητικά συστατικά που παρέχουν επιπλέον λειτουργικότητα στις υπηρεσίες πύλης (βάση δεδομένων και repository για τους κώδικες των χρηστών).

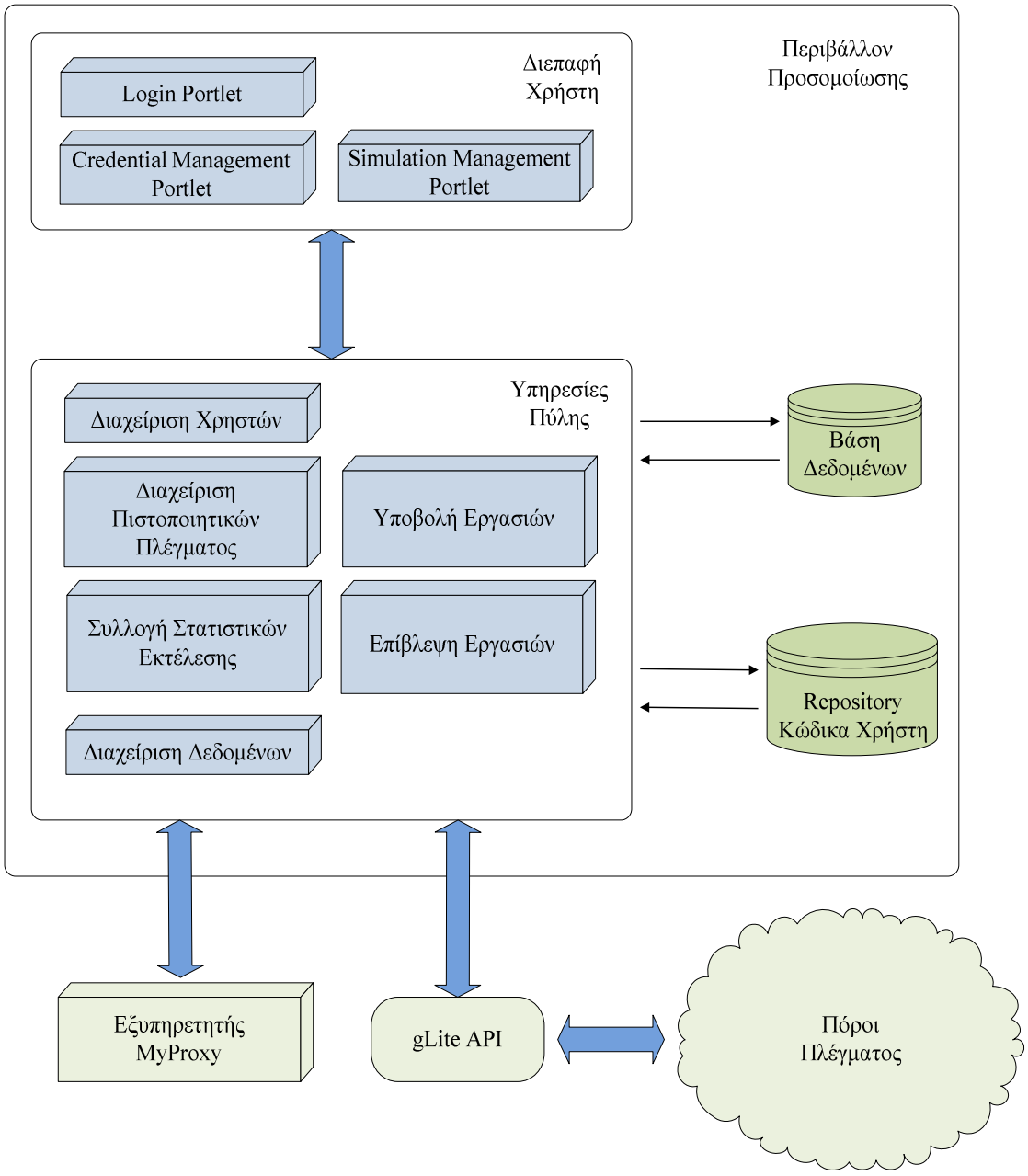


Σχήμα 38. Λειτουργία του συστήματος MyProxy σε εφαρμογή πύλης πλέγματος

Για την υλοποίηση και την υποστήριξη εκτέλεσης των portlets έγινε χρήση της έκδοσης 3.0.8 του GridSphere framework. Ως εξυπηρετητής ιστού της εφαρμογής χρησιμοποιήθηκε ο Apache Tomcat v.5.5.25. Ο σταθμός εργασίας που φιλοξενεί την πύλη είναι ένα linux box, το οποίο χρησιμοποιεί το λειτουργικό σύστημα Scientific Linux 3.0.7. και στο οποίο είναι εγκατεστημένο το λογισμικό διεπαφής χρήστη UI-PnP [102] του μεσισμικού gLite. Η πύλη χρησιμοποιεί τον εξυπηρετητή MyProxy "myproxy.grid.auth.gr" που διαχειρίζεται η

HellasGrid Task Force [103]. Για την επικοινωνία των υπηρεσιών της πύλης με το σύστημα MyProxy έγινε χρήση του Java CoG Kit [104], ενώ η επικοινωνία με το gLite πραγματοποιήθηκε μέσω του αντίστοιχου Java API που παρέχει το πρόγραμμα EGEE.

Στη συνέχεια περιγράφονται αναλυτικότερα τα συστατικά που απαρτίζουν το περιβάλλον προσομοίωσης.



Σχήμα 39. Συνολική Αρχιτεκτονική Εφαρμογής

6.5.1 Portlets

Η διεπαφή χρήστη του περιβάλλοντος προσομοίωσης αποτελείται από τρία portlets:

- Login Portlet: το portlet αυτό είναι προκατασκευασμένο και παρέχεται από το GridSphere framework. Χρησιμοποιείται για την πιστοποίηση των χρηστών κατά τα συνηθισμένα.
- Credential Management Portlet: αναπτύχθηκε ειδικά για τους σκοπούς της συγκεκριμένης εφαρμογής. Χρησιμοποιείται για τη διαχείριση των πιστοποιητικών πλέγματος των χρηστών, κάνοντας χρήση του συστήματος MyProxy που περιγράφτηκε σε προηγούμενη παράγραφο.
- Simulation Management Portlet: το συγκεκριμένο portlet επίσης αναπτύχθηκε ειδικά για τις ανάγκες της εφαρμογής. Μέσω αυτού γίνεται το setup της προσομοίωσης που επιθυμεί να εκτελέσει ο χρήστης στο πλέγμα, η υποβολή των εργασιών, καθώς και η επίβλεψή τους και η ανάκτηση των τελικών αποτελεσμάτων.

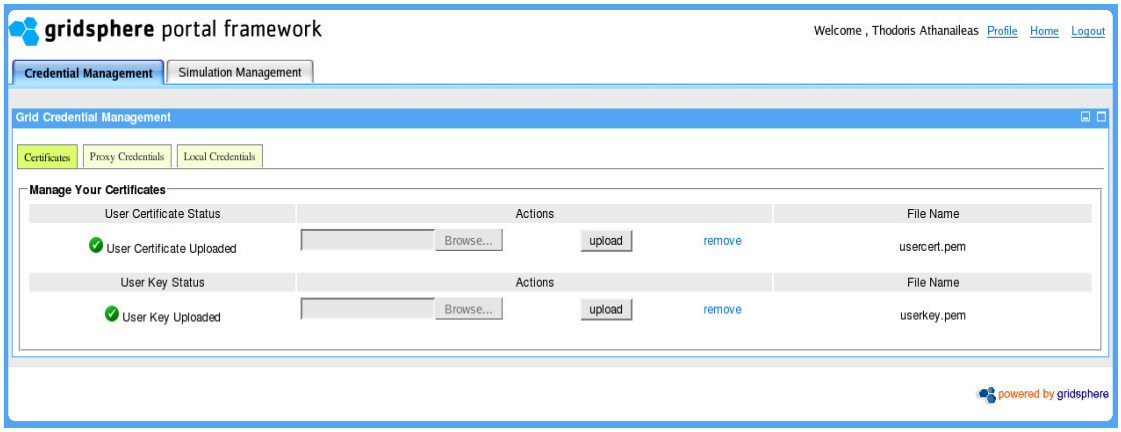
Στη συνέχεια περιγράφονται αναλυτικότερα τα δύο portlets που αναπτύχθηκαν ειδικά για το περιβάλλον προσομοίωσης.

6.5.1.1 Credential Management Portlet

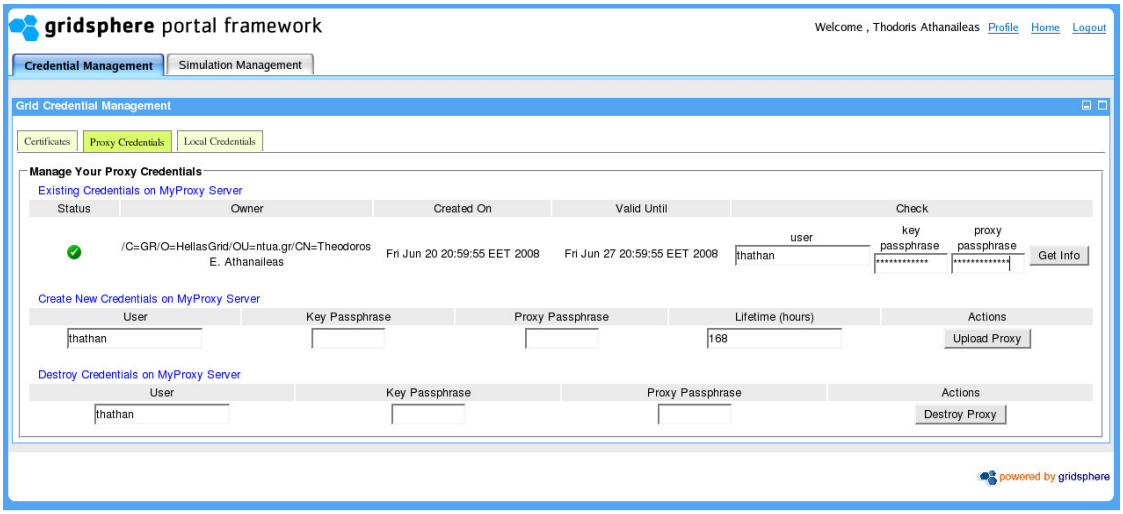
Το συγκεκριμένο portlet προσφέρει στο χρήστη μια διεπαφή για εκτέλεση ενεργειών που σχετίζονται με τη διαχείριση των πιστοποιητικών πλέγματος. Οι δυνατότητες που παρέχει είναι οι εξής:

- ανάκτηση πληροφοριών από τον εξυπηρετητή MyProxy σχετικά με την κατάσταση των πληρεξούσιων πιστοποιητικών που πιθανώς υπάρχουν διαθέσιμα σε αυτόν
- ανάκτηση πληρεξούσιων πιστοποιητικών από τον εξυπηρετητή MyProxy
- διαγραφή πληρεξούσιων πιστοποιητικών από τον εξυπηρετητή MyProxy
- ανέβασμα στην πύλη των μακροπρόθεσμων πιστοποιητικών του χρήστη
- δημιουργία νέων πληρεξούσιων πιστοποιητικών στον εξυπηρετητή MyProxy

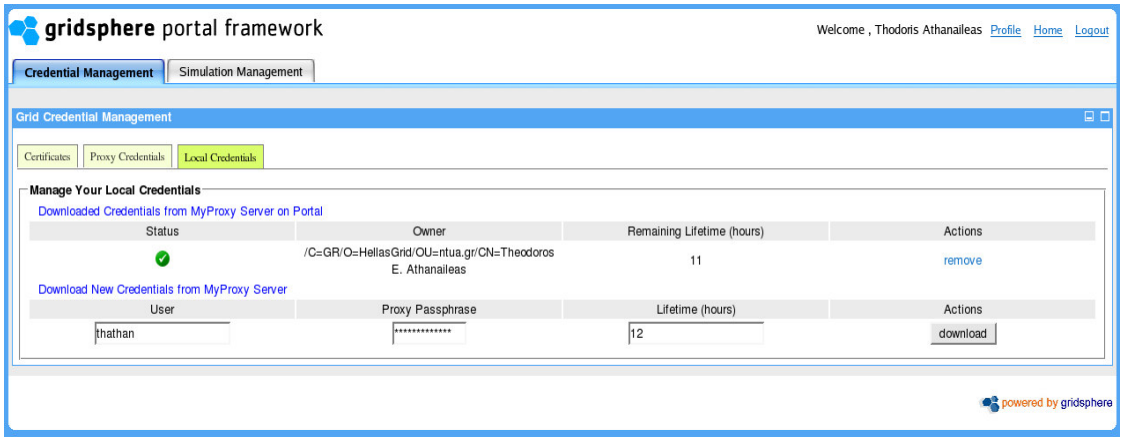
Ο ενδεδειγμένος τρόπος λειτουργίας του συστήματος υπαγορεύει την χρήση του MyProxy μέσω της πύλης μόνο για ανάκτηση πληρεξούσιων πιστοποιητικών που έχει δημιουργήσει ο χρήστης προηγουμένως με κάποιο ασφαλή τρόπο (π.χ. μέσω SSH session σε κάποιο UI του πλέγματος), όπως περιγράφτηκε σε προηγούμενη παράγραφο. Παρόλα αυτά, επιλέχθηκε να παρέχεται και η επιπλέον λειτουργικότητα που αναφέρεται παραπάνω.



Σχήμα 40. Σελίδα διαχείρισης μακροπρόθεσμων πιστοποιητικών στην πύλη



Σχήμα 41. Σελίδα διαχείρισης πληρεξούσιων πιστοποιητικών στον εξυπηρετητή MyProxy



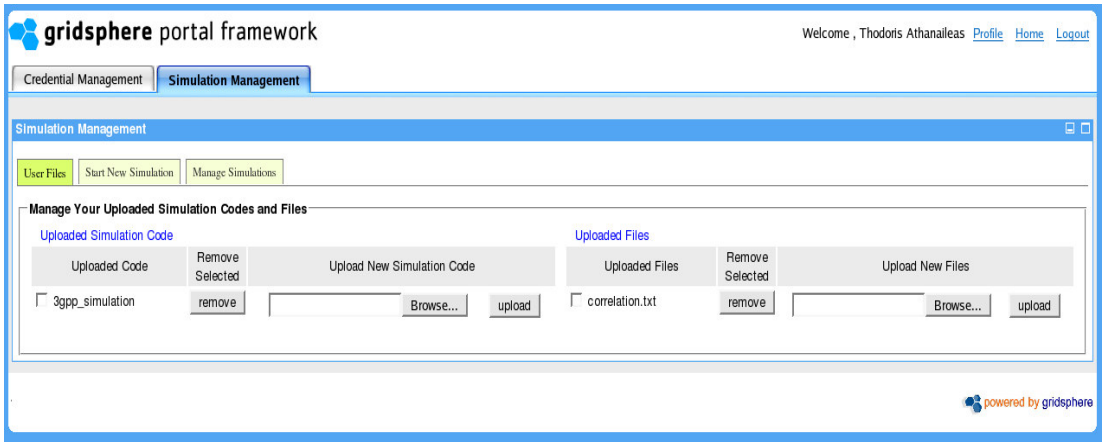
Σχήμα 42. Σελίδα διαχείρισης πληρεξούσιων πιστοποιητικών στην πύλη

Στα Σχήματα 40-42 φαίνονται οι σελίδες που παρέχονται για τη διαχείριση των διαφόρων τύπων πιστοποιητικών του χρήστη.

6.5.1.2 Simulation Management Portlet

Το Simulation Management Portlet παρέχει στο χρήστη τη διεπαφή για την εκτέλεση των ακόλουθων λειτουργιών:

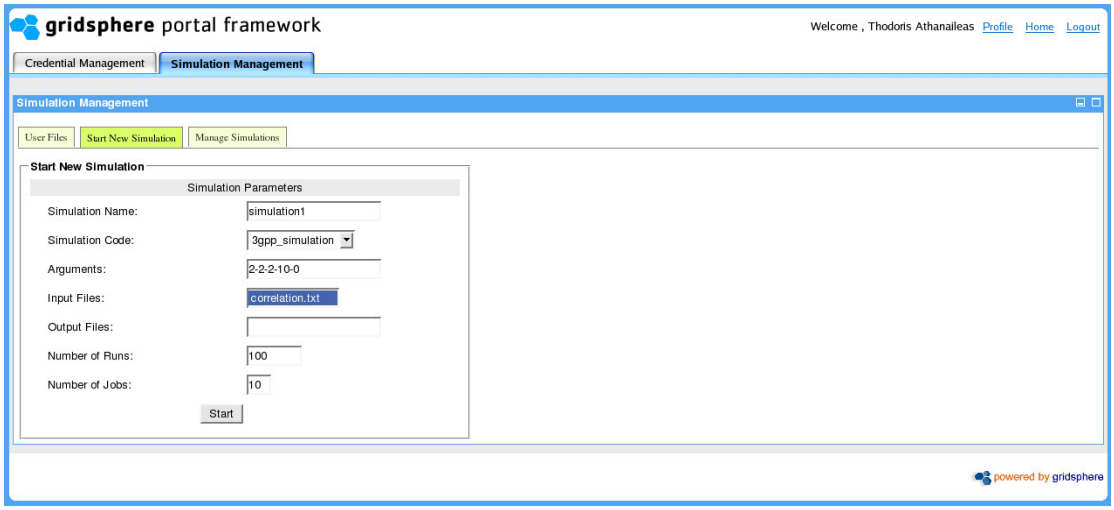
- διαχείριση των κωδίκων προσομοίωσης που επιθυμεί να εκτελέσει στο πλέγμα και των αρχείων που απαιτούνται για τις προσομοιώσεις
- δημιουργία και υποβολή προσομοιώσεων στο πλέγμα
- επίβλεψη της εξέλιξης των προσομοιώσεων και των επιμέρους εργασιών που τις απαρτίζουν
- ανάκτηση των τελικών αποτελεσμάτων



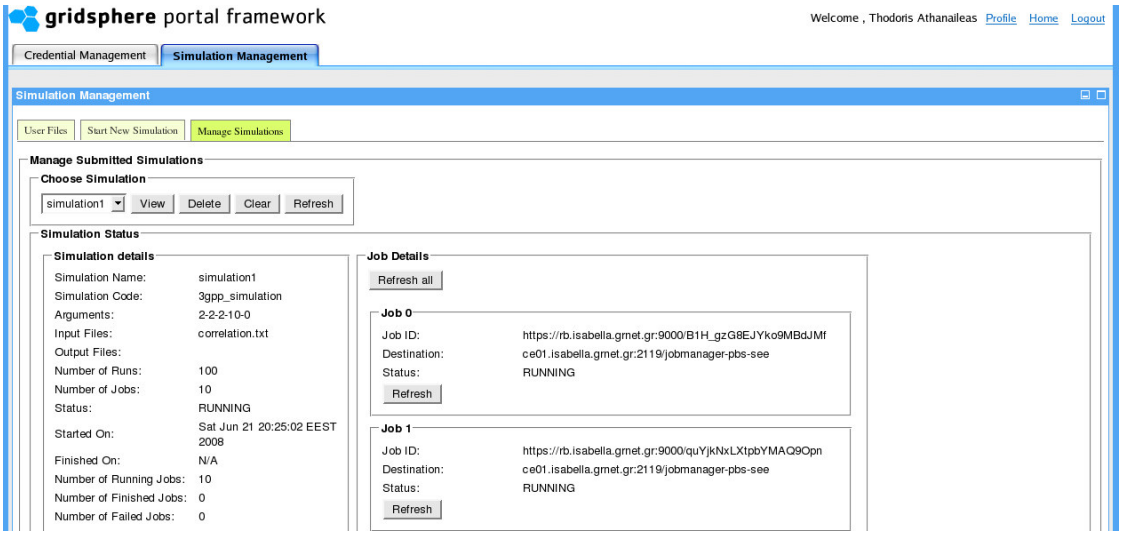
Σχήμα 43. Σελίδα διαχείρισης κωδίκων προσομοίωσης και αρχείων εισόδου στην πύλη

Στο Σχήμα 43 φαίνεται η φόρμα διαχείρισης των αρχείων του χρήστη. Ο χρήστης έχει τη δυνατότητα να ανεβάσει στην πύλη νέους κώδικες προσομοίωσης σε ένα συμπιεσμένο αρχείο, καθώς και οποιονδήποτε αρχείων μπορεί να απαιτούνται για μια προσομοίωση.

Στο Σχήμα 44 φαίνεται η φόρμα υποβολής νέας προσομοίωσης. Ο χρήστης επιλέγει κώδικα προσομοίωσης και αρχεία εισόδου και καθορίζει τα ορίσματα γραμμής εντολών της προσομοίωσης, καθώς και τον αριθμό των επιμέρους προσομοιώσεων Monte Carlo που θα εκτελεστούν και τον αριθμό των εργασιών που θα υποβληθούν για εκτέλεση στο πλέγμα.



Σχήμα 44. Σελίδα υποβολής νέας προσομοίωσης



Σχήμα 45. Σελίδα επίβλεψης και διαχείρισης υποβληθείσας προσομοίωσης

Η διαχείριση και επίβλεψη μιας υποβληθείσας προσομοίωσης γίνεται μέσω της φόρμας που φαίνεται στο Σχήμα 45. Ο χρήστης μπορεί να επιλέξει την προσομοίωση που επιθυμεί και να ανακτήσει την κατάσταση αυτής και των επιμέρους εργασιών, να την διαγράψει κτλ.

6.5.2 Υπηρεσίες Πύλης

Η σημαντικότερη ομάδα συστατικών της εφαρμογής είναι οι υπηρεσίες πύλης. Οι υπηρεσίες πύλης υλοποιούν τη λογική και τη λειτουργικότητα της εφαρμογής και συνεργάζονται με τα portlets της διεπαφής χρήστη για λήψη πληροφορίας εισόδου από το χρήστη και για παροχή πληροφοριών εκτέλεσης σε αυτόν. Επίσης, συνεργάζονται με τα

υπόλοιπα υποστηρικτικά συστατικά της πύλης (βάση δεδομένων και repository κώδικα χρήστη), καθώς και με το μεσισμικό πλέγματος για την πραγματοποίηση όλων των λειτουργιών που σχετίζονται με χρήση των πόρων. Οι υπηρεσίες πύλης ουσιαστικά είναι αυτές που εγκαθιδρύουν τη σύνδεση του επιπέδου παρουσίασης της εφαρμογής με το μεσισμικό του πλέγματος και τις υπηρεσίες βάσης δεδομένων και repository κώδικα. Οι υπηρεσίες πύλης είναι οι εξής:

- Υπηρεσίες Διαχείρισης Χρηστών: παρέχονται από το GridSphere framework και αναλαμβάνουν όλες εκείνες τις λειτουργίες που σχετίζονται με τους χρήστες (π.χ. πρόσθεση νέων χρηστών, διαγραφή χρηστών, πιστοποίηση χρηστών)
- Υπηρεσίες Διαχείρισης Πιστοποιητικών Πλέγματος: οι υπηρεσίες αυτές επικοινωνούν μέσω του Java CoG Kit API με τον εξυπηρετητή MyProxy και αναλαμβάνουν τις λειτουργίες που παρέχονται μέσω του Credential Management Portlet για τον κάθε χρήστη (π.χ. ανάκτηση πληρεξούσιου πιστοποιητικού από τον εξυπηρετητή).
- Υπηρεσίες Υποβολής Εργασιών: οι υπηρεσίες υποβολής εργασιών λαμβάνουν τις πληροφορίες για το configuration της προσομοίωσης που επιθυμεί να εκτελέσει ο χρήστης και στη συνέχεια αναλόγως δημιουργούν το κατάλληλο αρχείο περιγραφής για την υποβολή των εργασιών. Δημιουργούν το πακέτο συστατικών (κώδικας προσομοίωσης, αρχεία εισόδου κτλ) που απαιτούνται για την εκτέλεση της εργασίας και στη συνέχεια επικοινωνούν με το μεσισμικό πλέγματος μέσω του gLite Java API για την υποβολή. Σε όλες τις εργασίες που υποβάλλονται στο πλέγμα ανατίθεται ο ίδιος αριθμός επιμέρους προσομοιώσεων. Μια πρώτη προσέγγιση για την αποσύνθεση της προσομοίωσης είναι η ανάθεση μιας μοναδικής επιμέρους προσομοίωσης σε κάθε εργασία που υποβάλλεται στο πλέγμα. Αυτή η προσέγγιση μπορεί να αποδειχθεί μη αποδοτική όσον αφορά την απόδοση, καθώς ο μεγάλος αριθμός υποβαλλόμενων εργασιών συνεπάγεται αυξημένη επιβάρυνση λόγω των χρόνων αναμονής σε ουρές, πολλαπλή μεταφορά δεδομένων και συνολικά μεγαλύτερου χρόνου για εκκίνηση των εργασιών. Επιπλέον, η ανάθεση μιας μοναδικής επιμέρους προσομοίωσης ανά υποβαλλόμενη εργασία θα δημιουργούσε μεγάλη επιβάρυνση στην ίδια την πύλη, ειδικά αν λάβει κανείς υπόψη ότι μπορεί να χρησιμοποιείται από πολλούς χρήστες ταυτόχρονα. Σε αυτό το σημείο πρέπει να αναφερθεί ότι μια κρίσιμη απαίτηση για την παράλληλη και κατανεμημένη υλοποίηση στοχαστικών προσομοιώσεων είναι η αξιοπιστία των γεννητριών τυχαίων αριθμών (random

number generator – RNGs). Ένας αριθμός μεθόδων έχουν προταθεί για τη λήψη παράλληλων ροών τυχαίων αριθμών [105], [106]. Στην υλοποίηση του περιβάλλοντος προσομοίωσης επιλέχθηκε η μέθοδος των Ανεξάρτητων Ακολουθιών (Independent Sequences), κυρίως λόγω της απλότητας κατά την υλοποίηση. Σε κάθε εργασία που υποβάλλεται στο πλέγμα, δίνεται από τις Υπηρεσίες Υποβολής Εργασιών ένα μοναδικό αναγνωριστικό, το οποίο χρησιμοποιείται για την αρχικοποίηση της χρησιμοποιούμενης RNG.

- Υπηρεσίες Επίβλεψης Εργασιών: οι υπηρεσίες αυτές αναλαμβάνουν την λήψη πληροφοριών σχετικά με την κατάσταση της κάθε εργασίας μιας δεδομένης προσομοίωσης και την προώθησή τους στο Simulation Management Portlet, επικοινωνώντας για αυτό το σκοπό με την υπηρεσία LB του gLite. Η ανάκτηση πληροφορίας κατάστασης των εργασιών μπορεί να γίνεται είτε περιοδικά σε τακτά χρονικά διαστήματα που καθορίζει ο διαχειριστής της πύλης, είτε μετά από αίτηση του χρήστη.
- Υπηρεσίες Διαχείρισης Δεδομένων: αναλαμβάνουν τη γενική διαχείριση δεδομένων που παίρνουν μέρος στις λειτουργίες της πύλης, φροντίζοντας να γίνεται διαχωρισμός των δεδομένων που αφορούν τον κάθε χρήστη. Τα δεδομένα που διαχειρίζονται αυτές οι υπηρεσίες περιλαμβάνουν μεταξύ άλλων πιστοποιητικά χρηστών, αρχεία εισόδου προσομοιώσεων, υπολογιστικούς κώδικες και αποτελέσματα εργασιών.

6.5.3 Διαδικασία Εκτέλεσης Προσομοίωσης στο Πλέγμα

Το διάγραμμα καταστάσεων που φαίνεται στο Σχήμα 46 δείχνει τις αλληλεπιδράσεις μεταξύ του χρήστη και των συστατικών της πύλης κατά την εκτέλεση μιας προσομοίωσης στο πλέγμα. Όπως φαίνεται στο σχήμα, η αλληλεπίδραση του χρήστη με τις υπηρεσίες του πλέγματος επιτυγχάνεται μέσω των υπηρεσιών πύλης, οι οποίες χειρίζονται κατάλληλα τις αιτήσεις του χρήστη.

Η διαδικασία εκτέλεσης προσομοίωσης στο πλέγμα είναι η εξής: αφού ο χρήστης κάνει login στην πύλη, μπορεί να ανεβάσει σε αυτήν τους κώδικες προσομοίωσης που επιθυμεί να εκτελέσει, καθώς και τα οποιαδήποτε αρχεία εισόδου μπορεί να απαιτούνται για μια προσομοίωση (2). Ειδικά οι κώδικες προσομοίωσης τοποθετούνται στο code repository του κάθε χρήστη. Προτού προχωρήσει σε οποιαδήποτε αλληλεπίδραση με το πλέγμα, απαιτείται η λήψη των πληρεξούσιων πιστοποιητικών από τον εξυπηρετητή MyProxy (3). Αφού γίνει

η ανάκτηση των πιστοποιητικών από την πύλη, ο χρήστης μπορεί να προχωρήσει στην δημιουργία νέας προσομοίωσης ή στην επίβλεψη της κατάστασης προσομοιώσεων που έχουν υποβληθεί στο παρελθόν.

Για την υποβολή νέας προσομοίωσης (4), αρχικά δίνονται από το χρήστη οι παράμετροι της προσομοίωσης (κώδικας, αρχεία εισόδου, παράμετροι γραμμής εντολών κτλ). Με βάση αυτές τις παραμέτρους, καθορίζονται τα πακέτα με τα συστατικά που θα πρέπει να αποσταλούν στο πλέγμα και δημιουργούνται τα κατάλληλα αρχεία JDL των εργασιών. Ακολούθως γίνεται η υποβολή των εργασιών στο πλέγμα και οι πληροφορίες για την αναγνώριση της κάθε εργασίας αποθηκεύονται στη βάση δεδομένων. Αφού πραγματοποιηθεί η υποβολή της προσομοίωσης, ο χρήστης έχει τη δυνατότητα να επιβλέψει την εξέλιξη της εκτέλεσής της (5). Μετά την αίτηση του χρήστη, οι υπηρεσίες πύλης λαμβάνουν από τη βάση δεδομένων τα αναγνωριστικά των εργασιών που ανήκουν στη συγκεκριμένη προσομοίωση και ελέγχουν την κατάσταση της κάθε εργασίας ξεχωριστά, προωθώντας τα αποτελέσματα στο αντίστοιχο portlet. Τέλος, όταν όλες οι εργασίες της προσομοίωσης ολοκληρωθούν, ο χρήστης μπορεί να ανακτήσει τα συνολικά αποτελέσματα (6).

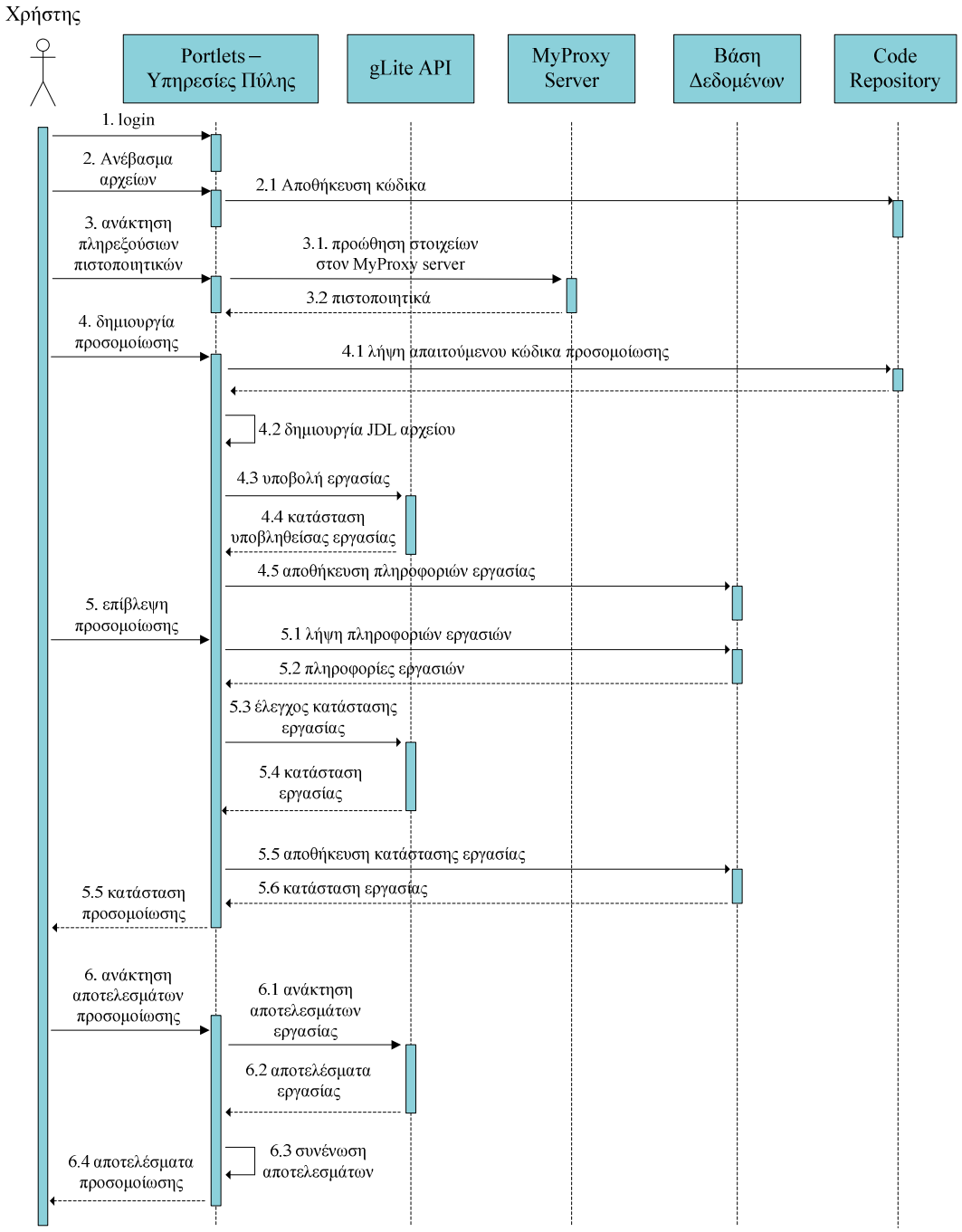
6.6 Στατιστικά Εκτέλεσης Προσομοιώσεων στο Πλέγμα

Το περιβάλλον προσομοίωσης που περιγράφτηκε στις προηγούμενες παραγράφους χρησιμοποιήθηκε για την εκτέλεση ενός μεγάλου αριθμού προσομοιώσεων συστημάτων ασύρματων επικοινωνιών. Ενδεικτικά παρουσιάζεται μια σειρά προσομοιώσεων συστήματος WCDMA.

Συγκεκριμένα, θεωρήθηκαν δύο διαφορετικές τοπολογίες δικτύου: (α) δίκτυο με τομείς όπου όλες οι κυψέλες χρησιμοποιούν συμβατικούς τομείς 120^o και (β) σταθερό πλέγμα λοβών (Fixed Grid of Beams – FGoB) στην κεντρική κυψέλη με 7 λοβούς ανά τομέα. Για κάθε τοπολογία δικτύου μέχρι τέσσερα tiers κυψελών θεωρούνται κατά τον υπολογισμό χωρητικότητας.

Για τις προσομοιώσεις χρησιμοποιήθηκε τυπική αστική περιοχή μεγέθους ~18 km^2, ενώ για τη διάδοση χρησιμοποιήθηκε πρόβλεψη διάδοσης με βάση την τεχνική ray tracing [45]. Για να μειωθεί η σημαντική πολυπλοκότητα που εισάγουν οι ray tracing τεχνικές, χρησιμοποιήθηκε μια υβριδική προσέγγιση για τη μοντελοποίηση διάδοσης: τα χαρακτηριστικά του ραδιοκαναλιού για την κεντρική κυψέλη παράγονται με τη βοήθεια της

ray tracing τεχνικής, ενώ για τις υπόλοιπες κυψέλες παράγονται με το μοντέλο Okumura Hata [107]. Σε όλες τις προσομοιώσεις ο μέγιστος βαθμός παρεμβολής (παράγοντας φορτίου – loading factor) είναι ίσος με 90%, που αντιστοιχεί σε πολύ φορτωμένο δίκτυο. Επίσης, λαμβάνονται υπόψη μόνο υπηρεσίες φωνής.



Σχήμα 46. Διάγραμμα καταστάσεων κατά την εκτέλεση προσομοίωσης στο πλέγμα

Κάθε Monte Carlo προσομοίωση αποτελείται από 1000 επιμέρους προσομοιώσεις, οι οποίες διαμοιράζονται ισομερώς σε 50 εργασίες (δηλαδή 20 επιμέρους προσομοιώσεις ανά εργασία). Οι προσομοιώσεις υποβλήθηκαν στο EGEE πλέγμα και εκτελέστηκαν στους πόρους του South Eastern Europe VO [108]. Οι παρακάτω πίνακες δείχνουν κάποια στατιστικά εκτέλεσης (σε δευτερόλεπτα) για διαφορετικούς αριθμούς tiers και διαφορετικούς παράγοντες φορτίου.

Σενάριο προσομοίωσης	Ελάχιστος Χρόνος αναμονής σε ουρά	Μέγιστος Χρόνος αναμονής σε ουρά	Μέσος Χρόνος αναμονής σε ουρά	Ελάχιστη επιβάρυνση εργασίας λόγω πλέγματος	Μέγιστη επιβάρυνση εργασίας λόγω πλέγματος	Μέση επιβάρυνση εργασίας λόγω πλέγματος
1 tier, 50% loading	81	537	166	102	639	232
1 tier, 90% loading	108	583	171	133	614	224
2 tiers, 50% loading	108	583	171	126	341	212
2 tiers, 90% loading	80	3978	1042	107	4001	1098
3 tiers, 50% loading	82	4580	1129	120	4637	1191
3 tiers, 90% loading	110	11937	546	138	11970	597
4 tiers, 50% loading	76	2844	257	132	2865	320
4 tiers, 90% loading	76	194	160	100	297	212

Πίνακας 10. Στατιστικά Εκτέλεσης στο πλέγμα (1)

Ο Πίνακας 10 δείχνει τους χρόνους αναμονής σε ουρές και τη συνολική επιβάρυνση λόγω πλέγματος σε κάθε εργασία. Η επιβάρυνση αποτελείται από το χρόνο αναμονής σε ουρές, το χρόνο μεταφοράς αρχείων εισόδου και εξόδου και το χρόνο που απαιτείται από το μεσισμικό πλέγματος για να ολοκληρώσει τις λειτουργίες του. Λόγω του γεγονότος ότι οι WCDMA προσομοιώσεις που εκτελέστηκαν δεν περιλαμβάνουν τη μεταφορά μεγάλων όγκων δεδομένων, ο κυρίαρχος παράγοντας επιβάρυνσης είναι ο χρόνος αναμονής σε ουρές, ο οποίος εξαρτάται αποκλειστικά από το υπολογιστικό φορτίο που έχει υποβληθεί στο πλέγμα τη στιγμή που υποβάλλεται η εργασία. Όπως φαίνεται στον παραπάνω πίνακα, οι χρόνοι αναμονής σε ουρές και κατά συνέπεια και η επιβάρυνση λόγω πλέγματος μπορούν να διαφέρουν πολύ κατά διαστήματα.

Ο Πίνακας 11 δείχνει το συνολικό χρόνο εκτέλεσης εργασίας, το χρόνο εκτέλεσης προσομοίωσης και τη συνολική επιτάχυνση. Ο χρόνος εκτέλεσης εργασίας είναι ο πραγματικός χρόνος που η εργασία εκτελεί τις επιμέρους προσομοιώσεις που της έχουν ανατεθεί. Η απόκλιση μεταξύ ελάχιστου και μέγιστου χρόνου εκτέλεσης οφείλονται κυρίως στη στοχαστική φύση των προσομοιώσεων (διαφορετικός αριθμός χρηστών συνεπάγεται διαφορετικούς χρόνους εκτέλεσης), αλλά και στην πιθανή ετερογένεια των κόμβων του πλέγματος, στους οποίους ανατέθηκαν τελικά οι εργασίες. Ο συνολικός χρόνος εκτέλεσης στο πλέγμα είναι ο συνολικός χρόνος (συμπεριλαμβανομένης της επιβάρυνσης πλέγματος) που απαιτείται για την ολοκλήρωση και της τελευταίας εργασίας και επομένως και για την ολοκλήρωση της Monte Carlo προσομοίωσης.

Σενάριο προσομοίωσης	Ελάχιστος χρόνος εκτέλεσης εργασίας	Μέγιστος χρόνος εκτέλεσης εργασίας	Μέσος χρόνος εκτέλεσης εργασίας	Συνολικός χρόνος εκτέλεσης στο πλέγμα	Επιτάχυνση (Speedup)
1 tier, 50% loading	393	513	421	1056	20
1 tier, 90% loading	1073	1130	1046	1634	32
2 tiers, 50% loading	483	619	542	912	30
2 tiers, 90% loading	2869	4079	3141	7129	22
3 tiers, 50% loading	990	1378	1208	5912	10
3 tiers, 90% loading	12239	14437	13236	24664	27
4 tiers, 50% loading	2711	4307	3552	6535	27
4 tiers, 90% loading	41034	50078	45909	50178	46

Πίνακας 11. Στατιστικά Εκτέλεσης στο Πλέγμα (2)

Η επιτάχυνση (speedup) υπολογίζεται ως εξής:

$$speedup = \frac{\sum_{i=1}^{no\_of\_jobs} (\text{Χρόνος\_Εκτέλεσης\_Εργασίας})_i}{\text{Συνολικός\_Χρόνος\_Εκτέλεσης\_στο\_Πλέγμα}}$$

Η έννοια του speedup δεν είναι αυτή που χρησιμοποιείται παραδοσιακά για την αποτίμηση της απόδοσης ενός παράλληλου αλγόριθμου, λόγω του ότι οι χρόνοι εκτέλεσης

στο πλέγμα εξαρτώνται σε μεγάλο βαθμό από το χρόνο που μια εργασία παραμένει στην ουρά. Παρόλα αυτά, το speedup, όπως παρουσιάζεται εδώ, είναι ένα μέγεθος που δείχνει πόσο επιταχύνεται η προσομοίωση από την πλευρά του χρήστη. Όπως φαίνεται στον παραπάνω πίνακα, το speedup κυμαίνεται από 10 έως 46, με μέση τιμή ίση με 27. Είναι φανερό ότι όσο πιο υπολογιστικά απαιτητική είναι η προσομοίωση, τόσο υψηλότερη θα είναι η τιμή του speedup που επιτυγχάνεται, καθώς οι πραγματικοί χρόνοι εκτέλεσης θα αυξάνονται επιπλέον.

7 Σύνοψη και Θέματα Μελλοντικής Έρευνας

7.1 Σύνοψη και Συμπεράσματα της Ερευνητικής Δραστηριότητας

Οι υπολογιστικές απαιτήσεις των προβλημάτων που σχετίζονται με την προσομοίωση, τη βελτιστοποίηση και τη σχεδίαση σύγχρονων συστημάτων ασύρματων και κινητών επικοινωνιών ολοένα και αυξάνονται, καθώς λαμβάνονται υπόψη προηγμένες τεχνικές και μοντέλα (για παράδειγμα τεχνικές MIMO και ντετερμινιστική πρόβλεψη διάδοσης). Αντικείμενο της παρούσας διδακτορικής διατριβής αποτέλεσε η αντιμετώπιση συγκεκριμένων προβλημάτων με χρήση τεχνικών παράλληλης και κατανεμημένης επεξεργασίας, καθώς και με χρήση των τεχνολογιών πλέγματος. Στα πλαίσια της διατριβής μελετήθηκαν τρία προβλήματα, για το καθένα από τα οποία ακολουθήθηκε διαφορετική προσέγγιση και αναπτύχθηκε αντίστοιχη υλοποίηση, με βάση τα χαρακτηριστικά και τις απαιτήσεις του.

Αρχικά αναπτύχθηκε παράλληλο μοντέλο για την πρόβλεψη διάδοσης σε ραδιοκανάλι με βάση την τεχνική ανίχνευσης ακτίνων. Το μοντέλο που θεωρήθηκε στηρίζεται στην ηλεκτρομαγνητική θεωρία των ειδώλων και παρουσιάζει ορισμένες δυσκολίες στην παραλληλοποίηση, λόγω των αλληλεξαρτήσεων που υπάρχουν στη ροή εκτέλεσης. Το παράλληλο σχήμα εκτέλεσης που αναπτύχθηκε στηρίζεται στην τμηματική και σταδιακή κατασκευή του δέντρου ειδώλων. Η παράλληλη υλοποίηση έγινε με βάση τις προδιαγραφές MPI και χρησιμοποιεί κατάλληλες τεχνικές για την ελαχιστοποίηση του κόστους επικοινωνίας. Πραγματοποιήθηκαν ένα σύνολο πειραμάτων για περιοχές διαφορετικού μεγέθους, καθώς και για διαφορετικό αριθμό ανακλάσεων. Τα αποτελέσματα που προέκυψαν χρησιμοποιώντας μέχρι και 17 επεξεργαστές δείχνουν ότι η παράλληλη υλοποίηση επιτυγχάνει πολύ καλή κλιμάκωση, ενώ η επιβάρυνση που επιφέρει στην

αντίστοιχη σειριακή παραμένει σε πολύ μικρά επίπεδα. Επίσης, πραγματοποιήθηκε μελέτη της επενέργειας διαφορετικών σχημάτων ανάθεσης εργασιών που έχουν προταθεί στη βιβλιογραφία στην απόδοση του παράλληλου μοντέλου. Το συμπέρασμα που προέκυψε είναι ότι τα διαφορετικά σχήματα δεν παρουσιάζουν μεγάλες διαφορές για σχετικά μικρό αριθμό επεξεργαστών, ωστόσο καθώς ο αριθμός αυτός αυξάνεται, οι διαφορές γίνονται πιο εμφανείς και ορισμένα σχήματα παρουσιάζονται καταλληλότερα.

Στη συνέχεια αναπτύχθηκε η κατανεμημένη πλατφόρμα Βελτιστοποίησης Νέφους Σωματιδίων με βάση το υπολογιστικό παράδειγμα των κινητών πρακτόρων. Χρησιμοποιώντας ως βασική προγραμματιστική διεπαφή το JADE API, δημιουργήθηκαν πράκτορες που εκμεταλλεύονται τη δυνατότητα επικοινωνίας τους για να υλοποιήσουν την απαραίτητη λειτουργικότητα και να αποτελέσουν έτσι ένα κατανεμημένο σμήνος. Ένα από τα βασικότερα πλεονεκτήματα της συγκεκριμένης προσέγγισης είναι η ανεξαρτησία μεταξύ της υλοποίησης του συστήματος των κινητών πρακτόρων και της υποδομής υλικού στην οποία αυτό θα εκτελεστεί. Η πλατφόρμα δοκιμάστηκε σε πρόβλημα βελτιστοποίησης διαγράμματος επίπεδης στοιχειοκεραίας. Τα αποτελέσματα που προέκυψαν δείχνουν ότι το κατανεμημένο σύστημα επιτυγχάνει τη μείωση του υπολογιστικού χρόνου που απαιτείται για την ολοκλήρωση του προβλήματος βελτιστοποίησης, παρόλα αυτά δεν παρουσιάζει την ιδανική συμπεριφορά που παρουσίασε το παράλληλο μοντέλο πρόβλεψης διάδοσης. Αυτό οφείλεται κατά κύριο λόγο στα πολλαπλά σημεία συγχρονισμού μεταξύ των υπολογιστικών κόμβων που επιβάλλονται σε κάθε επανάληψη του αλγορίθμου για την ανανέωση των θέσεων και ταχυτήτων των σωματιδίων, και λιγότερο στην αυξημένη επιβάρυνση επικοινωνίας που επιφέρει το σύστημα κινητών πρακτόρων σε σχέση με τις βιβλιοθήκες MPI.

Στο τελευταίο τμήμα της διατριβής παρουσιάστηκε ένα περιβάλλον για την προσομοίωση συστημάτων ασύρματων επικοινωνιών στο υπολογιστικό πλέγμα EGEE. Μελετήθηκαν οι μηχανισμοί που παρέχονται για την αλληλεπίδραση με το πλέγμα, οι οποίοι είναι αρκετά πολύπλοκοι για τον τελικό χρήστη. Για το λόγο αυτό αναπτύχθηκε μια ασφαλής δικτυακή πύλη που στηρίζεται στην τεχνολογία ιστού, η οποία λειτουργεί ως διεπαφή στο μεσισμικό πλέγματος (π.χ. για υποβολή και επίβλεψη εργασιών, ανάκτηση αποτελεσμάτων κλπ). Η υλοποίηση της πύλης βασίστηκε σε πρότυπες τεχνολογίες, η επιλογή των οποίων έγινε με βάση τις δυνατότητες για ευκολότερη μελλοντική επέκταση. Το περιβάλλον προσομοίωσης επιτρέπει την εκτέλεση στο πλέγμα Monte Carlo προσομοιώσεων, οι οποίες χρησιμοποιούνται συχνότατα για συστήματα ασύρματων επικοινωνιών. Τα στατιστικά εκτέλεσης που προέκυψαν από την εκτέλεση προσομοιώσεων

σεναρίων συστήματος WCDMA στο πλέγμα δείχνουν ότι η συγκεκριμένη προσέγγιση είναι αποδοτική για περιπτώσεις που το πρόβλημα μπορεί να αποσυντεθεί σε ένα σύνολο από εντελώς ανεξάρτητα τμήματα, όπως είναι για παράδειγμα μια Monte Carlo προσομοίωση.

7.2 Θέματα Μελλοντικής Έρευνας

Κάποια από τα θέματα που μελετήθηκαν στην παρούσα διατριβή μπορούν να αποτελέσουν τη βάση για περαιτέρω ερευνητική αναζήτηση. Οι προτάσεις για μελλοντική έρευνα, θα μπορούσαν να συνοψιστούν στα εξής σημεία:

7.2.1 Ανάπτυξη Υβριδικής Παράλληλης Υλοποίησης για τον Αλγόριθμο Πρόβλεψης Διάδοσης

Όπως προαναφέρθηκε, η ανάπτυξη του παράλληλου αλγορίθμου πρόβλεψης διάδοσης βασίζεται αποκλειστικά στο μοντέλο ανταλλαγής μηνυμάτων και τις προδιαγραφές MPI. Αν και τα αποτελέσματα από τη συγκεκριμένη υλοποίηση είναι πολύ κοντά στα ιδανικά, παρουσιάζει ενδιαφέρον η μελέτη υβριδικής προσέγγισης που συνδυάζει τα μοντέλα της διαμοιραζόμενης μνήμης και της ανταλλαγής μηνυμάτων (μια τέτοια υλοποίηση μπορεί για παράδειγμα να γίνει με βάση το μοντέλο των νημάτων και το MPICH). Το πλεονέκτημα αυτής της προσέγγισης είναι ότι κατά την εκτέλεση σε ένα σύστημα που αποτελείται από SMP κόμβους, η επικοινωνία μέσω δικτύου μπορεί να πραγματοποιείται μεταξύ μόνο μιας διεργασίας ανά κόμβο, ενώ ο συντονισμός της εκτέλεσης για τις διεργασίες που εκτελούνται σε έναν κόμβο γίνεται μέσω της διαμοιραζόμενης μνήμης. Αυτό έχει ως αποτέλεσμα τη μείωση της επιβάρυνσης επικοινωνίας πάνω από το δίκτυο και ενδέχεται να οδηγήσει σε βελτιωμένα αποτελέσματα.

7.2.2 Ανάπτυξη Παράλληλου Μοντέλου Πρόβλεψης Διάδοσης σε Κάρτες Γραφικών Γενικής Χρήσης

Μία πρόσφατη τεχνολογία που χρησιμοποιείται όλο και περισσότερο για την επιτάχυνση επιστημονικών εφαρμογών είναι ο προγραμματισμός μονάδων επεξεργασίας γραφικών (Graphics Processing Units – GPUs). Με την ανάπτυξη της αρχιτεκτονικής CUDA (Compute Unified Device Architecture) [109], παρέχεται πλέον στον προγραμματιστή η δυνατότητα εκμετάλλευσης της επεξεργαστικής ισχύος των σύγχρονων GPUs, οι οποίες αποτελούνται από εκατοντάδες πυρήνες, για την επιτάχυνση της εκτέλεσης εφαρμογών. Το

βασικότερο πλεονέκτημα από την προσέγγιση αυτή είναι το μικρό κόστος απόκτησης και συντήρησης του απαιτούμενου υλικού, αλλά και οι αυξημένες δυνατότητες κλιμάκωσης. Η νέα αυτή τεχνολογία μπορεί να χρησιμοποιηθεί για την ανάπτυξη κάποιας άλλης εκδοχής παράλληλου αλγορίθμου διάδοσης, που θα εκμεταλλεύεται την ισχύ των σύγχρονων GPUs. Βέβαια, η προσέγγιση αυτή συνεπάγεται την εκ νέου μελέτη του αλγορίθμου σε διαφορετικό επίπεδο, καθώς η διαδικασία παραλληλοποίησης σε αυτή την περίπτωση χρειάζεται να γίνει σε πολύ πιο "εκλεπτυσμένο" επίπεδο, με βάση το προγραμματιστικό μοντέλο της CUDA.

7.2.3 Υλοποίηση και Μελέτη Κατανεμημένων Υλοποιήσεων Διαφορετικών Εκδοχών του Αλγορίθμου PSO

Κατά την ανάπτυξη του κατανεμημένου συστήματος PSO που αναπτύχθηκε στα πλαίσια της διατριβής χρησιμοποιήθηκε η βασική έκδοση του αλγορίθμου, η οποία λειτουργεί με βάση την εύρεση του ολικού βέλτιστου σε κάθε επανάληψη για όλα τα σωματίδια του σμήνους. Αυτός είναι και ο λόγος που απαιτείται συγχρονισμός μετά το τέλος κάθε επανάληψης, ο οποίος περιορίζει την απόδοση της παράλληλης υλοποίησης. Στη βιβλιογραφία έχουν προταθεί διάφορες άλλες εκδοχές του PSO που στηρίζονται στην έννοια της "γειτονιάς", όπου το gbest υπολογίζεται για ένα υποσύνολο των σωματιδίων του σμήνους. Με βάση αυτές τις εκδοχές, υπάρχει δυνατότητα να περιοριστεί σε μεγάλο βαθμό η επιβάρυνση της επικοινωνίας θεωρώντας ως γειτονιά το σύνολο των σωματιδίων που αντιστοιχούν σε έναν υπολογιστικό κόμβο, πράγμα που μπορεί να οδηγήσει σε καλύτερη απόδοση της παράλληλης υλοποίησης.

7.2.4 Ανάπτυξη Ολοκληρωμένου Περιβάλλοντος για τη Βελτιστοποίηση και Σχεδίαση Συστημάτων Ασύρματων Επικοινωνιών στο Πλέγμα

Το περιβάλλον προσομοίωσης συστημάτων ασύρματων επικοινωνιών που αναπτύχθηκε στα πλαίσια της διατριβής μπορεί να αποτελέσει τη βάση για την ανάπτυξη ενός ολοκληρωμένου περιβάλλοντος που επιτρέπει τη σχεδίαση ενός δικτύου με κατανεμημένο και συνεργατικό (collaborative) τρόπο. Τα επιμέρους στοιχεία (κώδικες προσομοίωσης, κώδικες βελτιστοποίησης, αλγόριθμοι ray tracing, βάσεις δεδομένων για περιοχές, δεδομένα υπολογισμού διάδοσης, κτλ) μπορούν να παρέχονται σε μορφή υπηρεσιών. Με αυτόν τον τρόπο είναι δυνατή η ανάπτυξη μιας αρχιτεκτονικής με βάση τις υπηρεσίες και το πρότυπο OGSA των υπηρεσιών πλέγματος. Η πρόσβαση στις υπηρεσίες θα γίνεται μέσα από το

περιβάλλον της δικτυακής πύλης, όπου ο χρήστης θα μπορεί να χρησιμοποιήσει τις διάφορες διαθέσιμες υπηρεσίες για τη δημιουργία πολύπλοκων ροών εκτέλεσης στο πλέγμα, με σκοπό τη βελτιστοποίηση και τη σχεδίαση δικτύων με βάση τις απαιτήσεις που αυτός παρέχει.

8 Βιβλιογραφία

[1] D. Gesbert, M. Shafi, D. Shiu, P. Smith and A. Naguib, "From theory to practice: An overview of MIMO space-time coded wireless systems", IEEE Journal on Selected Areas in Communications, Vol. 21, Issue 3, pp. 281-302, April 2003

[2] M. Flynn, “Some Computer Organizations and Their Effectiveness”, IEEE Transactions on Computing., Vol. C, Issue 21, pp. 948, 1972

[3] R. Duncan, "A Survey of Parallel Computer Architectures", IEEE Computer, Vol. 23, Issue 2, pp. 5-16, 1990

[4] H. El-Rewini and M. Abd-El-Barr, “Advanced Computer Architecture and Parallel Processing”, John Wiley and Sons Publications, 2005

[5] Jack Dongarra, Ian Foster, Geoffrey Fox, William Gropp, Ken Kennedy, Linda Torczon and Andy White, “Sourcebook of parallel computing”, Morgan Kaufmann Publishers Inc, 2003

[6] David Butenhof, “Programming with POSIX threads”, Addison-Wesley Longman Publishing Co., Inc., 1997

[7] L. Dagum., R. Menon, “OpenMP: an industry standard API for shared-memory programming”, IEEE Computational Science and Engineering, Vol. 5, Issue 1, pp. 46-55, 1998

[8] Robit Chandra, Leonardo Dagum, Dave Kohr, Dror Maydan, Jeff McDonald, Ramesh Menon, “Parallel programming in OpenMP”, Morgan Kaufmann Publishers Inc., 2001

[9] Al Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek, Vaidy Sunderam, “PVM: Parallel Virtual Machine, A Users' Guide and Tutorial for Networked Parallel Computing”, The MIT Press, 1994

[10] Marc Snir, Steve Otto, David Walker , Jack Dongarra, Steven Huss-Lederman, “MPI: The Complete Reference”, MIT Press, Cambridge, MA, 1995

[11] Troy Downing, “Java RMI: Remote Method Invocation”, IDG Books Worldwide, Inc, 1998

[12] Ken Arnold, Robert Scheifler, Jim Waldo, Bryan O'Sullivan, Ann Wollrath, “Jini Specification”, Addison-Wesley Longman Publishing Co., Inc, 1999

[13] Alan Pope, “The CORBA reference guide: understanding the Common Object Request Broker Architecture”, Addison-Wesley Longman Publishing Co, Inc, 1998

[14] J.M. Bradshaw, “Software Agents“, MIT Press, 1997

[15] K. Rothermel, F. Hohl and N. Radouniklis, “Mobile Agent Systems: What is missing?“, International Working Conference on Distributed Applications and Interoperable Systems (DAIS'97), 1997

[16] A. Fuggetta, G.P. Picco and G.Vigna, “Understanding Code Mobility“, IEEE transactions on Software Engineering, Vol. 24, Issue 5, pp. 342-362, 1998

[17] C. Ghezzi and G. Vigna, “Mobile Code Paradigms and Technologies: A Case Study“, Proceedings of the First International Workshop on Mobile Agents, pp. 39-49, 1997

[18] C. Baumer, M. Breugst, S. Choy and T. Magedanz, “Grasshopper - a universal agent platform based on OMG MASIF and FIPA standards”, First International Workshop on Mobile Agent for Telecommunication Applications, pp. 1-18, 1999

[19] F. Bellifemine, G.Caire and D. Greenwood**, “**Developing Multi-Agent Systems with JADE**”,** John Wiley & Sons, 2007

[20] Foundation for Intelligent Physical Agents (FIPA): http://www.fipa.org/specifications/index.html

[21] Gene Amdahl, "Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities", AFIPS Conference Proceedings, pp. 483-485, 1967

[22] John Gustafson, **“**Reevaluating Amdahl's law**”,** Communications of the ACM, Vol. 31, Issue 5, pp 532-533, 1988

[23] C.A. Balanis, “Advanced Engineering Electromagnetics”, John Wiley & Sons, New York, NY, 1989

[24] J.B. Keller, “Geometrical Theory of Diffraction”, Journal of the Optical Society of America, Vol. 52, Issue 2, pp. 116-130, February 1962

[25] R. Kouyoumjian and P. Pathak, “A uniform geometric theory of diffraction for an edge on a perfectly conducting surface”, Proceedings of the IEEE, Vol. 62, Issue 11, pp. 1448-1461, November 1974

[26] R.J. Luebbers, “Finite Conductivity Uniform GTD versus Knife Edge Diffraction in Prediction of Propagation Path Loss”, IEEE Transactions on Antennas & Propagation, Vol. 32, Issue 1, pp. 70-76, January 1984

[27] H. Bertoni, "Radio Propagation for Modern Wireless Systems", Prentice-Hall PTR, Upper Saddle River, NJ, 2000

[28] K. Ng, E. Tameh and A. Nix, "An advanced multi-element microcellular ray tracing model", 1st International Symposium on Wireless Communication Systems, pp. 438-442, 2004

[29] M. Lawton and J. McGeehan, "The Application of a Deterministic Ray Launching Algorithm for the Prediction of Radio Channel Characteristics in Small - Cell Environments", IEEE Transactions on Vehicular Technology, Vol. 43, Issue 4, pp. 955-969, November 1994

[30] U. Dersch and E. Zollinger, "Propagation Mechanisms in Microcell and Indoor Environments", IEEE Transactions on Vehicular Technology, Vol. 43, Issue 4, pp. 1058-1066, November 1994

[31] S. Tan and H. Tan, "UTD Propagation Model in an Urban Street Scene for Microcellular Communications", IEEE Transactions on Electromagnetic Compatibility, Vol. 35, Issue 4, pp. 423-428, November 1993

[32] Z. Chen, A. Delis and H. Bertoni, "Building footprint simplification techniques and their effects on radio propagation predictions", The Computer Journal, Vol. 47, Issue 1, pp. 103-133, January 2004

[33] T. Kurner and A. Meier, "Prediction of Outdoor and Outdoor-to-Indoor Coverage in Urban Areas at 1.8 GHz", IEEE Journal on Selected Areas in Communications, Vol. 20, Issue 3, pp. 496-506, April 2002

[34] Z. Chen, H. Bertoni and A. Delis, "Progressive and Approximate Techniques in Ray-Tracing Based Radio Wave Propagation Prediction Models", IEEE Transactions on Antennas and Propagation, Vol. 52, Issue 1, pp. 240–251, January 2004

[35] P. Huttunen, J. Porras, J. Ikonen, and K. Sipila, "Using Cray T3E for the Parallel Calculation of Cellular Radio Coverage", Proceedings of the Eurosim'98, Helsinki, Finland, pp. 27-32, April 1998

[36] P. Huttunen, J. Ikonen, J. Porras, and K. Sipila, "Parallelization of Propagation Model Simulation", Science Technology: Science and Art, 10th European Simulation Symposium'98, Nottingham, United Kingdom, October 1998

[37] A. Verstak, J. He, L. Watson, T. Rappaport, C. Anderson, K. Bae, J. Jiang and W. Tranter, "S^4W: Globally Optimized Design of Wireless Communication Systems", Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'02), Fort Lauderdale, Florida, pp. 173-180, April 2002

[38] J. He, A. Verstak, L. Watson, T. Rappaport, C. Anderson, N. Ramakrishnan, C. Shaffer, W. Tranter, K. Bae and J. Jiang, "Global Optimization of Transmitter Placement in Wireless Communication Systems", Proceedings of High Performance Computing Symposium 2002, San Diego, CA, pp. 328-333, 2002

[39] J. He, A. Verstak, L. Watson, C. Stinson, N. Ramakrishnan, C. Shaffer, T. Rappaport, C. Anderson, K. Bae, J. Jiang, and W. Tranter, "Globally Optimal Transmitter Placement for Indoor Wireless Communication Systems", IEEE Transactions on Wireless Communications, Vol. 3, Issue 6, pp. 1906-1911, November 2004

[40] Z. Chen, A. Delis and H. Bertoni, "Radio-wave propagation predictions using ray-tracing techniques on a network of workstations (NOW)", Journal of Parallel and Distributed Computing, Vol. 64, Issue 10, pp. 1127-1156, October 2004

[41] A. Cavalcante, M. de Sousa, J. Costa, C. Frances, G. Protasio dos Santos Cavalcante and C. de Souza Sales, "3D ray-tracing parallel model for radio-propagation prediction", 2006 International Telecommunications Symposium, pp. 269-274, September 2006

[42] S. Seidel and T. Rappaport, "Site-Specific Propagation Prediction for Wireless In-Building Personal Communications System Design", IEEE Transactions on Vehicular Technology, Vol. 43, Issue 4, pp. 1058-1066, November 1994

[43] K. Schaubach and N. Davis, "Microcellular Radio-Channel Propagation Prediction", IEEE Antennas and Propagation Magazine, Vol. 36, Issue 4, pp. 25-34, August 1994

[44] G. Athanasiadou, “Study, Development and Evaluation of Novel Indoor and Outdoor Ray Tracing Radio-Wave Propagation Models”, Ph.D. Thesis, University of Bristol, 1997

[45] G. Athanasiadou, "Incorporating the Fresnel Zone Theory in Ray Tracing for Propagation Modelling of Fixed Wireless Access Channels", IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2007), pp. 1-5, 2007

[46] M. Lawton, J. McGeehan, "The Application of a Deterministic Ray Launching Algorithm for the Prediction of Radio Channel Characteristics in Small - Cell Environments", IEEE Transactions on Vehicular Technology, Vol. 43, Issue 4, pp. 955-969, November 1994

[47] B. Freisleben, D. Hartmann, and T. Kielmann, "Parallel Raytracing: A Case Study on Partitioning and Scheduling on Workstation Clusters", Proceedings of the Thirtieth International Conference on System Sciences, Maui, HI, pp. 596–605, 1997

[48] E. Reinhard and A. Chalmers, "Message Handling in Parallel Radiance", Proceedings of the 4th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface, pp. 486-493, 1997

[49] MPICH - A Portable Implementation of MPI,

http://www.mcs.anl.gov/research/projects/mpi/mpich1/

[50] CSELT S.p.A., TILab S.p.A., “JADE Programmer's Guide”, March 2005

[51] CSELT S.p.A., TILab S.p.A., “JADE Administrator's Guide”, March 2005

[52] J. Kennedy and R. Eberhart, “Particle swarm optimization”, in Proc. of IEEE International Conference on Neural Networks, Piscataway, NJ, pp. 1942-1948, 1995

[53] J. Kennedy and R. Eberhart, “Swarm Intelligence”, Morgan Kaufmann, San Francisco, CA, 2001

[54] B. Massingill, T. Mattson, and B. Sanders, “Patterns for Parallel Application Programs”, in Proc. of Sixth Pattern Languages of Programs Workshop, 1999

[55] K. Papadopoulos, C. Papagianni, I. Foukarakis et al., “Optimal Design of Switched Beam Antenna Arrays Using Particle Swarm Optimization”, in Proceedings of The European Conference on Antennas and Propagation: (EuCAP 2006), published on CDROM, p. 81.1, Nice, France, November 2006

[56] Ian Foster, Carl Kesselman: “The Grid: Blueprint for a Future Computing Infrastructure”, Morgan Kaufmann, 1998

[57] Ian Foster, Carl Kesselman, and Steven Tuecke: “The Anatomy of the Grid”, Lecture Notes in Computer Science, Vol. 2150, 2001

[58] Larry Smarr, Charles E. Catlett, "Metacomputing", Communications of the ACM, Vol. 35 , Issue 6, pp. 44-52, 1992

[59] FAFNER: Factoring via Network-Enabled Recursion, http://www.lehigh.edu/~bad0/fafner.html

[60] I. Foster, J. Geisler, W. Nickless, W. Smith, S. Tuecke “Software Infrastructure for the I-WAY High Performance Distributed Computing Experiment” in Proc. 5th IEEE Symposium on High Performance Distributed Computing. pp. 562-571, 1997

[61] David Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, Dan Werthimer, "SETI@home: an experiment in public-resource computing", Communications of the ACM, Vol. 45, Issue 11, pp. 56-61, 2002

[62] I. Foster and C. Kesselman, “Globus: A Metacomputing Infrastructure Toolkit”, International Journal of Supercomputer Applications, Vol. 11, Issue 2, pp. 115-128, 1997

[63] Andrew S. Grimshaw, William A. Wulf, “The Legion Vision of a Worldwide Virtual Computer”, Communications of the ACM, Vol. 40, Issue 1, pp. 39-45, 1997

[64] Foster, C. Kesselman, J. M. Nick, and S. Tuecke, "The Physiology of the Grid- An Open Grid Services Architecture for Distributed Systems Integration", The Globus Alliance, June 2002

[65] S. Tuecke et al., "Open Grid Services Infrastructure (OGSI) Version 1.0", Global Grid Forum, Proposed Recommendation GFD-R-P.15, 2003. http://www.ggf.org/documents/GFD.15.pdf

[66] K. Czajkowski, D. F. Ferguson, I. Foster, J. Frey, S. Graham, I. Sedukhin, D. Snelling, S. Tuecke, and W. Vambenepe, "The WS-Resource Framework", (March 2004), http://www-106.ibm.com/developerworks/library/ws-resource/ws-wsrf.pdf

[67] J. Joseph, M. Ernest, C. Fellenstein, "Evolution of grid computing architecture and grid adoption models", IBM Systems Journal, Vol. 43, Issue 4, pp. 624-645, 2004

[68] K. Czajkowski, D. F. Ferguson, I. Foster, J. Frey, S. Graham, I. Sedukhin, D. Snelling, S. Tuecke, and W. Vambenepe, "The WS-Resource Framework", (March 2004), http://www-106.ibm.com/developerworks/library/ws-resource/ws-wsrf.pdf

[69] Kevin Cline, Josh Cohen, Doug Davis, Donald F Ferguson, Heather Kreger, Raymond McCollum, Bryan Murray, Ian Robinson, Jeffrey Schlimmer, John Shewchuk, Vijay Tewari, William Vambenepe, "Toward Converging Web Service Standards for Resources, Events, and Management", http://download.boulder.ibm.com/ibmdl/pub/software/dw/webservices/Harmonization_Roadmap.pdf

[70] Ian Foster, "The Holy Grail: Industry-Wide System Management Standards at Last?", http://www.globus.org/wsrf/convergence.php

[71] Don Box, Erik Christensen, Francisco Curbera, Donald Ferguson, Jeffrey Frey, Marc Hadley, Chris Kaler, David Langworthy, Frank Leymann, Brad Lovering, Steve Lucco, Steve Millet, Nirmal Mukhi, Mark Nottingham, David Orchard, John Shewchuk, Eugène Sindambiwe, Tony Storey, Sanjiva Weerawarana, Steve Winkler, "Web Services Addressing (WS-Addressing) Standard", W3C, http://www.w3.org/Submission/ws-addressing/

[72] Organization for the Advancement of Structured Information Standards (OASIS), http://www.oasis-open.org

[73] WS-Resource Specification: http://docs.oasis-open.org/wsrf/wsrf-ws_resource-1.2-spec-os.pdf

[74] WS-ResourceLifetime Specification: http://docs.oasis-open.org/wsrf/wsrf-ws_resource_lifetime-1.2-spec-os.pdf

[75] WS-ServiceGroup Specification: http://docs.oasis-open.org/wsrf/wsrf-ws_service_group-1.2-spec-os.pdf

[76] WS-BaseFaults Specification: http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-spec-os.pdf

[77] Douglas Thain, Todd Tannenbaum, and Miron Livny, "Distributed Computing in Practice: The Condor Experience" Concurrency and Computation: Practice and Experience, Vol. 17, Issue 2-4, pp. 323-356, 2005

[78] The Legion Grid, home page: http://www.cs.virginia.edu/~legion/

[79] R. Buyya, D. Abramson, and J. Giddy, "Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid", HPC ASIA'2000, China, 2000

[80] Mathilde Romberg, “The UNICORE Grid infrastructure”, Scientific Programming, Vol. 10, Issue 2, pp. 149-157, 2002

[81] The gLite middleware, home page: http://glite.web.cern.ch/glite/

[82] I. Foster, “Globus Toolkit Version 4: Software for Service-Oriented Systems”, IFIP International Conference on Network and Parallel Computing, pp 2-13, 2005

[83] F. Gagdardi, B. Jones, F. Grey, M. Begin and M. Heikkurinen, "Building an infrastructure for scientific Grid computing : status and goals of the EGEE project", Philosophical transactions - Royal Society. Mathematical, physical and engineering sciences, Vol. 363, Issue 1833, pp. 1729-1742, 2005

[84] The Enabling Grids for E-sciEnces (EGEE) project, Homepage: http://www.eu-egee.org/

[85] P. Gkonis, G. Tsoulos and D. I. Kaklamani, “An Adaptive Admission Control Strategy for WCDMA Multicellular Networks with Non uniform Traffic”, 66th Vehicular Technical Conference, Baltimore, 2007

[86] C. A. J. Brew, F. F. Wilson, G. Castelli, T. Adye, E., Luppi, D. Andreotti, “BABAR Experience of Large Scale Production on the Grid”, Proceedings of the Second IEEE International Conference on e-Science and Grid Computing table of contents, page 151, 2006

[87] Shu Tezuka, Hiroki Murata, Shuji Tanaka and Shoji Yumae, ”Monte Carlo grid for financial risk management”, Future Generation Computer Systems, Special issue: Parallel computing technologies, Vol. 21, Issue 5, pp. 811-821, 2005

[88] J. Novotny, M. Russell, O. Wehrens, “Gridsphere: A portal framework for building collaborations, Concurrency and Computation: Practice and Experience”, Vol. 16, Issue 5, pp. 503-513, 2004

[89] JSR 168: Portlet Specification, http://jcp.org/en/jsr/detail?id=168

[90] LHC - The Large Hadron Collider, home page: http://lhc.web.cern.ch/lhc/

[91] The gLite 3 User Guide, https://edms.cern.ch/file/722398//gLite-3-UserGuide.html

[92] The GLUE schema, http://infnforge.cnaf.infn.it/glueinfomodel/

[93] The Globus Monitoring and Discovery System: http://globus.org/toolkit/mds/

[94] A. W. Cooke, A. J. G. Gray, W. Nutt et. al, “The Relational Grid Monitoring Architecture: Mediating Information about the Grid”, Journal of Grid Computing, Vol. 2, Issue 4, pp. 323 - 339, 2005

[95] Condor Classified Advertisements, http://www.cs.wisc.edu/condor/classad

[96] F. Pacini, Job Description Language How To, http://www.infn.it/workload-grid/docs/DataGrid-01-TEN-0102-0 2-Document.pdf

[97] GSIFTP Tools for the Data Grid, http://www.globus.org/toolkit/docs/2.4/datagrid/deliverables/gsiftp-tools.html

[98] JSR 168 - Portlet specification - http://www.jcp.org/en/jsr/detail?id=168

[99] J. Novotny, M. Russell, and O. Wehrens, "Gridsphere: A Portal Framework for Building Collaborations, Concurrency and Computation: Practice and Experience," Vol. 16, Issue 5, pp. 503-513, 2004

[100] J. Novotny, S. Tuecke, and V. Welch, "An Online Credential Repository for the Grid: MyProxy," Proceedings of the 10th International Symposium on High Performance Distributed Computing, New York, pp. 104-111, 2001

[101] GridLab: A Grid Application Toolkit and Testbed: http://www.gridlab.org/

[102] The GILDA gLite User Interface Plug & Play, https://gilda.ct.infn.it/UIPnP/

[103] The Hellas Grid - Task Force, http://www.hellasgrid.gr/

[104] Gregor von Laszewski, Ian Foster, Jarek Gawor and Peter Lane, “A Java commodity grid kit”, Concurrency and Computation: Practice and Experience, Vol. 13, Issue 8-9, pp.645-662, 2001

[105] F. James, “A Review of Pseudorandom Number Generators”, Computer Physics Communication. Vol. 60, Issue 3, pp. 329 – 344, 1990

[106] M. Traore, D. Hill, “The use of random number generation for stochastic distributed simulation: application to ecological modeling”, 13th European Simulation Symposium, Marseille, pp. 555-559, October 18-20, 2001

[107] S. Saunders, Antennas and Propagation for Wireless Communication Systems, JohnWiley & Sons, 1999

[108] The South-Eastern Europe Virtual Organization (SEE-VO), http://www.egee-see.org/see-vo.php?language=en

[109] J. Nickolls, I. Buck, M. Garland and K. Skadron, "Scalable Parallel Programming with CUDA", ACM Queue, Vol. 6 , Issue 2, pp. 40-53, March 2008

9 Παραρτήματα

9.1 Κατάλογος Δημοσιεύσεων

9.1.1 Διεθνή Επιστημονικά Περιοδικά

[1] T. Athanaileas, P. Gkonis, G. Athanasiadou, G. Tsoulos and D. Kaklamani, "Implementation and Evaluation of a Web-Based Grid-Enabled Environment for WCDMA Multibeam System Simulations", IEEE Antennas and Propagation Magazine, Vol. 50, issue 3, pp. 195-204, June 2008

9.1.2 Επιστημονικά Συνέδρια

[2] T. Athanaileas, P. Gkonis, G. Tsoulos and D. Kaklamani, "An Adaptive Framework for WCDMA System Analysis in the EGEE Grid Infrastructure", in proceedings of the 20th Open Grid Forum, Enabling Grids for E-Science(OGF20/EGEE), pp. 112, Manchester, UK, 9-11 May 2007

[3] T. Athanaileas, D. Dionysiou, G. Stamatakos, N. Mouravliansky, D. Kaklamani and N. Uzunoglu, "Applying Grid Technologies to In Silico Oncology", in 20th Open Grid Forum, Enabling Grids for E-Science(OGF20/EGEE), pp. 76, Manchester, UK, 9-11 May 2007

[4] P. Gkonis, T.Athanaileas, G.Tsoulos and D.Kaklamani, "Performance of WCDMA in a Multicellular Network for different Multiuser Detection Strategies," in proceedings of the 7th International Conference on ITS Telecommunications, pp. 195-200,Sofia Antipolis, 6-8 June 2007

[5] T. Athanaileas, K. Papadopoulos and D. Kaklamani, "A Multi-Agent Distributed Platform for Particle Swarm Optimization", in proceedings of Innovations in Intelligent Systems and Applications Symposium (ASYU-INISTA 2007), pp. 190-194, Constantinople, Turkey, 20-23 June 2007

[6] T. Athanaileas, G. Tsoulos and D. Kaklamani, "A Grid Enabled Problem Solving Environment for Monte Carlo Matlab Simulations", in proceedings of the 11th IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT'07), pp. 159-166, Chania, Crete Island, Greece, 22-24 October, 2007.

[7] D. Dionysiou, G. Stamatakos, T. Athanaileas, A. Menychtas, D. Kaklamani, T. Varvarigou and N. Uzunoglu, "In Silico Simulation of a Clinical Trial Concerning Tumour Response to Radiotherapy", in proceedings of the International Electronic Conference on Computer Science 2007 (IeCCS07), 14-17 December 2007(electronic conference)

[8] A. Menychtas, T. Athanaileas, D. Dionysiou, G. Stamatakos, D. Kaklamani, T. Varvarigou and N. Uzunoglu, "Development and adaptation of an in silico oncology application in grid environment", in 1st HellasGrid User Forum, Athens, Greece, 10-11 January 2008

[9] A. Menychtas, T. Athanaileas, D. Dionysiou, G. Stamatakos, D. Kaklamani, T. Varvarigou and N. Uzunoglu, "Development and adaptation of a web enabled in silico oncology application in grid environment", in 3rd Enabling Grid for E-Science (EGEE) User Forum, Clermont-Ferrand, France, 11-14 February, 2008

[10] T. Athanaileas, N. Tselikas, G. Tsoulos, and D. Kaklamani, "An Agent-based Framework for Integrating Mobility into Grid Services", in CD-ROM proceedings of the First International Conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications (MobilWare 2008), Innsbruck, Austria, February 13-15, 2008

[11] T. Athanaileas, A. Menychtas, D. Dionysiou, G. Stamatakos, D. Kaklamani, T. Varvarigou and N. Uzunoglu, "A Grid-Enabled Toolkit for In Silico Oncology Simulations", in CD-ROM proceedings of the First International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SIMUTools 2008), Marseille, France, March 3-7, 2008

[12] T. Athanaileas, A. Menychtas, D. Dionysiou, G. Stamatakos, D. Kaklamani, T. Varvarigou, I. Venieris and N. Uzunoglu, "Applying Grid Computing Technologies to In Silico Oncology", 3rd International Advanced Research Workshop on *In Silico Oncology*: Advances and Challenges (3rd IARWISO), September 23 - 24 2008, Istanbul, Turkey

9.1.3 Δημοσιεύσεις σε Διαδικασία Κρίσης

[1] T. Athanaileas, A. Menychtas, D. Dionysiou, D. Kyriazis, D. Kaklamani, T. Varvarigou, N. Uzunoglu and G. Stamatakos, "Exploiting Grid Technologies for the Simulation of Clinical Trials *In Silico*: the Paradigm of *In Silico* Radiation Oncology", Invited in:

SIMULATION: Transactions of The Society for Modeling and Simulation International, Major Revisions Requested

[2] K. Papadopoulos, T. Athanaileas and D. Kaklamani, “Using Java Mobile Agents and PSO for Implementing a Distributed Antenna Optimization Platform”, Submitted at IEEE Antennas and Propagation Magazine

[3] P. Gkonis, T. Athanaileas, G. Tsoulos, G. Athanasiadou, and D. Kaklamani, “Adaptive Beam-Centric Admission Control for WCDMA Multicell/Multiservice Scenarios with Non-Uniform Traffic", submitted at the Journal of Wireless Personal Communications (Kluwer).

9.1.4 Αναφορές από Τρίτους

[1] Laura Garcia-Lopez, Mario Reyes-Ayala, Sergio Vidal-Beltran, “Inter-cell Interference for Reverse Link in WCDMA”, WSEAS Transactions on Communications, Vol. 7, Issue 1, pp. 974 – 983, January 2008