**ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ - ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ**

# ΔΙΕΡΕΥΝΗΣΗ ΤΗΣ ΑΥΤΟΝΟΜΙΑΣ ΠΡΑΚΤΟΡΩΝ ΚΑΙ ΤΟΥ ΑΥΤΟΝΟΜΟΥ ΥΠΟΛΟΓΙΣΜΟΥ ΣΤΗΝ ΠΛΑΤΦΟΡΜΑ JADE

# EXPLORING AGENT AUTONOMY AND AUTOMATIC COMPUTING IN JADE

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Φύκας Ιωάννης Ηλίας

Επιβλέποντες Καθηγητές: Δασκαλοπούλου Ασπασία
Επίκουρη Καθηγήτρια Π.Θ.

Λάλης Σπυρίδων-Γεράσιμος
Αναπληρωτής Καθηγητής Π.Θ.

Βόλος, Φεβρουάριος 2013

## Ευχαριστίες

*Με την περάτωση της παρούσας διπλωματικής εργασίας θα ήθελα να ευχαριστήσω την κα Δασκαλοπούλου για την πολύτιμη βοήθεια και συνεχή καθοδήγηση που μου παρείχε όλους αυτούς τους μήνες, καθώς και για την εμπιστοσύνη που μου έδειξε.*

*Ένα μεγάλο ευχαριστώ οφείλω στην οικογένεια μου για την στήριξη τους και την πίστη τους όλα αυτά τα χρόνια. Χωρίς αυτούς τίποτα απ' ότι έχω κάνει εως τώρα δεν θα ήταν δυνατό.*

*Τέλος θέλω να ευχαριστήσω τους φίλους μου, οι οποίοι συνέβαλαν σημαντικά στο να γίνω ο άνθρωπος που είμαι σήμερα. Είναι εκεί για να μου θυμίζουν τι έχει πραγματική αξία στη ζωή και τους ευχαριστώ γι' αυτό.*

# Περίληψη

Κύριος στόχος της παρούσας διπλωματικής εργασίας είναι η μελέτη των μηχανισμών λειτουργίας ενός Cloud συστήματος που αποτελείται από αυτόνομους πράκτορες-μέλη. Πιο συγκεκριμένα, η μελέτη επικεντρώνεται στους μηχανισμούς αντιμετώπισης μιας πιθανής αποχώρησης ενός πράκτορα από το σύστημα κατά τη διάρκεια που το σύστημα έχει αναλάβει μία ομαδική εργασία.

Για την αποτελεσματικότερη μελέτη των μηχανισμών υλοποιήθηκαν δύο διαφορετικά Cloud συστήματα. Το πρώτο αποτελεί στην ουσία ένα κεντρικοποιημένο σύστημα όπου ένας πράκτορας έχει τον ρόλο του διαχειριστή και όλες οι επικοινωνίες εντός του συστήματος υποστηρίζονται από αυτόν. Το δεύτερο σύστημα είναι πιο ευέλικτο, καθώς υπάρχουν κανάλια επικοινωνίας ανάμεσα στα μέλη και έτσι δεν είναι απαραίτητη η διαμεσολάβιση του διαχειριστή για όλες τις επικοινωνίες εντός του συστήματος. Οι τοπολογίες των κόμβων στα δύο αυτά συστήματα σχεδιάστηκαν με τέτοιο τρόπο, ώστε να περιλαμβάνουν κάθε είδος σύνδεσης που μπορεί να υπάρξει ανάμεσα σε δύο κόμβους ενός συστήματος.

Η υλοποίηση έγινε στην πολυπρακτορική πλατφόρμα *JADE*, χρησιμοποιώντας αύτονομους νοήμονες πράκτορες για τα δομικά στοιχεία-μέλη των δύο συστημάτων.

## Abstract

The main objective of this thesis is to study the mechanisms of operation of a Cloud system consisting of autonomous agents-members. More specifically, the study focuses on the response mechanisms of a possible departure of an agent from the system during the system has undertaken a team work.

To better study the mechanisms, two different Cloud systems were implemented. The first is actually a centralized system where an agent has the role of the administrator and all communications within the system are supported by it. The second system is more flexible, and there are channels of communication between the members and so there is no need for the administrator to support every communication attempt within the system. The topologies of nodes in the two systems are designed in such a way as to include any type of connection that can exist between two nodes in a system.

The implementation was done in the multiagent platform *JADE*, using autonomous intelligent agents for components-members of the two systems.

# Πίνακας Περιεχομένων

# *Κεφάλαιο 1$^{ο}$*

## Εισαγωγή

### 1.1 Ασύρματα δίκτυα *ad hoc*

Ο αριθμός των ad hoc δικτύων ολοένα και αυξάνεται τα τελευταία χρόνια και η χρήση τους βρίσκει εφαρμογή σε πολλές πτυχές της καθημερινότητας, ακόμα και αν η παρουσία τους δεν γίνεται πάντα αντιληπτή. Τα συγκεκριμένα ασύρματα δίκτυα απασχολούν εδώ και μερικά χρόνια τους ερευνητές και θα συνεχίσουν να τους απασχολούν ακόμη περισσότερο λόγω της σημαντικής χρηστικότητας που παρουσιάζουν σε ένα μεγάλο εύρος εφαρμογών.

Ένα ασύρματο δίκτυο ad hoc είναι ένας αποκεντρωμένος τύπος δικτύου. Το δίκτυο ονομάζεται ad hoc επειδή δεν βασίζεται σε μία προϋπάρχουσα υποδομή (όπως είναι οι δρομολογητές στα ενσύρματα δίκτυα). Αντ' αυτού, κάθε κόμβος συμμετέχει στην δρομολόγηση προωθώντας δεδομένα για άλλους κόμβους και έτσι ο προσδιορισμός των κόμβων που προωθούν δεδομένα γίνεται δυναμικά με βάση την συνδεσιμότητα του δικτύου. Εκτός από την κλασσική δρομολόγηση, τα ασύρματα δίκτυα ad hoc μπορούν να χρησιμοποιήσουν και την τεχνική της πλημμύρας (flooding) για την προώθηση των δεδομένων. Ένα δίκτυο ad hoc αναφέρεται συνήθως σε οποιοδήποτε σύνολο δικτύων όπου όλες οι συσκευές είναι ισότιμες μέσα στο δίκτυο και είναι ελεύθερες να συνδεθούν και με όποια άλλη συσκευή ad hoc δικτύου είναι μέσα στο εύρος σύνδεσής τους.

Η αποκεντρωμένη φύση των ασύρματων δικτύων ad hoc τα καθιστά κατάλληλα για ποικίλες εφαρμογές (που δεν μπορούν να βασιστούν στους κεντρικούς κόμβους) και μπορεί να βελτιώσει την δυνατότητα κλιμάκωσης των δικτύων σε σύγκριση με τα ασύρματα δίκτυα κεντρικής διαχείρισης, αν και έχουν εντοπιστεί θεωρητικοί και πρακτικοί περιορισμοί στην συνολική χωρητικότητα των εν λόγω δικτύων. Ελάχιστη διαμόρφωση και γρήγορη ανάπτυξη κάνουν τα δίκτυα ad hoc κατάλληλα για καταστάσεις έκτακτης ανάγκης, όπως φυσικές καταστροφές ή πολεμικές συγκρούσεις. Η παρουσία δυναμικών και προσαρμοστικών πρωτοκόλλων δρομολόγησης επιτρέπει στα ad hoc δίκτυα να σχηματιστούν πολύ γρήγορα.

Τα ασύρματα δίκτυα ad hoc μπορούν να ταξινομηθούν περαιτέρω από την εφαρμογή τους σε:

- κινητά δίκτυα ad hoc (mobile ad hoc networks – MANET)
- ασύρματα δίκτυα πλέγματος (wireless mesh networks – WMN)
- ασύρματα δίκτυα αισθητήρων (wireless sensor networks – WSN)

## 1.2 Αντικείμενο διπλωματικής εργασίας

Στα πλαίσια αυτής της διπλωματικής εργασίας, έγινε μία μελέτη διαφορετικών πρωτοκόλλων που σχετίζονται τόσο με την κατανομή και την προώθηση μιας εργασίας στους κόμβους ενός ad hoc συστήματος, όσο και με τις πολιτικές που ακολουθούνται για την αντιμετώπιση μιας ενδεχόμενης απόχωρησης ενός κόμβου-μέλους του συστήματος. Στόχος της εργασίας είναι να εξετάσει τις εναλλακτικές επιλογές που μπορεί να υιοθετήσει ένα σύστημα για να αντιμετωπίσει ζητήματα που προκύπτουν συνεχώς λόγω της φύσης του συστήματος. Η εύρεση ολοένα και πιο αξιόπιστων και αποδοτικών πρωτοκόλλων αποτελεί βασικό συστατικό της περαιτέρω ανάπτυξης τέτοιων συστημάτων και η παρούσα διπλωματική εργασία επιχειρεί να κάνει ένα βήμα προς αυτή την κατεύθυνση.

Έτσι, υλοποιήθηκαν δύο Cloud συστήματα που προσομοιώνουν τις βασικές λειτουργίες ενός ασύρματου ad hoc δικτύου. Οι τοπολογίες των κόμβων στα δύο αυτά συστήματα σχεδιάστηκαν με τέτοιο τρόπο, ώστε να περιλαμβάνουν κάθε είδος σύνδεσης που μπορεί να υπάρξει ανάμεσα σε δύο κόμβους ενός συστήματος. Ένας από τους πιο βασικούς διαχωρισμούς που κάνει η μελέτη είναι η λειτουργία των δύο Cloud συστημάτων. Το ένα σύστημα παρουσιάζει ένα κεντρικοποιημένο μοντέλο επικοινωνίας, όπου ένας κόμβος αναλαμβάνει τα χρέη του διαχειριστή του συστήματος. Το δεύτερο σύστημα έχει έναν πιο αποκεντρικοποιημένο χαρακτήρα, γεγονός που αλλάζει σε σημαντικό βαθμό τόσο την λειτουργικότητά του, όσο και τα πρωτόκολλα που υιοθετεί.

Για τις ανάγκες της διπλωματικής, η μελέτη χωρίστηκε σε στάδια που παρουσιάζονται συνοπτικά παρακάτω:

- μελέτη της γλώσσας προγραμματισμού *Java*
- μελέτη της πολυπρακτορικής πλατφόρμας *JADE*
- μελέτη της γλώσσας *ACL*, που χρησιμοποιούν οι πράκτορες για να επικοινωνήσουν
- σχεδιασμός των δύο Cloud συστημάτων
- υλοποίηση οκτώ διαφορετικών σεναρίων εκτέλεσης και των αντίστοιχων πρωτοκόλλων
- αξιολόγηση της αποδοτικότητας των διαφορετικών πρωτοκόλλων

## 1.3 Οργάνωση κειμένου

Στα κεφάλαια που ακολουθούν θα γίνει αναλυτική περιγραφή των δύο Cloud συστημάτων που υλοποιήθηκαν, καθώς και των πρωτοκόλλων που μελετήθηκαν για τα δύο αυτά συστήματα. Πιο συγκεκριμένα, στο Κεφάλαιο 2 θα γίνει αναφορά στο απαραίτητο θεωρητικό υπόβαθρο που απαιτείται για την κατανόηση της συνολικής εργασίας. Ο σχεδιασμός των δύο Cloud συστημάτων παρουσιάζεται αναλυτικά στο 3ο Κεφάλαιο. Τα εναλλακτικά σενάρια εκτέλεσης και τα αντίστοιχα πρωτόκολλα συζητώνται στο Κεφάλαιο 4. Συνεχίζοντας, το 5ο Κεφάλαιο παρουσιάζει τα συμπεράσματα που εξήχθησαν από την αξιολόγηση της απόδοσης των πρωτοκόλλων. Τέλος στο Κεφάλαιο 6 είναι ο επίλογος, ενώ στο Κεφάλαιο 7 παρουσιάζεται η σχετική βιβλιογραφία που απαιτήθηκε.

# *Κεφάλαιο 2$^{ο}$*

## Θεωρητικό Υπόβαθρο

Για τις ανάγκες της παρούσας διπλωματικής εργασίας, απαιτήθηκαν γνώσεις οι οποίες είχαν καθοριστικό ρόλο στην υλοποίησή της. Οι γνώσεις αυτές χωρίζονται σε δύο κατηγορίες.

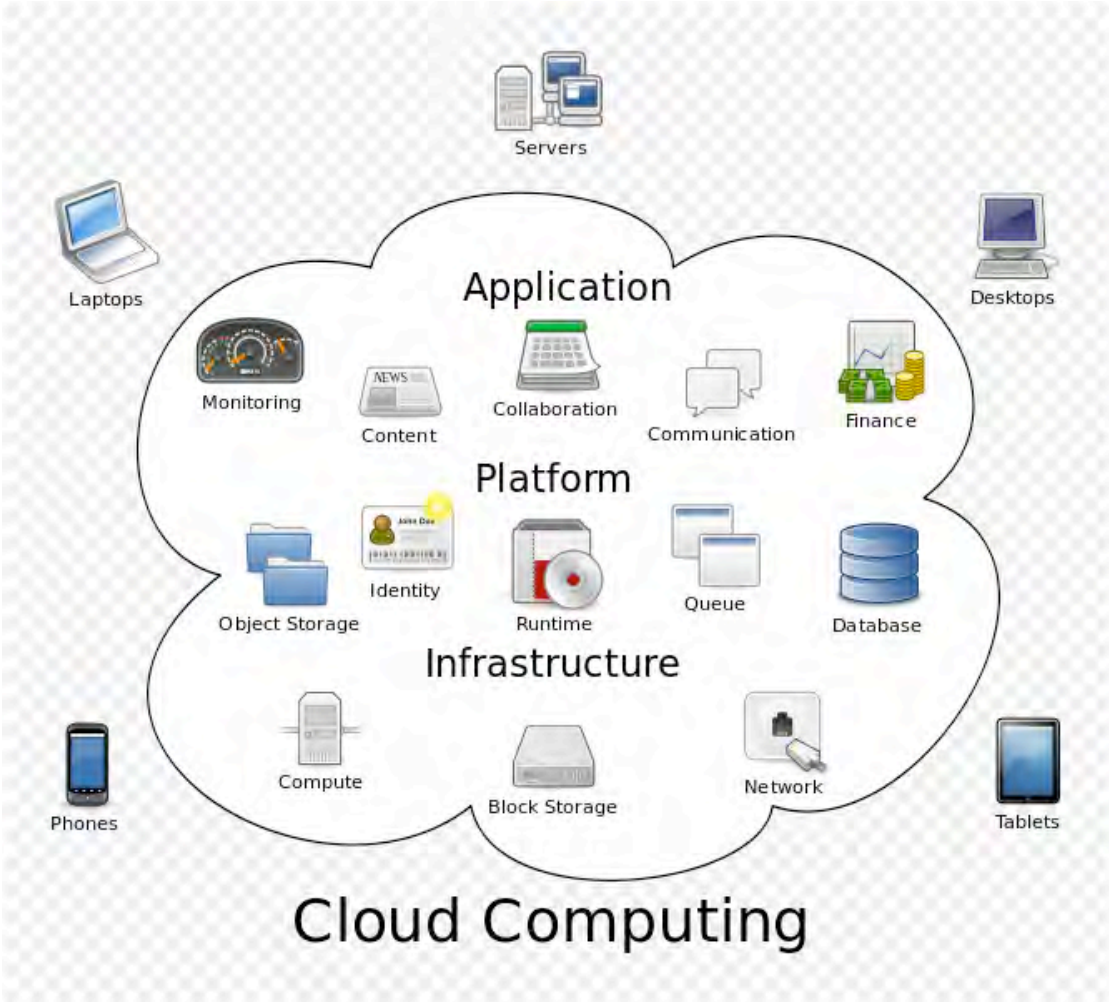
Αρχικά ήταν αναγκαία η θεωρητική μελέτη των λειτουργιών ενός Cloud συστήματος, καθώς και των διάφορων συμπεριφορών των μελών ενός τέτοιου συστήματος. Η απόκτηση των παράπανω γνώσεων ήταν κρίσιμη στον σχεδιασμό των δύο Cloud συστημάτων που υλοποιήθηκαν, όπως επίσης και στην κατασκευή των διαφορετικών πολιτικών που χρησιμοποιήθηκαν για την λήψη κρίσιμων αποφάσεων στα σενάρια εκτέλεσης.

Στο επόμενο στάδιο της εργασίας, οι γνώσεις που απαιτήθηκαν δεν ήταν θεωρητικές αλλά πιο πολύ προγραμματιστικής φύσεως και είχαν να κάνουν με την έννοια του πράκτορα και των πολυπρακτορικών συστημάτων γενικότερα. Επιπλέον, για την ανάγκη της υλοποίησης των Cloud συστημάτων ήταν σημαντική η μελέτη των πρακτόρων της πολυπρακτορικής πλατφόρμας JADE και της γλώσσας επικοινωνίας των πρακτότων ACL.

Στις επόμενες ενότητες ακολουθεί μία ανάλυση των βασικών εννοιών που είναι απαραίτητες για την κατανόηση της συνολικής εργασίας.

## 2.1 Συστήματα Cloud Computing

Το Cloud Computing είναι η χρήση των υπολογιστικών πόρων (υλικού και λογισμικού) που παραδίδονται ως υπηρεσία μέσω δικτύου (συνήθως το Internet). Το όνομα προέρχεται από την χρήση ενός συμβόλου σε σχήμα συννέφου για την αφαιρετική αναπαράσταση της σύνθετης υποδομής που περιέχει ένα τέτοιο σύστημα. Ένα σύστημα Cloud αναθέτει εξ' αποστάσεως υπηρεσίες με τα δεδομένα του χρήστη.




Λογικό Διάγραμμα Cloud Computing

Το Cloud Computing έχει αναδειχθεί ως ένα δημοφιλές μοντέλο υπολογισμού για την υποστήριξη επεξεργασίας μεγάλου όγκου δεδομένων χρησιμοποιώντας συστάδες (clusters) υπολογιστών. Ο στόχος του συγκεκριμένου μοντέλου είναι να βελτιώσει την χρήση των κατανεμημένων πόρων ώστε *συντονισμένα* να χρησιμοποιούνται για την επίτευξη υψηλότερης απόδοσης και να αντιμετωπίζουν υπολογιστικά προβλήματα μεγάλης κλίμακας. Σε ένα σύστημα Cloud Computing εξετάζονται διαφορετικές βασικές αρχές,

όπως το virtualization, την κλιμάκωση (scalability), τη διαλειτουργικότητα (interoperability), την ποιότητα των υπηρεσιών, τους μηχανισμούς ανάνηψης από σφάλματα και τα cloud μοντέλα παράδοσης. Δυστυχώς μέχρι και σήμερα δεν υπάρχει τυπικός ορισμός για το Cloud Computing.

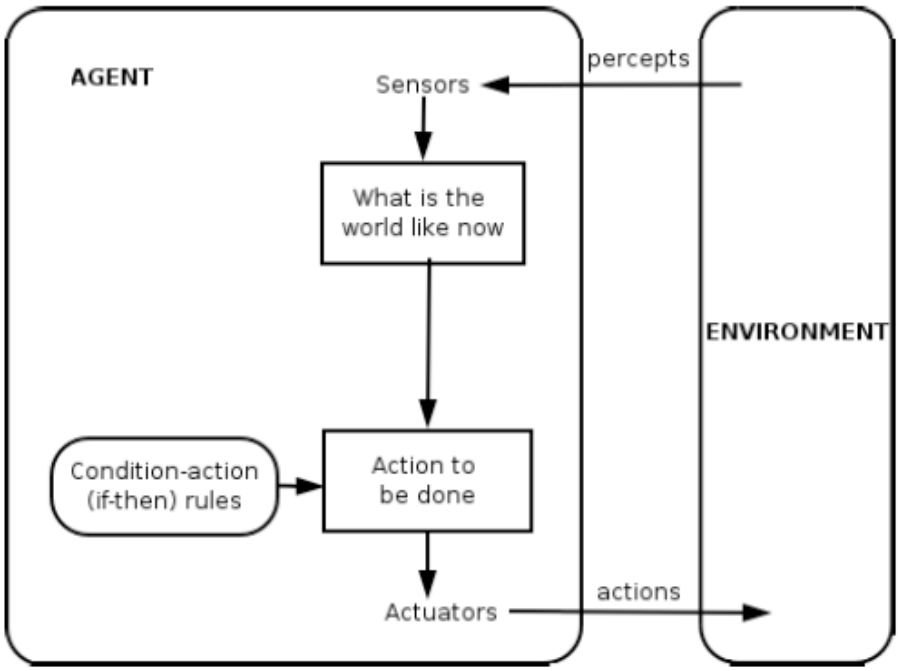
Η ταξινόμηση των Cloud συστημάτων γίνεται με βάση τα εξής κριτήρια:

- αρχιτεκτονική του Cloud
- διαχείρηση του virtualization
- υπηρεσίες
- ανοχή σε σφάλματα
- εξισορρόπηση φορτίου
- διαλειτουργικότητα
- κλιμακούμενη αποθήκευση δεδομένων

Αυτή τη στιγμή υπάρχουν πολλά συστήματα Cloud Computing, όπως τα Amazon EC2, Google App Engine (GAE), Microsoft Azure, Apple iCloud και αρκετά άλλα. Οι χρήστες παρόλα αυτά είναι αρκετά επιφυλακτικοί γιατί τα αρχεία, οι εφαρμογές τους και οι υπολογιστικοί πόροι δεν είναι πλέον στον δικό τους έλεγχο αποκλειστικά.

## 2.2 Ο Πράκτορας Λογισμικού

Στην επιστήμη των υπολογιστών, ένας πράκτορας λογισμικού (software agent) είναι ένα πρόγραμμα υπολογιστή που λειτουργεί για λογαριασμό ενός χρήστη ή ενός άλλου προγράμματος σε μια σχέση αντιπροσώπευσης. Μία τέτοια ‘’δράση για λογαριασμό κάποιου’’ δίνει στον πράκτορα την εξουσία να αποφασίσει ποια δράση, εάν υπάρχει, είναι η πιο κατάλληλη.

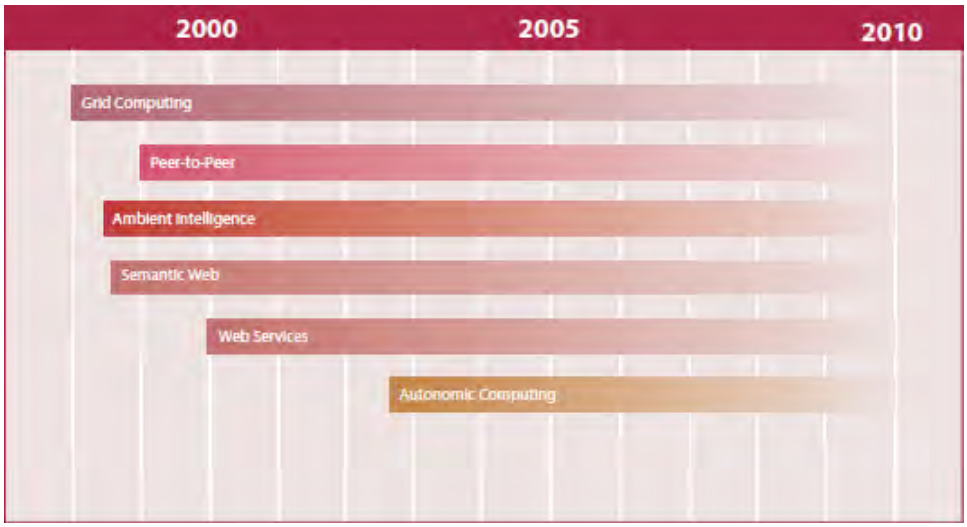


Ο όρος πράκτορας λογισμικού είναι αρκετά γενικός και περιλαμβάνει διάφορους τύπους πρακτόρων όπως οι ευφυείς πράκτορες (χρησιμοποιούνται σε πολλές πτυχές της Τεχνητής Νοημοσύνης όπως η μάθηση και η λογική), οι αυτόνομοι πράκτορες (μπορούν να προσαρμόσουν τον τρόπο με τον οποίο θα πετύχουν τους στόχους τους), οι κατανεμημένοι πράκτορες (που εκτελούνται σε διαφορετικούς υπολογιστές) και οι κινητοί πράκτορες (που μπορούν να μεταφέρουν την εκτέλεσή τους σε διαφορετικούς επεξεργαστές).

Η έννοια του πράκτορα έχει συνδεθεί με ένα ευρύ φάσμα κλάδων της Τεχνολογίας των Πληροφοριών (Information Technology), όπως:

- Computer Networks
- Software Engineering
- Artificial Intelligence
- Human-Computer Interaction
- Distributed and Concurrent Systems
- Mobile Systems
- Control Systems
- Decision Support
- Information Retrieval and Management

Σε γενικές γραμμές, είναι σαφές ότι η τεχνολογική εξέλιξη στον τομέα των κατανεμημένων υπολογισμών ολοένα και περισσότερο αντιμετωπίζει προβλήματα που έχουν διερευνηθεί στο πλαίσιο της ερευνητικής κοινότητας των πρακτόρων. Είναι λογικό να προβάλλονται τα μεγάλα συστήματα από πλευράς των υπηρεσιών που προσφέρουν και κατά συνέπεια, από πλευράς των οντοτήτων ή πρακτόρων που παρέχουν ή καταναλώνουν υπηρεσίες. Οι τομείς αυτοί αντικατοπτρίζουν τις τάσεις και τους "οδηγούς" για τις εφαρμογές στις οποίες συνήθως πολλοί πράκτορες και υπηρεσίες μπορούν να συμμετέχουν και να εξαπλωθούν ευρέως σε γεωγραφικά κατανεμημένο περιβάλλον. Η παρακάτω εικόνα απεικονίζει την εξέλιξη αυτών των τομέων (που επιβάλλουν τη χρήση των τεχνολογιών πράκτορα) διαχρονικά, γεγονός που υποδηλώνει ότι η ωριμότητα τους είναι πιθανό να είναι μερικά χρόνια μακριά.



Εμφάνιση agent-related τομέων με την πάροδο του χρόνου

## 2.3 Η Πλατφόρμα JADE

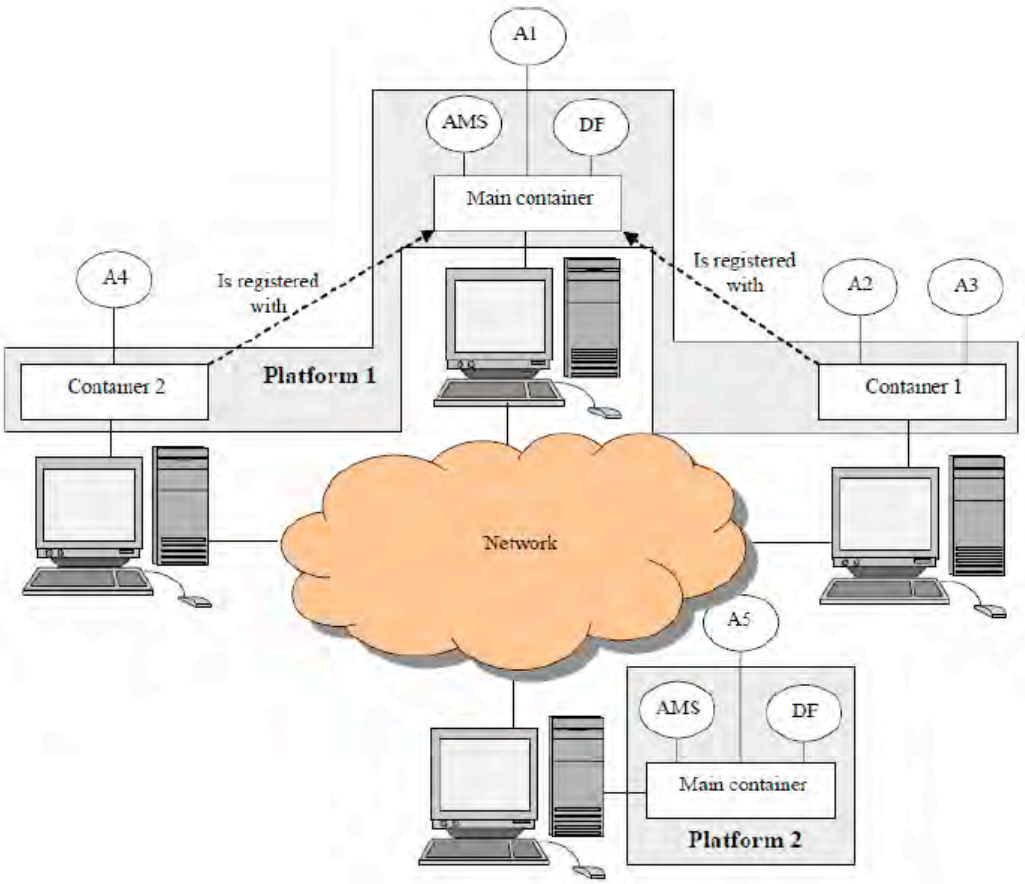
Η υλοποίηση των Cloud συστημάτων, καθώς και των σεναρίων εκτέλεσης έγινε με την χρήση της πολυπρακτορικής πλατφόρμας εκτέλεσης και επικοινωνίας JADE. Με την χρήση των JADE πρακτόρων κατέστη δυνατή η αξιοποίηση επιπλέον λειτουργιών και εργαλείων που κάνουν την χρήση τους ευκολότερη. Παρακάτω παρουσιάζονται τα βασικά χαρακτηριστικά της πλατφόρμας και των πρακτόρων της.

### 2.3.1 Βασικά χαρακτηριστικά της πλατφόρμας *JADE*

Το JADE (Java Agent Development Framework) είναι ένα ελεύθερο λογισμικό που έχει υλοποιηθεί πλήρως στην γλώσσα προγραμματισμού Java. Απλοποιεί την υλοποίηση πολυπρακτορικών συστημάτων μέσω μιας πλατφόρμας εκτέλεσης (middleware) που πληροί τις προδιαγραφές FIPA και μέσω ενός συνόλου εργαλείων που υποστηρίζουν την αποσφαλμάτωση (debugging) και την ανάπτυξη (deploying). Η πλατφόρμα πρακτόρων μπορεί να κατανεμηθεί σε πολλούς υπολογιστές (οι οποίοι δεν χρειάζεται καν να ‘’τρέχουν’’το ίδιο λειτουργικό σύστημα) και η διαμόρφωση μπορεί να ελεγχθεί μέσω ενός απομακρυσμένου GUI. Η διαμόρφωση μπορεί να αλλάξει ακόμα και κατά την διάρκεια της εκτέλεσης με την δημιουργία νέων πρακτόρων ή με την μετακίνηση πρακτόρων από ένα σύστημα σε κάποιο άλλο, όπως και όταν απαιτείται. Η μόνη απαίτηση του συστήματος είναι η έκδοση Java RunTime 5 ή μεταγενέστερη. Η πλατφόρμα εκτέλεσης JADE περιλαμβάνει:

- περιβάλλον εκτέλεσης (runtime environment), όπου ‘’ζουν’’ οι πράκτορες
- βιβλιοθήκη κλάσεων που απαιτούνται για την ανάπτυξη των πρακτόρων
- σουίτα γραφικών εργαλείων (GUI), που επιτρέπει την διαχείρηση και επίβλεψη της λειτουργίας των πρακτόρων

Κάθε στιγμιότυπο εκτέλεσης του περιβάλλοντος JADE λέγεται *Container*, το οποίο μπορεί να περιέχει πολλούς πράκτορες. Το σύνολο από ενεργά containers ονομάζεται Platform. Σε κάθε τέτοιο platform απαιτείται να υπάρχει ένας *Main Container*, όπου όλοι οι άλλοι containers πρέπει να εγγραφούν σε αυτόν για να χρησιμοποιήσουν την πλατφόρμα. Αν δημιουργηθεί ένας άλλος main container αμέσως συνιστά μία άλλη πλατφόρμα. Εκτός από την δυνατότητα αποδοχής καταχωρήσεων από άλλα containers, ο main container διαφέρει από τα υπόλοιπα καθώς περιλαμβάνει και δύο ειδικούς πράκτορες (AMS και DF) οι οποίοι ξεκινούν την λειτουργία τους αυτόματα μόλις ξεκινήσει και ο main container. Αυτοί οι δύο ειδικοί πράκτορες παρουσιάζονται πιο αναλυτικά σε παρακάτω ενότητα.



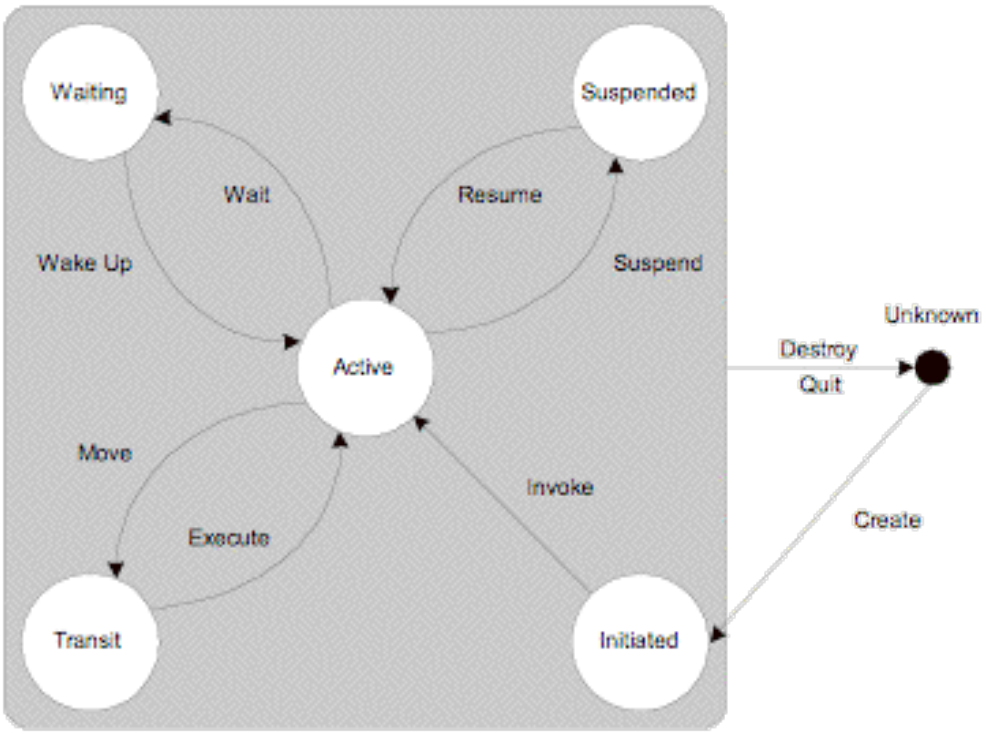
Platforms και Containers

### 2.3.2 Η κλάση *agent*

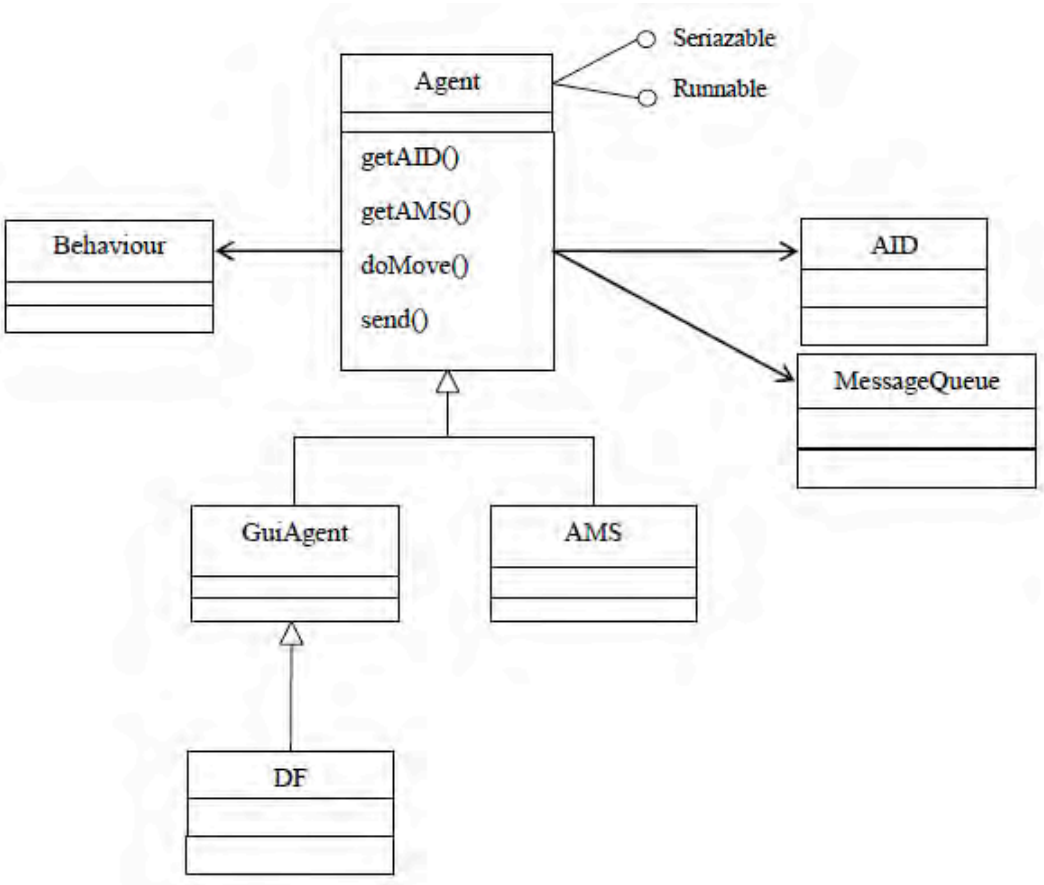
Η κλάση agent που προσφέρεται από την βιβλιοθήκη κλάσεων του περιβάλλοντος JADE χρησιμοποιείται κατά κόρον στην υλοποίηση των Cloud συστημάτων της εργασίας και περιγράφει την λειτουργικότητα ενός πράκτορα. Κατά την δημιουργία ενός αντικειμένου της κλάσης, ένα σύνολο από κατασκευαστές (constructors) προσφέρουν πολλές δυνατότητες, όπως

- η περιγραφή ενός πράκτορα
- ο ορισμός ή η εύρεση της διεύθυνσής του
- ο ορισμός ή η εύρεση του ονόματός του
- ο ορισμός ή η εύρεση του πρωτοκόλλου επικοινωνίας του με άλλους πράκτορες

Η δομή της κλάσης και ο κύκλος ζωής ενός πράκτορα περιγράφονται από τα παρακάτω διαγράμματα.




FIPA Agent life cycle




Agent Class diagram

Κάθε πράκτορας που δημιουργείται και εισέρχεται στην πλατφόρμα εκτέλεσης λαμβάνει ένα μοναδικό αναγνωριστικό (AID) και με αυτό το αναγνωριστικό μπορεί να δέχεται και να στέλνει μηνύματα, ανεξάρτητα από την τοποθεσία του.

Όπως αναφέρθηκε σε προηγούμενη ενότητα, στην πλατφόρμα JADE υπάρχουν δύο ειδικοί πράκτορες που περιλαμβάνονται στον main container κάθε πλατφορμας:

1. Ο ***AMS*** *(Agent Management System)* παρέχει μία ''υπηρεσία ονόματος'' (naming service), η οποία εξασφαλίζει ότι κάθε πράκτορας της πλατφόρμας έχει μοναδικό αναγνωριστικό. Επίσης ο ειδικός αυτός πράκτορας ασκεί την εξουσία της πλατφόρμας. Μπορεί δηλαδή να δημιουργεί και να τερματίζει τη λειτουργία των υπόλοιπων πρακτόρων.
2. Ο ***DF*** *(Directory Facilitator)* παρέχει μία υπηρεσία ''χρυσού οδηγού'' (yellow pages), μέσω της οποίας ένας πράκτορας μπορεί να εντοπίσει άλλους πράκτορες που παρέχουν υπηρεσίες για την επίτευξη των στόχων του.

### 2.3.3 Η κλάση behaviour

Η συμπεριφορά ενός πράκτορα είναι μία αφηρημένη κλάση που προσφέρει η βιβλιοθήκη κλάσεων του περιβάλλοντος JADE και χρησιμοποιείται για την μοντελοποίηση των εργασιών που εκτελεί ένας πράκτορας. Με αυτό τον τρόπο, ένα σύνολο συμπεριφορών προγραμματίζονται και εκτελούνται ανάλογα με τις ανάγκες και τις υποχρεώσεις του κάθε πράκτορα. Οι πράκτορες της κλάσης agent που παρουσιάστηκε παραπάνω έχουν την δυνατότητα να προσθέσουν ή και να αφαιρέσουν συμπεριφορές καλώντας δύο συγκεκριμένες μεθόδους, τις *addBehaviour(Behaviour)* και *removeBehaviour(Behaviour)*. Ο πράκτορας μπορεί να εκτελεί πολλές εργασίες ανταποκρινόμενος σε διαφορετικά εξωτερικά ερεθίσματα παράλληλα. Η κλάση behaviour δίνει συνεπώς την δυνατότητα στον πράκτορα να ανταποκρίνεται κατάλληλα στα διάφορα ερεθίσματα που δέχεται. Η εκτέλεση των συμπεριφορών δεν γίνεται παράλληλα, αλλά διαδοχικά. Κάθε φόρα που ολοκληρώνεται ένας κύκλος εκτέλεσης μιας συμπεριφοράς συνεχίζει η επόμενη. Το τέλος του κύκλου εκτέλεσης καθορίζεται από τον χρήστη μέσα από συγκεκριμένη μέθοδο. Συγκεκριμένα, η κλάση behaviour προσφέρει μεθόδους για διάφορες λειτουγίες όπως:

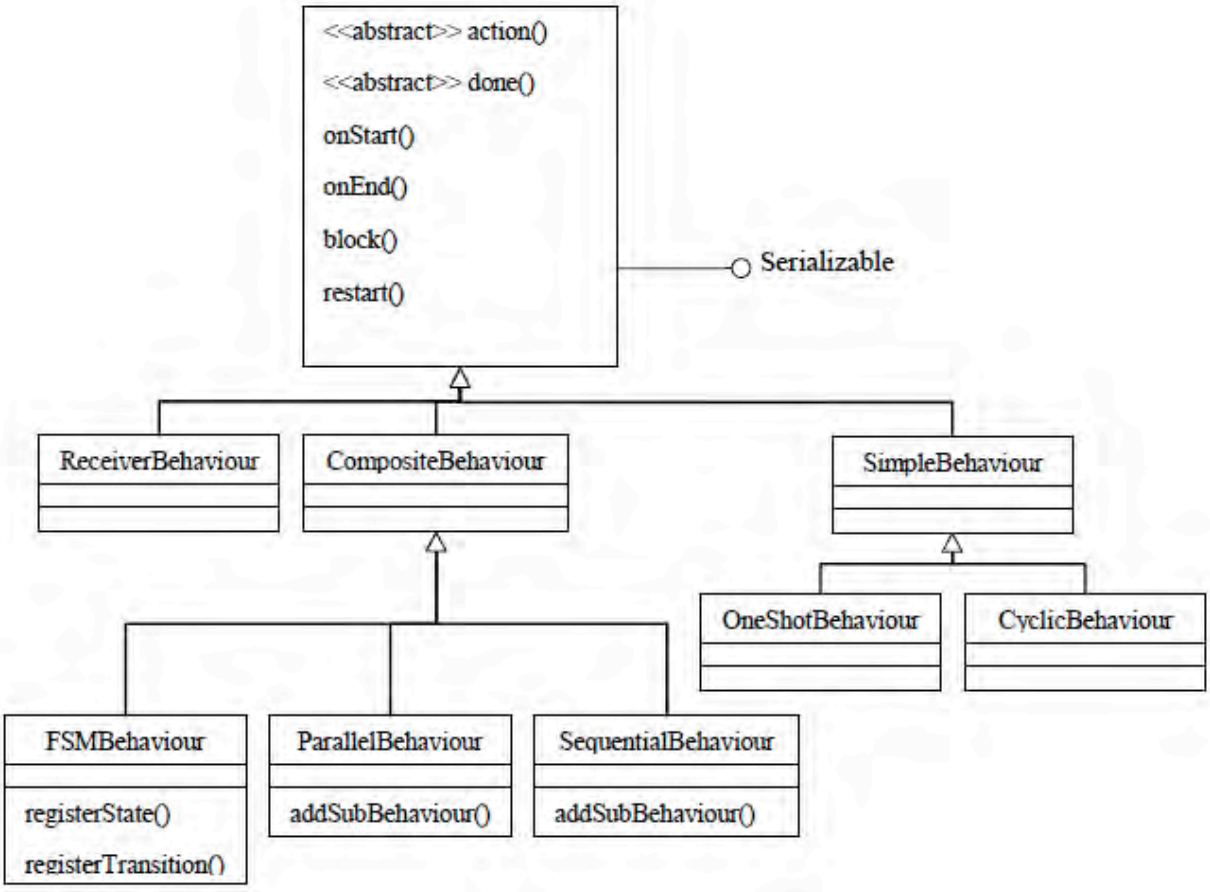
- η **action()**, για την ενεργοποίηση της συμπεριφοράς
- η **done()**, για την ολοκλήρωση της συμπεριφοράς και
- η **restart()**, για την επανεκκίνηση της συμπεριφορας.

Η βιβλιοθήκη του περιβάλλοντος JADE προσφέρει διάφορες κατηγορίες behaviour για την διευκόλυνση του προγραμματιστή και οι οποίες τονίζονται παρακάτω.

- **SimpleBehaviour**: πρόκειται για την πιο απλή συμπεριφορά, η οποία εκτελείται είτε μία φορά (*OneShotBehaviour*) είτε περιοδικά (*CyclicBehaviour*).

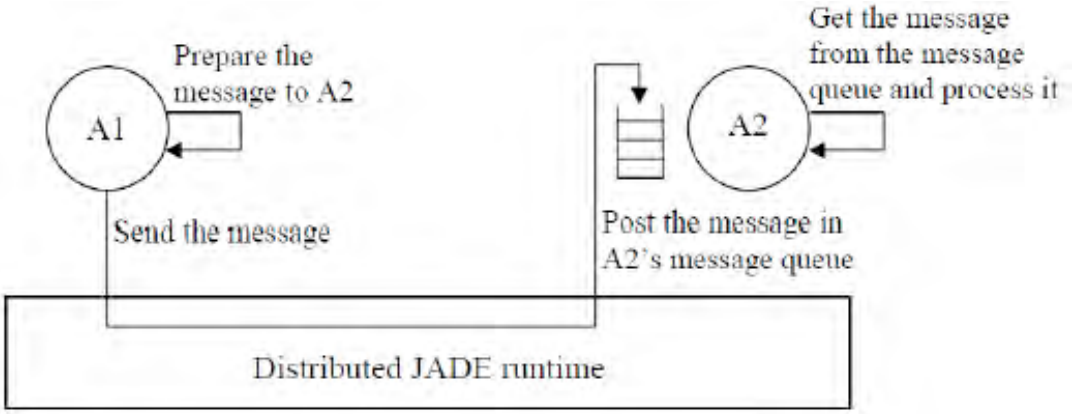
- **ReceiverBehaviour**: είναι η συμπεριφορά που εκτελείται κάθε φορά που έρχεται ένα μήνυμα στην ουρά του πράκτορα και ταιριάζει σε ένα δεδομένο Message Template (που έχει οριστεί κατά την αρχικοποίηση της συμπεριφοράς).
- **CompositeBehaviour**: αποτελεί την πιο σύνθετη συμπεριφορά, η οποία αποτελείται από πολλές υπο-συμπεριφορές που εκτελούνται ανάλογα με τις ανάγκες του πράκτορα.

Στο παρακάτω διάγραμμα παρουσιάζεται η δομή της κλάσης behaviour.



### 2.3.4 Επικοινωνία μεταξύ πρακτόρων

Ένα από τα σημαντικότερα χαρακτηριστικά του περιβάλλοντος JADE είναι η επικοινωνία μεταξύ πρακτόρων. Ο τρόπος επικοινωνίας που χρησιμοποιείται είναι η ασύγχρονη ανταλλαγή μηνυμάτων.




Ασύγχρονη επικοινωνία στην πλατφόρμα JADE

Η δομή της επικοινωνίας προσφέρει ευέλικτη και αποτελεσματική ανταλλαγή μηνυμάτων, αφού το περιβάλλον JADE δημιουργεί και διαχειρίζεται μία ουρα εισερχόμενων ACL μηνυμάτων, ιδιωτική για κάθε πράκτορα. Οι πράκτορες με τη σειρά τους μπορούν να έχουν πρόσβαση στην ουρά τους μέσω συνδυασμού διαφόρων τρόπων, όπως blocking, polling, timeout και pattern matching. Το πλήρες μοντέλο επικοινωνίας FIPA (Foundation for Intelligent Physical Agents) έχει υπολοιηθεί και τα συστατικά του έχουν ενσωματωθεί πλήρως.

### 2.3.5 Η γλώσσα επικοινωνίας *ACL*

Τα μηνύματα που ανταλλάσσονται μεταξύ των πρακτόρων στο περιβάλλον JADE ακολουθούν την τυποποίηση της γλώσσας ACL (που έχει καθοριστεί από την FIPA). Η τυποποίηση αυτή καθορίζει τα παρακάτω πεδία:

- τον *αποστολέα* του μηνύματος,
- την λίστα με τους *παραλήπτες*,
- την *πρόθεση* της επικοινωνίας, δηλαδή τον λόγο για τον οποίο επιδιώκεται η επικοινωνία (Request, Inform, Query If, Call for proposals),
- το *περιεχόμενο*,
- την *γλώσσα περιεχομένου*,
- την *οντολογία*, που αναφέρεται στην έννοια των συμβόλων που χρησιμοποιούνται από τους αποστολείς και τους παραλήπτες,
- *πεδία* που ρυθμίζουν επιμέρους συζητήσεις που πραγματοποιούνται ταυτόχρονα (conversation-id, reply-with, in-reply-to κ.α.).

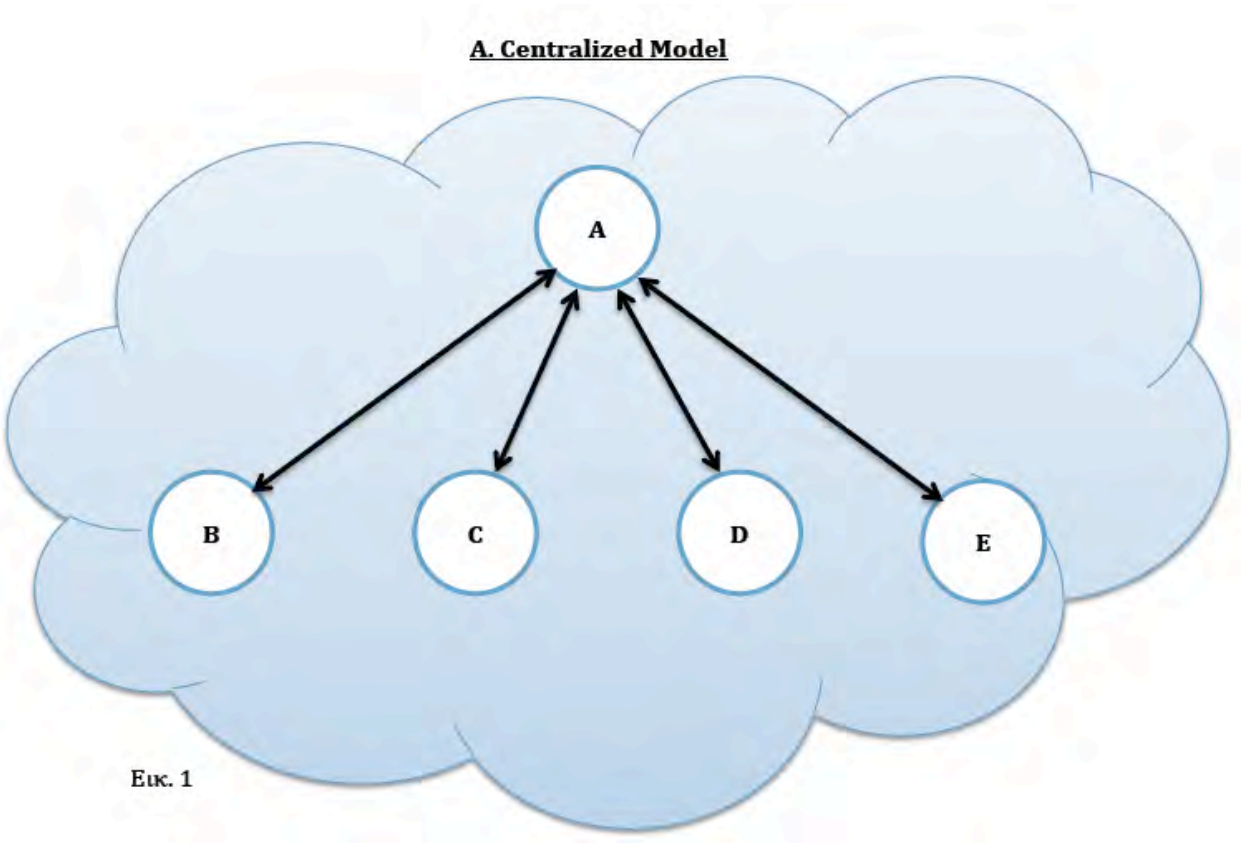
# *Κεφάλαιο 3ο*

## Σχεδιασμός Συστήματος

Η υλοποίηση της πειραματικής εφαρμογής έγινε στην γλώσσα προγραμματισμού Java και στηρίχθηκε εξ' ολοκλήρου πάνω στην πολυπρακτορική πλατφόρμα JADE. Η εφαρμογή επικεντρώνεται στους μηχανισμούς αποχώρησης ενός μέλους του Cloud συστήματος κατά τη διάρκεια που το σύστημα έχει αναλάβει μία εργασία σαν ομάδα. Η εργασία αυτή χωρίζεται σε δεκαπέντε (15) υποεργασίες, κατανέμεται στα μέλη της ομάδας με τυχαίο τρόπο και εκτελείται σειριακά. Σε αυτό το κεφάλαιο θα γίνει η παρουσίαση των επιμέρους αρχείων που απαρτίζουν την εφαρμογή.

### 3.1 Αρχιτεκτονική

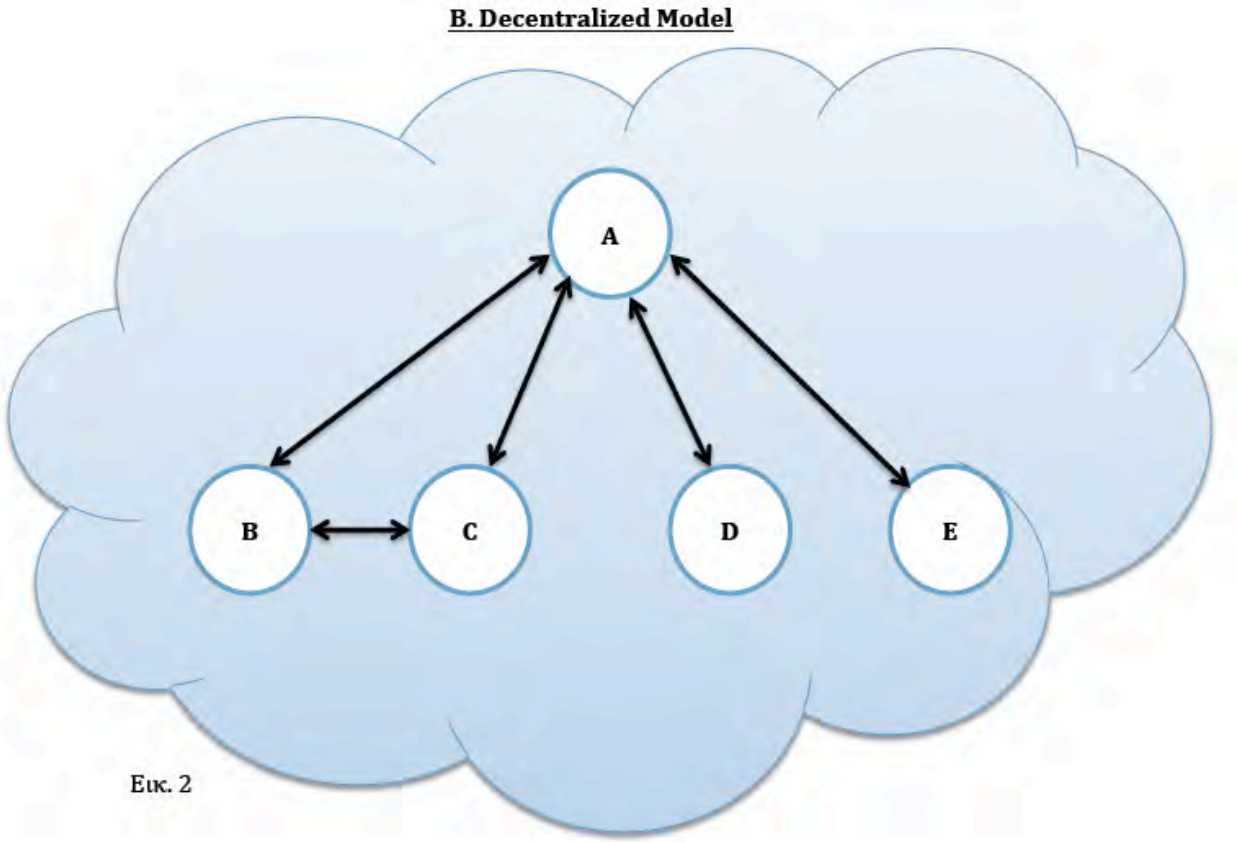
Για τις ανάγκες της μελέτης κατασκευάστηκαν δύο ξεχωριστά Cloud συστήματα πρακτόρων που παρουσιάζονται στις παρακάτω εικόνες.




Εικ. 1

Σε αυτό το σύστημα, το μέλος Α έχει την ιδιότητα του διαχειριστή και είναι υπεύθυνος για την κατανομή των υποεργασιών και την διαμεσολάβηση στην επικοινωνία μεταξύ των υπόλοιπων μελών του συστήματος. Πρόκειται

συνεπώς για ένα κεντρικοποιημένο μοντέλο συστήματος, όπου όλα είναι σε άμεση εξάρτηση με τον administrator.



Εικ. 2

Η επικοινωνία στο παραπάνω σύστημα είναι αποκεντρικοποιημένη και σχεδιασμένη με τέτοιο τρόπο, ώστε να συμπεριλαμβάνονται όλα τα πιθανά σενάρια επικοινωνίας μεταξύ των μελών του Cloud συστήματος. Έτσι κάποια από τα μέλη έχουν την δυνατότητα να επικοινωνούν απευθείας χωρίς την βοήθεια του μέλους Α (που διατηρεί παρόλα αυτά την αποκλειστική ευθύνη κατανομής της εργασίας στα μέλη του συστήματος).

Και στα δύο αυτά συστήματα, η εργασία που χωρίζεται σε κομμάτια και κατανέμεται στα μέλη της ομάδας αποτελεί ένα αφαιρετικό μηχανισμό που χρησιμεύει στην εξαγωξή συμπερασμάτων και δεν είναι κάποια συγκεκριμένη δουλειά που αναλαμβάνει το σύστημα. Είναι ένα γενικό μοντέλο κατανεμημένης εργασίας που θα μπορούσε να προσομοιώσει ένα μεγάλο εύρος εργασιών. Επίσης, η αποχώρηση ενός μέλους του συστήματος γίνεται με βάση το μοντέλο Fail-Stop. Σύμφωνα με αυτό, η αποχώρηση ενός μέλους από το σύστημα γίνεται σίγουρα γνωστή σε κάποιο άλλο μέλος και με αυτό τον τρόπο αποφεύγεται το ενδεχόμενο της αστοχίας ή της μη ολοκλήρωσης της εργασίας.

## 3.2 Περιγραφή αρχείων κεντρικοποιημένου μοντέλου

Για την λειτουργία του Centralized Cloud συστήματος υλοποιήθηκαν οι εξής πέντε τύποι πρακτόρων:

- AgentA
- AgentB
- AgentC
- AgentD
- AgentE

Ο κάθε ένας από αυτούς αντιπροσωπεύει το αντίστοιχο μέλος του συστήματος.

### 3.2.1 Η κλάση *AgentA*

Η κλάση αυτή επεκτείνει την κλάση Agent της πλατφόρμας JADE και αποτελείται από μεθόδους που υλοποιούν τις απαραίτητες λειτουργίες που απαιτούνται για τον διαχειριστή του συστήματος (κατανομή υποεργασιών, διαμεσολάβιση στην επικοινωνία των μελών).

#### **3.2.1.1** Η μέθοδος ***setup***

Η βασική μέθοδος κάθε κλάσης που επεκτείνει την κλάση Agent χρησιμεύει για την υλοποίηση βασικών λειτουργιών. Πιο συγκεκριμένα, με τη μέθοδο αυτή ο πράκτορας Α:

- Εισέρχεται στο σύστημα και λαμβάνει ένα μοναδικό αναγνωριστικό AID με το οποίο μπορούν οι υπόλοιποι πράκτορες να τον εντοπίσουν
- Εγγράφεται στην πλατφόρμα μέσω του DF Agent και καταχωρεί τις υπηρεσίες που αναλαμβάνει
- Προσθέτει τις δύο συμπεριφορές που υλοποιούν τις λειτουργίες του διαχειριστή.

#### **3.2.1.2** Η μέθοδος ***takeDown***

Η μέθοδος αυτή χρησιμοποιείται από τον πράκτορα Α κατά την αποχώρηση του από το σύστημα. Αρχικά απεγγράφεται από την πλατφόρμα πάλι μέσω του ειδικού πράκτορα DF Agent και στην συνέχεια εκπέμπει ένα μήνυμα αποχώρησης και φεύγει από το σύστημα.

#### **3.2.1.3** Η συμπεριφορά ***AgentFinder***

Με την χρήση αυτής της συμπεριφοράς ο πράκτορας Α ανακαλύπτει τα υπόλοιπα μέλη που απαρτίζουν το Cloud σύστημα. Αφού περάσει ένα χρονικό διάστημα κατά το οποίο μπορούν τα μέλη που το επιθυμούν να εγγραφουν στο σύστημα, ο διαχειριστής αναζητά για μέλη και ανανεώνει την λίστα των γνωστών μελών του συστήματος. Στη συνέχεια χωρίζει την εργασία που έχει ανατεθεί στο σύστημα σε δεκαπέντε υποεργασίες και *με τυχαίο τρόπο* κατανέμει τις υποεργασίες στα μέλη της λίστας. Σε περίπτωση που κανένας πράκτορας δεν έχει εκδηλώσει ενδιαφέρον να συμμετέχει στο σύστημα και άρα η λίστα

μελών παραμείνει κενή, τότε ο πράκτορας Α καλεί την μέθοδο takeDown και τερματίζει.

#### **3.2.1.4** Η συμπεριφορά ***RequestPperformer***

Η συμπεριφορά αυτή εκτελείται μέχρι να ολοκληρωθεί όλη η εργασία που έχει αναλάβει το σύστημα. Ο πράκτορας Α χρησιμοποιεί αυτή την συμπεριφορά για να επικοινωνεί με τα υπόλοιπα μέλη του συστήματος ή για να προωθεί αιτήσεις μεταξύ των μελών (αφού δεν υπάρχει απευθείας επικοινωνία μεταξύ τους). Η επικοινωνία γίνεται με την χρήση ACL μηνυμάτων που υποστηρίζονται από την πλατφόρμα JADE. Πιο συγκεκριμένα, ο διαχειριστής του συστήματος ειδοποιεί το μέλος που τυχαία ανέλαβε την πρώτη υποεργασία και περίμένει από αυτό ένα μήνυμα ολοκλήρωσης της συγκεκριμένης υποεργασίας. Στη συνέχεια αποστέλλει ένα αντίστοιχο μήνυμα ειδοποίησης στο μέλος που είναι υπεύθυνο για την δεύτερη υποεργασία, περίμένει και πάλι ένα μήνυμα επιβεβαίωσης και η διαδικασία συνεχίζεται μέχρι να ολοκληρωθεί όλη η εργασία. Ο administrator στέλνει τα μηνύματα με πανεκπομπή (broadcast) και κάθε μέλος είναι σε θέση να αναγνωρίσει ποιο από τα μηνύματα αφορά το ίδιο, αγνοώντας τα υπόλοιπα μηνύματα.

### **3.2.2 Η κλάση *AgentB***

Όμοια με την κλάση του διαχειριστή, αυτή η κλάση επεκτείνει την κλάση Agent της πλατφόρμας JADE και προσομοιώνει τις βασικές λειτουργίες ενός απλού μέλους του Cloud συστήματος που φέρνει εις πέρας τις υποεργασίες που του αναθέτει ο διαχειριστής. Η κλάση αποτελείται από τις παρακάτω μεθόδους και συμπεριφορές.

#### **3.2.2.1** Η μέθοδος ***setup***

Σε αντιστοιχία με την ανάλογη μέθοδο της κλάσης AgentA, αυτή η μέθοδος υλοποιεί τις αρχικές βασικές λειτουργίες που είναι απαραίτητες. Εγγραφει δηλαδή τον πράκτορα Β στην πλατφόρμα με την βοήθεια του DF Agent και προσθέτει την συμπεριφορά που υλοποιεί την βασική λειτουργικότητα του πράκτορα.

#### **3.2.2.2** Η μέθοδος ***takeDown***

Με τη μέθοδο αυτή ο πράκτορας απεγγράφεται από τις υπηρεσίες yellow pages που προσφέρει ο DF Agent, εκπέμπει ένα μήνυμα αποχώρησης και στη συνέχεια αποχωρεί από το σύστημα.

#### **3.2.2.3** Η συμπεριφορά ***FunctionPerformer***

Η συμπεριφορά αυτή είναι κυκλική. Αυτό πρακτικά σημαίνει ότι εκτελείται συνεχώς μέχρι ο πράκτορας να τερματίσει την λειτουργεία του. Ο πράκτορας Β περιμένει μέχρι κάποιο μήνυμα από τον διαχειριστή να φτάσει και στην συνέχεια αποφασίζει αν το μήνυμα τον αφορά ή όχι. Στην δεύτερη περίπτωση, αν το μήνυμα προορίζεται για κάποιο άλλο μέλος του συστήματος,

τότε το αγνοεί και συνεχίζει όπως πριν να περιμένει για το επόμενο μήνυμα του διαχειριστή. Όταν ένα μήνυμα που τον αφορά φτάσει, ο πράκτορας το διαβάζει και εκτελεί την υποεργασία που συμπεριλαμβάνεται στο μήνυμα. Μόλις ολοκληρώσει με επιτυχία την υποεργασία έχει την υποχρέωση να ενημερώσει τον administrator για την ομαλή περάτωσή της. Έπειτα επιστρέφει στην αρχική κατάσταση αναμονής για νέο μήνυμα από τον διαχειριστή.

#### **3.2.2.4** Η κλάση ***MatchBOntology***

Πρόκειται για μια private κλάση η οποία υλοποιεί (implements) την διεπαφή *MessageTemplate.MatchExpression* της κλάσης *MessageTemplate*. Αυτή η κλάση χρησιμοποιείται από τον πράκτορα Β για να έχει την δυνατότητα να ξεχωρίζει τα μήνυματα που τον αφορούν από το σύνολο των μηνυμάτων που κάνει broadcast ο διαχειριστής. Ο πράκτορας Β καλεί αυτή την κλάση, κατασκευάζει ένα πρότυπο αναγνώρισης μηνυμάτων που τον αφορούν και χρησιμοποιεί αυτό το πρότυπο για την αναγνώριση των μηνυμάτων.

### **3.2.3 Οι κλάσεις *AgentC*, *AgentD* και *AgentE***

Σε απόλυτη αναλογία με τον τρόπο κατασκευής της κλάσης AgentB, αυτές οι τρεις κλάσεις χρησιμοποιούν τις ίδιες μεθόδους (setup, takeDown) και συμπεριφορές (FunctionPerformer) και υλοποιούν την ίδια ακριβώς λειτουργικότητα. Περιέχουν επίσης η καθεμία ξεχωριστά μία private κλάση κατασκευής προτύπου μηνυμάτων (MatchXOntology), η οποία λειτουργεί όπως ακριβώς περιγράφτηκε παραπάνω για τον πράκτορα Β.

## **3.3 Περιγραφή αρχείων αποκεντρικοποιημένου μοντέλου**

Όμοια με το κεντρικοποιημένο μοντέλο, στο Decentralized Cloud σύστημα υλοποιήθηκαν οι ίδιες πέντε κλάσεις πρακτόρων. Η βασική διαφορά αυτού του συστήματος από το προηγούμενο είναι η προσθήκη της δυνατότητας επικοινωνίας μεταξύ των μελών όπως αυτή παρουσιάζεται στην Εικόνα 2 παραπάνω. Με τη δυνατότητα της επικοινωνίας μεταξύ των μελών, ο διαχειριστής απαλλάσσεται από ένα μέρος του φόρτου επικοινωνίας που ήταν υποχρεωμένος να εξυπηρετεί όταν τα υπόλοιπα μέλη του συστήματος δεν είχαν τρόπο άμεσης επικοινωνίας. Πλέον όταν ο πράκτορας Β εκτελεί μία υποεργασία και η επόμενη υποεργασία πρέπει να προωθηθεί στον πράκτορα C, δεν είναι αναγκαία η παρέμβαση του διαχειριστή για την προώθηση αλλά ο πράκτορας Β το κάνει αυτόματα μόλις ολοκληρώσει την δική του υποεργασία. Η ίδια διαδικασία μπορεί να συμβεί και αντίστροφα από τον πράκτορα C στον πράκτορα Β.

### **3.3.1 Η κλάση *Task***

Η κλάση αυτή υλοποιεί την διεπαφή *java.io.Serializable* και σκοπός της είναι να δώσει την δυνατότητα στους πράκτορες-μέλη του συστήματος να κατασκευάσουν ενα αντικείμενο (object) τύπου Task. Αυτό το αντικείμενο χρησιμοποιείται στη συνέχεια για την άμεση προώθηση των υποεργασιών και

των υπολοίπων στοιχείων που απαιτούνται ανάμεσα στα μέλη χωρίς την παρεμβολή του διαχειριστή. Από τη στιγμή που η κλάση αυτή στην ουσία δημιουργεί ένα νέο τύπο αντικειμένου (object type) Task, η δομή της περιλαμβάνει και τους αντίστοιχους κατασκευαστές-constructors.

### 3.3.2 Η κλάση *AgentA*

Ο πράκτορας Α, που υλοποιείται από αυτή την κλάση, εκτελεί και πάλι χρέη διαχειριστή αλλά οι αρμοδιότητες του σε σχέση με το κεντρικοποιημένο μοντέλο είναι εμφανώς περιορισμένες. Έχει και πάλι την ευθυνη καταμερισμού της συνολικης εργασίας στα μέλη του συστήματος, αλλά ο τρόπος επικοινωνίας και προώθησης των υποεργασιών πλέον μοιράζεται σε όλα τα μέλη ανάλογα με την δυνατότητα άμεσης επικοινωνίας που έχουν μεταξύ τους.

#### **3.3.2.1** Η μέθοδος ***setup***

Η μέθοδος αυτή, όπως και σε κάθε άλλη κλάση που υλοποιεί πράκτορα, χρησιμοποιείται για τις βασικές αρχικές λειτουργίες που απαιτούνται κατά την έναρξη λειτουργίας ενός πράκτορα. Εκτός όμως από τις ήδη γνωστές λειτουργίες του διαχειριστή (εγγραφή στην πλατφόρμα, απόκτηση αναγνωριστικού AID, αναζήτηση άλλων πρακτόρων-μελών, τυχαία κατανομή των υποεργασιών), ο πράκτορας Α εδώ κατασκευάζει ένα δικό του αντικείμενο τύπου Task, στο οποίο εισάγει πληροφορίες όπως η συσχέτιση κάθε υποεργασίας με κάποιον πράκτορα-μέλος και ο αριθμός των υποεργασιών που έχουν ολοκληρωθεί. Στην συνέχεια προσθέτει τις τρεις βασικές συμπεριφορές που υλοποιούν τις λειτουργίες που του αναλογούν.

#### **3.3.2.2** Η μέθοδος ***takeDown***

Σε αυτή τη μέθοδο εκτελούνται όλες οι διαδικασίες που περιγράφτηκαν παραπάνω (απεγγραφή, μήνυμα ειδοποίησης, αποχώρηση) και αφορούν την αποχώρηση του πράκτορα Α από το σύστημα.

#### **3.3.2.3** Η συμπεριφορά ***AgentFinder***

Η λειτουργικότητα αυτής της OneShot συμπεριφοράς είναι ταυτόσημη με την λειτουργικότητα της αντίστοιχης συμπεριφοράς του πράκτορα Α στο κεντρικοποιημένο σύστημα. Χρησιμοποιείται δηλαδή για τον εντοπισμό των άλλων μελών του συστήματος, την δημιουργία της λίστας μελών και τον τυχαίο καταμερισμό της συνολικής εργασίας με βάση αυτή τη λίστα.

#### **3.3.2.4** Η συμπεριφορά ***InitialRequestPerformer***

Η συμπεριφορά αυτή είναι μίας εκτέλεσης (OneShot Behaviour) και εκτελείται από τον διαχειριστή μόλις κατανείμει τις επιμέρους εργασίες στα μέλη του συστήματος. Ο σκοπός της είναι να προωθήσει την πρώτη υποεργασία στον πράκτορα-μέλος που είναι υπεύθυνος για την ολοκλήρωσή της. Η ειδοποίηση του μέλους δεν γίνεται με πολυεκπομπή, όπως στο κεντρικοποιημένο μοντέλο, αλλά με την αποστολή ACL μηνύματος απευθείας

στον αντίστοιχο πράκτορα. Στο συγκεκριμένο μήνυμα επισυνάπτεται το αντικείμενο τύπου Task που έχει κατασκευάσει νωρίτερα ο πράκτορας Α. Στη συνέχεια η συμπεριφορά τερματίζει, καθώς η περαιτέρω επικοινωνία μεταξύ των πρακτόρων και η προώθηση υποεργασιών είναι μία διαδικασία που αφορά όλα τα μέλη και όχι μόνο τον διαχειριστή πράκτορα Α. Συνεπώς χρειάζεται για αυτή τη λειτουργία μία ξεχωριστή συμπεριφορά, που περιγράφεται παρακάτω.

#### **3.3.2.5** Η συμπεριφορα ***ForwardPerformer***

Η συμπεριφορά αυτή είναι κυκλική και χρησιμοποιείται από τον πράκτορα Α καθ' όλη την διάρκεια επεξεργασίας της συνολικής εργασίας του συστήματος. Πιο συγκεκριμένα, ο πράκτορας Α αμέσως μόλις προωθήσει την πρώτη υποεργασία στο αρμόδιο μέλος, περιμένει μέχρι κάποιο μήνυμα να φτάσει από κάποιο άλλο μέλος. Το συγκεκριμένο ACL μήνυμα, το οποίο είναι τύπου REQUEST και περιέχει ένα αντικείμενο τύπου Task με όλες τις απαραίτητες πληροφορίες, φτάνει στον διαχειριστή όταν κάποιο μέλος του συστήματος έχει ολοκληρώσει μία υποεργασία αλλά αδυνατεί (λόγω της συγκεκριμένης πολιτικής επικοινωνίας του συστήματος) να προωθήσει μόνο του την επόμενη υποεργασία στον αντίστοιχο πράκτορα. Το μέλος δηλαδή στέλνει το αίτημα στον διαχειριστή (που έχει την δυνατότητα να επικοινωνεί με όλα τα μέλη) και αυτός με τη σειρά του προωθεί το μήνυμα στον ενδιαφερόμενο πράκτορα. Ο πράκτορας Α εμπλέκεται σε αυτή τη διαδικασία μόνο στην περίπτωση που δύο πράκτορες-μέλη δεν έχουν άμεση επικοινωνία μεταξύ τους.

### **3.3.3 Η κλάση *AgentB***

Η κλάση που υλοποιεί τον πράκτορα Β είναι σε πολλά σημεία όμοια με εκείνη που υλοποιεί τον πράκτορα Β στο κεντρικοποιημένο μοντέλο. Υπαρχουν όμως μερικές βασικές διαφορές που έχουν να κάνουν με το κομμάτι της επικοινωνίας που εισήχθη σε αυτό το αποκεντρικοποιημένο σύστημα.

#### **3.3.3.1** Η μέθοδος ***setup***

Όπως και στην αντίστοιχη μέθοδο του centralized συστήματος, αυτή η μέθοδος υλοποιεί τις ίδιες βασικές λειτουγίες του πράκτορα και προσθέτει την συμπεριφορά που χειρίζεται την λειτουργικότητά του.

#### **3.3.3.2** Η μέθοδος ***takeDown***

Υλοποιεί τις λειτουργίες αποχώρησης του πράκτορα από το σύστημα, ακριβώς όπως αυτές έχουν περιγραφεί πιο πάνω.

#### **3.3.3.3** Η συμπεριφορά ***FunctionPerformer***

Η κυκλική αυτή συμπεριφορά εκτελεί την λειτουργικότητα του πράκτορα Β, λαμβάνοντας υπόψη και τους περιορισμούς επικοινωνίας που έχει ο πράκτορας. Πιο συγκεκριμένα, ο πράκτορας Β περιμένει όπως και πριν για κάποιο μήνυμα, χωρίς όμως να ψάχνει για κάποιο συγκεκριμένο πρότυπο ή αποστολέα όπως πριν. Με την προσθήκη επιπλέον επικοινωνίας στο σύστημα

κατέστη δυνατή η παραλαβή μηνύματος και από τον πράκτορα C, αφού τα δύο αυτά μέλη επικοινωνούν πλέον άμεσα χωρίς την διαμεσολάβηση του διαχειριστή. Όταν λοιπόν ένα τέτοιο μήνυμα φτάσει στον πράκτορα B, αυτός αποκωδικοποιεί την πληροφορία που περιέχεται στο αντικείμενο τύπου Task που ήταν συννημένο και εκτελει την υποεργασία που του ζητείται. Στη συνέχεια προσπαθεί να προωθήσει την επόμενη υποεργασία προς εκτέλεση στο αρμόδιο μέλος. Αν αυτό το μέλος είναι ο πράκτορας C με τον οποίο επικοινωνεί άμεσα, τότε στέλνει το αίτημα απευθείας. Στην περίπτωση όμως που ο αρμόδιος πράκτορας είναι κάποιος από τους D ή E, τότε αφού δεν μπορεί να επικοινωνήσει κατευθείαν μαζί τους στέλνει ένα αίτημα προώθησης στον διαχειριστή.

### 3.3.4 Η κλάση *AgentC*

Η κλάση και οι μέθοδοι που υλοποιούν τον πράκτορα C είναι ακριβώς όμοιοι σε υλοποίηση και λειτουργικότητα με εκείνες που χρησιμοποιεί ο πράκτορας B. Αυτο συμβαίνει για τον λόγο ότι εκτός από τις βασικές λειτουργίες που είναι όμοιες σε όλους τους πράκτορες, ακόμα και η πολιτική επικοινωνίας του πράκτορα C είναι δυϊκή της πολιτικής του πράκτορα B. Συνεπώς, η συμπεριφορά που καθορίζει την λειτουργικότητα του πράκτορα C είναι αντίστοιχη με εκείνη του πράκτορα B.

### 3.3.5 Οι κλάσεις *AgentD* και *AgentE*

Οι δύο αυτές κλάσεις που υλοποιούν τους πράκτορες D και E, έχουν τις ίδιες μεθόδους και συμπεριφορές για τους ίδιους λόγους που και οι πράκτορες B και C είχαν τις ίδιες κλάσεις. Η βασική διαφορά των κλάσεων αυτών των δύο πρακτόρων από τις κλάσεις των άλλων δύο εντοπίζεται στην πολιτική επικοινωνίας. Όπως φαίνεται στην Εικόνα 2, οι πράκτορες D και E επικοινωνούν μόνο με τον διαχειριστή πράκτορα A. Για αυτό το λόγο, σε αντίθεση με τους πράκτορες B και C, εάν η επόμενη υποεργασία προς εκτέλεση δεν αφορά τους ίδιους τότε το μόνο που έχουν να κάνουν είναι να στείλουν αίτημα προώθησης στον διαχειριστή, καθώς είναι ο μόνος που μπορεί να επικοινωνήσει με τα άλλα μέλη του Cloud συστήματος. Αυτή είναι και η μόνη διαφορά στην υλοποίηση της συμπεριφοράς FunctionPerformer σε σχέση με την αντίστοιχη συμπεριφορά των πρακτόρων B και C.

# *Κεφάλαιο 4ο*

## Παραδειγματικές Εκτελέσεις

Με βάση τα δύο Cloud συστήματα που περιγράφτηκαν αναλυτικά στο Κεφάλαιο 3, δημιουργήθηκε μία σειρά σεναρίων εκτέλεσης τα οποία έχουν σαν κοινή συνισταμένη την αποχώρηση ενός πράκτορα-μέλους από το σύστημα κατά την διάρκεια διεκπεραίωσης της εργασίας που έχει ανατεθεί στο σύστημα. Τα σενάρια αυτά χωρίζονται, για προφανή λόγο, σε δύο βασικές κατηγορίες (Version 1 και 2) που έχουν να κάνουν με τον τύπο του συστήματος που εξετάζουμε. Σε κάθε μία από αυτές τις κατηγορίες αναπτύσσονται σενάρια εκτέλεσης που διαφέρουν ως προς το μέλος που επιθυμεί να αποχωρήσει από το σύστημα και ως προς τον μηχανισμό αντιμετώπισης της αποχώρησης από τη μεριά του συστήματος.

### 4.1 Σενάρια εκτέλεσης κεντρικοποιημένου συστήματος

Στο κεντρικοποιημένο μοντέλο του Cloud συστήματος, η αποχώρηση ενός πράκτορα κατά τη διάρκεια που το σύστημα σαν ομάδα έχει αναλάβει κάποια εργασία προσθέτει μία σειρά ενεργειών που πρέπει να γίνουν για να μπορέσει το σύστημα να συνεχίσει χωρίς προβλήματα την ολοκλήρωση της εργασίας. Τα παρακάτω σενάρια εκτέλεσης δίνουν έμφαση σε μερικά από τα πιο σημαντικά ζητήματα που μπορεί να προκύψουν από την αποχώρηση ενός μέλους από το σύστημα. Για την ευκολότερη κατανόηση των σεναρίων στο υπόλοιπο του κεφαλαίου, ο πράκτορας Α μπορεί να αναφερθεί ως *administrator* ή *allocator* (επειδή είναι επιφορτισμένος με την τυχαία κατανομή των υποεργασιών στους άλλους πράκτορες και με την διαμεσολάβηση στην επικοινωνία μεταξύ των υπόλοιπων μελών) και οι πράκτορες B,C,D και E μπορεί να αναφερθούν ως *workers* ή *executors* (επειδή κύρια λειτουργία τους είναι να ολοκληρώσουν ο καθένας ξεχωριστά τις υποεργασίες που τους αναλογούν).

#### 4.1.1 Σενάριο εκτέλεσης *I*

Ένα από τα πρώτα σενάρια που υλοποιήθηκε ήταν η οικειοθελής αποχώρηση από το σύστημα ενός από τους τέσσερις workers (Agent B,C,D ή E). Το μείζον ζήτημα που προκύπτει από μία τέτοια αποχώρηση έχει να κάνει με τις υποεργασίες που είχε αναλάβει ο προς αποχώρηση πράκτορας. Επειδη όμως το μοντέλο του Cloud συστήματος έχει κεντρικοποιημένη δόμη, η αντιμετώπιση αυτής της κατάστασης μπορεί να γίνει πολύ απλά και αποτελεσματικά από τον διαχειριστή. Πιο συγκεκριμένα, υποθέτοντας ότι κάποιος από τους τέσσερις workers εκδηλώσει την επιθυμία του να φύγει κάποια στιγμή στη διάρκεια διεκπεραίωσης της ομαδικής εργασίας, θα πρέπει να ειδοποιήσει αναγκαστικά τον πράκτορα Α που εκτελεί χρέη διαχειριστή. Αυτός με τη σειρά του για να μην υπάρξει αστοχία και εν τέλει αποτυχία στην σειριακή εκτέλεση των υποεργασιών, κατανέμει τις υπόλοιπες υποχρεώσεις-εργασίες του

αποχωρήσαντα πράκτορα στα υπόλοιπα μέλη. Αυτά πλέον πρέπει να ‘’σηκώσουν’’ περισσότερο φόρτο εργασίας, καλύπτοντας με αυτό τον τρόπο την απουσία του πράκτορα. Ο διαχειριστής είναι σε θέση να γνωρίζει τις υπόλοιπες υποχρεώσεις του μέλους που έφυγε επειδή ακριβώς ήταν ο ίδιος που έκανε την αρχική τυχαία κατανομή υποεργασιών στο σύστημα.

### 4.1.2 Σενάριο εκτέλεσης *II*

Πρόκειται για ένα σενάριο που είναι δυϊκό του πρώτου και προσωμοιώνει την αποχώρηση του πράκτορα Α που αποτελεί τον διαχειριστή του συστήματος. Επειδή ο ρόλος του συγκεκριμένου μέλους είναι κομβικός για την ομαλή λειτουργία του συστήματος, τα ζητήματα που χρίζουν αντιμετώπισης σε μία ενδεχόμενη αποχώρησή του είναι ασφαλώς περισσότερα και πιο σημαντικά. Το πρώτο εύλογο ερώτημα που προκύπτει είναι το ποιος θα αναλάβει πλέον την ευθύνη διαχείρησης του συστήματος και την επίβλεψη της διαδικασίας ολοκλήρωσης της εργασίας που αύτο έχει ομαδικά αναλάβει. Στο συγκεκριμένο σενάριο εκτέλεσης υλοποιήθηκε ένας σαφής μηχανισμός που χρησιμοποιείται για την αντιμετώπιση μιας τέτοιας κατάστασης. Υποθέτοντας ότι ο διαχειριστής-πράκτορας Α θέλει να φύγει από το σύστημα ενώ βρίσκεται σε εξέλιξη η διαδικασία ολοκλήρωσης της ομαδικής εργασίας, το πρώτο που κάνει είναι να αποφασίσει ποιο από τα υπόλοιπα μέλη θα τον διαδεχτεί στο ρόλο του διαχειριστή και επιβλέποντα της εργασίας. Σε αυτή την εκτέλεση η απόφαση του πράκτορα Α βασίζεται αποκλειστικά στον πίνακα κατανομής των υποεργασιών. Αν, για παράδειγμα, ο πράκτορας Α αποφασίσει ότι θέλει να φύγει ενώ μόλις έχει ολοκληρωθεί η υποεργασία Ν, τότε χρίζει ως νέο administrator το μέλος που έχει την υποχρέωση εκτέλεσης της υποεργασίας Ν+1. Στη συνέχεια, αντί να στείλει σε αυτό το μέλος την επόμενη υποεργασία προς εκτέλεση όπως θα έπρεπε να κάνει, ειδοποιεί τον συγκεκριμένο πράκτορα για την επιθυμία αποχώρησής του και τον ενημερώνει ότι αναλαμβάνει πλέον χρέη διαχειριστή. Παράλληλα του στέλνει πληροφορία για την πορεία της υπο εκτέλεσης εργασίας για να μπορέσει ο νέος διαχειριστής να συνεχίσει την επίβλεψη. Επίσης, ο νέος διαχειριστής λαμβάνει από τον παλιό όλες τις απαραίτητες πληροφορίες για να μπορεί να επικοινωνεί ο ίδιος πλεόν με τα υπόλοιπα μέλη του Cloud σύστηματος. Μετά την αποχώρηση του πράκτορα Α, ο νέος administrator κατανέμει εκ νέου τις εναπομείνουσες υποεργασίες στους υπόλοιπους workers, αλλά ο ίδιος δεν αναλαμβάνει πλέον καμία υποχρέωση εκτέλεσης αφού ο ρόλος του και οι λειτουργίες του είναι διαφορετικές. Μόλις η τυχαία ανακατανομή ολοκληρωθεί, η διαδικασία διεκπεραίωσης της εργασίας συνεχίζεται κανονικά με τον νέο διαχειριστή να επιβλέπει την ομαλή διεξαγωγή της.

### 4.1.3 Παρατηρήσεις

Ενώ στο κεντρικοποιημένο μοντέλο τα ζητήματα που προκύπτουν από την αποχώρηση ενός μέλους είναι σημαντικά και απαιτούν ειδική μεταχείριση για την εξασφάλιση της ομαλής λειτουργίας του συστήματος, το πιο σημαντικό ίσως πρόβλημα που μπορεί να προκύψει σε τέτοια συστήματα είναι από τη φύση του μοντέλου λυμένο. Το πρόβλημα αυτό δεν είναι άλλο από την επικοινωνία μεταξύ των μελών. Στο centralized σύστημα, όλοι οι πράκτορες

επικοινωνούν απευθείας μόνο με τον διαχειριστή. Συνεπώς, ένας πράκτορας μόνο (ο διαχειριστής) είναι ικανός να αντιμετωπίσει τα προβλήματα επικοινωνίας που μπορεί να προκύψουν.

## 4.2 Σενάρια εκτέλεσης αποκεντρικοποιημένου συστήματος

Σε αντίθεση με το κεντρικοποιημένο μοντέλο, ένα σύστημα με αποκεντρικοποιημένη λειτουργία, όπως αυτό της Εικόνας 2, πρέπει να αντιμετωπίσει αποτελεσματικά (εκτός των άλλων) και τυχόν ζητήματα επικοινωνίας που θα προκύψουν με μια ενδεχόμενη αποχώρηση ενός μέλους. Οι παρακάτω παραδειγματικές εκτελέσεις βασίζονται στο σύστημα που περιγράφτηκε στο Κεφάλαιο 3 και εξετάζει περιπτώσεις αποχώρησης ενός worker αλλά και του administrator. Ιδιαίτερη προσοχή πρέπει να δωθεί στο γεγονός ότι πλέον οι πράκτορες-μέλη B και C έχουν δυνατότητα άμεσης επικοινωνίας, μια δυνατότητα που τους διαφοροποιεί σε σημαντικό βαθμό από τους άλλους δύο πράκτορες-workers (Agent D και E) του συστήματος.

### 4.2.1 Σενάριο εκτέλεσης *III*

Το συγκεκριμένο σενάριο προσομοιώνει την αποχώρηση ενός μέλους-worker που έχει δυνατότητα επικοινωνίας μόνο με τον διαχειριστή. Τέτοια μέλη είναι οι πράκτορες D και E. Ο τρόπος αντιμετώπισης των ζητημάτων που προκύπτουν από την αποχώρηση ενός εκ των δύο είναι ανάλογος με εκείνο που χρησιμοποιήθηκε στο πρώτο σενάριο εκτέλεσης (4.1.1). Η βασική διαφορά είναι ότι επειδή η λειτουργία του συστήματος είναι αποκεντρικοποιημένη, κάθε πράκτορας ''κουβαλά'' όλες τις πληροφορίες που απαιτούνται για την ομαλή εκτέλεση της ομαδική εργασίας. Έτσι, όταν κάποιος από τους δύο συγκεκριμένους πράκτορες αποφασίσει να αποχωρήσει μόλις ολοκληρώσει κάποια υποεργασία ειδοποιεί τον διαχειριστή για την πρόθεση του και του στέλνει τις πληροφορίες που έχει για την πορεία εκτέλεσης της εργασίας. Στην συνέχεια, ο διαχειριστής-πράκτορας A καλείται να ανακατανείμει τις υποεργασίες που αντιστοιχούν στο μέλος που δεν υπάρχει πλέον στο σύστημα. Ο administrator είναι σε θέση να κάνει την ανακατανομή βασισμένος στην πληροφορία για την πορεία της εργασίας που του έστειλε ο πράκτορας πριν αποχωρήσει. Μετά από αυτή την διαδικασία, τα υπόλοιπα μέλη-workers αναλαμβάνουν το φόρτο εργασίας του αποχωρήσαντα και η εξυπηρέτηση των υποεργασιών συνεχίζεται κανονικά.

### 4.2.2 Σενάριο εκτέλεσης *IV*

Η επισήμανση των επιπλέον ζητημάτων που προκύπτουν όταν αποχωρήσει ένα μέλος-worker με αυξημένες δυνατότητες επικοινωνίας, όπως οι πράκτορες B και C, είναι κύριος στόχος του συγκεκριμένου σεναρίου εκτέλεσης. Αν γίνει η υπόθεση ότι το μέλος C εκφράσει επιθυμία αποχώρησης από το σύστημα, τότε ένα πρόβλημα που πρέπει να αντιμετωπιστεί (εκτός αυτών που αναλύθηκαν στο προηγούμενο σενάριο) είναι το θέμα της επικοινωνίας μεταξύ του πράκτορα B με τον πράκτορα C που φεύγει. Όπως και στο τρίτο σενάριο

εκτέλεσης (4.2.1), το μέλος που αποχωρεί ενημερώνει τον διαχειριστή για την πρόθεσή του και του στέλνει όλες τις πληροφορίες που χρειάζονται για την ομαλή συνέχιση της εξυπηρέτησης της εργασίας. Επιπλέον ενημερώνει τον γείτονά του, μέλος B, ότι αποχωρεί και διακόπτει την απευθείας επικοινωνία μαζί του.Αυτό που είναι απαραίτητο όμως είναι να υπάρξει μέριμνα για το τι θα γίνει με το κανάλι επικοινωνίας που υπήρχε ανάμεσα στους πράκτορες B και C. Σε αυτό το σενάριο, ο διαχειριστής επιλέγει να μην ανακατανείμει τις υποεργασίες που αναλογούν στο μέλος που αποχώρησε, αλλά να προωθήσει όλες τις υποχρεώσεις του πράκτορα C στον παλιό του γείτονα (μέλος B). Πλέον ο πράκτορας B δεν γειτονεύει με κάποιο μέλος και επικοινωνεί άμεσα μόνο με τον διαχειριστή. Συνέπεια της παραπάνω πολιτικής είναι ο εκφυλισμός του συστήματος σε ένα κεντρικοποιημένο μοντέλο, όπως ακριβώς το centralized μοντέλο που παρουσιάστηκε στο Κεφάλαιο 3. Η λειτουργία του συστήματος συνεχίζεται έπειτα κανονικά, αλλά πλέον όλες οι απαραίτητες επικοινωνίες εξυπηρετούνται μέσω του διαχειριστή πράκτορα A.

### 4.2.3 Σενάριο εκτέλεσης *V*

Το συγκεκριμένο σενάριο εκτέλεσης αποτελεί μία παραλλαγή του προηγούμενου σεναρίου (4.2.2) και επιχειρεί να αντιμετωπίσει το πρόβλημα εκφυλισμού που προέκυψε από την πολιτική αποχώρησης που ακολουθήθηκε προηγουμένως. Υποθέτοντας και πάλι ότι το μέλος C αποχωρεί από το σύστημα, ο διαχειριστής επιλέγει ένα διαφορετικό τρόπο αντιμετώπισης του ζητήματος επικοινωνίας μεταξύ του μέλους B με τον πράκτορα C που αποχωρεί. Πιο συγκεκριμένα, ο πράκτορας A όταν ειδοποιηθεί από τον worker C για την αποχώρηση του τελευταίου, ενημερώνει το μέλος-worker D και του παρέχει τις απαραίτητες πληροφορίες για να μπορέσει αυτό να συνδεθεί άμεσα με τον πράκτορα B. Με αυτό τον τρόπο τα μέλη B και D είναι πλέον γείτονες και η επικοινωνία του Cloud συστήματος έχει διατηρηθεί αμετάβλητη ως προς τη δομή της. Στη συνέχεια ο διαχειριστής ανακατανέμει στους υπόλοιπους τρεις workers τις υποεργασίες που αντιστοιχούσαν στο μέλος που αποχώρησε και η διαδικασία εξυπηρέτησης της ομαδικής εργασίας συνεχίζεται ομαλά. Με την υιοθέτηση αυτής της πολιτικής, η δομή του συστήματος παρέμεινε αναλλοίωτη ενώ παράλληλα η εξυπηρέτηση της ομαδικής εργασίας συνεχίστηκε ομαλά.

### 4.2.4 Σενάριο εκτέλεσης *VI*

Σε όλα τα σενάρια που ακολουθούν από εδώ και κάτω, η προσοχή στρέφεται στην προσομοίωση καταστάσεων στις οποίες εισέρχεται το σύστημα ως σύνολο όταν ο διαχειριστής-πράκτορας A εκφράσει επιθυμία αποχώρησης. Τα προβλήματα που προκύπτουν σε μία τέτοια κατάσταση είναι ασφαλώς περισσότερα και σημαντικότερα σε σχεσή με τα ζητήματα που απασχολούν το σύστημα όταν ένα μέλος-worker αποχωρεί. Στην συγκεκριμένη εκτέλεση, ο administrator A αποφασίζει να αποχωρήσει από το Cloud σύστημα ενώ η διαδικασία ολοκλήρωσης της ομαδικής εργασίας είναι σε εξέλιξη. Η πολιτική αποχώρησης που ακολουθεί ξεκινά ειδοποιώντας τον πράκτορα D για την πρόθεσή του. Αφού του στείλει όλες τις απαραίτητες πληροφορίες που απαιτούνται για να αναλάβει το μέλος D το ρόλο του διαχειριστή (πληροφορίες για την επικοινωνία μεταξύ των μελών και για την πορεία ολοκλήρωσης της

εργασίας), αποχωρεί από το σύστημα. Ο νέος διαχειριστής ενημερώνει στη συνέχεια τα υπόλοιπα μέλη του συστήματος για την αλλαγή του administrator. Η διαδικασία εξυπηρέτησης της εργασίας συνεχίζεται κανονικά με το μέλος D να μεσολαβεί στην επικοινωνία των μελών που δεν έχουν απευθείας κανάλι επικοινωνίας. Επιπλέον, ο νεος διαχειριστής-πράκτορας D ανακατανέμει τις υποεργασίες που του αναλογούν στους υπόλοιπους workers, αφού ο λειτουργικός του ρόλος έχει διαφοροποιηθεί.

### 4.2.5 Σενάριο εκτέλεσης *VII*

Στο σενάριο αυτό ο διαχειριστής-πράκτορας A αποχωρεί κατά τη διάρκεια ολοκλήρωσης της ομαδικής εργασίας και ειδοποιεί τον worker C για να τον διαδεχτεί στο ρόλο του administrator. Η πολιτική που ακολουθεί ο πράκτορας C είναι αντίστοιχη με εκείνη που ακολούθησε ο πράκτορας A στο σενάριο 4.2.2, όπου είχε αποχωρήσει ο worker C. Πιο συγκεκριμένα, ο νέος διαχειριστής ενημερώνει τα υπόλοιπα μέλη για την αλλαγή στο σύστημα μετά την αποχώρηση του πράκτορα A και κλείνει το κανάλι άμεσης επικοινωνίας που υπήρχε μεταξύ των B και C. Η διαδικασία εξυπηρέτησης της ομαδικής εργασίας συνεχίζεται κανονικά, με τον διαχειριστή-μέλος C να προωθεί όλες τις υποεργασίες που του αναλογούν στον πρώην γείτονά του, πράκτορα B. Συνέπεια της συγκεκριμένης πολιτικής, όπως και της αντίστοιχης του σεναρίου 4.2.2, είναι ο εκφυλισμός του Cloud συστήματος σε ένα κεντρικοποιημένο μοντέλο, όπου όλες οι ανάγκες επικοινωνίας εξυπηρετούνται μέσω του διαχειριστή και τον καταστούν αυτομάτως σημείο συμφόρησης για την επικοινωνία στο σύστημα.

### 4.2.6 Σενάριο εκτέλεσης *VIII*

Ο εκφυλισμός του Cloud συστήματος που προέκυψε από την πολιτική που ακολουθήθηκε προηγουμένως στο σενάριο 4.2.5 οδήγησε στην υλοποίηση του συγκεκριμένου σεναρίου, το οποίο προσπαθεί να μην αλλοιώσει την δομή του συστήματος μετά την αποχώρηση του πράκτορα A. Με ένα ανάλογο στόχο είχε υλοποιηθεί και το σενάριο 4.2.3 που περιγράφτηκε παραπάνω. Στην ίδια λογική κινείται και η πολιτική που υιοθετείται σε αυτό το σενάριο. Καθώς ο διαχειριστής-μέλος A αποχωρεί, χρίζει διάδοχό του ρόλου του τον worker-πράκτορα C. Ο τελευταίος, σε αντίθεση με ότι κάνει στο σενάριο 4.2.5, ενημερώνει τα άλλα μέλη για την αλλαγή στο σύστημα και ταυτόχρονα στέλνει στον πράκτορα D τις απαραίτητες πληροφορίες για να εγκαταστήσει κανάλι άμεσης επικοινωνίας με τον πράκτορα B, που είχε μείνει χωρίς γείτονα μετά την αλλαγή διαχειριστή. Επιπλέον, το μέλος C ανακατανέμει όλες τις υπόλοιπες υποεργασίες που του αναλογούν στους εναπομείναντες workers, αντί να τις προωθήσει στον πράκτορα B όπως προηγουμένως, και η διαδικασία ολοκλήρωσης της ομαδικής εργασίας συνεχίζεται ομαλά. Με αυτή την πολιτική αντιμετώπισης της αποχώρησης του πράκτορα A επιτυγχανεται η συνέχιση της ομαλής λειτουργίας του Cloud συστήματος, αλλά ταυτόχρονα η λειτουργική και επικοινωνιακή δομή του συστήματος παραμένει αναλλοίωτη και δεν υπάρχει εκφυλισμός.

# *Κεφάλαιο 5$^{ο}$*

## Συμπεράσματα

Όπως περιγράφτηκε στο προηγούμενο κεφάλαιο, διαφορετικά σενάρια εκτέλεσης υλοποιήθηκαν ξεχωριστά για κάθε ένα από τα δύο Cloud συστήματα. Στο κάθε σενάριο εφαρμόστηκε και μία διαφορετική πολιτική που αφορά είτε την ανακατανομή των υποεργασιών είτε τον επανασχεδιασμό των καναλιών επικοινωνίας μεταξύ των πρακτόρων-μελών. Σε κάποια σενάρια ήταν αναγκαία η λήψη αποφάσεων και για τα δύο αυτά ζητήματα. Όπως είναι φανερό, μετά την υλοποίηση των διαφόρων σεναρίων και πολιτικών απόφασης, το επόμενο βήμα ήταν η εξαγωγή συμπερασμάτων για την αποδοτικότητα της κάθε πολιτικής. Για το λόγο αυτό σε όλα τα σενάρια εκτέλεσης εισήχθη ένα είδος ρολογιού. Πιο συγκεκριμένα, με βάση την τοπική ώρα του υπολογιστή έγινε μία χρονομέτρηση για κάθε ένα από τα σενάρια εκτέλεσης. Η χρονομέτρηση ξεκινούσε από τον διαχειριστή του Cloud συστήματος ταυτόχρονα με την αποστολή του πρώτου μηνύματος επικοινωνίας προς το πρώτο μέλος-worker που είχε υποχρέωση να διεκπεραιώσει την πρώτη υποεργασία. Υπεύθυνο για την λήξη της χρονομέτρησης ήταν το μέλος του συστήματος που λάμβανε το τελευταίο μήνυμα που επιβεβαιώνει την ομαλή ολοκλήρωση της συνολικής εργασίας. Κρατώντας ως δεδομένο ότι η συνολική εργασία που αναλαμβάνουν τα δύο Cloud συστήματα απαιτεί δεκαπέντε δευτερόλεπτα (15 sec) για να ολοκληρωθεί, ενδεικτικοί χρόνοι εκτέλεσης παρουσιάζονται στον παρακάτω πίνακα.

| **Παραδειγματική Εκτέλεση** | **Χρόνος Εκτέλεσης** |
|---|---|
| Σενάριο Εκτέλεσης *I* | 15,069 sec |
| Σενάριο Εκτέλεσης *II* | 15,125 sec |
| Σενάριο Εκτέλεσης *III* | 15,124 sec |
| Σενάριο Εκτέλεσης *IV* | 15,118 sec |
| Σενάριο Εκτέλεσης *V* | 15,127 sec |
| Σενάριο Εκτέλεσης *VI* | 15,115 sec |
| Σενάριο Εκτέλεσης VII | 15,126 sec |
| Σενάριο Εκτέλεσης *VIII* | 15,122 sec |

Για την μέτρηση του χρόνου εκτέλεσης χρησιμοποιήθηκε η κλάση *TimeUnit* και η χρονομέτρηση έγινε σε nanoseconds. Στον παραπάνω πίνακα οι χρόνοι εκτέλεσης έχουν στρογγυλοποιηθεί στο τρίτο δεκαδικό ψηφίο.

Όπως μπορεί να παρατηρηθεί από τη μελέτη του πίνακα, οι χρόνοι εκτέλεσης των σεναρίων εκτέλεσης διαφέρουν σε τόσο μικρό βαθμό που δεν γίνεται αντιληπτή η διαφορά. Πιο συγκεκριμένα, το *εύρος* των χρόνων εκτέλεσης είναι περίπου *58 milliseconds* και η *μέση τιμή* είναι περίπου *15,116 sec*. Συνεπώς, το συμπέρασμα που προέκυψε είναι ότι όλες οι πολιτικές που υλοποιήθηκαν ήταν εξαιρετικά αποδοτικές επιβαρύνοντας (με τις πολιτικές αποφάσεων) κατά μέσο όρο μόνο *116 milliseconds* τον συνολικό χρόνο εκτέλεσης των σεναρίων.

Αν γίνει μία πιο προσεκτική παρατήρηση των χρόνων εκτέλεσης, όπως αυτοί παρουσιάζονται στον πιο πάνω πίνακα, μπορεί κανείς να ξεχωρίσει τις δύο ακραίες περιπτώσεις:

1. το σενάριο εκτέλεσης *I* έχει τον πιο μικρό χρόνο εκτέλεσης, κάτι που είναι αναμενόμενο καθώς (όπως τονίστηκε αναλυτικά στο Κεφάλαιο 4) ο διαχειριστής του συστήματος ειδοποιείται άμεσα για την πρόθεση αποχώρησης και ανακατανέμει τις υπόλοιπες υποεργασίες του υπό αποχώρηση πράκτορα στα υπόλοιπα μέλη χωρίς να περιπλέκει η κατάσταση περισσότερο
2. τα σενάρια εκτέλεσης *V* και *VII* είναι οι δύο παραδειγματικές εκτελέσεις που κοστίζουν περισσότερο χρόνο, γιατί οι πολιτικές που υλοποιούν προβλέπουν την διατήρηση της δομής του Cloud συστήματος ως έχει και έτσι υπάρχει περισσότερη επικοινωνία μεταξύ των μελών για να δημιουργηθούν νέα κανάλια επικοινωνίας και να ανακατανεμηθούν οι υποεργασίες του αποχωρήσαντα πράκτορα.

Σε κάθε περίπτωση όμως, οι διαφορές στους χρόνους εκτέλεσης είναι της τάξεως των millisecond και δεν μπορούν να αποτελέσουν ασφαλή κριτηριο για την προτίμηση μιας πολιτικής απόφασης έναντι κάποιας άλλης.

# *Κεφάλαιο 6$^{ο}$*

## Επίλογος

### 6.1 Σύνοψη

Βασική επιδίωξη της παρούσας διπλωματικής εργασίας ήταν η διερεύνηση των διαφορετικών μηχανισμών που μπορούν να υιοθετηθούν από ένα Cloud σύστημα, όταν κάποιο μέλος του συστήματος αποχωρεί από αυτό ενώ του έχουν ανατεθεί συγκεκριμένες υποχρεώσεις. Συνεπώς, ιδιαίτερη σημασία δόθηκε στον σχεδιασμό τόσο των δύο Cloud συστημάτων όσο και των πολιτικών απόφασης που τα μέλη των συστημάτων χρησιμοποίησαν για την αντιμετώπιση των ζητημάτων που προέκυψαν με την αποχώρηση. Η υλοποίηση έγινε στην γλώσσα προγραμματισμού Java και στηρίχθηκε στην πολυπρακτορική πλατφόρμα JADE, όπως περιγράφεται στα προηγούμενα κεφάλαια.

### 6.2 Μελλοντικές Επεκτάσεις

Η μελέτη αυτή προέκυψε λόγω της ολοένα αυξανόμενης χρήσης δυναμικά σχηματιζόμενων συστημάτων σε πολλούς κλάδους της τεχνολογίας. Έτσι, η ανάγκη αξιολόγησης των ήδη υπάρχοντων πρωτοκόλλων αποχώρησης και η δημιουργία καινούριων σχετίζεται άμεσα με την περαιτέρω ανάπτυξη ανάλογων συστημάτων. Συνεπώς, είναι προφανές ότι η παρούσα μελέτη επιδέχεται επεκτάσεων και προσθηκών για να καλύψει τις ραγδαίες εξελίξεις του τομέα. Πιο συγκεκριμένα, κάποιες προτάσεις για μελλοντική δουλειά είναι:

- η υλοποίηση επιπλέον πολιτικών απόφασης που αφορούν την ανακατανομή των υποεργασιών του μέλους που αποχωρεί
- η υλοποίηση επιπλέον πολιτικών απόφασης που έχουν να κάνουν με την εγκατάσταση νέων καναλιών επικοινωνίας μετά την απόχωρηση ενός μέλους
- η λεπτομερής μελέτη της περίπτωσης το μέλος που αποχωρεί από το σύστημα να μην ειδοποιήσει για την πρόθεση αποχώρησης
- η αντικατάσταση της συνολικής εργασίας (που στην παρούσα μελέτη χρησιμοποιήθηκε σε αφαιρετικό επίπεδο) με μία εργασία που έχει ένα συγκεκριμένο στόχο να υλοποιήσει
- η μελέτη της περίπτωσης που υπάρχει ανάγκη επικοινωνίας και με πράκτορες που είναι μέλη άλλου συστήματος για την ολοκλήρωση της

συνολικής ομαδικής εργασίας

- η βελτιστοποίηση των υπάρχοντων πολιτικών απόφασης με σκοπό την βελτίωση της συνολικής απόδοσης

- η μελέτη της δυϊκής περίπτωσης, δηλαδή της περίπτωσης κατά την οποία ένας νέος πράκτορας εμφανιστεί και εκφράσει την επιθυμία να γίνει μέλος του συστήματος και να αναλάβει και αυτός υποχρεώσεις και εργασίες του συστήματος

# *Κεφάλαιο 7ο*

## Βιβλιογραφία


[1] http://en.wikipedia.org/wiki/Wireless_ad_hoc_network

[2] http://en.wikipedia.org/wiki/Cloud_computing

[3] http://en.wikipedia.org/wiki/Software_agent

[4] Giovanni Caire (TILAB, formerly CSELT). "JADE Tutorial – JADE Programming for Beginners". http://jade.tilab.com/doc/tutorials/JADEProgramming-Tutorial-for-beginners.pdf

[5] Fabio Bellifemine, Giovanni Caire, Tiziana Trucco (TILAB S.p.A., formerly CSELT), Giovanni Rimassa (FRAMeTech s.r.l.), Roland Mungenast (PROFACTOR GmbH). "JADE Administrator's Guide". http://jade.tilab.com/doc/administratorsguide.pdf

[6] Fabio Bellifemine, Giovanni Caire, Tiziana Trucco (TILAB, formerly CSELT), Giovanni Rimassa (University of Parma). "JADE Programer's Guide". http://jade.tilab.com/doc/programmersguide.pdf

[7] http://jade.tilab.com/doc/api/jade/core/Agent.html

[8] http://www.iro.umontreal.ca/~vaucher/Agents/Jade/primer4.html