ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ, ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

Διπλωματική Εργασία

Πολυπρακτορικά Συστήματα για την Ευφυή Διαχείριση Πόρων

Φωτεινή Γεωργίου

Επιβλέποντες: Ασπασία Δασκαλοπούλου

Επίκουρη Καθηγήτρια

Χρήστος Δ. Αντωνόπουλος

Επίκουρος Καθηγητής

Βόλος, Φεβρουάριος 2013

Ευχαριστίες

Η ολοκλήρωση της διπλωματικής μου εργασίας σηματοδοτεί και το τέλος των προπτυχιακών μου σπουδών και θα ήθελα να ευχαριστήσω κάποια άτομα που ήταν δίπλα μου όλα αυτά τα χρόνια.

Αρχικά, θα ήθελα να ευχαριστήσω την κα Ασπασία Δασκαλοπούλου, που ως επιβλέπουσα τις διπλωματικής εργασίας μου, με βοήθησε με την καθοδήγηση και τις πολύτιμες συμβουλές της. Καθ' όλη τη διάρκεια της προσπάθειας μου με στήριξε, προτείνοντας λύσεις σε τυχόν προβλήματα που προέκυπταν, ακούγοντας τις ιδέες και τις προτάσεις μου και δείχνοντας πάντα ενδιαφέρον αλλά και εμπιστοσύνη στη δουλειά μου. Η συνεργασία μαζί της μου έμαθε πολλά και συνέβαλε στο να ανταποκριθώ με τον καλύτερο δυνατό τρόπο στις ανάγκες της παρούσας εργασίας.

Αναμφισβήτητα, η σημαντικότερη βοήθεια προήλθε από την οικογένειά μου. Πάντα στο πλευρό μου, στις ευχάριστες αλλά κυρίως στις δύσκολες στιγμές, δεν σταμάτησαν να μου δείχνουν την αγάπη, την πίστη και την υποστήριξή τους όλα αυτά τα χρόνια των σπουδών μου.

Ασφαλώς ένα μεγάλο ευχαριστώ οφείλω και στους φίλους μου, οι οποίοι ήταν πάντα δίπλα μου και στη διάθεσή μου όταν τους είχα ανάγκη, πρόθυμοι να με ακούσουν, να με συμβουλεύσουν και να με ενθαρρύνουν.

Φωτεινή Γεωργίου

Βόλος, Φεβρουάριος 2013

Περίληψη

Η διαχείριση πόρων αποτελεί ένα πρόβλημα που συναντάται σε πολλούς τομείς, από απλές καθημερινές δραστηριότητες μέχρι και πιο σύνθετες, μεγαλύτερου επιστημονικού ενδιαφέροντος, εφαρμογές. Οι πόροι μπορεί να είναι οποιασδήποτε φύσης: φυσικοί, ανθρώπινοι, πόροι σε κτίρια, σε παραγωγικά συστήματα, υλικοί ή άυλοι, υπολογιστικοί, βιοτικοί ή αβιοτικοί, ανανεώσιμοι ή μη, κλπ. Για την ευφυή διαχείρισή τους έχουν επινοηθεί κατά καιρούς διάφοροι αλγόριθμοι και έχουν κατασκευαστεί συστήματα των οποίων η πολυπλοκότητα ποικίλλει ανάλογα με τον τομέα εφαρμογής τους και το μέγεθος του εκάστοτε προβλήματος.

Η Κατανεμημένη Τεχνητή Νοημοσύνη (Distributed Artificial Intelligence) είναι ένας κλάδος στα πλαίσια του οποίου έχει μελετηθεί ιδιαίτερα το πρόβλημα της διαχείρισης πόρων και έχουν επινοηθεί πολύ αξιόλογες και χρήσιμες λύσεις με αξιοποίηση των τεχνικών και δυνατοτήτων που αυτός παρέχει. Τα Συστήματα Κατανεμημένης Τεχνητής Νοημοσύνης (DAI Systems) χαρακτηρίζονται από δύο βασικά ζητήματα: το συντονισμό και τη συνεργασία. Σε πολλές περιπτώσεις μελέτης προβλημάτων που σχετίζονται με τη διαχείριση πόρων και το πώς αυτή μπορεί να γίνει με ευφυή τρόπο για την επίτευξη ενός επιθυμητού αποτελέσματος, παρατηρείται η προσπάθεια να απαντηθεί το εξής ερώτημα: πώς μπορεί ένα σύνολο από γεωγραφικά κατανεμημένους πράκτορες, χρησιμοποιώντας τους διαθέσιμους πόρους, να συντονιστεί και να κατανείμει εργασίες (tasks) στα μέλη που το αποτελούν, με στόχο την εξυπηρέτηση κάποιας ανάγκης αλλά και την ικανοποίηση συγκεκριμένων περιορισμών;

Στην παρούσα διπλωματική, μελετάται το πρόβλημα της διαχείρισης πόρων από τη σκοπιά επίλυσής του με τη βοήθεια ενός πολυπρακτορικού συστήματος. Πιο συγκεκριμένα, προσομοιώνοντας κατάλληλα το περιβάλλον του προβλήματος, τον πόρο που υπάρχει σε αυτό, τους καταναλωτές του πόρου και τους πράκτορες-εξυπηρετητές, σχεδιάστηκε ένας αλγόριθμος ο οποίος βάσει κάποιων δεδομένων και παραδοχών, παρουσιάζει έναν από τους πιθανούς τρόπους αντιμετώπισης ενός τέτοιου προβλήματος. Στόχος του αλγορίθμου είναι οι πράκτορες να χρησιμοποιήσουν τους πόρους που διαθέτουν και να εξυπηρετήσουν με όσο το δυνατόν καλύτερο τρόπο τις ανάγκες των καταναλωτών, βάσει συγκεκριμένων κριτηρίων: της απόστασης που απέχει ο κάθε πράκτορας από τους καταναλωτές (πρωτεύον κριτήριο) και της ποσότητας πόρου που έχει ο κάθε πράκτορας στη διάθεσή του (δευτερεύον κριτήριο). Αυτή η εξυπηρέτηση γίνεται είτε απ' ευθείας, επειδή κάποιοι καταναλωτές έχουν εξ' αρχής ανατεθεί σε συγκεκριμένους πράκτορες για να εξυπηρετηθούν, είτε μετά από επικοινωνία και διαπραγμάτευση μεταξύ των πρακτόρων. Η διαπραγμάτευση βασίζεται κυρίως στη λογική του πρωτοκόλλου Contract Net (CNET Protocol) και πραγματοποιείται με ανταλλαγή μηνυμάτων. Στην ουσία, για κάθε καταναλωτή που περιμένει εξυπηρέτηση, πραγματοποιείται ένας γύρος διαπραγμάτευσης μεταξύ των πρακτόρων, κατά τον οποίο ανταλλάσσουν μηνύματα με την απαιτούμενη κάθε φορά πληροφορία, μέχρι να καταλήξουν σε μία απόφαση για το ποιος θα αναλάβει τον συγκεκριμένο καταναλωτή.

Ο αλγόριθμος υλοποιήθηκε στη γλώσσα προγραμματισμού Java, χρησιμοποιώντας την πλατφόρμα JADE, μέσω της οποίας έγινε δυνατή η δημιουργία των πρακτόρων και η αξιοποίηση των γνωρισμάτων που τους χαρακτηρίζουν. Η πλατφόρμα αυτή παρέχει διάφορα χρήσιμα εργαλεία, μερικά από τα οποία βοήθησαν πολύ στη συγκεκριμένη υλοποίηση ώστε να υπάρχει μία ολοκληρωμένη εικόνα για το περιβάλλον και τα στοιχεία που περιλαμβάνει, αλλά και για να γίνεται η παρακολούθηση της συνομιλίας μεταξύ των πρακτόρων (πόσα μηνύματα ανταλλάσσονται, τι είδους κτλ).

Μετά τη σχεδίαση και την υλοποίηση του αλγορίθμου στα πλαίσια της παρούσας εργασίας, έγιναν διάφορες παρατηρήσεις σχετικά με κάποιες βελτιώσεις που θα μπορούσαν να εφαρμοστούν σε αυτόν αλλά και με κάποιες επεκτάσεις που θα μπορούσαν να απεικονίσουν καλύτερα και πιο ολοκληρωμένα ένα γεγονός ή μία κατάσταση του πραγματικού κόσμου στο οποίο συναντάται το πρόβλημα της διαχείρισης πόρων. Οι επεκτάσεις απαιτούν τροποποιήσεις στον αλγόριθμο και σίγουρα αυξάνουν την πολυπλοκότητά του. Ωστόσο, παρουσιάζουν ιδιαίτερο ερευνητικό ενδιαφέρον και μπορούν να αποτελέσουν αντικείμενο μελλοντικής έρευνας.

Abstract

The problem of resource allocation and management is encountered in various fields, including simple, daily activities as well as more complex applications of high scientific interest. The origin of the resources can vary: physical, human, land, biological, economic, tangible or intangible, computational, biotic or abiotic, renewable or non-renewable, resources related to buildings, to production systems, etc. For their intelligent management, a lot of algorithms have been designed and systems have been constructed, whose complexity depends on the application field and on the size and importance of the problem each time encountered.

Distributed Artificial Intelligence is a field in the context of which a lot of research has been conducted on the resource management problem. In addition, the exploitation of the techniques and tools offered by this field has led to various remarkable and useful solutions. Distributed Artificial Intelligence Systems are characterized by two basic issues: coordination and cooperation. In many cases, the study of problems related to resource management and how it can be achieved in the best way, creates the need for an answer to the following question: how can a group of geographically distributed agents properly allocate a set of tasks among themselves in order to serve a need, while using only the available resources and satisfying different types of constraints?

The goal of this diploma thesis is the study of the resource allocation and management problem and how it can be solved by a multi-agent system. More specifically, after the suitable simulation of the components of the problem (environment, resource consumers, agents that serve the consumers' needs), an algorithm was designed that presents a possible solution to the problem, based on certain data and assumptions. The aim of the algorithm is the coordination of the agents so that they can use the resources they have at their disposal in order to serve the consumers' needs in the best possible way, while satisfying some criteria: the criterion that is related to the distance that each consumer has from the agents (primary criterion) and the criterion that is related to the amount of resource that an agent has at his disposal at any time (secondary criterion). The service of the needs is done either directly, as some consumers (tasks) are assigned from the beginning to certain agents, or after communication and negotiation among the agents about who will serve whom. The agents' cooperation and negotiation is based mostly on the idea of Contract Net Protocol (CNET Protocol) and is implemented with message exchange. Basically, for each consumer that waits to be served, a negotiation round takes place, during which the agents exchange messages containing the required information. At the end of each round, a decision is made on which agent will serve the specific consumer.

The algorithm was implemented in Java programming language and the JADE (Java Agent DEvelopment Framework) platform helped in the creation of the agents and the exploitation of their features. The platform provides various graphical tools, some of which were proved to be really useful for the specific implementation, as they helped having a good understanding of the environment and the components that constitute the problem, but

also offered a way to track the conversation between the agents (how many messages are exchanged, of what type, etc.).

The design and the implementation of the algorithm resulted in some conclusions, but at the same time a lot of observations were made on the optimizations and the extensions that could be applied to it. All these could certainly cover a wider range of problems and lead to a better representation of an event or situation of the real world that deals with resource allocation and management. The difficult part of the extension of the algorithm is that it can cause changes which will probably increase its complexity. However, they are of significant research interest and they can form the core of future research in a higher level.

Περιεχόμενα

1. Εισαγωγή

Η κατανομή και η διαχείριση πόρων αποτελεί ένα πρόβλημα που συναντάται σε πολλούς τομείς, από απλές καθημερινές δραστηριότητες μέχρι και πιο σύνθετες, μεγαλύτερου επιστημονικού ενδιαφέροντος εφαρμογές. Σχετίζεται με τον αποτελεσματικό προγραμματισμό των δραστηριοτήτων που πρέπει να γίνουν και με την έξυπνη χρήση των πόρων που απαιτούνται από αυτές, λαμβάνοντας υπ' όψιν τη διαθεσιμότητα των πόρων αλλά και το χρόνο στον οποίο πρέπει να ολοκληρωθούν οι συγκεκριμένες δραστηριότητες. Οι πόροι μπορεί να είναι οποιασδήποτε φύσης: φυσικοί, ανθρώπινοι, πόροι σε κτίρια, σε παραγωγικά συστήματα, υλικοί ή άυλοι, υπολογιστικοί, βιοτικοί ή αβιοτικοί, ανανεώσιμοι ή μη κλπ. Για την ευφυή διαχείρισή τους έχουν επινοηθεί κατά καιρούς διάφοροι αλγόριθμοι και έχουν κατασκευαστεί συστήματα των οποίων η πολυπλοκότητα ποικίλλει ανάλογα με τον τομέα εφαρμογής τους και το μέγεθος του εκάστοτε προβλήματος.

Στην παρούσα εργασία μελετάται το πρόβλημα της διαχείρισης των πόρων και η επίλυσή του από ένα πολυπρακτορικό σύστημα, στα πλαίσια ενός κατάλληλα προσομοιωμένου περιβάλλοντος, στο οποίο υπάρχουν πόροι και καταναλωτές που έχουν ανάγκη για αυτούς. Σε αυτή την περίπτωση, στόχος είναι οι πράκτορες, που έχουν στη διάθεσή τους κάποιες μονάδες πόρου, να επικοινωνήσουν και να συνεργαστούν κατάλληλα, ώστε να εξυπηρετήσουν όσο το δυνατόν πιο αποδοτικά τους καταναλωτές του περιβάλλοντος, βάσει συγκεκριμένων κριτηρίων. Έτσι, σχεδιάστηκε ένας αλγόριθμος, ο οποίος δείχνει τη διαδικασία με την οποία γίνεται η εξυπηρέτηση των καταναλωτών, είτε επειδή έχουν ανατεθεί αρχικά ως αρμοδιότητα κάποιου πράκτορα, είτε επειδή ανατίθενται κατά τη διάρκεια εκτέλεσης του αλγορίθμου σε κάποιον πράκτορα, ως αποτέλεσμα κατάλληλης επικοινωνίας και σύναψης συμφωνίας μεταξύ των πρακτόρων. Η επικοινωνία για τη λήψη απόφασης σχετικά με το ποιος θα εξυπηρετήσει ένα συγκεκριμένο καταναλωτή, βασίζεται κυρίως στη λογική του πρωτοκόλλου Contract Net. Η υλοποίηση του αλγορίθμου έγινε στη γλώσσα προγραμματισμού Java και βασίστηκε στην πλατφόρμα ανάπτυξης πρακτόρων JADE.

Στο ***2ο Κεφάλαιο*** γίνεται μία συνοπτική εισαγωγή στους πράκτορες, τα πολυπρακτορικά συστήματα και αναφέρονται βασικές έννοιες σχετικά με την επικοινωνία, τη συνεργασία τους και τη σύναψη συμφωνιών μεταξύ αυτών. Γίνεται, επίσης, αναφορά σε συστήματα που έχουν σχεδιαστεί για να επιλύσουν το πρόβλημα της διαχείρισης πόρων σε κάποιο συγκεκριμένο τομέα που περιλαμβάνει δραστηριότητες του πραγματικού κόσμου.

Στο ***3ο Κεφάλαιο*** δίνεται μία γενική περιγραφή του αλγορίθμου που χρησιμοποιήθηκε για την επίλυση του προβλήματος που μελετάται στην παρούσα εργασία. Γίνεται μία αναφορά στις τρεις φάσεις που στην ουσία συνθέτουν τον αλγόριθμο και εξηγείται τι συμβαίνει σε κάθε μία από αυτές, ώστε τελικά να επιτευχθεί το επιθυμητό αποτέλεσμα.

Στο ***4ο Κεφάλαιο*** περιγράφεται η υλοποίηση του αλγορίθμου. Αρχικά, δίνονται κάποιες γενικές πληροφορίες σχετικά με την πλατφόρμα ανάπτυξης πρακτόρων JADE και τα εργαλεία που αυτή παρέχει και χρησιμοποιήθηκαν για τους σκοπούς της παρούσας εργασίας. Στη συνέχεια, παρουσιάζονται τα διαγράμματα UML των Java κλάσεων, οι οποίες

δημιουργήθηκαν για τους σκοπούς του συγκεκριμένου αλγορίθμου. Βάσει αυτών περιγράφεται αναλυτικά η διαδικασία της υλοποίησης, οι παραδοχές που έγιναν, η προσομοίωση του περιβάλλοντος του προβλήματος αλλά και των στοιχείων που το αποτελούν, η επικοινωνία μεταξύ των πρακτόρων και η επιλογή κατάλληλων τύπων μηνυμάτων ανάλογα με τις εκάστοτε συνθήκες, ενώ παρουσιάζεται ο ρόλος των κλάσεων και οι δραστηριότητες που λαμβάνουν χώρα σε κάθε μία από αυτές.

Στο ***5ο Κεφάλαιο*** αναφέρονται διάφορες παρατηρήσεις που έγιναν κατά τη διάρκεια της παρούσας εργασίας, οι οποίες σχετίζονται με βελτιώσεις που θα μπορούσαν να γίνουν στον αλγόριθμο που σχεδιάστηκε και χρησιμοποιήθηκε για την επίλυση του προβλήματος που μελετάται. Επίσης, δίνονται ιδέες και προτάσεις για μελλοντικές επεκτάσεις που θα μπορούσαν να διευρύνουν τις δυνατότητες του αλγορίθμου, ώστε να μπορεί να εφαρμοστεί σε ένα μεγαλύτερο φάσμα προβλημάτων, τα οποία αντιπροσωπεύουν εφαρμογές που συναντώνται σε ποικίλες ανθρώπινες δραστηριότητες.

2. Θεωρητικό Υπόβαθρο

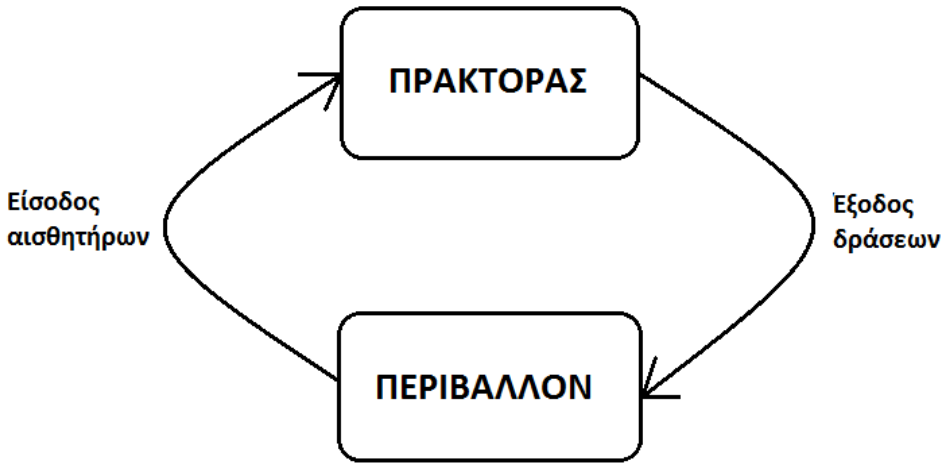
Ο τομέας των έξυπνων πρακτόρων παρουσιάζει ιδιαίτερο ερευνητικό ενδιαφέρον. Πρόκειται για υπολογιστικά συστήματα ικανά για ανεξάρτητη δράση μέσα σε περιβάλλοντα για λογαριασμό των χρηστών ή ιδιοκτητών τους. Η ανάγκη για κατασκευή τέτοιου είδους συστημάτων αποτέλεσε βασική πρόκληση στην επιστήμη των υπολογιστών και ήταν αυτή που οδήγησε στην εμφάνιση του πεδίου των πολυπρακτορικών συστημάτων (multi-agent systems).

2.1 Εισαγωγή σε πράκτορες – Ορισμοί και χαρακτηριστικά

Υπάρχουν διάφοροι ορισμοί και κατηγοριοποιήσεις για τους πράκτορες [Franklin & Graesser, 1997]. Μέρος της δυσκολίας που συναντάται στο να δοθεί ένας καθολικά αποδεκτός ορισμός για τον πράκτορα είναι ότι τα χαρακτηριστικά του γνωρίσματα, που σχετίζονται με τη δράση του, έχουν διαφορετική σημασία για τους διάφορους τομείς. Ωστόσο, ο ορισμός που φαίνεται να είναι περισσότερο αποδεκτός είναι αυτός που δίνει ο Wooldridge [Wooldridge & Jennings, 1995] σύμφωνα με τον οποίο:

> *Ο πράκτορας είναι ένα υπολογιστικό σύστημα που βρίσκεται σε κάποιο περιβάλλον, και το οποίο είναι ικανό για αυτόνομη δράση μέσα σε αυτό το περιβάλλον προκειμένου να ικανοποιήσει τους σχεδιαστικούς του στόχους.*

Η *Εικόνα 2.1* δίνει μία αφαιρετική άποψη ενός πράκτορα. Απεικονίζεται ο πράκτορας και το περιβάλλον στο οποίο βρίσκεται. Ο πράκτορας, ως αυτόνομη οντότητα, παρατηρεί και λαμβάνει είσοδο από το περιβάλλον του μέσω αισθητήρων, και παράγει ως έξοδο δράσεις που επηρεάζουν το περιβάλλον.



Εικόνα 2.1

Για να μπορέσει να χαρακτηριστεί ένας πράκτορας έξυπνος, θα πρέπει να διαθέτει κάποιες συγκεκριμένες ικανότητες. Αυτές αναφέρονται στην παρακάτω λίστα που προτάθηκε από τους [Wooldridge & Jennings, 1995]:

- ***Αυτονομία* (autonomy)**: Ένας ευφυής πράκτορας έχει στόχους και λειτουργεί εκ μέρους του χρήστη ή ιδιοκτήτη του, χωρίς όμως την παρέμβαση του.
- ***Αντιδραστικότητα* (reactivity)**: Ένας ευφυής πράκτορας μπορεί να αντιλαμβάνεται το περιβάλλον του και να ανταποκρίνεται έγκαιρα σε αλλαγές που συμβαίνουν σε αυτό, προκειμένου να επιτύχει τους σχεδιαστικούς του στόχους.
- ***Ενεργητικότητα* (proactiveness)**: Ένας ευφυής πράκτορας έχει την ικανότητα να λαμβάνει πρωτοβουλίες, ανάλογα με τις μεταβολές που παρατηρούνται στο περιβάλλον του, ώστε να ικανοποιήσει τους στόχους του.
- ***Κοινωνική ικανότητα* (social ability)**: Ένας ευφυής πράκτορας μπορεί να αλληλεπιδρά με άλλους πράκτορες (ή πιθανώς και με ανθρώπους), μέσω κάποιας κατάλληλα ορισμένης γλώσσας επικοινωνίας, ώστε να συνεργαστεί και να εκπληρώσει τους στόχους του.

Ανάλογα με την κατηγορία στην οποία μπορεί να ανήκει ένας πράκτορας, μπορεί να εμφανίζει κι άλλες δευτερεύουσες ικανότητες.

2.2 Πολυπρακτορικά Συστήματα

Στον κλάδο της Τεχνητής Νοημοσύνης (Artificial Intelligence), η τεχνολογία των συστημάτων βασισμένων σε πράκτορες, αποτελεί ένα νέο παράδειγμα σύλληψης, σχεδίασης και υλοποίησης συστημάτων λογισμικού. Οι πράκτορες είναι οντότητες που δρουν αυτόνομα και για λογαριασμό των χρηστών τους σε ανοιχτά και κατανεμημένα περιβάλλοντα, ώστε να επιλύσουν σύνθετα προβλήματα. Ωστόσο, υπάρχουν ολοένα και περισσότερες εφαρμογές, οι οποίες απαιτούν πολλαπλούς πράκτορες, οι οποίοι μπορούν να δουλέψουν μαζί στο ίδιο περιβάλλον. Ένα πολυπρακτορικό σύστημα (multi-agent system) είναι ουσιαστικά ένα δίκτυο από πράκτορες λογισμικού που αλληλεπιδρούν, δηλαδή συνεργάζονται, επικοινωνούν και διαπραγματεύονται, για να λύσουν προβλήματα τα οποία δεν μπορούν να επιλυθούν μόνο με εκμετάλλευση των ικανοτήτων και της γνώσης που μπορεί να έχει ένας μεμονωμένος πράκτορας.

Στα πολυπρακτορικά συστήματα συναντώνται συνήθως δύο διαφορετικές περιπτώσεις. Η πρώτη περιλαμβάνει πράκτορες, οι οποίοι ανταλλάσσουν πληροφορίες, αλλά εργάζονται αυτόνομα προς την κατεύθυνση επίτευξης δικών τους ανεξάρτητων στόχων. Η δεύτερη περιλαμβάνει πράκτορες οι οποίοι επικοινωνούν και συνεργάζονται, ώστε οι λύσεις που δίνουν σε επιμέρους προβλήματα να συνδυαστούν σε μία τελική λύση για κάποιο πιο σύνθετο πρόβλημα. Το στοιχείο εκείνο που χαρακτηρίζει τους πράκτορες της δεύτερης περίπτωσης είναι η δυνατότητα που έχουν να χρησιμοποιούν μία κατάλληλη και κοινά αποδεκτή μεταξύ τους γλώσσα επικοινωνίας, ώστε να διαπραγματευτούν, να επιλύσουν ενδεχόμενες συγκρούσεις που εμφανίζονται και τελικά να καταλήξουν σε κάποια συμφωνία, που οδηγεί στην καλύτερη δυνατή αντιμετώπιση ενός προβλήματος.

Ένα πολυπρακτορικό σύστημα παρουσιάζει τα εξής πλεονεκτήματα σε σχέση με τη χρήση ενός μοναδικού πράκτορα ή την επιλογή κάποιας κεντρικοποιημένης προσέγγισης:

- Μπορεί να κατανείμει υπολογιστικούς πόρους και δυνατότητες σε ένα δίκτυο από διασυνδεδεμένους πράκτορες. Σε αντίθεση με ένα κεντρικοποιημένο σύστημα, το οποίο μπορεί να αντιμετωπίζει περιορισμούς σε πόρους, σημεία συμφόρησης στην απόδοση ή κρίσιμες αποτυχίες, ένα πολυπρακτορικό σύστημα δεν «υποφέρει» από το πρόβλημα του «μοναδικού σημείου αποτυχίας» (“single point of failure” problem).
- Επιτρέπει τη διασύνδεση και τη διαλειτουργικότητα πολλαπλών, υπαρχόντων συστημάτων. Κατασκευάζοντας τον κατάλληλο wrapper πράκτορα για αυτά τα συστήματα, μπορούν να μετατραπούν σε μία κοινωνία πρακτόρων.
- Επιτρέπει τη συμμετοχή ετερογενών πρακτόρων σχεδιασμένων από διαφορετικές ομάδες και σχεδιαστές.
- Παρέχει τη δυνατότητα κλιμάκωσης με τη δυναμική διείσδυση ή έξοδο πρακτόρων από το σύστημα.
- Μοντελοποιεί προβλήματα βάσει της ιδέας των αυτόνομων πρακτόρων που αλληλεπιδρούν. Αυτό αποτελεί ένα πιο φυσικό και κατανοητό τρόπο για την αναπαράσταση προβλημάτων που σχετίζονται με την κατανομή εργασιών, τον προγραμματισμό μιας ομάδας, τις διαφορετικές προτιμήσεις χρηστών, τα ανοιχτά περιβάλλοντα, κλπ.
- Ανακτά, φιλτράρει, συνδυάζει και συντονίζει πληροφορία που μπορεί να προέρχεται από πηγές χωρικά κατανεμημένες.
- Παρέχει λύσεις σε καταστάσεις όπου η εμπειρία και η τεχνογνωσία είναι χωρικά και προσωρινά κατανεμημένη.
- Βελτιώνει τη συνολική απόδοση του συστήματος και πιο συγκεκριμένα σε επίπεδο υπολογιστικής απόδοσης, αξιοπιστίας, επεκτασιμότητας, δυνατότητας συντήρησης, απόκρισης, ευελιξίας και επαναχρησιμοποίησης.

Εκτός όμως από τα παραπάνω πλεονεκτήματα, υπάρχουν και κάποιες δυσκολίες κατά το σχεδιασμό και την υλοποίηση ενός πολυπρακτορικού συστήματος, στις οποίες πρέπει να δοθεί ιδιαίτερη προσοχή ώστε να αντιμετωπιστούν όσο γίνεται πιο αποτελεσματικά. Αυτές οι δυσκολίες σχετίζονται κυρίως με την επικοινωνία μεταξύ των πρακτόρων και τον τρόπο συνεργασίας τους, αλλά και με την ικανότητα μάθησης ώστε να μπορούν να αλλάζουν κατάλληλα τη συμπεριφορά τους και να προσαρμόζονται στις νέες συνθήκες που επικρατούν στο περιβάλλον τους. Η δυνατότητα μάθησης είναι πολύ σημαντική και κρίσιμη, γιατί κατά το σχεδιασμό των πρακτόρων είναι αδύνατο να προβλεφθούν όλες οι δυνατές καταστάσεις που μπορεί να αντιμετωπίσουν στο περιβάλλον τους, ώστε να προσδιοριστεί και να τους αποδοθεί εκ των προτέρων η ιδανική συμπεριφορά για κάθε μία από αυτές.

Οι βασικές ιδέες των πολυπρακτορικών συστημάτων δεν βρίσκουν εφαρμογή σε ένα μοναδικό πεδίο, αλλά σε πληθώρα διαφορετικών θεμάτων. Εκτός από την Επιστήμη των Υπολογιστών, οι τεχνολογίες που βασίζονται σε πράκτορες συναντώνται σε διάφορους άλλους, θεωρητικούς και εφαρμοσμένους τομείς. Σύμφωνα με το [Luck et al., 2003], οι

κυριότερες συνδέσεις παρατηρούνται με τους εξής επιστημονικούς κλάδους: Φιλοσοφία, Λογική, Οικονομικά, Κοινωνικές Επιστήμες και Βιολογία.

2.3 Επικοινωνία, συνεργασία και σύναψη συμφωνιών μεταξύ των πρακτόρων

Η επικοινωνία αποτελεί ένα θέμα κεντρικής σημασίας στην Επιστήμη των Υπολογιστών και έχουν αναπτυχθεί πολλές μέθοδοι τυποποίησης για την αναπαράσταση των ιδιοτήτων των συστημάτων που επικοινωνούν. Για την επικοινωνία μεταξύ των πρακτόρων, έχουν αναπτυχθεί διάφοροι μηχανισμοί, οι κυριότεροι από τους οποίους περιλαμβάνουν:

- ανταλλαγή μηνυμάτων με χρήση κάποιας γλώσσας επικοινωνίας πρακτόρων (message passing)
- επικοινωνία με χρήση του μοντέλου του μαυροπίνακα (blackboard model)

Υπάρχουν γλώσσες που αναπτύχθηκαν ειδικά για την επικοινωνία μεταξύ των πρακτόρων και βασίζονται στην ανταλλαγή μηνυμάτων. Μία τέτοια γλώσσα είναι η *KQML* (Knowledge Query and Manipulation Language). Αυτή η γλώσσα ορίζει μία κοινή μορφή για τα μηνύματα, τα οποία μπορούν να θεωρηθούν ως αντικείμενα με την έννοια του αντικειμενοστραφούς προγραμματισμού, καθώς διαθέτουν ένα τελεστικό (που μπορεί να θεωρηθεί ως κλάση) και έναν αριθμό από παραμέτρους (που μπορούν να θεωρηθούν μεταβλητές στιγμιότυπου). Αν και η KQML υιοθετήθηκε σε μεγάλο βαθμό από την κοινότητα των ερευνητών και χρησιμοποιήθηκε ως βάση για πολλές υλοποιήσεις, διάφοροι λόγοι οδήγησαν στην ανάπτυξη μίας νέας γλώσσας από την ομάδα FIPA. Το Ίδρυμα για Ευφυείς Φυσικούς Πράκτορες (Foundation for Intelligent Physical Agents, FIPA) ανέπτυξε μία γλώσσα επικοινωνίας πρακτόρων που αποκαλείται *FIPA ACL* [FIPA, 1999]. Αυτή είναι παρόμοια με την KQML, με μόνη σημαντική διαφορά το σύνολο των τελεστικών που παρέχει κάθε μία από αυτές. Η επικοινωνία μεταξύ των πρακτόρων στα πλαίσια της παρούσας εργασίας γίνεται με χρήση της γλώσσας FIPA ACL και σε επόμενη ενότητα παρουσιάζεται ένα μέρος από τα τελεστικά της που χρησιμοποιούνται για τους σκοπούς του αλγορίθμου που σχεδιάστηκε. Ολόκληρη η λίστα με τα τελεστικά που παρέχονται από τη γλώσσα και τη σημασία που έχει το καθένα μπορεί να βρεθεί στο κείμενο FIPA (1999).

Το μοντέλο του μαυροπίνακα είναι ένας από τους πιο σημαντικούς προδρόμους στην ιστορία των πολυπρακτορικών συστημάτων [Engelmore και Morgan, 1988]. Σύμφωνα με αυτό, μία ομάδα «πηγών γνώσης» (πρακτόρων), προκειμένου να επιλύσουν συλλογικά κάποιο πρόβλημα, χρησιμοποιούν μια κοινή δομή δεδομένων που ονομάζεται μαυροπίνακας.

Η βασική διαφορά ανάμεσα στους δύο μηχανισμούς επικοινωνίας, είναι ότι στην περίπτωση της ανταλλαγής μηνυμάτων, οι πράκτορες ανταλλάσουν μεταξύ τους προσωπικά, ιδιωτικά μηνύματα, ενώ στα μοντέλο του μαυροπίνακα όλα τα μηνύματα αποθηκεύονται σε ένα χώρο που είναι κοινός για όλους τους πράκτορες του συστήματος, οπότε αυτά είναι ορατά και προσπελάσιμα από τον καθένα.

Σε μία κοινότητα πρακτόρων, εκτός από το μηχανισμό επικοινωνίας, είναι πολύ σημαντικό να οριστεί και το λεγόμενο πρωτόκολλο επικοινωνίας. Το πρωτόκολλο μπορεί να καθορίσει τον τύπο, τη δομή, το περιεχόμενο και τη σημασιολογία των μηνυμάτων που ανταλλάσουν οι πράκτορες, αλλά και να ορίσει το σύνολο των επιτρεπτών ενεργειών σε κάθε περίσταση.

Όπως σε κάθε ανθρώπινη κοινωνία, έτσι και στα πολυπρακτορικά συστήματα υπάρχει αλληλεπίδραση μεταξύ των μελών τους, τα οποία μπορεί να λειτουργούν ιδιοτελώς. Σε μία τέτοια περίπτωση, θα πρέπει οι πράκτορες να επισυνάψουν μεταξύ τους κάποιου είδους αμοιβαία επωφελούς συμφωνίας για ζητήματα κοινού ενδιαφέροντος. Η σύναψη συμφωνιών αποτελεί θεμελιώδη ικανότητα των αυτόνομων ευφυών πρακτόρων και η δυνατότητα διαπραγμάτευσης (negotiation) και επιχειρηματολογίας (argumentation) έχουν κεντρικό ρόλο σε αυτή. Μία ειδική κατηγορία πρωτοκόλλων επικοινωνίας πρακτόρων είναι τα πρωτόκολλα διαπραγμάτευσης. Αυτά ορίζουν τους «κανόνες της αναμέτρησης» ανάμεσα στους πράκτορες [Rosenschein και Zlotkin, 1994] και σχεδιάζονται έτσι ώστε μία συγκεκριμένη διαπραγμάτευση να έχει κάποιες επιθυμητές ιδιότητες.

Υπάρχουν περιπτώσεις προβλημάτων στα οποία γίνεται η υπόθεση της αγαθής προαίρεσης (benevolence), ότι δηλαδή οι πράκτορες ενός συστήματος μοιράζονται έμμεσα ένα κοινό στόχο και έτσι δεν υπάρχει ενδεχόμενο διένεξης μεταξύ τους. Σε αυτή την υπόθεση στηρίζεται και η συνεργατική κατανεμημένη επίλυση προβλήματος, ΣΚΕΠ, (Cooperative Distributed Problem Solving, CDPS) που ξεκίνησε με δουλειά του Lesser και των συναδέλφων του. Η ΣΚΕΠ μελετά τον τρόπο με τον οποίο ένα χαλαρά συνδεδεμένο δίκτυο επιλυτών προβλημάτων (πρακτόρων) μπορεί να συνεργαστεί για να επιλύσει προβλήματα που είναι πέρα από τις ατομικές ικανότητες των επιλυτών αυτών [Durfee et al., 1989b, σελ. 63]. Μπορεί να θεωρηθεί ως μία δραστηριότητα τριών σταδίων:

1. Αποσύνθεση προβλήματος
2. Επίλυση προβλήματος
3. Σύνθεση λύσης

Με δεδομένο αυτό το γενικό πλαίσιο για τη ΣΚΕΠ, υπάρχουν δύο συγκεκριμένες συνεργατικές δραστηριότητες επίλυσης προβλημάτων: η κοινοχρησία εργασιών και η κοινοχρησία αποτελεσμάτων [Smith και Davis, 1980].

- *Κοινοχρησία εργασιών*: Πραγματοποιείται όταν ένα πρόβλημα αποσυντίθεται σε μικρότερα υποπροβλήματα που ανατίθενται σε διαφορετικούς πράκτορες. Το βασικό ζήτημα είναι ο τρόπος με τον οποίο γίνεται η ανάθεση εργασιών σε κάθε πράκτορα το οποίο εξαρτάται σε μεγάλο βαθμό από το αν οι πράκτορες είναι ομοιογενείς ή μη ως προς τις ικανότητές τους. Υπάρχουν και οι περιπτώσεις, όπου οι πράκτορες είναι εντελώς αυτόνομοι και κατά συνέπεια μπορούν να αρνηθούν την εκτέλεση εργασιών. Τότε, δεν ισχύει η υπόθεση της αγαθής προαίρεσης που αναφέρθηκε παραπάνω και είναι αναγκαία η σύναψη συμφωνιών μεταξύ των πρακτόρων για να πραγματοποιηθεί η ανάθεση των εργασιών.
- *Κοινοχρησία αποτελεσμάτων*: Απαιτεί οι πράκτορες να μοιράζονται πληροφορίες σχετικά με τα υποπροβλήματά τους. Αυτές οι πληροφορίες μπορεί να διαμοιράζονται ενεργητικά (ένας πράκτορας στέλνει πληροφορίες σε κάποιον άλλο πράκτορα επειδή πιστεύει ότι θα ενδιαφερθεί για αυτές) ή αντιδραστικά (ένας

πράκτορας στέλνει πληροφορίες σε κάποιον άλλο ως απάντηση σε μία αίτηση που ο ίδιος έλαβε νωρίτερα).

Φυσικά, υπάρχουν και περιπτώσεις προβλημάτων, στα οποία γίνεται ένας συνδυασμός κοινοχρησίας εργασιών και κοινοχρησίας αποτελεσμάτων.

Το πρωτόκολλο Contract Net (CNET) είναι ένα πρωτόκολλο υψηλού επιπέδου για την επίτευξη αποδοτικής συνεργασίας μέσω της κοινοχρησίας εργασιών σε δίκτυα επικοινωνούντων επιλυτών προβλημάτων [Smith, 1977, 1980a,b· Smith και Davis, 1980]. Η βασική έννοια που χρησιμοποιείται στο CNET είναι η σύναψη συμβολαίων (contracts), όπως υποδηλώνει και το όνομά του και θα μπορούσε να θεωρηθεί ότι αποτελείται από τις εξής φάσεις:

1. *Αναγνώριση προβλήματος*: Οι πράκτορες ενός συστήματος αναγνωρίζουν το πρόβλημα το οποίο καλούνται να επιλύσουν.
2. *Ανακοίνωση εργασίας*: Ένας πράκτορας που παράγει μία εργασία τη διαφημίζει προς τους υπόλοιπους και στη συνέχεια ενεργεί ως διαχειριστής αυτής της εργασίας, για όσο διαρκέσει. Η ανακοίνωση μπορεί να γίνει είτε με γενική εκπομπή, είτε με ανακοίνωση από σημείο προς σημείο ανάλογα με το αν είναι γνωστές ή όχι, οι ικανότητες των άλλων πρακτόρων και κατά συνέπεια η καταλληλότητά τους να ολοκληρώσουν τη συγκεκριμένη εργασία.
3. *Υποβολή προσφορών*: Οι πράκτορες «ακούν» την ανακοίνωση, αξιολογούν τους πόρους (υλικού και λογισμικού) που έχουν στη διάθεσή τους και αν κρίνουν ότι είναι κατάλληλοι για μία συγκεκριμένη εργασία, υποβάλλουν την προσφορά τους.
4. *Ανάθεση συμβολαίου*: Ο διαχειριστής λαμβάνει όλες τις προσφορές και επιλέγει τον πράκτορα που είναι καταλληλότερος για την εκτέλεση μιας εργασίας. Στη συνέχεια, διαβιβάζει την επιλογή του σε αυτόν μέσω ενός μηνύματος απονομής. Ο επιλεγμένος πράκτορες ονομάζεται ανάδοχος για αυτή την εργασία.

Όταν ολοκληρωθεί μία εργασία, ο ανάδοχος στέλνει μία αναφορά στο διαχειριστή [Smith, 1980b, σελ. 60, 61].

Η διαδικασία αυτή πραγματοποιείται για όλες τις πιθανές εργασίες που μπορεί να υπάρχουν σε μία κοινωνία πρακτόρων και υπάρχουν περιπτώσεις στις οποίες μπορεί να απλοποιηθεί, με αποτέλεσμα μία βελτίωση στην αποδοτικότητα του πρωτοκόλλου.

Ο αλγόριθμος που σχεδιάστηκε για το σκοπό της παρούσας εργασίας, στηρίζεται στη συνεργασία μεταξύ των πρακτόρων για το συγκεκριμένο πρόβλημα που μελετάται με κοινοχρησία εργασιών. Συγκεκριμένα, ορίζονται δύο κριτήρια βάσει των οποίων το σύνολο των εργασιών που πρέπει να ολοκληρωθούν για την επίτευξη του στόχου, μοιράζεται στους πράκτορες. Τα κριτήρια αυτά δεν σχετίζονται με τις ικανότητες των πρακτόρων, καθώς αυτοί θεωρούνται ότι είναι ομοιογενείς. Επίσης, ισχύει η υπόθεση της αγαθής προαίρεσης, οπότε δεν χρειάζεται ποτέ να γίνει κάποια συμφωνία μεταξύ των πρακτόρων για την επίλυση πιθανής διένεξης. Η ουσιαστική διαφορά που παρουσιάζει η λογική του αλγορίθμου σε σχέση με την ιδέα του πρωτοκόλλου Contract Net είναι ότι δεν υπάρχει κάποιος πράκτορας που να παίζει το ρόλο του διαχειριστή ανακοινώνοντας εργασίες, λαμβάνοντας προσφορές και αναθέτοντας εργασίες σε κατάλληλους πράκτορες. Αντίθετα,

ο κάθε πράκτορας εξετάζει μία-μία τις εργασίες που πρέπει να γίνουν στα πλαίσια του κόσμου του από συγκεκριμένη λίστα, ελέγχει τις δυνατότητες του κι αν είναι ικανός να διεκπεραιώσει μία εργασία, στέλνει (υποβάλλει) την προσφορά τους στους υπόλοιπους πράκτορες. Η ανάθεση μίας εργασίας στον πιο κατάλληλο πράκτορα γίνεται με αυτόματο τρόπο: ο κάθε πράκτορας ελέγχει τις προσφορές που λαμβάνει, τις συγκρίνει με τη δική του κι αν είναι ο πιο κατάλληλος αυτο-ανακηρύσσεται ανάδοχος της συγκεκριμένης εργασίας. Η μόνη περίπτωση που μπορεί να παρουσιάζει ιδιαιτερότητα είναι αυτή στην οποία υπάρχει ισοπαλία σε σχέση με τις προσφορές που υποβάλλονται. Τότε, γίνεται παροχή βοήθειας από έναν πράκτορα (Dummy Agent), ο οποίος αποφασίζει τυχαία ποιος θα γίνει ανάδοχος της εργασίας και το ανακοινώνει σε όλους τους πράκτορες. Ο Dummy Agent έχει αποκλειστικά αυτόν το ρόλο και δεν συμμετέχει ποτέ στη διαδικασία υποβολής ή αξιολόγησης κάποιας προσφοράς για μία εργασία.

2.4 Πολυπρακτορικά Συστήματα και Διαχείριση Πόρων

Η κατανομή και διαχείριση πόρων (resource allocation and management) είναι ένα πρόβλημα, που συναντάται σε πολλούς τομείς, από απλές καθημερινές δραστηριότητες, μέχρι και εφαρμογές μεγαλύτερου επιστημονικού ενδιαφέροντος. Αυτό αποτελεί και το αντικείμενο μελέτης της παρούσας διπλωματικής. Οι πόροι μπορεί να είναι οποιασδήποτε φύσης: φυσικοί, ανθρώπινοι, πόροι σε κτίρια, σε αυτόματα συστήματα σπιτιών, σε παραγωγικά συστήματα, υλικοί ή άυλοι, υπολογιστικοί, βιοτικοί ή αβιοτικοί, ανανεώσιμοι ή μη, κλπ. Για την ευφυή διαχείρισή τους έχουν επινοηθεί πολλών ειδών αλγόριθμοι και έχουν κατασκευαστεί συστήματα με δυνατότητες ανάλογες με τις απαιτήσεις του κάθε προβλήματος και με τις ανάγκες που παρουσιάζονται σε κάθε τομέα εφαρμογής.

Η Κατανεμημένη Τεχνητή Νοημοσύνη (Distributed Artificial Intelligence) είναι ένας κλάδος στα πλαίσια του οποίου έχει μελετηθεί ιδιαίτερα το πρόβλημα της διαχείρισης πόρων και έχουν επινοηθεί πολύ αξιόλογες και χρήσιμες λύσεις με αξιοποίηση των τεχνικών και δυνατοτήτων που αυτός παρέχει. Τα Συστήματα Κατανεμημένης Τεχνητής Νοημοσύνης (DAI Systems), τα οποία αποτελούνται από ένα σύνολο από κατανεμημένους, έξυπνους πράκτορες λήψης απόφασης, χαρακτηρίζονται από δύο βασικά ζητήματα: το συντονισμό και τη συνεργασία. Σε πολλές περιπτώσεις μελέτης προβλημάτων που σχετίζονται με τη διαχείριση των πόρων και το πώς αυτή μπορεί να γίνει με ευφυή τρόπο για την επίτευξη ενός επιθυμητού αποτελέσματος, γίνεται η προσπάθεια να δοθεί απάντηση στο εξής ερώτημα: πώς ένα σύνολο από γεωγραφικά κατανεμημένους πράκτορες μπορεί να κατανείμει σωστά μία συλλογή από εργασίες (tasks) για την επίτευξη ενός στόχου, λύνοντας τις οποιεσδήποτε συγκρούσεις (conflicts) που μπορεί να εμφανιστούν, αλλά και ικανοποιώντας συγκεκριμένους περιορισμούς που πιθανώς να έχουν τεθεί στα πλαίσια του συγκεκριμένου προβλήματος.

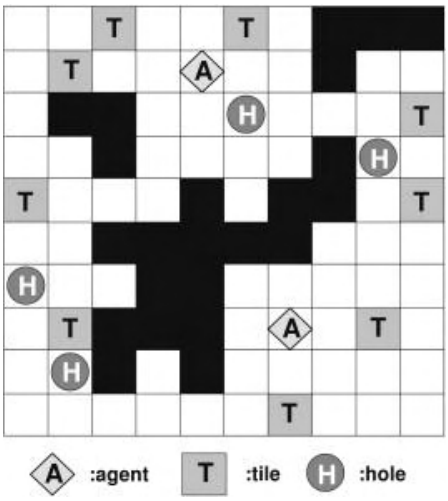
2.5 Παραδείγματα εφαρμογών και συστημάτων

Η ιδέα μελέτης του προβλήματος της διαχείρισης πόρων και ο τρόπος προσομοίωσης του στα πλαίσια της παρούσας εργασίας, καθώς και η ιδέα σχεδίασης του αλγορίθμου για την αντιμετώπισή του, προήλθαν έπειτα από αναζήτηση και εύρεση σε σχετική βιβλιογραφία άλλων εφαρμογών που χρησιμοποιούν την τεχνολογία των πολυπρακτορικών συστημάτων είτε για το ίδιο πρόβλημα είτε για άλλα προβλήματα που συναντώνται σε ένα πλήθος από διαφορετικούς τομείς. Στις επόμενες ενότητες, γίνεται αναφορά σε παραδείγματα τέτοιου είδους εφαρμογών και συστημάτων.

2.5.1 Αναφορά στο Tileworld

Το Tileworld είναι ένα σύστημα το οποίο αποτελείται από προσομοιωμένους πράκτορες-ρομπότ και από ένα προσομοιωμένο περιβάλλον, το οποίο είναι δυναμικό και μη προβλέψιμο. Οι πράκτορες, καθώς και το περιβάλλον, μπορούν να παραμετροποιηθούν, δίνοντας έτσι τη δυνατότητα στους χρήστες τους να ελέγχουν συγκεκριμένα χαρακτηριστικά του καθενός.

Το περιβάλλον του Tileworld είναι ένα πλέγμα που μοιάζει με σκακιέρα (chessboard-like grid), στο οποίο υπάρχουν πράκτορες (agents), πλακίδια (tiles), εμπόδια (obstacles) και τρύπες (holes). Στην *Εικόνα 2.2* παρουσιάζεται μία τυπική αρχική κατάσταση για το σύστημα Tileworld. Ένας πράκτορας είναι ένα μοναδιαίο τετράγωνο, με τη δυνατότητα να μετακινείται πάνω, κάτω, δεξιά και αριστερά κατά ένα κελί (cell) τη φορά. Ένα πλακίδιο είναι ένα μοναδιαίο τετράγωνο που συμπεριφέρεται σαν ένα πραγματικό πλακίδιο, δηλαδή μπορεί να ολισθαίνει. Επίσης, γραμμές από πλακίδια μπορούν να ωθούνται από έναν πράκτορα. Ένα εμπόδιο είναι μία συλλογή από κελιά του πλέγματος (grid cells) τα οποία δεν μετακινούνται. Μία τρύπα είναι ένα κελί του πλέγματος, το οποίο μπορεί να «γεμίσει» από ένα πλακίδιο, όταν αυτό μετακινείται στην κορυφή του κελιού με την τρύπα. Με αυτόν τον τρόπο, το πλακίδιο και η τρύπα εξαφανίζονται, αφήνοντας στη θέση τους ένα κενό κελί. Εάν μία τρύπα γεμίσει, ο πράκτορας παίρνει πόντους για την ενέργειά του. Επομένως, στόχος του κάθε πράκτορα είναι να βρίσκει πλακίδια και να «γεμίζει» τρύπες για να μαζέψει όσο το δυνατόν περισσότερους πόντους.



Εικόνα 2.2

Μία προσομοίωση του Tileworld λαμβάνει χώρα δυναμικά: ξεκινά από μία κατάσταση η οποία παράγεται τυχαία από τον προσομοιωτή, σύμφωνα με ένα σύνολο από παραμέτρους, και αλλάζει συνεχώς με την πάροδο του χρόνου. Τα αντικείμενα (τρύπες, πλακίδια και εμπόδια) εμφανίζονται και εξαφανίζονται σε ποσοστά τα οποία υπολογίζονται από παραμέτρους που θέτει ο ερευνητής, ενώ την ίδια στιγμή ο πράκτορας μετακινείται στο χώρο και σπρώχνει πλακίδια σε τρύπες. Το δυναμικό στοιχείο που χαρακτηρίζει μία προσομοίωση του συστήματος Tileworld είναι αυτό που το ξεχωρίζει από πολλούς προγενέστερους τομείς που έχουν χρησιμοποιηθεί για τη μελέτη AI σχεδιασμού, όπως π.χ. blocks-world.

Το σύστημα Tileworld αποτελεί μία αφαίρεση του Robot Delivery Domain, στο οποίο ένας πράκτορας περιφέρεται στο χώρο ενός γραφείου, μεταφέροντας μηνύματα και αντικείμενα ως απάντηση σε αιτήσεις ανθρώπων. Υπάρχει μία στενή αντιστοιχία ανάμεσα στους δύο τομείς: η ύπαρξη μιας τρύπας αντιστοιχεί σε μία αίτηση, η τρύπα αντιπροσωπεύει μία τοποθεσία διανομής, τα πλακίδια αντιστοιχούν σε μηνύματα ή αντικείμενα που μεταφέρονται, το πλέγμα αντιστοιχεί στους διαδρόμους του γραφείου και ο χρόνος προσομοίωσης αντιστοιχεί στον πραγματικό χρόνο.

Το Tileworld δημιούργησε την ιδέα για τον τρόπο προσομοίωσης του προβλήματος αυτής της εργασίας. Όπως θα παρουσιαστεί πιο αναλυτικά σε επόμενα κεφάλαια, το περιβάλλον προσομοιώνεται ως ένα δισδιάστατο πλέγμα και οι οντότητες που ζουν σε αυτό (πράκτορες και καταναλωτές) βρίσκονται ανά πάσα στιγμή σε κάποιο από τα κελιά του πλέγματος. Οι πόροι του περιβάλλοντος προσομοιώνονται σαν ακέραιες ποσότητες που είναι κατανεμημένες στα κελιά του. Πριν την εκτέλεση του αλγορίθμου, το περιβάλλον βρίσκεται σε μία αρχική κατάσταση, ενώ μετά την εκτέλεσή του προκύπτει μία νέα κατάσταση για το περιβάλλον, η οποία δεν είναι κάθε φορά η ίδια (λόγω της τυχαιότητας που έχει εισαχθεί σε κάποια στοιχεία).

2.5.2 Αναφορά σε άλλα συστήματα

Τα πολυπρακτορικά συστήματα έχουν εφαρμογές σε πάρα πολλούς διαφορετικούς τομείς και ανθρώπινες δραστηριότητες όπως: συντήρηση αεροσκαφών, ηλεκτρονική αγορά βιβλίων, στρατιωτική εκκαθάριση ναρκοπεδίων, ασύρματη συνεργασία και επικοινωνία, στρατιωτικός σχεδιασμός υλικοτεχνικής υποστήριξης, διαχείριση της αλυσίδας εφοδιασμού, διαχείριση υδάτινων πόρων, μοντελοποίηση οίκο-περιβαλλοντικών προβλημάτων, προσομοίωση και διαχείριση οικοσυστήματος, κλπ. Ενδεικτικά αναφέρονται κάποια παραδείγματα συστημάτων, τα οποία στηρίζονται σε αρχές της Τεχνητής Νοημοσύνης και εκμεταλλεύονται τις ιδιότητες και τα χαρακτηριστικά των πρακτόρων.

- *Κατανεμημένη πλατφόρμα εναέριου πολέμου* [N.V. Findler και G.D. Elder, 1995]: Πρόκειται για ένα ευέλικτο υπολογιστικό σύστημα το οποίο χρησιμοποιείται για τη διεξαγωγή έρευνας σχετικής με τη χρήση κατανεμημένων έξυπνων πρακτόρων για την επίλυση προβλημάτων κατανομής πόρων μέσα σε ένα περιβάλλον εναέριου πολέμου. Η πλατφόρμα έχει αναπτυχθεί και χρησιμοποιείται για την απόκτηση ποσοτικών και ποιοτικών αποτελεσμάτων που αποδεικνύουν την εφικτότητα της

προσέγγισης των κατασκευαστών της σχετικά με τη δυναμική κατανομή πόρων και εργασιών.

Το πρόβλημα που εξετάζεται είναι πώς οι κατανεμημένοι πράκτορες θα μπορέσουν να κατανείμουν σωστά εργασίες μεταξύ τους ικανοποιώντας συγκεκριμένους περιορισμούς, αλλά και πώς θα επιλύσουν τυχόν διενέξεις που προκύπτουν ώστε να ολοκληρώσουν το σύνολο των εργασιών επιτυχώς. Για την κατανομή των εργασιών μεταξύ των πρακτόρων, χρησιμοποιείται μία καινούρια μορφή διαπραγμάτευσης, κατά την οποία οι πράκτορες επιβάλλουν την προσφορά τους για κάποια εργασία, βασιζόμενοι στις ικανότητές τους. Ο πράκτορας που είναι ο πιο κατάλληλος αναλαμβάνει την εργασία και οι υπόλοιπες μπαίνουν πάλι σε διαδικασία διαπραγμάτευσης, καθώς οι άλλοι πράκτορες βελτιώνουν συνεχώς τις προσφορές τους με την πάροδο του χρόνου. Όσο για την επίλυση πιθανών διενέξεων και συγκρούσεων, έχει υλοποιηθεί μία τεχνική η οποία ακολουθεί έναν επαναληπτικό τρόπο επίλυσης και βασίζεται σε μία ιεραρχία από προτεραιότητες που αποδίδονται στις εργασίες.

- *ARIES* [Villa et al., BioEcon 2009]: Πρόκειται για μία καινούρια μεθοδολογία και διαδικτυακή εφαρμογή που έχει σαν στόχο να εκτιμήσει τις υπηρεσίες οικοσυστήματος (Ecosystem Services) και να προβάλλει την αξία τους στους ανθρώπους, ώστε να κάνει τις περιβαλλοντικές αποφάσεις πιο εύκολες και πιο αποτελεσματικές. Δημιουργώντας πιθανοτικά μοντέλα (χωρικά δίκτυα Bays) πρόβλεψης και χρήσης των ES σε μία περιοχή ενδιαφέροντος, και αντιστοιχίζοντας τις πραγματικές φυσικές ροές από τα οφέλη που αυτά παρέχουν προς τους δικαιούχους τους, το ARIES συμβάλλει στην κατανόηση και την ποσοτικοποίηση περιβαλλοντικών εκτιμήσεων και στην ανακάλυψη των παραγόντων που επηρεάζουν την αξία τους βάσει ρητών αναγκών και προτεραιοτήτων.

- *Σύστημα MCOE* [D.A. Callegari και F. Moreira de Oliveira, 2000]: Τα έξυπνα διδακτικά συστήματα αλλά και τα έξυπνα περιβάλλοντα μάθησης αποτελούν μοντέλα συστημάτων διδασκαλίας-μάθησης βασισμένα σε υπολογιστές. Η χρήση των πολυπρακτορικών συστημάτων είναι ολοένα και αυξανόμενη σε αυτό τον τομέα και πολλές τεχνικές Machine Learning εφαρμόζονται σε τέτοιου είδους συστήματα. Μία από τις πιο υποσχόμενες τεχνικές είναι η λεγόμενη Reinforcement Learning (RL), και ειδικά για περιβάλλοντα πολλαπλών πρακτόρων. Το σύστημα MCOE είναι ένα έξυπνο και συνεργατικό περιβάλλον μάθησης για διδασκαλία Οικολογίας σε παιδιά. Έχει αναπτυχθεί πάνω σε μία πολυπρακτορική αρχιτεκτονική, συμπεριλαμβάνοντας πράκτορες δράσης και γνώσης. Το περιβάλλον του παιχνιδιού είναι ένα τεχνητό, απλοποιημένο οικοσύστημα που αποτελείται από μία λίμνη με ζωντανά στοιχεία (ψάρια, μικροοργανισμοί, φυτά), ενώ κάποια ρυπογόνα στοιχεία ενεργοποιούνται τυχαία σε διάφορες χρονικές στιγμές. Οι πράκτορες δράσης χρησιμοποιούνται για να προσομοιώσουν στοιχεία του περιβάλλοντος, όπως είναι τα φυτά, τα ζώα, αλλά και τα ρυπογόνα στοιχεία. Οι πράκτορες γνώσης χρησιμοποιούνται για να προσομοιώσουν τους πράκτορες που αντιστοιχούν στους μαθητές και στους δασκάλους. Στόχος είναι οι μαθητές να συνεργαστούν και να

επιλέξουν τα κατάλληλα εργαλεία , ώστε να αντιμετωπίσουν τα ρυπογόνα στοιχεία και να διατηρήσουν την ισορροπία του οικοσυστήματος.

- *WRCAASM* [Z. Jianshi, W. Zhongjing, W. Wenbin, 2003]: Το μοντέλο WRCAASM (Water Resource Complex Adaptive Allocation System Model) στηρίζεται στην ακόλουθη ιδέα: περιέχει ως βασικά στοιχεία προσαρμοστικούς πράκτορες με ικανότητα αυτο-μελέτης και κατασκευάζεται πάνω σε κοινωνικά και οικονομικά πλαίσια. Ερευνά τους κανόνες μακροοικονομικής εξέλιξης, μέσα από προσομοιώσεις των συμπεριφορών που σχετίζονται με την χρήση υδάτινων πόρων, καθώς και τα αναπτυξιακά μοντέλα της κοινωνίας και της οικονομίας με το μοντέλο της κατανομής υδάτινων πόρων κάτω από συγκεκριμένους περιορισμούς. Σε αυτή την περίπτωση, οι πράκτορες χωρίζονται σε τρείς κατηγορίες: πράκτορες κοινωνικής παραγωγής, κοινωνικής διαχείρισης και οικολογίας. Κάθε μία από αυτές τις κατηγορίες προσομοιώνεται κατάλληλα για τους σκοπούς του συγκεκριμένου μοντέλου και τους αποδίδονται συγκεκριμένα χαρακτηριστικά και ιδιότητες.

3. Σχεδιασμός

3.1 Γενική περιγραφή του αλγορίθμου

Για την επίλυση του προβλήματος που μελετάται στην παρούσα εργασία, σχεδιάστηκε ένας αλγόριθμος. Στόχος αυτού του αλγορίθμου είναι η εκτέλεσή του να οδηγήσει στην ικανοποίηση των αναγκών που έχουν κάποιοι καταναλωτές για τον πόρο μέσω της συνεργασίας και της επικοινωνίας μεταξύ των πρακτόρων-εξυπηρετητών που υπάρχουν στο περιβάλλον. Για την πραγματοποίηση αυτής της δραστηριότητας, λαμβάνονται υπ' όψιν δύο κριτήρια: η απόσταση των πρακτόρων από κάθε καταναλωτή (πρωτεύον κριτήριο) και η εκάστοτε ποσότητα πόρου που έχουν οι πράκτορες στη διάθεσή τους ανά πάσα στιγμή (δευτερεύον κριτήριο).

Ο αλγόριθμος αποτελείται από 3 κύριες φάσεις, οι οποίες περιγράφονται πιο αναλυτικά στη συνέχεια. Έστω ότι η εξυπηρέτηση των αναγκών όλων των καταναλωτών αντιστοιχεί στην ολοκλήρωση της εργασίας T από τους πράκτορες του περιβάλλοντος.

Στην πρώτη φάση, γίνεται ανάθεση εργασίας στους δύο πράκτορες με κριτήριο την απόσταση και πιο συγκεκριμένα την άμεση γειτνίαση. Με άλλα λόγια, ο κάθε πράκτορας αναλαμβάνει να εξυπηρετήσει τους καταναλωτές που βρίσκονται το πολύ κατά μία θέση μακριά σε σχέση με τη θέση στην οποία βρίσκεται ο ίδιος. Για τους πράκτορες γίνεται η υπόθεση της ορθογώνιας κίνησης, δηλαδή ότι μπορούν να μετακινηθούν στο περιβάλλον πάνω, κάτω, δεξιά και αριστερά σε σχέση με τη θέση στην οποία βρίσκονται, αλλά όχι διαγώνια. Αυτό σημαίνει ότι οι καταναλωτές που θεωρούνται άμεσοι γείτονες ενός πράκτορα απέχουν απόσταση το πολύ ένα από αυτόν μόνο σε οριζόντια ή κατακόρυφη διεύθυνση και όχι σε διαγώνια. Σε αυτή τη φάση, λοιπόν, σε κάθε πράκτορα ανατίθεται η εργασία $T_j = \{t_i \mid t_i, i$ ένας άμεσος γείτονας του πράκτορα $j\}$, όπου t_i είναι μία υπο-εργασία που αντιστοιχεί στην εξυπηρέτηση του καταναλωτή i από τον πράκτορα j. Σε αυτό το σημείο, τονίζεται ότι στη γενική περίπτωση ισχύει $\text{Union}(T_j) \neq T$, για κάθε j που αποτελεί το αναγνωριστικό (id) ενός πράκτορα, δηλαδή η συνολική εργασία που πρέπει να πραγματοποιηθεί δεν ταυτίζεται απαραίτητα με το σύνολο των εργασιών που ανατίθενται στους πράκτορες σε αυτή τη φάση. Βέβαια, υπάρχει και η πιθανότητα να ισχύει η ισότητα ($\text{Union}(T_j) = T$), καθώς οι θέσεις των πρακτόρων και των καταναλωτών στο περιβάλλον, καθορίζονται τυχαία.

Στη δεύτερη φάση, στόχος είναι να ολοκληρωθεί το σύνολο των εργασιών $\{T- \text{Union}(T_j)\}$, δηλαδή να εξυπηρετηθούν όλοι εκείνοι οι καταναλωτές που δεν αποτελούσαν γείτονες κανενός πράκτορα στην πρώτη φάση κι έτσι εξακολουθεί να υπάρχει η ανάγκη τους για τον πόρο. Εάν οι πράκτορες έχουν ακόμα στη διάθεσή τους κάποια ποσότητα πόρου, σε αυτό το σημείο αρχίζει η επικοινωνία μεταξύ τους με ανταλλαγή μηνυμάτων. Μέσω αυτής γίνεται διαπραγμάτευση για το ποιος πράκτορας θα αναλάβει να εξυπηρετήσει τον κάθε καταναλωτή, βάσει των κριτηρίων που αναφέρθηκαν παραπάνω. Εάν, όμως, κάποιος από τους πράκτορες έχει πλέον στη διάθεση του μηδενική ποσότητα πόρου δεν θα ξεκινήσει καθόλου ή θα διακοπεί η επικοινωνία μεταξύ τους, καθώς δεν θα έχει νόημα κανενός είδους διαπραγμάτευση.

Η διαπραγμάτευση έχει ως στόχο την κατανομή της συνολικής εργασίας ανάμεσα στους πράκτορες, λαμβάνοντας υπ' όψιν κάθε φορά την απόσταση και τη διαθέσιμη ποσότητα πόρου. Αυτή η διαδικασία απαιτεί τη συνεργασία και το συντονισμό των πρακτόρων. Υπάρχει ένα ευρύ φάσμα τεχνικών μέσω των οποίων αυτό μπορεί να επιτευχθεί. Κάποιες τεχνικές είναι πιο κεντρικοποιημένες, καθώς υπάρχει ένας συγκεκριμένος, μοναδικός πράκτορας (master agent) ο οποίος ενεργεί ως διαχειριστής (contractor) της συνολικής εργασίας και αναλαμβάνει να αναθέτει μερίδια αυτής στους υπόλοιπους πράκτορες. Άλλες τεχνικές βασίζονται σε στρατηγικές διαπραγμάτευσης και στη συλλογική λήψη απόφασης για την κατανομή εργασίας από όλους τους πράκτορες.

Στην παρούσα εργασία, η επίλυση του προβλήματος που μελετάται στηρίζεται στην ιδέα της διαπραγμάτευσης των πρακτόρων και στην ενίσχυση της συνεργατικής συμπεριφοράς, χρησιμοποιώντας και κάποια στοιχεία από τη λογική του πρωτοκόλλου Contract Net (CNET). Οι πράκτορες εκτιμούν την ικανότητά τους να ολοκληρώσουν μία εργασία (δηλαδή να εξυπηρετήσουν ένα συγκεκριμένο καταναλωτή) και ανταλλάσσουν αυτή την πληροφορία, η οποία αποτελεί στην ουσία την προσφορά (bid) του κάθε πράκτορα για την εργασία. Ο πράκτορας που θα έχει την καλύτερη προσφορά (best bid) είναι και αυτός που θα γίνει υπεύθυνος για την ολοκλήρωση της συγκεκριμένης εργασίας. Η καλύτερη προσφορά μπορεί να ισοδυναμεί, βάσει των κριτηρίων που έχουν αναφερθεί παραπάνω, είτε στη μικρότερη απόσταση ενός πράκτορα από ένα συγκεκριμένο καταναλωτή είτε στη μεγαλύτερη ποσότητα πόρου που ένας πράκτορας έχει στη διάθεσή του μία συγκεκριμένη χρονική στιγμή. Με αυτόν τον τρόπο, λοιπόν, η κατανομή εργασίας προκύπτει μετά από επικοινωνία των πρακτόρων και ως απόφαση του συνόλου τους και δεν βασίζεται στη δράση ενός μοναδικού πράκτορα-διαχειριστή, ο οποίος απλά συλλέγει τις προσφορές όλων των υπόλοιπων πρακτόρων και στη συνέχεια τους αναθέτει εργασίες βάσει αυτών. Στο σημείο αυτό πρέπει, επίσης, να τονιστεί ότι στην επίλυση του προβλήματος που μελετάται σε αυτήν την εργασία, οι πράκτορες έχουν ένα κοινό στόχο: την ικανοποίηση των αναγκών των καταναλωτών, και συνεργάζονται για να τον πετύχουν. Δεν υπάρχουν αντικρουόμενα συμφέροντα τα οποία επιδιώκουν να εξυπηρετήσουν οι πράκτορες προς όφελός τους, οπότε η επικοινωνία τους διέπεται από έναν συνεργατικό παρά ανταγωνιστικό χαρακτήρα.

Στην τρίτη και τελευταία φάση, στόχος είναι να εξυπηρετηθούν οι καταναλωτές των οποίων οι ανάγκες παρέμειναν ανικανοποίητες μετά το τέλος των δύο προηγούμενων φάσεων. Σε αυτό το σημείο, δεν υπάρχει επικοινωνία μεταξύ των πρακτόρων. Εάν υπάρχει ακόμα πόρος στη διάθεση των πρακτόρων, τότε ο καθένας από αυτούς προσπαθεί να εξυπηρετήσει καταναλωτές που αποτελούσαν αρμοδιότητα άλλου πράκτορα σε προηγούμενη φάση, αλλά λόγω έλλειψης της αναγκαίας ποσότητας πόρου δεν κατάφερε να ικανοποιήσει τις ανάγκες τους. Εάν κάποιος πράκτορας δεν έχει πλέον μονάδες πόρου στη διάθεσή του, οι υπόλοιποι (με μη μηδενική ποσότητα πόρου) θα πρέπει να αναλάβουν την εξυπηρέτηση των καταναλωτών που έχουν απομείνει. Τέλος, εάν κανένας πράκτορας δεν διαθέτει την αναγκαία ποσότητα πόρου για εξυπηρέτηση, πιθανόν να υπάρξουν καταναλωτές των οποίων οι ανάγκες δεν θα ικανοποιηθούν ποτέ.

Επομένως, αφού ολοκληρωθεί και η τρίτη φάση, ολοκληρώνεται στην ουσία και ο αλγόριθμος. Το αποτέλεσμα, σε αυτό το σημείο, μπορεί να είναι η μη ύπαρξη

καταναλωτών με ανικανοποίητες ανάγκες για τον πόρο ή η ύπαρξη κάποιων καταναλωτών που δεν έχουν εξυπηρετηθεί και δεν πρόκειται να εξυπηρετηθούν από κάποιον πράκτορα.

4. Υλοποίηση

Για την υλοποίηση του αλγορίθμου που περιγράφηκε προηγουμένως, ώστε να μελετηθεί σε ένα πιο πρακτικό επίπεδο η επίλυση του προβλήματος της διαχείρισης πόρων από ένα πολυπρακτορικό σύστημα, χρησιμοποιήθηκε η γλώσσα προγραμματισμού Java και επιλέχθηκε η πλατφόρμα ανάπτυξης πολυπρακτορικών εφαρμογών JADE (4.2.0 version). Η ανάπτυξη έγινε στο ολοκληρωμένο περιβάλλον ανάπτυξης εφαρμογών (integrated development environment - IDE) NetBeans 7.2 .

4.1 Πλατφόρμα Ανάπτυξης Πρακτόρων JADE

Το JADE (Java Agent DEvelopment Framework) είναι μία εφαρμογή ενδιάμεσου στρώματος (middleware) που αναπτύχθηκε πλήρως σε Java από την Telecom Italia Lab. Χρησιμοποιείται ως περιβάλλον ανάπτυξης κατανεμημένων πολυπρακτορικών εφαρμογών βασισμένων στην peer-to-peer αρχιτεκτονική επικοινωνίας. Η πληροφορία, οι πόροι και ο έλεγχος μπορούν να κατανεμηθούν πλήρως σε κινητά τερματικά (mobile terminals), όπως επίσης και σε υπολογιστές σταθερού δικτύου (fixed network). Το περιβάλλον μπορεί να εξελίσσεται δυναμικά με ομότιμους (peers), που στο JADE αποκαλούνται πράκτορες (agents), οι οποίοι εμφανίζονται και εξαφανίζονται από το σύστημα, ανάλογα με τις ανάγκες και τις απαιτήσεις της εκάστοτε εφαρμογής. Η επικοινωνία μεταξύ των ομοτίμων, ανεξάρτητα από το αν τρέχουν πάνω σε ασύρματο ή ενσύρματο δίκτυο, είναι απόλυτα συμμετρική, δίνοντας έτσι τη δυνατότητα στον καθέναν από αυτούς να έχει το ρόλο είτε του εκκινητή μίας συνομιλίας, είτε αυτού που ανταποκρίνεται σε μία πρόσκληση για συνομιλία που έχει λάβει από κάποιον άλλο.

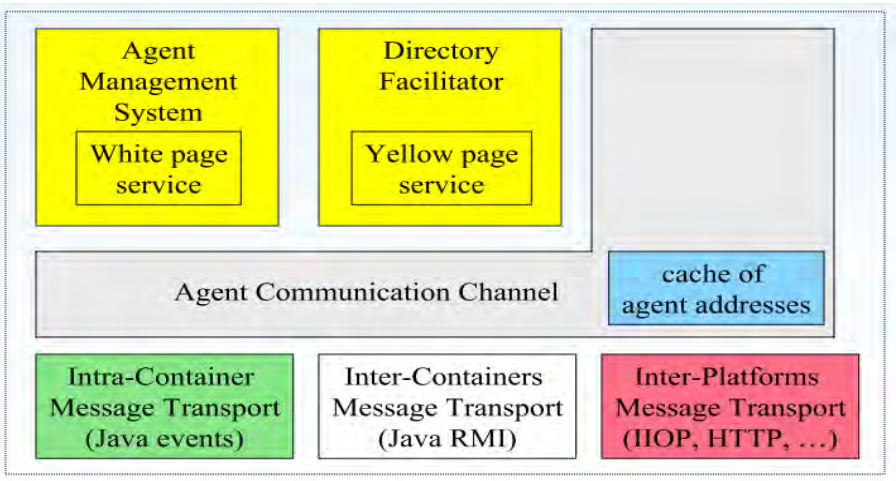
Το JADE στηρίζεται στις εξής βασικές αρχές:

- ***Διαλειτουργικότητα (interoperability)****:* Είναι συμβατό με το πρότυπο FIPA. Αυτό σημαίνει ότι οι πράκτορες που υποστηρίζονται από το JADE μπορούν να αλληλεπιδρούν με άλλους πράκτορες που ακολουθούν το ίδιο πρότυπο.
- ***Ομοιομορφία (uniformity) και μεταφερσιμότητα (portability):*** Παρέχει διεπαφές (APIs) ομοιογενείς και ανεξάρτητες από το υφιστάμενο δίκτυο και από την έκδοση της Java που χρησιμοποιείται. Αυτό έχει σαν αποτέλεσμα να είναι πολύ εύκολη η μεταφορά εφαρμογών του JADE από μία υπολογιστική πλατφόρμα σε μία άλλη.
- ***Ευκολία χρήσης (easy-to-use):*** Η πολυπλοκότητα του JADE κρύβεται πίσω από ένα απλό και φιλικό περιβάλλον διεπαφής της πλατφόρμας.
- ***Φιλοσοφία pay-as-you-go:*** Οι προγραμματιστές των εφαρμογών δεν χρειάζεται να γνωρίζουν και να χρησιμοποιούν όλα τα τεχνικά χαρακτηριστικά που παρέχονται από την πλατφόρμα, γεγονός που δεν αυξάνει την υπολογιστική πολυπλοκότητα.

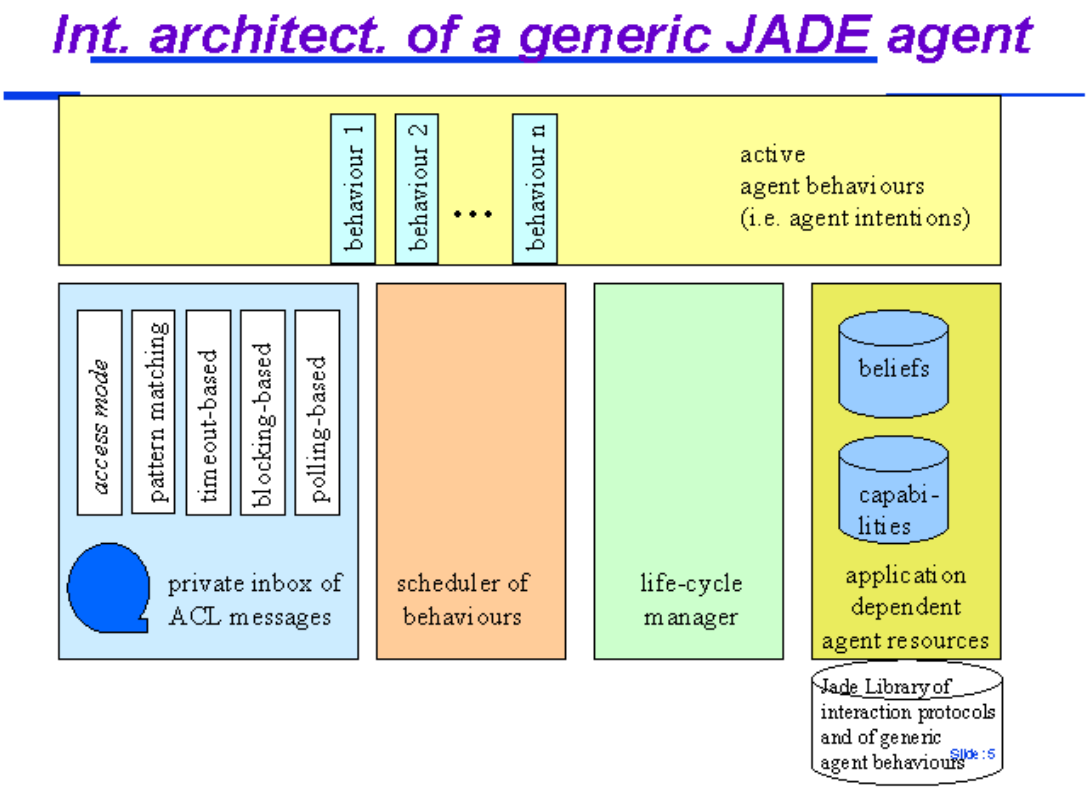
Στόχος του JADE είναι να απλοποιήσει την ανάπτυξη εφαρμογών πολυπρακτορικών συστημάτων, διασφαλίζοντας συμβατότητα με το πρότυπο FIPA, μέσα από ένα σύνολο υπηρεσιών συστήματος και πρακτόρων. Για να πετύχει αυτό το στόχο, το JADE προσφέρει στον προγραμματιστή την ακόλουθη λίστα από χαρακτηριστικά και χρήσιμα εργαλεία:

- Μία πλατφόρμα πρακτόρων, συμβατή με το πρότυπο FIPA, η οποία περιλαμβάνει τρεις πράκτορες που ενεργοποιούνται αυτόματα κατά την έναρξη λειτουργίας της πλατφόρμας: AMS (Agent Management System), DF (Directory Facilitator) και ACC (Agent Communication Channel).
- Δυνατότητα κατανομής της πλατφόρμας σε ένα πλήθος από hosts. Μόνο μία Java εφαρμογή και κατ' επέκταση μόνο μία JVM (Java Virtual Machine) εκτελείται σε κάθε host. Οι πράκτορες υλοποιούνται σαν Java νήματα (threads) και τα γεγονότα (Java events) χρησιμοποιούνται για αποτελεσματική επικοινωνία μεταξύ των πρακτόρων που βρίσκονται στον ίδιο host.
- Υποστήριξη και διαχείριση μέσω γραφικού περιβάλλοντος του κύκλου ζωής του κάθε πράκτορα.
- Μηχανισμό μεταφοράς και διεπαφή για την αποστολή και παραλαβή μηνυμάτων μεταξύ των πρακτόρων.
- Υποστήριξη διαφόρων πρωτοκόλλων επικοινωνίας (FIPA ACL) και πρωτοκόλλων διαπραγμάτευσης (Contract Net).
- Μεταφορά ACL μηνυμάτων μέσα στην ίδια πλατφόρμα πρακτόρων. Τα μηνύματα μεταφέρονται κωδικοποιημένα ως Java αντικείμενα, αντί ως αλφαριθμητικά (strings). Αν ο αποστολέας και ο παραλήπτης δεν ανήκουν στην ίδια πλατφόρμα, το μήνυμα μετατρέπεται αυτόματα από/σε μία μορφή συμβατή με το πρότυπο FIPA. Με αυτόν τον τρόπο, αυτή η μετατροπή δεν είναι ορατή στους υλοποιητές των πρακτόρων, οι οποίοι έχουν πάντα να διαχειριστούν την ίδια κλάση από Java αντικείμενα.
- Υπηρεσίες καταλόγου (white, yellow pages) για την εύρεση πρακτόρων και εφαρμογών.
- Υπηρεσία ονοματοδοσίας σε συμβατότητα με το πρότυπο FIPA: κατά την εκκίνηση, οι πράκτορες αποκτούν ένα GUID (Globally Unique Identifier) από την πλατφόρμα, δηλαδή ένα αναγνωριστικό που τους χαρακτηρίζει.
- Γραφικό περιβάλλον για το χρήστη (GUI) για τη διαχείριση πολλών πρακτόρων και πλατφορμών από τον ίδιο πράκτορα. Η δραστηριότητα της κάθε πλατφόρμας μπορεί να παρακολουθείται και να καταγράφεται.
- Υποστήριξη της μετακίνησης πρακτόρων (κώδικα, κατάσταση εκτέλεσης) σε διαφορετικές τοποθεσίες (containers, platforms).

Στις παρακάτω εικόνες φαίνεται η εσωτερική αρχιτεκτονική του JADE run-time (*Εικόνα 4.1*) και ενός γενικού JADE πράκτορα (*Εικόνα 4.2*).

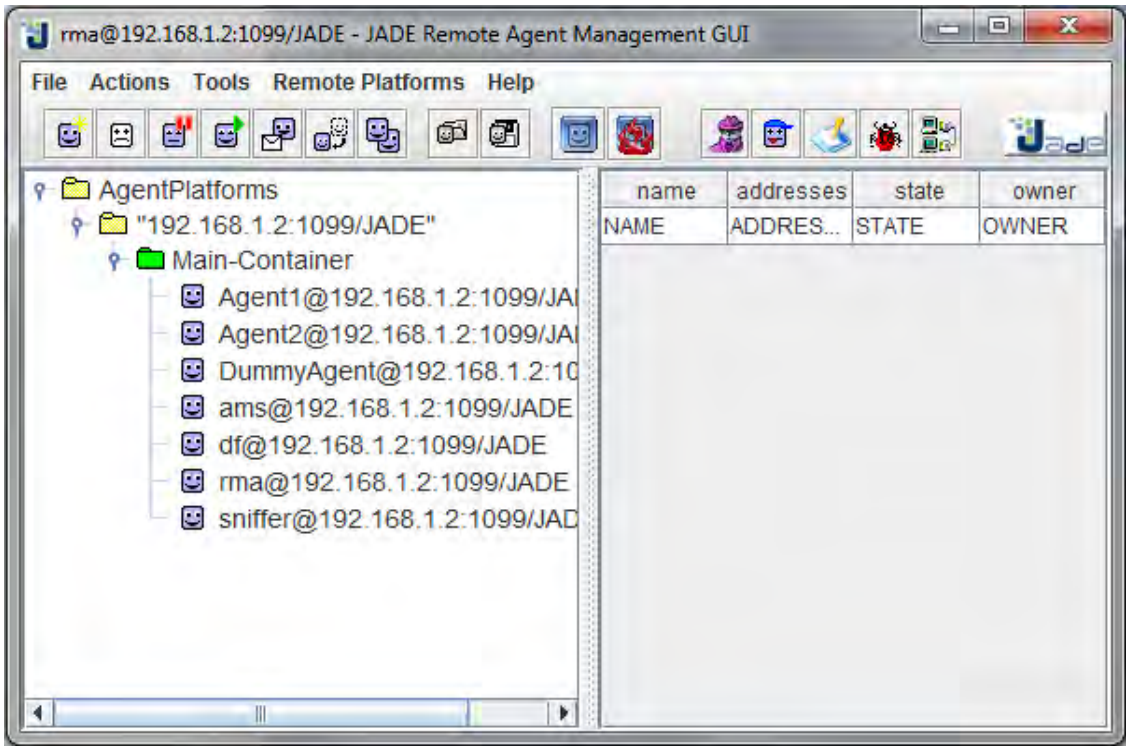


Εικόνα 4.1



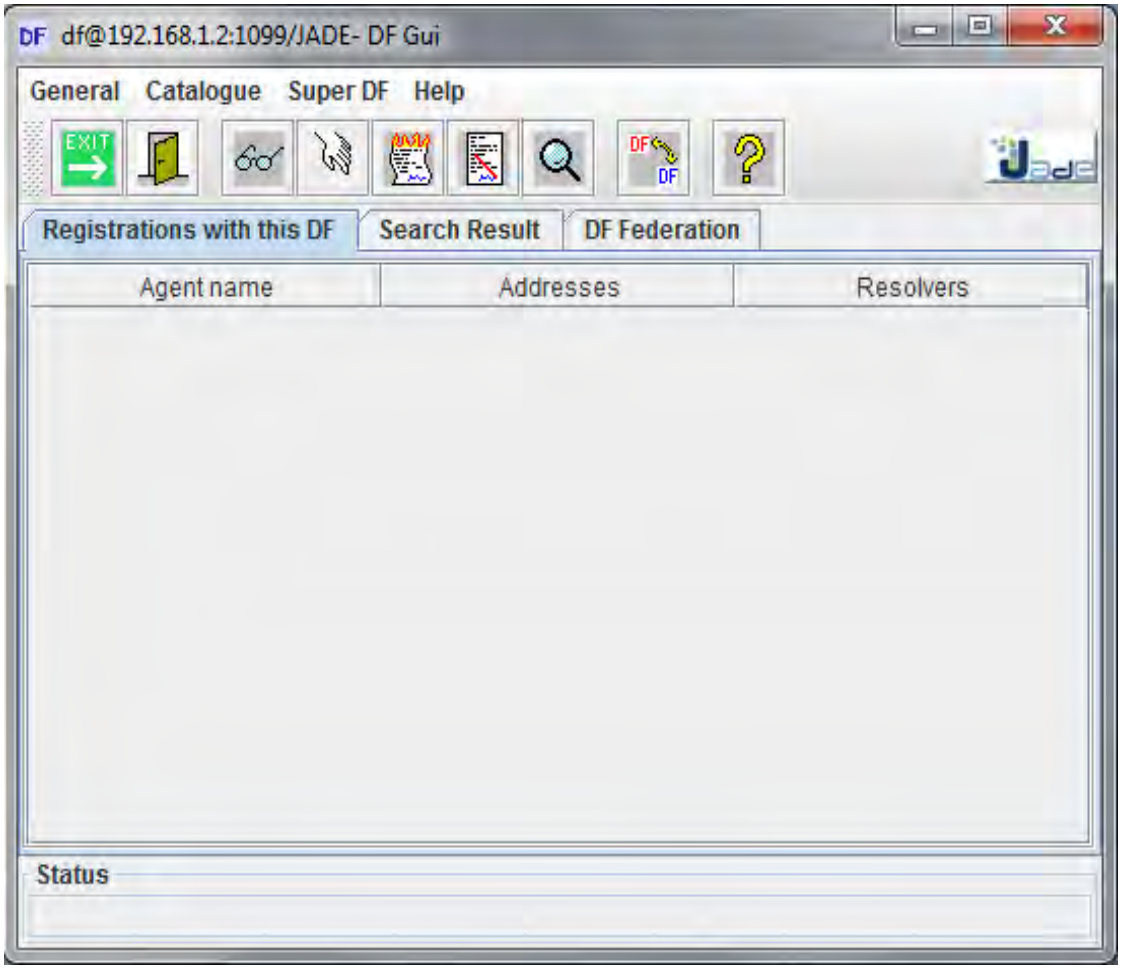
Εικόνα 4.2

Το GUI που παρέχεται από την πλατφόρμα φαίνεται στην *Εικόνα 4.3*. Διευκολύνει την απομακρυσμένη διαχείριση και τον έλεγχο της κατάστασης των πρακτόρων, επιτρέποντας το σταμάτημα και την επανεκκίνησή τους. Επίσης, επιτρέπει τη δημιουργία και την έναρξη εκτέλεσης ενός πράκτορα σε έναν απομακρυσμένο host, καθώς και τον έλεγχο απομακρυσμένων πλατφορμών συμβατών με το πρότυπο FIPA.



Εικόνα 4.3 RMA GUI

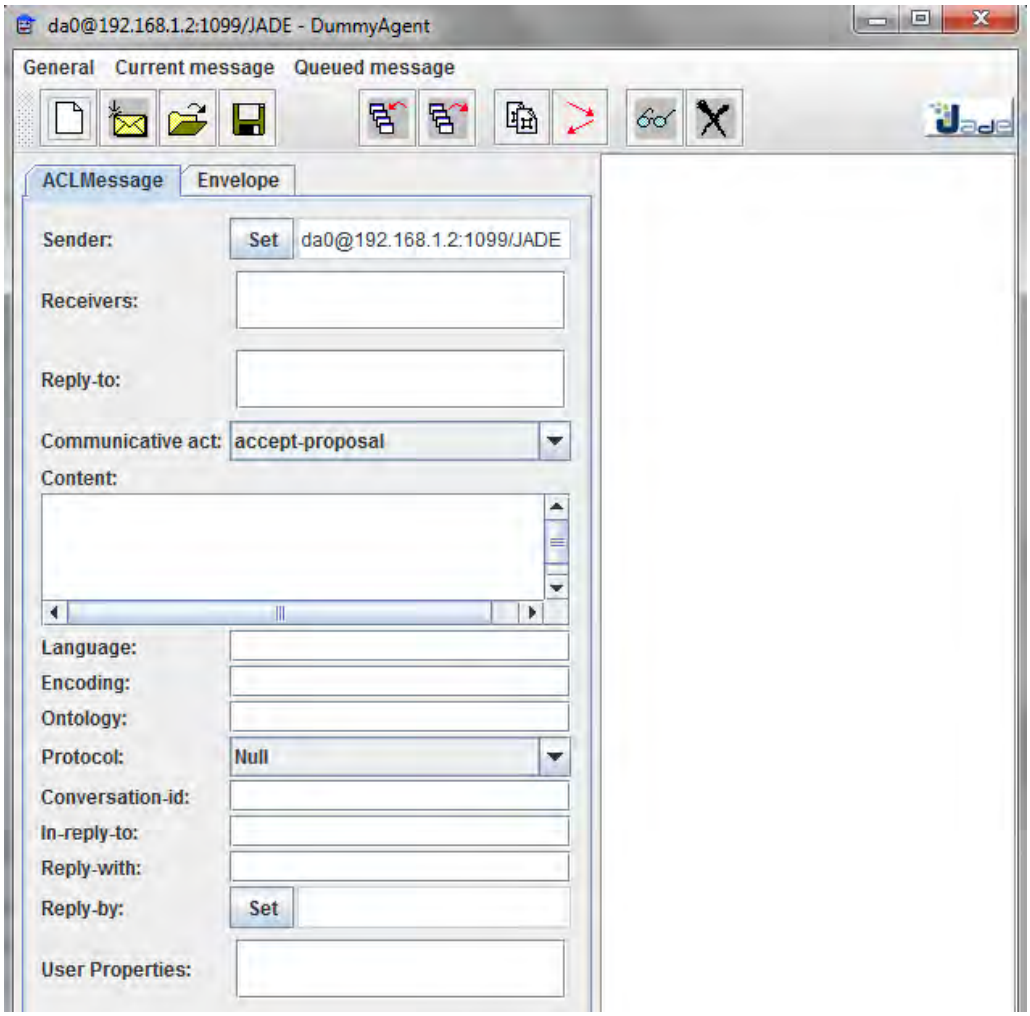
Μέσα από τις δυνατότητες που παρέχει το RMA, μπορεί να γίνει εκκίνηση ενός GUI του DF (Directory Facilitator) πράκτορα (*Εικόνα 4.4*). Αυτός ο πράκτορας παρέχει την υπηρεσία yellow pages, μέσω της οποίας ένας πράκτορας μπορεί να βρει άλλους πράκτορες που παρέχουν τις υπηρεσίες που αυτός χρειάζεται για να πετύχει τους στόχους του. Μέσω αυτής της διεπαφής, ο πράκτορας μπορεί να αναζητήσει, να δει, να τροποποιήσει τις περιγραφές των πρακτόρων που έχουν εγγραφεί, να πραγματοποιήσει εγγραφή ή να διαγράψει κάποιον πράκτορα.



Εικόνα 4.4 DF GUI

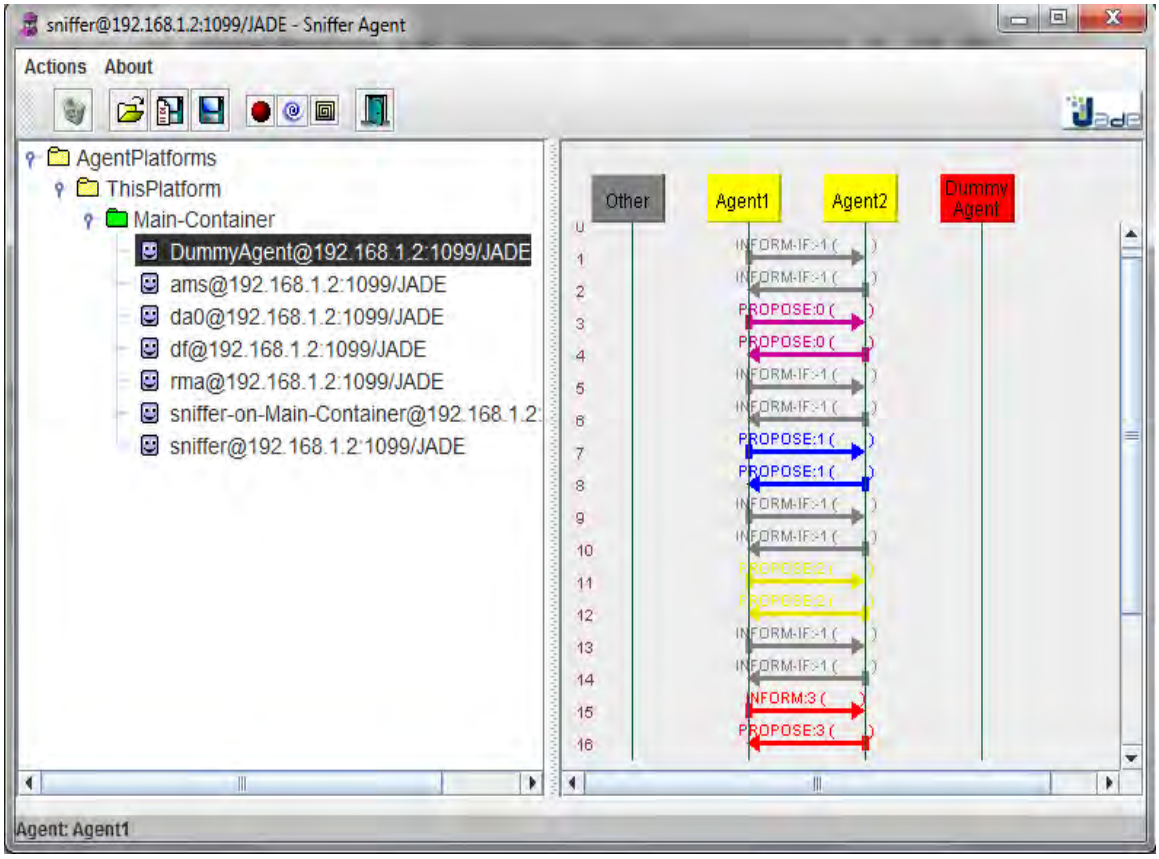
Ένα σύνολο από γραφικά εργαλεία έχει υλοποιηθεί με στόχο την υποστήριξη της φάσης της εκσφαλμάτωσης και διαχείρισης της εφαρμογής. Στη συνέχεια αναφέρονται μερικά από αυτά, ενώ οι αντίστοιχες εικόνες απεικονίζουν το περιβάλλον τους.

- ***Dummy Agent***
  Είναι ένα πολύ χρήσιμο εργαλείο για την παρακολούθηση της ανταλλαγής μηνυμάτων μεταξύ των πρακτόρων. Προσφέρει δυνατότητα σύνθεσης, επεξεργασίας και αποστολής ACL μηνυμάτων σε πράκτορες, παραλαβής μηνυμάτων και αποθήκευσης/φόρτωσης μηνυμάτων σε/από τον δίσκο. Το περιβάλλον του φαίνεται στην *Εικόνα 4.5.*



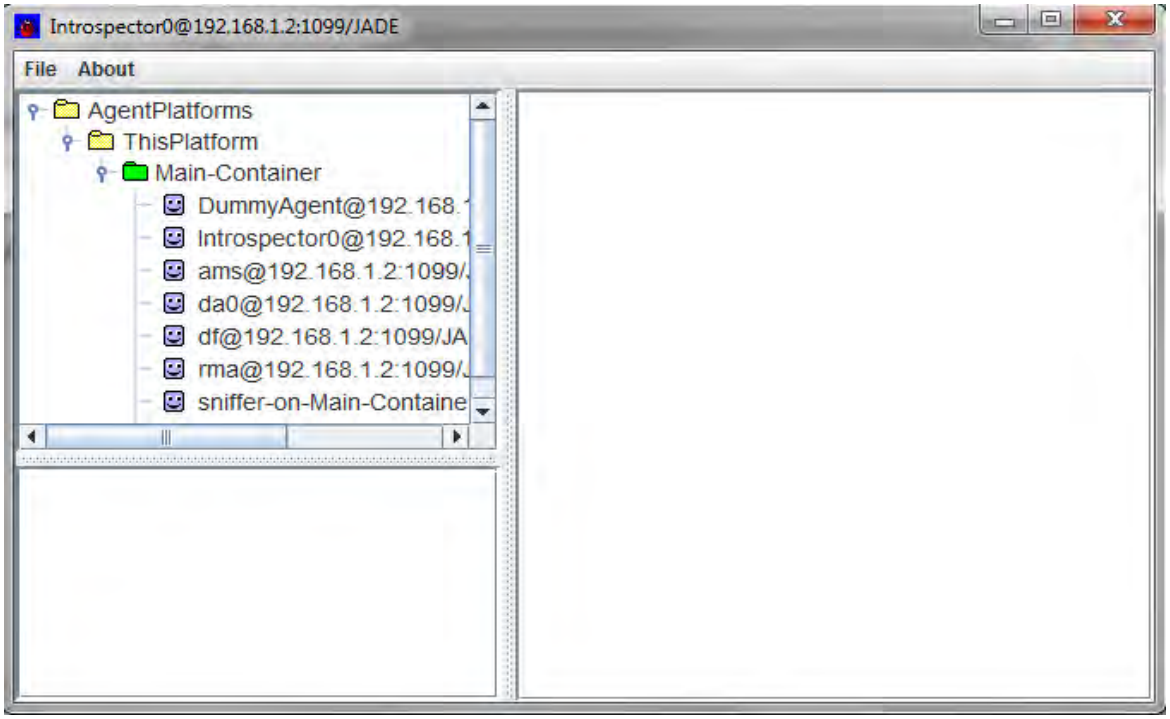
Εικόνα 4.5 Dummy Agent

- ***Sniffer Agent***
  Επιτρέπει την παρακολούθηση των μηνυμάτων που ανταλλάσσονται σε μία JADE πλατφόρμα πρακτόρων. Όταν ένας χρήστης αποφασίσει να παρακολουθήσει έναν πράκτορα, κάθε μήνυμα που προορίζεται για αυτόν ή που προέρχεται από αυτόν, παρουσιάζεται στο παράθυρο του πράκτορα sniffer. Ο χρήστης μπορεί να δει, να αποθηκεύσει και να φορτώσει κάθε μήνυμα για μεγαλύτερη ανάλυση. Το περιβάλλον του φαίνεται στην *Εικόνα 4.6*.



Εικόνα 4.6 Sniffer Agent

- ***Introspector Agent:***
  Επιτρέπει την παρακολούθηση και τον έλεγχο του κύκλου ζωής ενός πράκτορα και των μηνυμάτων που ανταλλάσσει (τις ουρές των απεσταλμένων και των ληφθέντων μηνυμάτων). Το περιβάλλον του απεικονίζεται στην *Εικόνα 4.7*.



Εικόνα 4.7 Introspector Agent

4.2 Η γλώσσα επικοινωνίας ACL

Το Ίδρυμα για Ευφυείς Φυσικούς Πράκτορες (Foundation for Intelligent Physical Agents, FIPA) ασχολήθηκε με την ανάπτυξη προτύπων για συστήματα πρακτόρων. Αποτέλεσμα των προσπαθειών ήταν η ανάπτυξη μίας γλώσσας επικοινωνίας πρακτόρων που αποκαλείται FIPA ACL (FIPA, 1999), η οποία επιφανειακά παρουσιάζει πολλές ομοιότητες με την KQML. Είναι μία γλώσσα που βασίζεται σε μηνύματα, χρησιμοποιεί 22 τελεστικά που ορίζουν την επιδιωκόμενη ερμηνεία των μηνυμάτων, και δεν υπαγορεύει καμία συγκεκριμένη γλώσσα για το περιεχόμενο των μηνυμάτων.

Ένα FIPA ACL μήνυμα περιλαμβάνει ένα σύνολο από μία ή περισσότερες παραμέτρους. Το ποιες παράμετροι χρειάζονται για την αποτελεσματική επικοινωνία μεταξύ πρακτόρων εξαρτάται από την εκάστοτε κατάσταση. Η μόνη παράμετρος που είναι απαραίτητη σε όλα τα μηνύματα είναι το τελεστικό (performative), ενώ αναμενόμενο θεωρείται ότι όλα τα μηνύματα θα έχουν αποστολέα (sender), παραλήπτη (receiver) και περιεχόμενο (content). Στον *Πίνακα 4.1* παρουσιάζεται το σύνολο όλων των πιθανών παραμέτρων που μπορεί να έχει ένα ACL μήνυμα:

Parameter	Category of Parameters
performative	Type of communicative acts
sender	Participant in communication
receiver	Participant in communication
reply-to	Participant in communication
content	Content of message
language	Description of Content
encoding	Description of Content
ontology	Description of Content
protocol	Control of conversation
conversation-id	Control of conversation
reply-with	Control of conversation
in-reply-to	Control of conversation
reply-by	Control of conversation

Πίνακας 4.1 Παράμετροι FIPA ACL μηνύματος

Από τα 22 τελεστικά που παρέχονται από τη γλώσσα επικοινωνίας FIPA ACL και των οποίων η σημασία περιγράφεται αναλυτικά στο κείμενο FIPA Communicative Art Library Specification [FIPA00037], παρακάτω αναφέρονται μόνο εκείνα που έχουν χρησιμοποιηθεί στα πλαίσια της παρούσας εργασίας για τους σκοπούς του αλγορίθμου. Πιο συγκεκριμένα:

- ***INFORM-IF***: Ένα τέτοιο μήνυμα υποδηλώνει αν μία πρόταση είναι αληθής ή ψευδής. Στον αλγόριθμο χρησιμοποιείται για να δηλώνουν οι πράκτορες πριν την έναρξη κάθε γύρου επικοινωνίας και διαπραγμάτευσης, αν είναι αληθές ή ψευδές το ότι η ποσότητα του πόρου που έχουν στη διάθεσή τους είναι ακόμη μη μηδενική.

- ***INFORM***: Με ένα τέτοιο μήνυμα ο αποστολέας θέλει να πληροφορήσει τον παραλήπτη ότι μία συγκεκριμένη πρόταση είναι αληθής. Στον αλγόριθμο, ένας πράκτορας στέλνει ένα μήνυμα τέτοιου τύπου σε έναν άλλο για να τον ενημερώσει ότι σε ένα γύρο διαπραγμάτευσης που εξετάζεται ένας συγκεκριμένος καταναλωτής, η ποσότητα πόρου που έχει στη διάθεσή του δεν επαρκεί για να τον εξυπηρετήσει ακόμα κι αν αποτελεί δική του αρμοδιότητα. Μηνύματα τέτοιου τύπου αποστέλλονται επίσης από τον πράκτορα Dummy Agent, με σκοπό όταν ζητηθεί η βοήθειά του, να ενημερώσει τους άλλους πράκτορες ποιος είναι αυτός που τελικά αναλαμβάνει την εξυπηρέτηση ενός καταναλωτή.

- ***PROPOSE***: Ένα τέτοιο μήνυμα αντιπροσωπεύει την υποβολή μίας πρότασης για την εκτέλεση μίας ενέργειας, υπό κάποιες προϋποθέσεις. Στην παρούσα εργασία, ένα μήνυμα τέτοιου τύπου στέλνεται από έναν πράκτορα και αντιπροσωπεύει την πρόταση-προσφορά του για τον καταναλωτή για τον οποίο γίνεται κάθε φορά διαπραγμάτευση.

- ***REQUEST***: Ο αποστολέας στέλνει ένα τέτοιο μήνυμα για να ζητήσει από τον παραλήπτη να εκτελέσει κάποια συγκεκριμένη ενέργεια. Στον αλγόριθμο της παρούσας εργασίας, υπάρχουν δύο περιπτώσεις που ένας πράκτορας στέλνει ένα τέτοιο μήνυμα. Η πρώτη περίπτωση είναι όταν ελέγχεται το πρωτεύον κριτήριο της διαπραγμάτευσης αλλά δεν μπορεί να παρθεί απόφαση. Τότε ο κάθε πράκτορας ζητάει από τον άλλο να του στείλει κάποιο καινούριο στοιχείο για να αποφασιστεί ποιος θα ικανοποιήσει τελικά τις ανάγκες ενός καταναλωτή. Η δεύτερη περίπτωση είναι όταν και ο έλεγχος του δευτερεύοντος κριτηρίου της διαπραγμάτευσης οδηγεί σε αδυναμία λήψης απόφασης για το ποιος πράκτορας θα αναλάβει την εξυπηρέτηση ενός συγκεκριμένου καταναλωτή. Τότε και οι δύο πράκτορες στέλνουν ένα τέτοιο μήνυμα στο πράκτορα Dummy Agent με σκοπό να αποφασίσει εκείνος και κατόπιν να τους ενημερώσει.

4.3 Αναλυτική περιγραφή της υλοποίησης

Η υλοποίηση, που πραγματοποιήθηκε για την πρακτική μελέτη του προβλήματος διαχείρισης πόρων από ένα πολυπρακτορικό σύστημα, περιλαμβάνει τα εξής βασικά στοιχεία:

- το περιβάλλον, το οποίο περιέχει τον διαθέσιμο πόρο
- τους καταναλωτές του πόρου, οι οποίοι ζουν μέσα στο περιβάλλον και έχουν ανάγκη για τον πόρο
- τους πράκτορες-εξυπηρετητές, οι οποίοι ζουν και κινούνται μέσα στο περιβάλλον, ενώ έχουν στη διάθεσή τους μονάδες πόρου με τις οποίες θα καλύψουν τις ανάγκες των καταναλωτών

και παρουσιάζει την επικοινωνία μεταξύ των πρακτόρων και τις ενέργειές τους με στόχο την καλύτερη δυνατή εξυπηρέτηση των αναγκών των καταναλωτών.

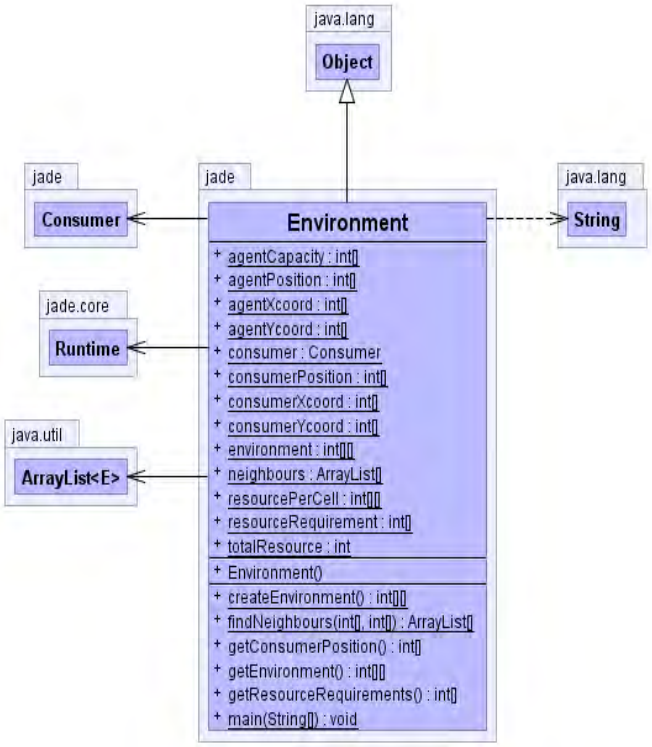
Τα παραπάνω στοιχεία δημιουργούνται αντίστοιχα από τις εξής τρεις Java classes: Environment class, Consumer class και Agents class. Η υλοποίηση χρησιμοποιεί επίσης τις κλάσεις GlobalData class και DummyAgent class. Η λειτουργία όλων αυτών των κλάσεων περιγράφεται αναλυτικά σε επόμενη ενότητα.

4.3.1 UML Class Diagrams

Αρχικά, στις *Εικόνες 4.8* έως *4.12,* παρουσιάζεται το UML Class Diagram για κάθε κλάση που χρησιμοποιείται στην υλοποίηση.

Environment class:

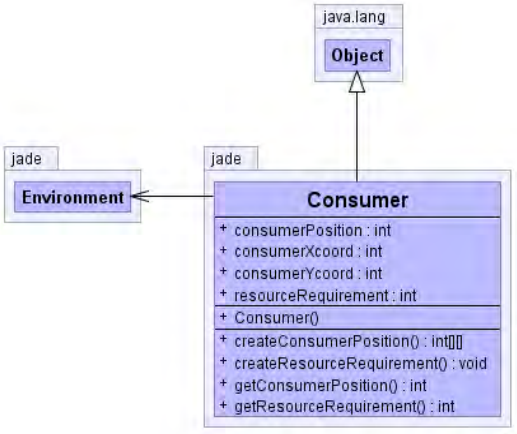
```
public class Environment
extends java.lang.Object
```



Εικόνα 4.8

Consumer class:

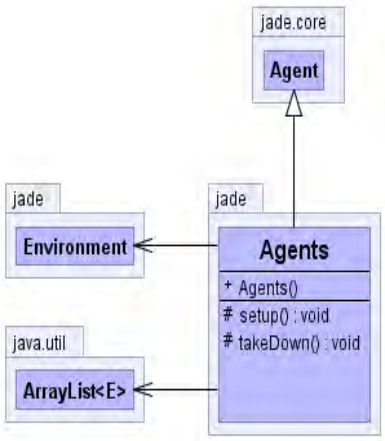
```
public class Consumer
extends java.lang.Object
```



Εικόνα 4.9

Agents class:

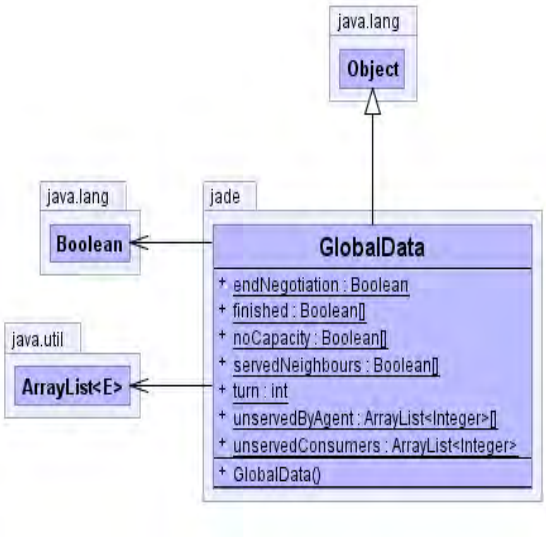
```
public class Agents
extends jade.core.Agent
```



Εικόνα 4.10

GlobalData class:

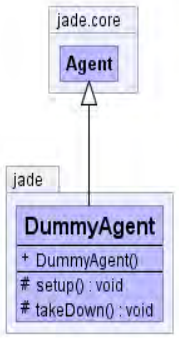
```
public class GlobalData
extends java.lang.Object
```



Εικόνα 4.11

DummyAgent Class:

```
public class DummyAgent
extends jade.core.Agent
```



Εικόνα 4.12

Για τη δημιουργία διαγραμμάτων UML απ' ευθείας από τον κώδικα Java χρησιμοποιήθηκε μία Javadoc επέκταση, το yWorks UML Doclet. Προσθέτοντας την κατάλληλη βιβλιοθήκη στο project στο NetBeans, δίνεται η δυνατότητα παραγωγής ενός Javadoc το οποίο περιέχει διαγράμματα UML για τις κλάσεις του project.

4.3.2 Αναλυτική περιγραφή των κλάσεων

Σε αυτή την ενότητα θα γίνει αναλυτική περιγραφή των κλάσεων που χρησιμοποιήθηκαν στην υλοποίηση του αλγορίθμου για την επίλυση του προβλήματος διαχείρισης πόρων από ένα πολυπρακτορικό σύστημα, αλλά και των συγκεκριμένων δεδομένων που χρησιμοποιήθηκαν για να μελετηθεί η εφαρμογή του.

Environment class:

Αρχικά, δημιουργείται ένα δισδιάστατο πλέγμα (2-D grid) διαστάσεων 10x10, το οποίο προσομοιώνει το περιβάλλον (κόσμο) του προβλήματος μέσω της συνάρτησης *createEnvironment()*. Το κάθε κελί του πλέγματος αποτελεί μία πιθανή θέση είτε για έναν καταναλωτή είτε για έναν πράκτορα. Στο περιβάλλον, υπάρχει διαθέσιμος μόνο ένας τύπος πόρου και η συνολική του ποσότητα (totalResource) είναι V = 100 μονάδες. Γίνεται η

υπόθεση ότι ο πόρος είναι ισοκατανεμημένος στα 100 κελιά του πλέγματος, οπότε σε κάθε κελί θα αντιστοιχεί ποσότητα πόρου ίση με 100/V = 100/100 = 1 μονάδα (resourcePerCell).

Στη συνέχεια, δημιουργούνται τέσσερις καταναλωτές. Αυτό γίνεται με τη βοήθεια της κλάσης Consumer, από την οποία δημιουργούνται τέσσερα στιγμιότυπα: Consumer1, Consumer2, Consumer3, Consumer4. Αφού ολοκληρωθεί η δημιουργία των καταναλωτών και καθοριστούν οι θέσεις τους στο περιβάλλον, καθώς και οι απαιτήσεις τους για τον πόρο, δημιουργούνται δύο πράκτορες-εξυπηρετητές: Agent1, Agent2. Για κάθε έναν από αυτούς, υπολογίζεται η θέση του στο περιβάλλον (agentPosition) και η ποσότητα του πόρου που έχει στη διάθεσή του (agentCapacity). Η θέση του πράκτορα καθορίζεται με τυχαίο τρόπο και δεν μεταβάλλεται κατά τη διάρκεια εκτέλεσης της εφαρμογής, παρά μόνο όταν ο πράκτορας πρέπει να μετακινηθεί στο χώρο για να εξυπηρετήσει κάποιον καταναλωτή. Η ποσότητα πόρου του κάθε πράκτορας έχει στη διάθεσή του, έχει άμεση σχέση με τη θέση του στο περιβάλλον, καθώς υπολογίζεται από το άθροισμα της ποσότητας πόρου που υπάρχει στα κελιά που περιβάλλουν το δικό του κελί σε απόσταση 1 (δηλαδή τα κελιά που βρίσκονται πάνω, κάτω, δεξιά, αριστερά αλλά και διαγώνια σε σχέση με το κελί του πράκτορα). Η διαθέσιμη ποσότητα πόρου ενός πράκτορα μεταβάλλεται κατά τη διάρκεια εκτέλεσης της εφαρμογής, μόνο σε περίπτωση που αυτός εξυπηρετήσει τις ανάγκες κάποιου καταναλωτή.

Η επόμενη διαδικασία που πραγματοποιείται σε αυτή την κλάση είναι η εύρεση των καταναλωτών που ανήκουν στη γειτονιά του κάθε πράκτορα. Δεδομένης της θέσης του πράκτορα, ένας καταναλωτής θεωρείται γείτονάς του, εάν ο πράκτορας μπορεί να φτάσει το κελί του με μία μόνο κίνηση σε συγκεκριμένη κατεύθυνση. Θεωρούμε, όπως αναφέρθηκε σε προηγούμενη ενότητα, ότι η κίνηση των πρακτόρων γίνεται μόνο προς μία κατεύθυνση - πάνω, κάτω, δεξιά, αριστερά – κάθε φορά. Αυτό σημαίνει ότι για να έχει ένας πράκτορας πρόσβαση σε κάποιο κελί που βρίσκεται διαγώνια σε σχέση με το δικό του, θα πρέπει να πραγματοποιήσει 2 κινήσεις: μία οριζόντια (δεξιά ή αριστερά) και μία κατακόρυφη (πάνω ή κάτω). Για αυτόν το λόγο, οι καταναλωτές που βρίσκονται σε κελιά διαγώνια σε σχέση με το κελί ενός πράκτορα δεν θεωρούνται άμεσοι γείτονές του.

Ο πίνακας neighbours[] περιλαμβάνει δύο λίστες, τις neighbours[0] και neighbours[1], στις οποίες τοποθετούνται οι καταναλωτές που είναι γείτονες του πρώτου και του δεύτερου πράκτορα αντίστοιχα, ύστερα από σχετικό έλεγχο των θέσεων τους στο περιβάλλον. Αυτοί οι καταναλωτές θα είναι οι πρώτοι που θα εξυπηρετηθούν από τους δύο πράκτορες ($1^{η}$ φάση του αλγορίθμου). Οι καταναλωτές που δεν αποτελούν γείτονα κανενός πράκτορα, τοποθετούνται στη λίστα unservedConsumers της κλάσης GlobalData. Αυτοί θα εξυπηρετηθούν αργότερα, έπειτα από επικοινωνία και διαπραγμάτευση μεταξύ των δύο πρακτόρων ($2^{η}$ φάση του αλγορίθμου).

Μετά το τέλος αυτής της διαδικασίας, η κλάση Environment δημιουργεί τους δύο πράκτορες που θα υπάρχουν στο περιβάλλον του προβλήματος. Γίνεται χρήση της μεθόδου *createNewAgent()*, η οποία παρέχεται μέσω της βιβλιοθήκης jade. Η μέθοδος αυτή δημιουργεί κάθε φορά ένα στιγμιότυπο της κλάσης Agents (στη συγκεκριμένη περίπτωση καλείται δύο φορές) και στέλνει σε αυτή ως ορίσματα τη θέση ενός πράκτορα στο περιβάλλον, την ποσότητα πόρου που έχει στη διάθεσή του, τη λίστα με τους καταναλωτές-

γείτονες του και ένα id που τον διαφοροποιεί από τον άλλο πράκτορα. Σε αυτό το σημείο, πρέπει να τονιστεί ότι η επιλογή δημιουργίας των δύο πρακτόρων από την ίδια κλάση οφείλεται στο γεγονός ότι στη συγκεκριμένη εφαρμογή είναι πανομοιότυποι και πραγματοποιούν ακριβώς τις ίδιες ενέργειες για την επίτευξη ενός κοινού στόχου. Εάν ο καθένας από αυτούς είχε έναν διακριτό ρόλο στη διαδικασία επίλυσης του προβλήματος, ο οποίος συνοδευόταν και από διαφορετικές ενέργειες για τον καθένα, τότε θα έπρεπε να δημιουργούνται από διαφορετικές κλάσεις.

Εκτός από αυτούς τους δύο πράκτορες που συμμετέχουν ενεργά στην επίλυση του προβλήματος που μελετάται, δημιουργούνται και κάποιοι άλλοι πράκτορες με βοηθητικό ρόλο. Ένας από αυτούς είναι ο Dummy Agent, ο οποίος δημιουργείται από την κλάση DummyAgent και ο ρόλος του θα περιγραφεί παρακάτω. Επίσης, δημιουργούνται δύο πράκτορες ελέγχου: ο rma και ο sniffer, από κλάσεις που παρέχονται μέσω της βιβλιοθήκης jade. Ο rma πράκτορας παρέχει διάφορα εργαλεία με τα οποία μπορεί να γίνει έλεγχος της εφαρμογής, ενώ ο sniffer πράκτορας χρησιμοποιείται για να παρακολουθείται η επικοινωνία (ανταλλαγή μηνυμάτων) μεταξύ των δύο πρακτόρων.

Consumer class:

Όπως αναφέρθηκε και παραπάνω, από αυτήν την κλάση δημιουργούνται τέσσερα στιγμιότυπα που αναπαριστούν τους καταναλωτές του πόρου στο περιβάλλον. Για κάθε έναν από αυτούς, είναι απαραίτητο να προσδιοριστούν δύο στοιχεία: η θέση του στο περιβάλλον (consumerPosition) και οι απαιτήσεις του για τον πόρο (resourceRequirement).

Η μέθοδος *createConsumerPosition()* της κλάσης Consumer χρησιμοποιείται για να καθοριστεί η θέση ενός καταναλωτή, η οποία είναι τυχαία (οποιοδήποτε κελί στο 2D πλέγμα) και δεν μεταβάλλεται κατά τη διάρκεια εκτέλεσης της εφαρμογής.

Η μέθοδος *createResourceRequirement()* της κλάσης Consumer χρησιμοποιείται για να καθοριστεί η ποσότητα του πόρου που έχει ανάγκη ο καταναλωτής. Η ποσότητα αυτή ορίζεται με τυχαίο τρόπο και μπορεί να μεταβληθεί, συγκεκριμένα να μειωθεί, μόνο αν ο καταναλωτής εξυπηρετηθεί από κάποιον πράκτορα.

Agents class:

Από αυτήν την κλάση, δημιουργούνται οι 2 πράκτορες της εφαρμογής και καθορίζονται οι ενέργειες που πρέπει να κάνουν για την επίλυση του προβλήματος. Αφού γίνουν οι απαραίτητες αρχικοποιήσεις, στον κάθε πράκτορα ανατίθενται κάποιες συμπεριφορές (behaviours) οι οποίες αντιστοιχούν στις εργασίες (tasks) που πρέπει να πραγματοποιήσει.

Η ανάθεση της πρώτης συμπεριφοράς γίνεται μέσω της κλήσης της μεθόδου *addBehaviour(new ServeNeighbours())*. Σκοπός αυτής της συμπεριφοράς είναι ο πράκτορας να εξυπηρετήσει αρχικά τους καταναλωτές που ανήκουν στη λίστα με τους γείτονές του (neighbours[i], i = 0, 1) και αποτελεί την $1^{η}$ φάση του αλγορίθμου, έτσι όπως περιγράφηκε

σε προηγούμενη ενότητα. Εάν ο πράκτορας έχει τη δυνατότητα να εξυπηρετήσει έναν καταναλωτή-γείτονα (δηλαδή, agentCapacity >= need(consumer)), τότε ο πράκτορας μετακινείται στη θέση του συγκεκριμένου καταναλωτή και τον εξυπηρετεί μηδενίζοντας τις απαιτήσεις του για τον πόρο. Επίσης, μειώνει την ποσότητα του πόρου που έχει αυτός πλέον στη διάθεση του κατά το ποσό που παρείχε στον συγκεκριμένο καταναλωτή και αλλάζει και την τιμή της τρέχουσας θέσης του. Αν ο πράκτορας δεν έχει τη δυνατότητα να εξυπηρετήσει έναν καταναλωτή-γείτονα, απλά τον τοποθετεί στη λίστα unservedByAgent_i (i = 0, 1). Αυτή τη λίστα θα την επεξεργαστεί ο άλλος πράκτορας, αφού τελειώσει με τις δικές του εργασίες και εφόσον η ποσότητα του πόρου που έχει στη διάθεσή του δεν έχει μηδενιστεί. Εάν κατά τη διάρκεια αυτής της διαδικασίας, μηδενιστεί η ποσότητα του πόρου που έχει ένας πράκτορας, τότε θα πρέπει να αναλάβει την εξυπηρέτηση όλων των υπόλοιπων καταναλωτών ο άλλος πράκτορας. Εάν μηδενιστεί η ποσότητα πόρου και των δύο πρακτόρων, τότε ο αλγόριθμος δεν περνάει στη $2^η$ φάση και αν υπάρχουν καταναλωτές με μη μηδενική ανάγκη, θα παραμείνουν ανικανοποίητοι για πάντα.

Η ανάθεση της δεύτερης συμπεριφοράς γίνεται μέσω της κλήσης της μεθόδου *addBehaviour(new ServeDuringNegotiation())*. Σκοπός αυτής της συμπεριφοράς είναι οι δύο πράκτορες να επικοινωνήσουν μεταξύ τους, προκειμένου να ικανοποιήσουν τις απαιτήσεις όλων των καταναλωτών που δεν αποτελούσαν γείτονες κανενός στην $1^η$ φάση κι έτσι δεν εξυπηρετήθηκαν, δηλαδή στην ουσία να αδειάσουν τη λίστα unservedConsumers. Η διαπραγμάτευση πραγματοποιείται βάσει των κριτηρίων που έχουν ήδη αναφερθεί σε προηγούμενη ενότητα και αυτή η διαδικασία αποτελεί τη $2^η$ φάση του αλγορίθμου.

Αρχικά, ο κάθε πράκτορας, πριν προβεί στην εξυπηρέτηση ενός καταναλωτή από τη λίστα unservedConsumers, στέλνει στον άλλο πράκτορα ένα μήνυμα τύπου INFORM_IF με περιεχόμενο το agentCapacity που του αντιστοιχεί για να τον ενημερώσει για την κατάστασή του. Οι πράκτορες λαμβάνουν τα μηνύματα και ανάλογα με το περιεχόμενο που διαβάζουν δρουν με διαφορετικό τρόπο. Διακρίνονται οι εξής περιπτώσεις:

i. Αν και για τους δύο πράκτορες ισχύει agentCapacity>=1, τότε αρχίζει η μεταξύ τους διαπραγμάτευση για να αποφασιστεί ποιος θα αναλάβει να εξυπηρετήσει ένα συγκεκριμένο καταναλωτή.
ii. Αν για έναν από αυτούς ισχύει agentCapacity=0, τότε αυτός τερματίζει, εφόσον δεν μπορεί να προσφέρει τίποτα, και η επικοινωνία διακόπτεται. Ο άλλος πράκτορας καταλαβαίνει ότι πρέπει να αναλάβει μόνος του να εξυπηρετήσει όλους τους υπόλοιπους καταναλωτές. Τους τοποθετεί σε μία αύξουσα σειρά βάσει της απόστασής τους από αυτόν, εξυπηρετεί όσους του επιτρέπει το agentCapacity του και στο τέλος τερματίζει κι αυτός.
iii. Αν και για τους δύο πράκτορες ισχύει agentCapacity=0, διακόπτεται και πάλι η μεταξύ τους επικοινωνία και τερματίζουν και οι δύο. Εάν υπάρχουν καταναλωτές στις λίστες unservedByAgent_0, unservedByAgent_1, unservedConsumers, θα μείνουν ανικανοποίητοι για πάντα.

Έστω ότι ισχύει η περίπτωση i. Ο κάθε πράκτορας παίρνει από τη λίστα unservedConsumers τον πρώτο καταναλωτή και ελέγχει αν μπορεί να τον εξυπηρετήσει. Εάν ναι, τότε υπολογίζει την απόστασή του από αυτόν. Στη συνέχεια, στέλνει ένα μήνυμα τύπου

PROPOSE στον άλλο πράκτορα, περιεχόμενο του οποίου είναι η απόσταση (πρωτεύον κριτήριο διαπραγμάτευσης) που υπολόγισε προηγουμένως. Αυτό το μήνυμα αποτελεί ουσιαστικά την προσφορά (bid) του κάθε πράκτορα για τον εκάστοτε καταναλωτή που ελέγχει. Αν προκύψει ότι ο πράκτορας δεν είναι ικανός να εξυπηρετήσει το συγκεκριμένο καταναλωτή, στέλνει ένα μήνυμα τύπου INFORM στον άλλο πράκτορα, ενημερώνοντάς τον για αυτό. Το περιεχόμενο αυτού του μηνύματος είναι και πάλι η απόσταση του πράκτορα από το συγκεκριμένο καταναλωτή που ελέγχεται.

Στη συνέχεια, ο κάθε πράκτορας περιμένει να λάβει ένα μήνυμα (τύπου PROPOSE ή τύπου INFORM) από τον άλλο πράκτορα και δεν προχωράει στην αποστολή επόμενου μηνύματος που θα σχετίζεται με άλλον καταναλωτή της λίστας unservedConsumers.

Αν ο πράκτορας λάβει μήνυμα τύπου PROPOSE και η agentCapacity του είναι μεγαλύτερη από το resourceRequirement του συγκεκριμένου καταναλωτή που ελέγχεται, τότε διαβάζει το περιεχόμενό του (απόσταση του άλλου πράκτορα από τον συγκεκριμένο καταναλωτή) και το συγκρίνει με τη δική του απόσταση. Σε αυτό το σημείο διακρίνονται οι εξής περιπτώσεις:

i. Η απόστασή του είναι μικρότερη σε σχέση με το περιεχόμενο του μηνύματος που έλαβε.
   Αν ισχύει κάτι τέτοιο, δεδομένου ότι η απόσταση είναι πρωτεύον κριτήριο για τη διαπραγμάτευση, σημαίνει ότι αυτός βρίσκεται πιο κοντά στο συγκεκριμένο καταναλωτή. Οπότε, αναλαμβάνει να τον εξυπηρετήσει. Αλλάζει την τιμή της θέσης του (agentPosition), αφού μετακινείται προς τον καταναλωτή, μειώνει την ποσότητα του πόρου που έχει στη διάθεσή του (agentCapacity) κατά το ποσό που απαιτεί ο καταναλωτής, μηδενίζει την ανάγκη του (resourceRequirement(consumer)=0) και τέλος τον αφαιρεί από τη λίστα unservedConsumers.
ii. Η απόστασή του είναι μεγαλύτερη σε σχέση με το περιεχόμενο του μηνύματος που έλαβε.
   Αν ισχύει κάτι τέτοιο, σημαίνει ότι αυτός βρίσκεται πιο μακριά από τον συγκεκριμένο καταναλωτή σε σχέση με τον άλλο πράκτορα. Οπότε, σε αυτή την περίπτωση δεν κάνει τίποτα, καθώς η εξυπηρέτηση αυτού του καταναλωτή δεν αποτελεί μερίδιο της δουλειάς του τη συγκεκριμένη στιγμή.
iii. Η απόστασή του είναι ίση με το περιεχόμενο του μηνύματος που έλαβε. Σε αυτήν την περίπτωση ξεκινάει ένας καινούριος γύρος ανταλλαγής μηνυμάτων. Ο κάθε πράκτορας στέλνει στον άλλο ένα μήνυμα τύπου REQUEST, περιεχόμενο του οποίου είναι η ποσότητα του πόρου που έχει εκείνη τη στιγμή στη διάθεσή του (δευτερεύον κριτήριο διαπραγμάτευσης).

Με την ίδια λογική που χρησιμοποιήθηκε και προηγουμένως, όταν ένας πράκτορας λάβει ένα μήνυμα τύπου REQUEST, συγκρίνει το περιεχόμενό του με τη ποσότητα πόρου που έχει στη διάθεση του. Εδώ, διακρίνονται και πάλι τρεις περιπτώσεις:

i. Η διαθέσιμη ποσότητα πόρου του πράκτορα να είναι μεγαλύτερη σε σχέση με το περιεχόμενο του μηνύματος που έλαβε.

Αν ισχύει κάτι τέτοιο, αυτός πρέπει να αναλάβει την εξυπηρέτηση αυτού του καταναλωτή. Αυτό το κάνει μειώνοντας το agentCapacity του κατά την ποσότητα που απαιτεί ο καταναλωτής και στη συνέχεια μηδενίζοντας την ανάγκη του για τον πόρο (resourceRequirement(consumer)=0). Αφού μετακινείται προς αυτόν, αλλάζει και την τιμή της τρέχουσας θέσης του, ενώ τέλος, αφαιρεί τον καταναλωτή από τη λίστα unservedConsumers, αφού πλέον έχει εξυπηρετηθεί.

ii. Η διαθέσιμη ποσότητα πόρου του πράκτορα είναι μικρότερη σε σχέση με το περιεχόμενο του μηνύματος που έλαβε.
Αν ισχύει αυτό, την εξυπηρέτηση του καταναλωτή πρέπει να την αναλάβει ο άλλος πράκτορας καθώς είναι αυτός που έχει το μεγαλύτερο agentCapacity.

iii. Η διαθέσιμη ποσότητα πόρου του πράκτορα είναι ίση με το περιεχόμενο του μηνύματος που έλαβε.
Σε αυτή την περίπτωση, δεν πραγματοποιείται νέος γύρος ανταλλαγής μηνυμάτων μεταξύ των πρακτόρων. Αντίθετα, και οι δύο πράκτορες στέλνουν σε ένα τρίτο, τον Dummy Agent, ένα μήνυμα τύπου REQUEST για να το ενημερώσουν ότι έγινε έλεγχος και των δύο κριτηρίων της διαπραγμάτευσης, αλλά δεν μπορεί να ληφθεί απόφαση για το ποιος θα εξυπηρετήσει το συγκεκριμένο καταναλωτή. Το περιεχόμενο αυτού του μηνύματος είναι το id του κάθε πράκτορα. Ο Dummy Agent όταν λάβει και τα δύο μηνύματα, κάνει τις απαραίτητες ενέργειες και ενημερώνει τους πράκτορες για το ποιος πρέπει να κινηθεί προς τον καταναλωτή ώστε να ικανοποιήσει τις απαιτήσεις του για τον πόρο. Αυτό το κάνει στέλνοντας τους ένα μήνυμα τύπου INFORM, περιεχόμενο του οποίου είναι το id του πράκτορα που επιλέχθηκε να εξυπηρετήσει τον καταναλωτή. Όταν και οι δύο πράκτορες λάβουν ένα μήνυμα τύπου INFORM με αποστολέα τον Dummy Agent, τότε ελέγχουν το περιεχόμενο του μηνύματος και το συγκρίνουν με το id τους. Για όποιον αυτά τα δύο συμπίπτουν, αυτός αναλαμβάνει να εξυπηρετήσει τον καταναλωτή. Μετακινείται προς αυτόν αλλάζοντας την τιμή του agentPosition, μειώνει το agentCapacity κατά την ποσότητα που απαιτεί ο καταναλωτής, μηδενίζει το resourceRequirement(consumer) και αφαιρεί τον καταναλωτή από τη λίστα unservedConsumers.

Έστω ότι οι πράκτορες είναι οι i και j. Αν ο πράκτορας i λάβει ένα μήνυμα τύπου INFORM με αποστολέα τον άλλο πράκτορα και η agentCapacity του είναι μεγαλύτερη από το resourceRequirement του συγκεκριμένου καταναλωτή που ελέγχεται τότε καταλαβαίνει ότι ο πράκτορας j δεν έχει την αναγκαία ποσότητα πόρου για να εξυπηρετήσει τον συγκεκριμένο καταναλωτή. Σε αυτή την περίπτωση, συγκρίνει το περιεχόμενο του μηνύματος που έλαβε με την απόστασή του και διακρίνονται τρεις περιπτώσεις:

i. Η απόσταση του πράκτορα i είναι μικρότερη ή ίση από το περιεχόμενο του μηνύματος.
Αυτό σημαίνει ότι ο συγκεκριμένος καταναλωτής είναι αρμοδιότητα του πράκτορα i. Οπότε, αλλάζει την τιμή της τρέχουσας θέσης του, μειώνει το agentCapacity του κατά την ποσότητα πόρου που απαιτεί ο καταναλωτής, μηδενίζει το resourceRequirement(consumer) και τέλος αφαιρεί τον καταναλωτή από τη λίστα unservedConsumers.

ii. Η απόσταση του πράκτορα i είναι μεγαλύτερη από το περιεχόμενο του μηνύματος.
Αυτό σημαίνει ότι ο καταναλωτής αποτελεί αρμοδιότητα πράκτορα j, αλλά αυτός δεν είναι σε θέση να τον ικανοποιήσει. Οπότε, θα πρέπει να τον αναλάβει ο πράκτορας i. Τον προσθέτει στη λίστα unservedByAgent_j και θα προσπαθήσει να τον εξυπηρετήσει, αφού τελειώσει με την εξυπηρέτηση των καταναλωτών που αποτελούν δική του αρμοδιότητα.

Αυτή η ανταλλαγή μηνυμάτων πραγματοποιείται με τη σειρά και για τους τέσσερις καταναλωτές του περιβάλλοντος και συνεχίζεται όσο δεν έχει μηδενιστεί η ποσότητα πόρου που έχει κάποιος πράκτορας στη διάθεσή του.

Η τελευταία συμπεριφορά που ανατίθεται στους δύο πράκτορες γίνεται μέσω κλήσης της μεθόδου *addBehaviour(new ServeAfterNegotiation())*. Η εκτέλεση αυτής της συμπεριφοράς αποτελεί την $3^η$ φάση του αλγορίθμου επίλυσης του προβλήματος. Με αυτή, ο πράκτορας i προσπαθεί να εξυπηρετήσει τους καταναλωτές που υπάρχουν στη λίστα unservedByAgent_j, δηλαδή τους καταναλωτές που αποτελούσαν κανονικά αρμοδιότητα του πράκτορα j, αλλά λόγω έλλειψης πόρου δεν κατάφερε να τους ικανοποιήσει (i, j=0,1). Αυτό, φυσικά, γίνεται μόνο στην περίπτωση που ο πράκτορας έχει ακόμα στη διάθεσή του κάποιες μονάδες πόρου. Μετά το τέλος αυτής της φάσης, οι δύο πράκτορες τερματίζουν και φαίνεται αν τελικά κατάφεραν ή όχι να εξυπηρετήσουν όλους τους καταναλωτές του περιβάλλοντος.

Παρατηρήσεις:

- Οι πράκτορες δημιουργούνται από την ίδια κλάση και λειτουργούν σαν δύο νήματα (threads) που εκτελούνται παράλληλα. Για να μπορέσει, λοιπόν, να γίνει σωστά η επικοινωνία μεταξύ τους και να τελειώνει η διαπραγμάτευση σχετικά με την εξυπηρέτηση ενός καταναλωτή πριν προχωρήσουν στον έλεγχο του επόμενου, πρέπει να υπάρξει ένας συντονισμός μεταξύ τους. Αυτό επιτυγχάνεται με τη βοήθεια μίας καθολικής (global) μεταβλητής, της turn. Αυτή καθορίζει κάθε φορά ποιανού πράκτορα είναι η σειρά να πραγματοποιήσει μία ενέργεια, ώστε να γίνονται σωστά οι αλλαγές τιμών των διαφόρων καθολικών και τοπικών μεταβλητών, αλλά και για να υπάρχει μία λογική συνέχεια στην επικοινωνία μεταξύ των πρακτόρων και στα διαγνωστικά μηνύματα που εκτυπώνονται στην κονσόλα.

- Η σειρά με την οποία μπαίνουν οι καταναλωτές στις λίστες unservedByAgent_i (i=0, 1) και unservedConsumers, σχετίζεται αποκλειστικά και μόνο με τη σειρά «γέννησής» τους στο περιβάλλον. Αυτό σε συνδυασμό με το γεγονός ότι οι θέσεις τους είναι τυχαίες στο χώρο και αλλάζουν σε κάθε εκτέλεση του αλγορίθμου, οδηγεί στο συμπέρασμα ότι η σειρά με την οποία ελέγχονται για να εξυπηρετηθούν από τους πράκτορες χαρακτηρίζεται από τυχαιότητα και δεν οδηγεί ποτέ στη λιμοκτονία (starvation) ενός συγκεκριμένου καταναλωτή λόγω έλλειψης πόρου μέχρι να έρθει η σειρά εξυπηρέτησής του.

GlobalData class:

Η κλάση αυτή δεν προσφέρει κάποια λειτουργικότητα. Χρησιμοποιείται μόνο για κάποια καθολικά (global) δεδομένα που χρειάζονται για την υλοποίηση του αλγορίθμου.

DummyAgent class:

Από αυτήν την κλάση δημιουργείται ο πράκτορας Dummy Agent και ανατίθεται σε αυτόν μία συμπεριφορά. Η εργασία που πρέπει να πραγματοποιήσει είναι να αποφασίσει ποιος πράκτορας θα εξυπηρετήσει ένα συγκεκριμένο καταναλωτή, όταν ο έλεγχος και των δύο κριτηρίων (πρωτεύον, δευτερεύον) της διαπραγμάτευσης έχουν καταλήξει σε ισότητα και επομένως σε αδυναμία λήψης απόφασης από τους δύο πράκτορες.

Αρχικά, περιμένει να λάβει δύο μηνύματα τύπου REQUEST, ένα από τον Agent1 και ένα από τον Agent2, τα οποία περιέχουν το id του κάθε πράκτορα. Όταν τα λάβει και τα δύο, διαβάζει το περιεχόμενο τους και τυχαία διαλέγει κάποιο από τα δύο id. Στη συνέχεια, ετοιμάζει ένα μήνυμα τύπου INFORM, με παραλήπτες τους δύο πράκτορες και περιεχόμενο το id που επέλεξε. Στέλνει το μήνυμα και μετά παραμένει ανενεργός μέχρι οι δύο πράκτορες να χρειαστούν και πάλι κάποια στιγμή τη βοήθειά του.

4.3.3 Σενάρια εκτέλεσης του αλγορίθμου

Παρακάτω παρουσιάζονται κάποια ενδεικτικά σενάρια τα οποία προέκυψαν από την εφαρμογή του αλγορίθμου σε ένα περιβάλλον που περιέχει τέσσερις καταναλωτές και δύο πράκτορες και ισχύουν όλα τα δεδομένα και οι υποθέσεις που αναφέρθηκαν παραπάνω.

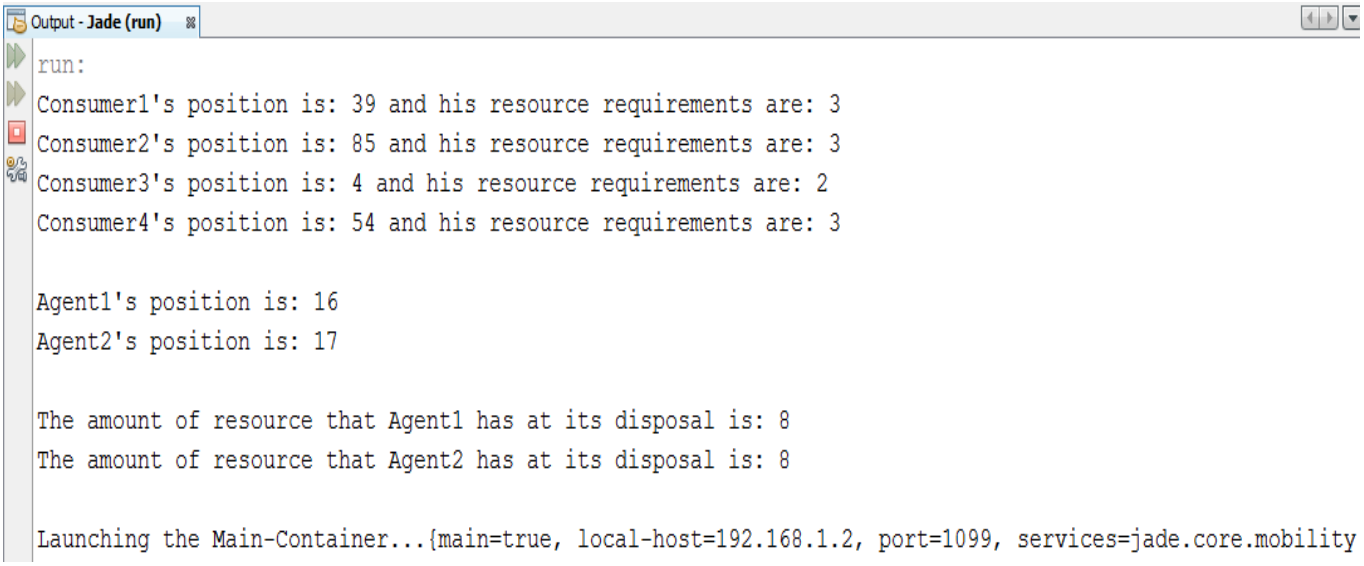
Στις εικόνες που ακολουθούν, απεικονίζονται τα αποτελέσματα από την εκτέλεση του αλγορίθμου. Για κάθε σενάριο παρουσιάζονται τα μηνύματα που εκτυπώνονται στην κονσόλα του NetBeans, τα οποία δείχνουν την πορεία εξέλιξης του αλγορίθμου, καθώς επίσης παρουσιάζεται και το αποτέλεσμα της δουλειάς του Sniffer agent, που είναι στην ουσία η καταγραφή των μηνυμάτων που ανταλλάσσονται μεταξύ των δύο πρακτόρων κατά τη $2^{η}$ φάση (επικοινωνία – επίτευξη συμφωνιών) του αλγορίθμου, ενώ απεικονίζεται ο τύπος και το αναγνωριστικό τους. Η σημασία του κάθε τύπου έχει αναφερθεί νωρίτερα στην *Ενότητα 4.2*. Τα αναγνωριστικά 0-3 (στα χρωματιστά βέλη) υποδηλώνουν τα μηνύματα που ανταλλάσσονται σε κάθε γύρο διαπραγμάτευσης που σχετίζεται με έναν καταναλωτή, όπου το 0 αντιστοιχεί στον πρώτο καταναλωτή που ελέγχεται, το 1 στον δεύτερο, κ.ο.κ. . Τα γκρι βέλη με το αναγνωριστικό -1 δεν έχουν καμία σχέση με τη διαδικασία της διαπραγμάτευσης. Αντιπροσωπεύουν μηνύματα που ανταλλάσσονται πριν από κάθε γύρο μόνο ως επιβεβαίωση ότι οι πράκτορες έχουν στη διάθεσή τους μη μηδενική ποσότητα πόρου και ότι μπορούν και έχει νόημα να συνεχίσουν την επικοινωνία.

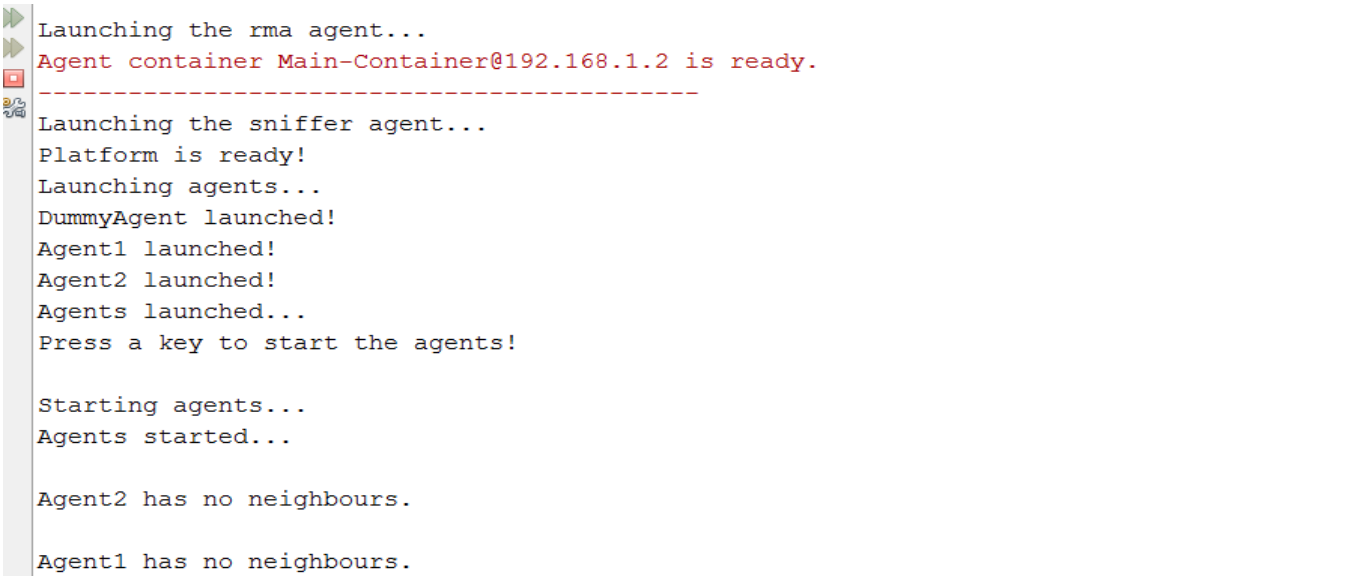
Σενάριο 1:

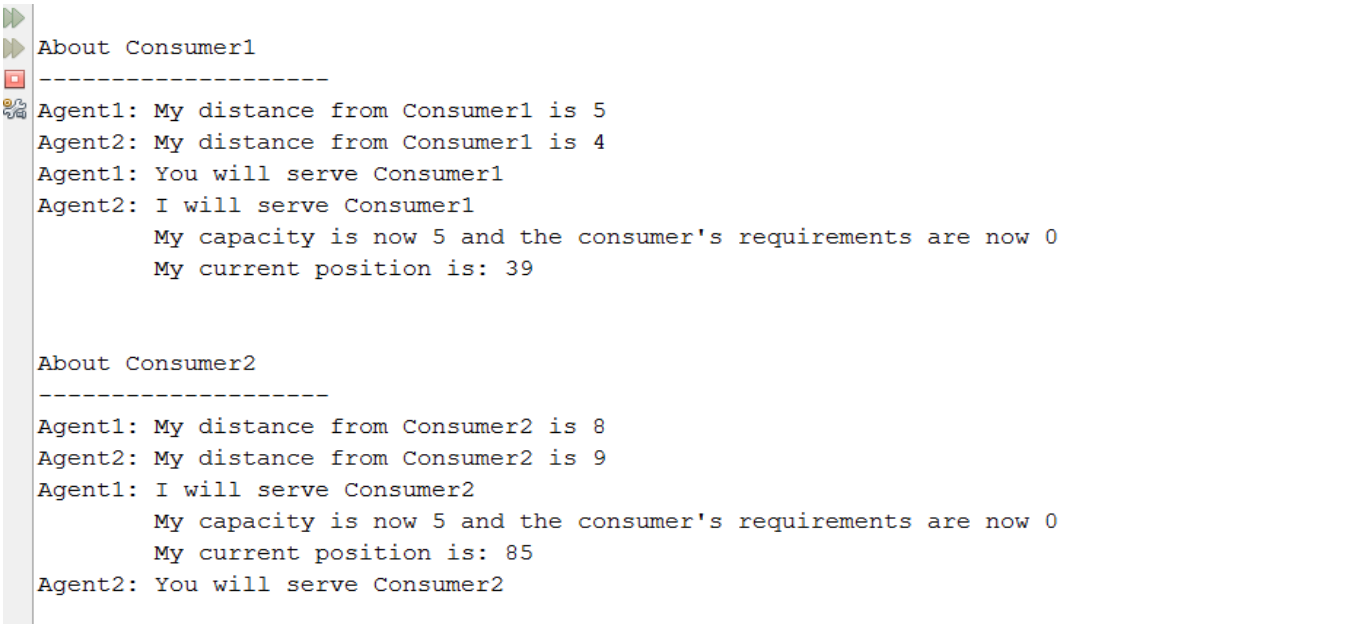
Σε αυτό το σενάριο κανένας από τους δύο πράκτορες δεν έτυχε να έχει κάποιον καταναλωτή ως άμεσο γείτονά του, έτσι όπως καθορίστηκαν οι θέσεις όλων στο περιβάλλον. Αυτό σημαίνει ότι δεν πραγματοποιείται η $1^{η}$ φάση και οι πράκτορες περνούν κατ' ευθείαν στη $2^{η}$, ξεκινώντας τη διαπραγμάτευση. Στην *Εικόνα 4.14* φαίνεται ο τρόπος με τον οποίο οι δύο πράκτορες επικοινωνούν μεταξύ τους για να συμφωνήσουν ποιος από τους δύο θα εξυπηρετήσει τον καταναλωτή που ελέγχεται κάθε φορά, ενώ η *Εικόνα 4.15* δείχνει πιο συγκεκριμένα τα μηνύματα που ανταλλάσσονται μεταξύ των δύο πρακτόρων, το τελεστικό τους και το αναγνωριστικό που έχουν. Οι *Εικόνες 4.13* και *4.16* δείχνουν αντίστοιχα την αρχική και την τελική κατάσταση του περιβάλλοντος.

i \ j	0	1	2	3	4	5	6	7	8	9
0					C3					
1							A1	A2		
2										
3										C1
4										
5					C4					
6										
7										
8						C2				
9										

Εικόνα 4.13 Αρχική κατάσταση του περιβάλλοντος
Ai, i=1,2 : οι δύο πράκτορες
Cj, j=1,2,3,4 : οι τέσσερις καταναλωτές
(πράσινο: έχει εξυπηρετηθεί, κόκκινο: δεν έχει εξυπηρετηθεί)
Υπολογισμός θέσης πράκτορα ή καταναλωτή: i*10 + j


Output - Jade (run)
run:
Consumer1's position is: 39 and his resource requirements are: 3
Consumer2's position is: 85 and his resource requirements are: 3
Consumer3's position is: 4 and his resource requirements are: 2
Consumer4's position is: 54 and his resource requirements are: 3
Agent1's position is: 16
Agent2's position is: 17
The amount of resource that Agent1 has at its disposal is: 8
The amount of resource that Agent2 has at its disposal is: 8
Launching the Main-Container...{main=true, local-host=192.168.1.2, port=1099, services=jade.core.mobility

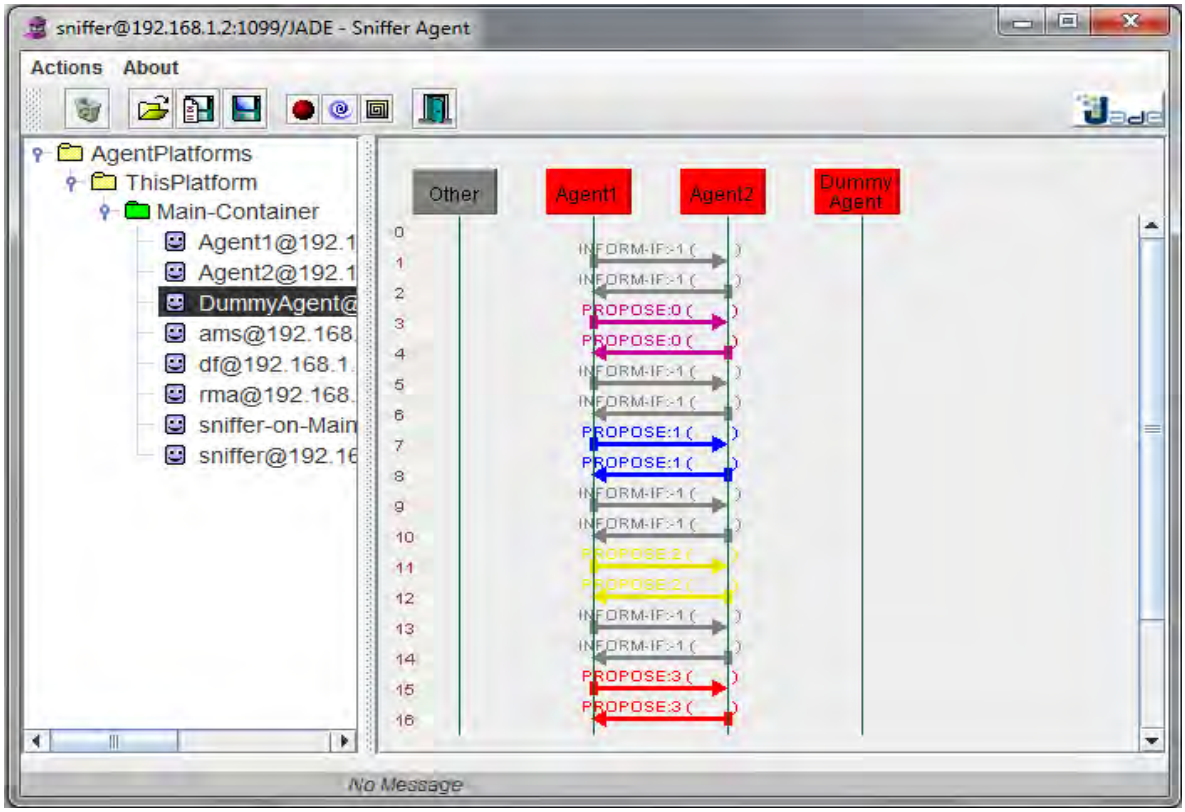

Launching the rma agent...
Agent container Main-Container@192.168.1.2 is ready.
Launching the sniffer agent...
Platform is ready!
Launching agents...
DummyAgent launched!
Agent1 launched!
Agent2 launched!
Agents launched...
Press a key to start the agents!
Starting agents...
Agents started...
Agent2 has no neighbours.
Agent1 has no neighbours.


About Consumer1
Agent1: My distance from Consumer1 is 5
Agent2: My distance from Consumer1 is 4
Agent1: You will serve Consumer1
Agent2: I will serve Consumer1
My capacity is now 5 and the consumer's requirements are now 0
My current position is: 39
About Consumer2
Agent1: My distance from Consumer2 is 8
Agent2: My distance from Consumer2 is 9
Agent1: I will serve Consumer2
My capacity is now 5 and the consumer's requirements are now 0
My current position is: 85
Agent2: You will serve Consumer2

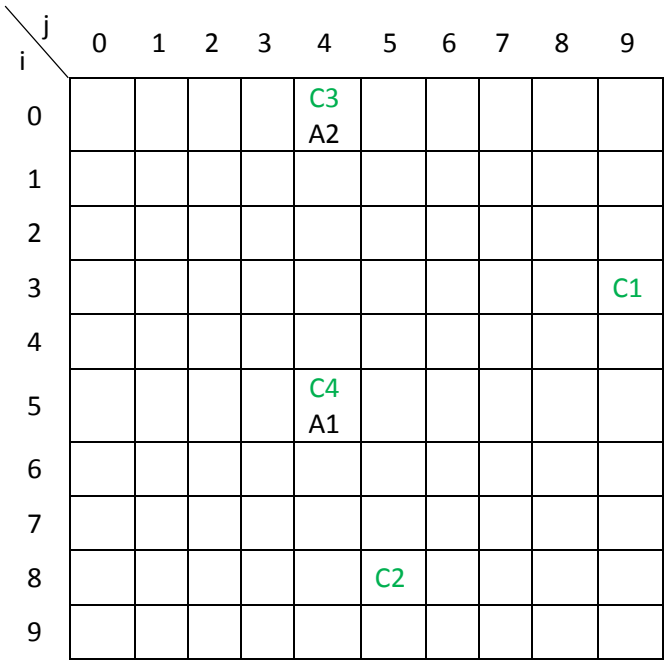
```
About Consumer3
--------------------
Agent1: My distance from Consumer3 is 9
Agent2: My distance from Consumer3 is 8
Agent1: You will serve Consumer3
Agent2: I will serve Consumer3
        My capacity is now 3 and the consumer's requirements are now 0
        My current position is: 4


About Consumer4
--------------------
Agent1: My distance from Consumer4 is 4
Agent2: My distance from Consumer4 is 5
Agent1: I will serve Consumer4
        My capacity is now 2 and the consumer's requirements are now 0
        My current position is: 54
Agent2: You will serve Consumer4
```

```
After the phase of negotiation:
--------------------------------
There are no unserved consumers.
The agents managed to satisfy the needs of all of them.
```

Εικόνα 4.14 Η διαδικασία εξυπηρέτησης των καταναλωτών



Εικόνα 4.15 Η ανταλλαγή μηνυμάτων μεταξύ των πρακτόρων για τη λήψη απόφασης



Εικόνα 4.16 Τελική κατάσταση του περιβάλλοντος
Ai, i=1,2 : οι δύο πράκτορες
Cj, j=1,2,3,4 : οι τέσσερις καταναλωτές
(πράσινο: έχει εξυπηρετηθεί, κόκκινο: δεν έχει εξυπηρετηθεί)
Υπολογισμός θέσης πράκτορα ή καταναλωτή: i*10 + j

Σενάριο 2:

Σε αυτήν την περίπτωση, ο Agent1 τυχαίνει να έχει ένα γείτονα, τον Consumer1 και ο Agent2 τυχαίνει να έχει τον Consumer3. Γι' αυτό το λόγο περνάνε από την $1^η$ φάση και τους εξυπηρετούν. Στη συνέχεια, και οι δύο πράκτορες βρίσκονται στη $2^η$ φάση, όπου επικοινωνούν για να εξυπηρετήσουν τους καταναλωτές που έχουν απομείνει. Όπως φαίνεται στην *Εικόνα 4.18,* η εξυπηρέτηση των δύο καταναλωτών που έχουν απομείνει πραγματοποιείται κανονικά. Οι *Εικόνες 4.17* και *4.20* δείχνουν την αρχική και την τελική κατάσταση του περιβάλλοντος, ενώ η *Εικόνα 4.19,* δείχνει τα μηνύματα που ανταλλάσσονται μεταξύ των δύο πρακτόρων, κατά τη $2^η$ φάση του αλγορίθμου.

i \ j	0	1	2	3	4	5	6	7	8	9
0				A2			C4			
1				C3						
2										
3					C2					
4										
5										
6										
7										
8										
9						C1	A1			

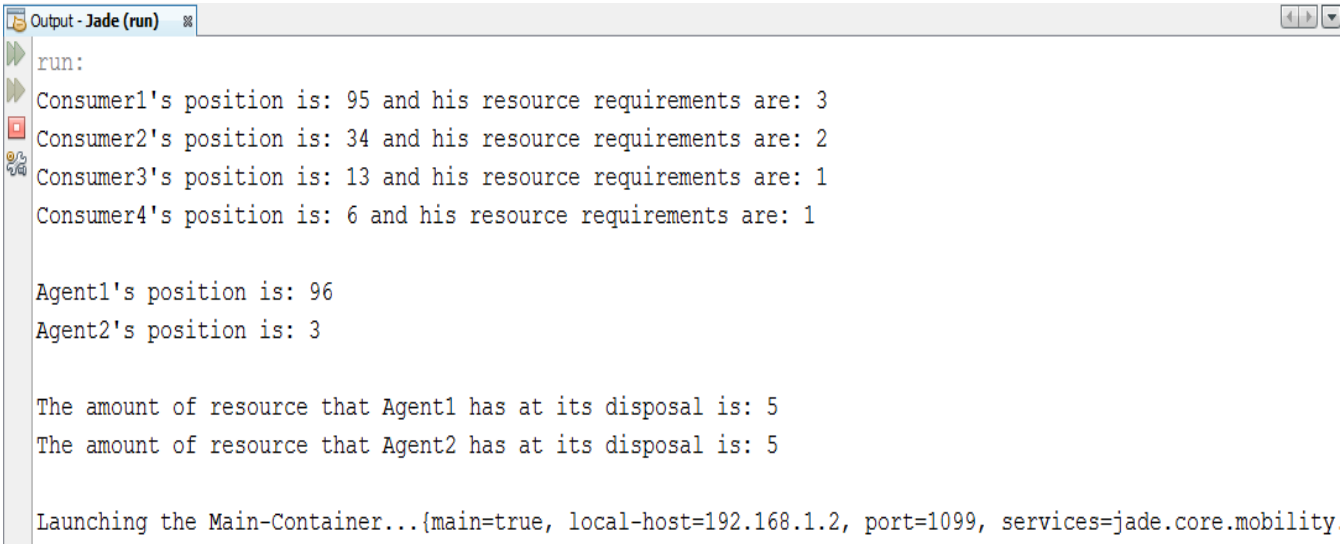
Εικόνα 4.17 Αρχική κατάσταση του περιβάλλοντος

Ai, i=1,2 : οι δύο πράκτορες

Cj, j=1,2,3,4 : οι τέσσερις καταναλωτές

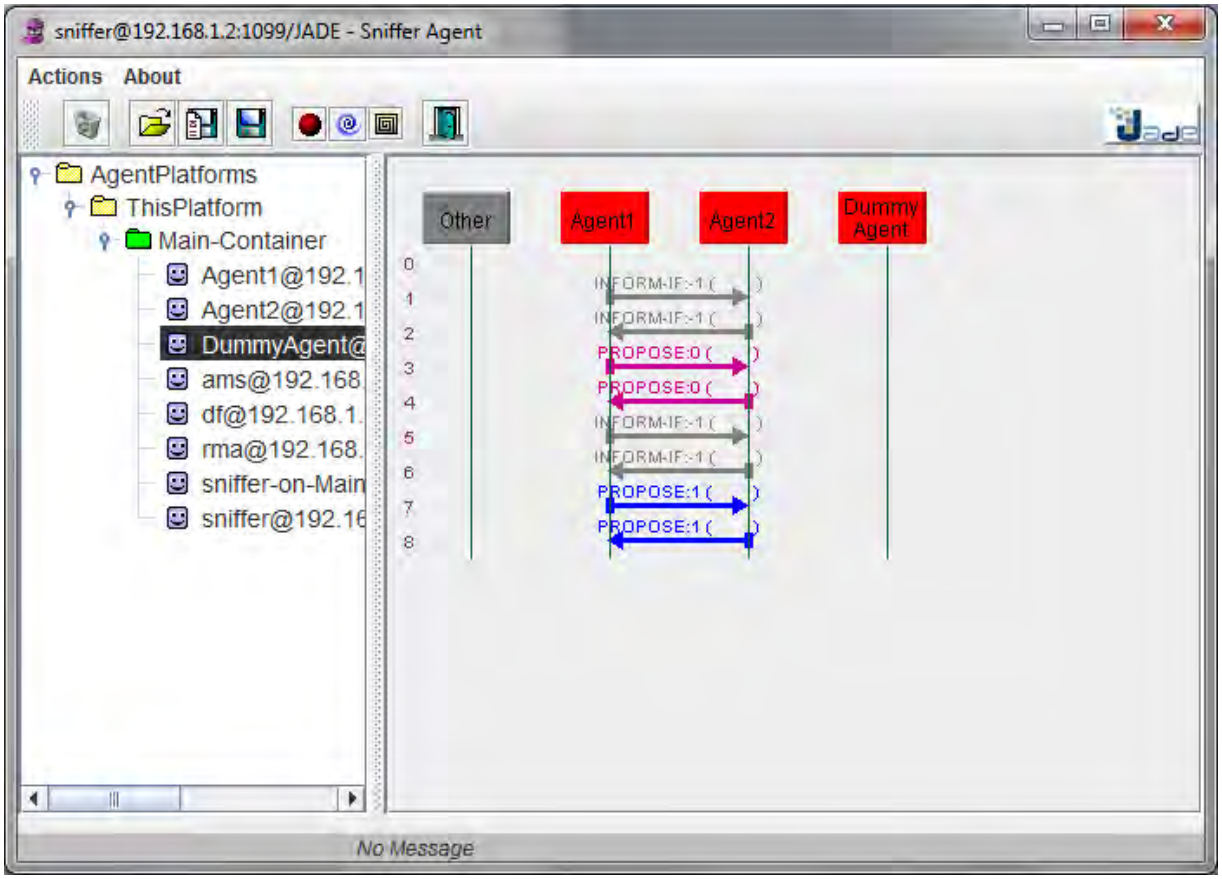
(πράσινο: έχει εξυπηρετηθεί, κόκκινο: δεν έχει εξυπηρετηθεί)

Υπολογισμός θέσης πράκτορα ή καταναλωτή: i*10 + j



```
Launching the sniffer agent...
Platform is ready!
Launching agents...
DummyAgent launched!
Agent1 launched!
Agent2 launched!
Agents launched...
Press a key to start the agents!

Starting agents...
Agents started...

The following consumers are Agent1's neighbours:
Consumer1

Agent1 served his neighbour Consumer1 and now his capacity is: 2
His current position is: 95

Agent1's capacity is 2 after serving his neighbours.

The following consumers are Agent2's neighbours:
Consumer3

Agent2 served his neighbour Consumer3 and now his capacity is: 4
His current position is: 13

Agent2's capacity is 4 after serving his neighbours.


About Consumer2
--------------------
Agent1: My distance from Consumer2 is 7
Agent2: My distance from Consumer2 is 3
Agent1: You will serve Consumer2
Agent2: I will serve Consumer2
        My capacity is now 2 and the consumer's requirements are now 0
        My current position is: 34

About Consumer4
--------------------
Agent1: My distance from Consumer4 is 10
Agent2: My distance from Consumer4 is 5
Agent1: You will serve Consumer4
Agent2: I will serve Consumer4
        My capacity is now 1 and the consumer's requirements are now 0
        My current position is: 6


After the phase of negotiation:
--------------------------------
There are no unserved consumers.
The agents managed to satisfy the needs of all of them.
```

Εικόνα 4.18 Η διαδικασία εξυπηρέτησης των καταναλωτών



Εικόνα 4.19 Η ανταλλαγή μηνυμάτων μεταξύ των πρακτόρων για τη λήψη απόφασης

i \ j	0	1	2	3	4	5	6	7	8	9
0							C4 A2			
1				C3						
2										
3					C2					
4										
5										
6										
7										
8										
9						C1 A1				

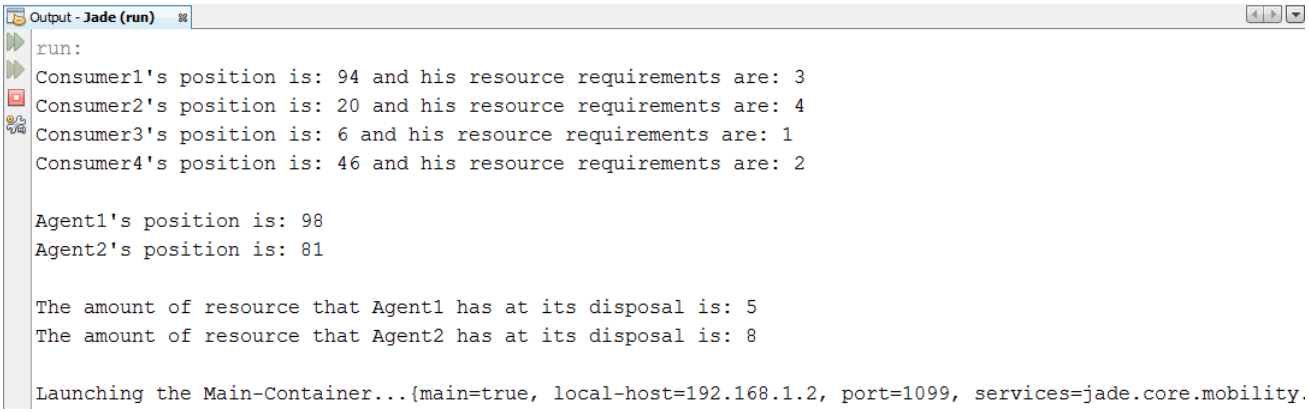
Εικόνα 4.20 Τελική κατάσταση του περιβάλλοντος
Ai, i=1,2 : οι δύο πράκτορες
Cj, j=1,2,3,4 : οι τέσσερις καταναλωτές
(πράσινο: έχει εξυπηρετηθεί, κόκκινο: δεν έχει εξυπηρετηθεί)
Υπολογισμός θέσης πράκτορα ή καταναλωτή: i*10 + j

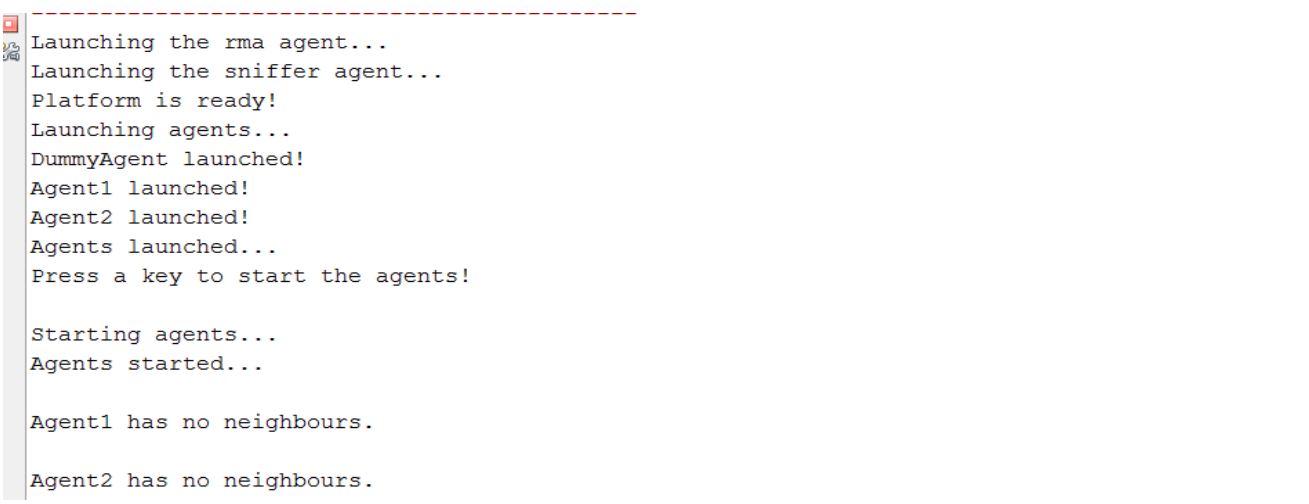
Σενάριο 3:

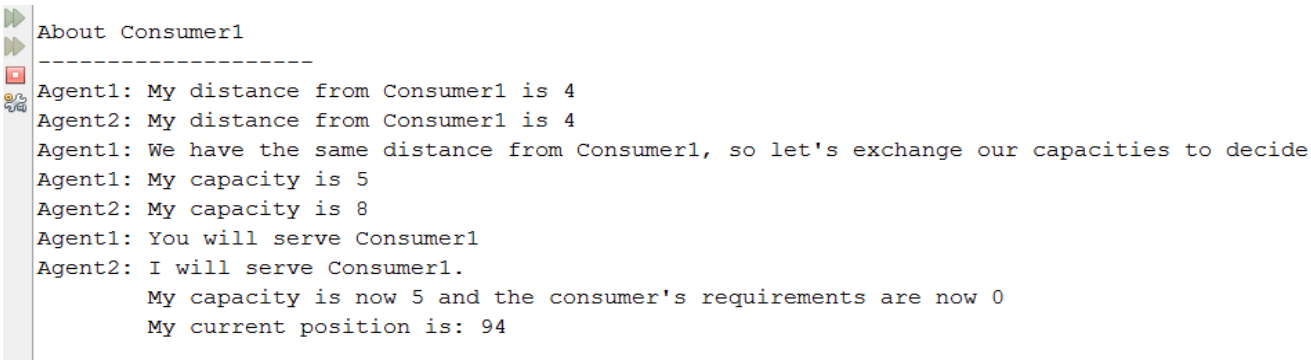
Σε αυτή την περίπτωση παρουσιάζεται τι γίνεται, αν κατά τη φάση επικοινωνίας μεταξύ των δύο πρακτόρων προκύψει ότι δεν μπορεί να ληφθεί απόφαση για το ποιος θα εξυπηρετήσει ένα συγκεκριμένο καταναλωτή με τον έλεγχο του πρωτεύοντος κριτηρίου. Οι πράκτορες θα πρέπει να ελέγξουν και τον δευτερεύον κριτήριο και να ανταλλάξουν μηνύματα με την ποσότητα πόρου που έχει ο καθένας στη διάθεσή του. Στο συγκεκριμένο σενάριο αυτό συμβαίνει κατά τον έλεγχο του καταναλωτή Consumer1. Ο έλεγχος των καταναλωτών Consumer2 και Consumer3 εξελίσσεται κανονικά. Μετά την εξυπηρέτηση, όμως, του $3^{ου}$ καταναλωτή, η ποσότητα πόρου που έχει ο πράκτορας Agent2 στη διάθεσή του μηδενίζεται, γεγονός το οποίο οδηγεί σε διακοπή επικοινωνίας και σε τερματισμό της $2^{ης}$ φάσης του αλγορίθμου. Στην $3^{η}$ φάση, ο πράκτορας Agent1 αναλαμβάνει την εξυπηρέτηση των καταναλωτών που έχουν απομείνει με μη μηδενικές ανάγκες, που στη συγκεκριμένη περίπτωση είναι μόνο ο καταναλωτής Consumer4. Εφόσον έχει τη δυνατότητα, τον εξυπηρετεί κι έτσι τελικά ικανοποιούνται οι ανάγκες όλων των καταναλωτών. Η *Εικόνα 4.22* δείχνει τις φάσεις από τις οποίες περνάει ο αλγόριθμος, ενώ η *Εικόνα 4.23* δείχνει αναλυτικά τα μηνύματα που ανταλλάσσουν οι δύο πράκτορες στη $2^{η}$ φάση. Οι *Εικόνες 4.21* και *4.24* απεικονίζουν αντίστοιχα την αρχική και την τελική κατάσταση του περιβάλλοντος.

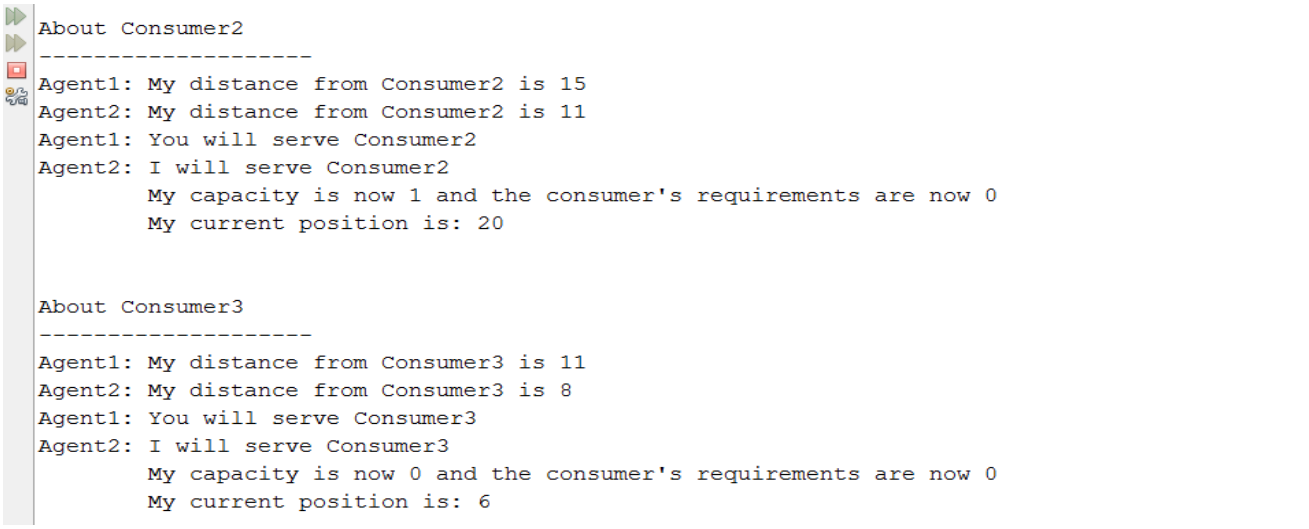
i \ j	0	1	2	3	4	5	6	7	8	9
0							C3			
1										
2	C2									
3										
4							C4			
5										
6										
7										
8		A2								
9					C1				A1	

Εικόνα 4.21 Αρχική κατάσταση του περιβάλλοντος
Ai, i=1,2 : οι δύο πράκτορες
Cj, j=1,2,3,4 : οι τέσσερις καταναλωτές
(πράσινο: έχει εξυπηρετηθεί, κόκκινο: δεν έχει εξυπηρετηθεί)
Υπολογισμός θέσης πράκτορα ή καταναλωτή: i*10 + j


Output - Jade (run)
run:
Consumer1's position is: 94 and his resource requirements are: 3
Consumer2's position is: 20 and his resource requirements are: 4
Consumer3's position is: 6 and his resource requirements are: 1
Consumer4's position is: 46 and his resource requirements are: 2
Agent1's position is: 98
Agent2's position is: 81
The amount of resource that Agent1 has at its disposal is: 5
The amount of resource that Agent2 has at its disposal is: 8
Launching the Main-Container...{main=true, local-host=192.168.1.2, port=1099, services=jade.core.mobility.

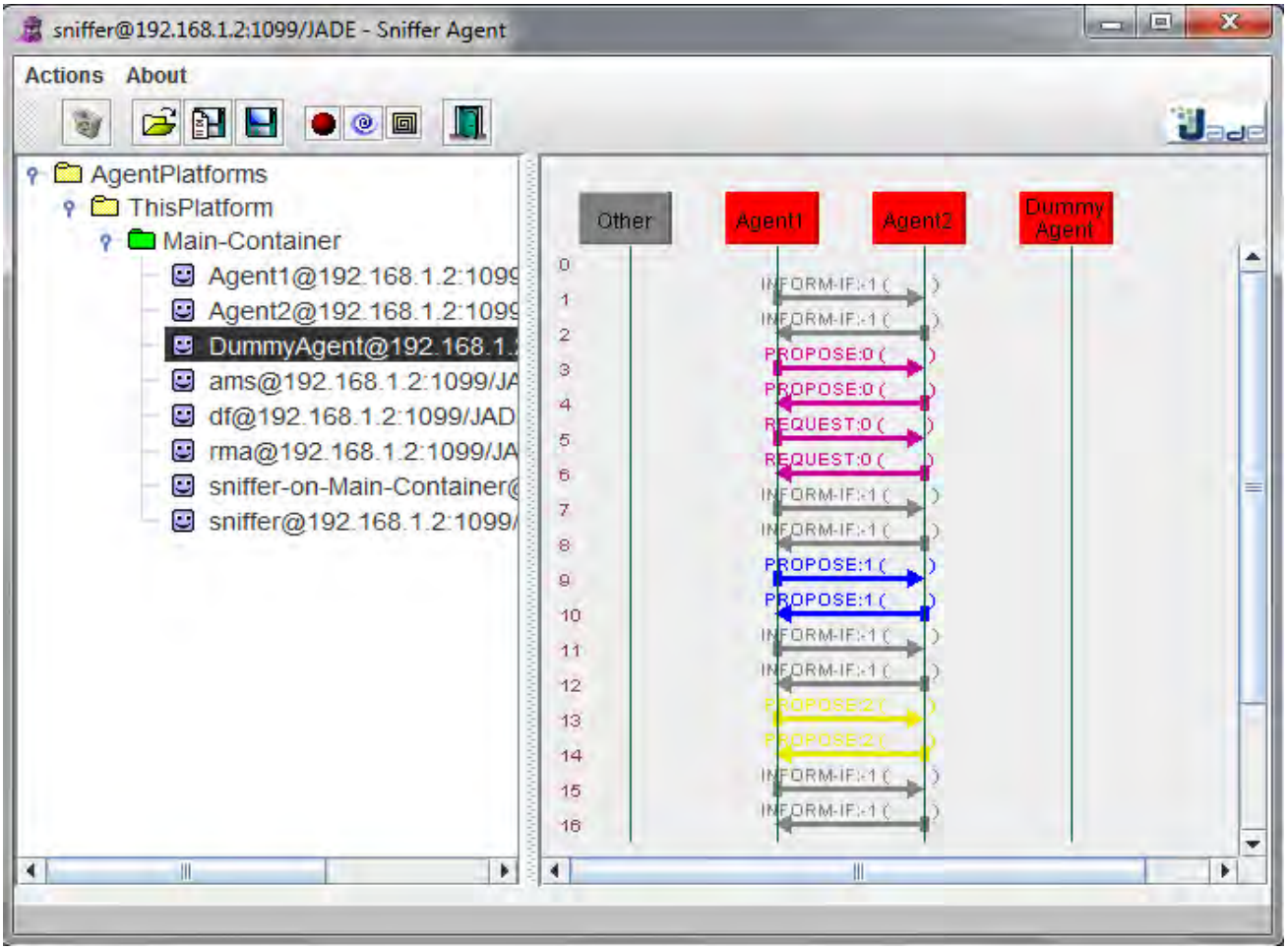

Launching the rma agent...
Launching the sniffer agent...
Platform is ready!
Launching agents...
DummyAgent launched!
Agent1 launched!
Agent2 launched!
Agents launched...
Press a key to start the agents!
Starting agents...
Agents started...
Agent1 has no neighbours.
Agent2 has no neighbours.


About Consumer1
Agent1: My distance from Consumer1 is 4
Agent2: My distance from Consumer1 is 4
Agent1: We have the same distance from Consumer1, so let's exchange our capacities to decide
Agent1: My capacity is 5
Agent2: My capacity is 8
Agent1: You will serve Consumer1
Agent2: I will serve Consumer1.
My capacity is now 5 and the consumer's requirements are now 0
My current position is: 94


About Consumer2
Agent1: My distance from Consumer2 is 15
Agent2: My distance from Consumer2 is 11
Agent1: You will serve Consumer2
Agent2: I will serve Consumer2
My capacity is now 1 and the consumer's requirements are now 0
My current position is: 20
About Consumer3
Agent1: My distance from Consumer3 is 11
Agent2: My distance from Consumer3 is 8
Agent1: You will serve Consumer3
Agent2: I will serve Consumer3
My capacity is now 0 and the consumer's requirements are now 0
My current position is: 6

```
Agent1 will try to serve the remaining consumers, as the other agent's capacity is 0.

The consumers that are unserved are:
Consumer4

Agent1 served Consumer4
His capacity is now 3
His current position is 46
```

Εικόνα 4.22 Η διαδικασία εξυπηρέτησης των καταναλωτών



Εικόνα 4.23 Η ανταλλαγή μηνυμάτων μεταξύ των πρακτόρων για τη λήψη απόφασης

i \ j	0	1	2	3	4	5	6	7	8	9
0							C3 A2			
1										
2	C2									
3										
4							C4 A1			
5										
6										
7										
8										
9					C1					

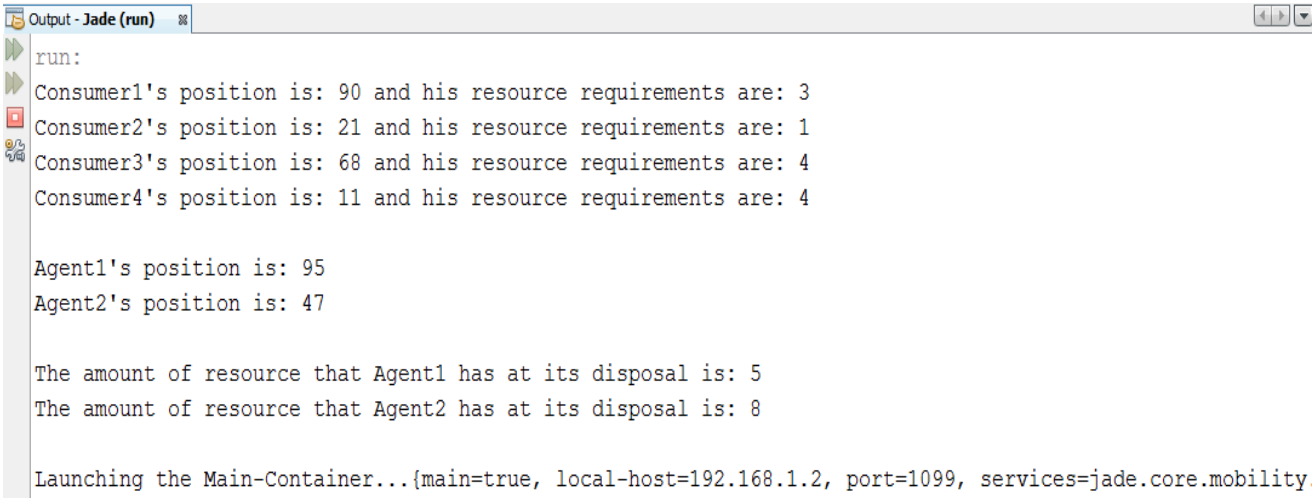
Εικόνα 4.24 Τελική κατάσταση του περιβάλλοντος
Ai, i=1,2 : οι δύο πράκτορες
Cj, j=1,2,3,4 : οι τέσσερις καταναλωτές
(πράσινο: έχει εξυπηρετηθεί, κόκκινο: δεν έχει εξυπηρετηθεί)
Υπολογισμός θέσης πράκτορα ή καταναλωτή: i*10 + j

Σενάριο 4:

Σε αυτό το σενάριο παρουσιάζεται τι γίνεται εάν κατά τη φάση επικοινωνίας προκύψει για κάποιον καταναλωτή ότι κανένας πράκτορας δεν έχει την απαιτούμενη ποσότητα πόρου ώστε να τον εξυπηρετήσει. Στο συγκεκριμένο παράδειγμα, αυτό ισχύει για τον καταναλωτή Consumer4. Οπότε, οι ανάγκες του συγκεκριμένου καταναλωτή θα μείνουν ανικανοποίητες. Η *Εικόνα 4.26* δείχνει αναλυτικά την εξέλιξη του αλγορίθμου, ενώ η *Εικόνα 4.27* δείχνει τα μηνύματα που ανταλλάσσουν οι δύο πράκτορες. Οι *Εικόνες 4.25* και *4.28* απεικονίζουν αντίστοιχα την αρχική και τελική κατάσταση του περιβάλλοντος.

i \ j	0	1	2	3	4	5	6	7	8	9
0										
1		C4								
2		C2								
3										
4								A2		
5										
6									C3	
7										
8										
9	C1					A1				

Εικόνα 4.25 Αρχική κατάσταση του περιβάλλοντος
Ai, i=1,2 : οι δύο πράκτορες
Cj, j=1,2,3,4 : οι τέσσερις καταναλωτές
(πράσινο: έχει εξυπηρετηθεί, κόκκινο: δεν έχει εξυπηρετηθεί)
Υπολογισμός θέσης πράκτορα ή καταναλωτή: i*10 + j

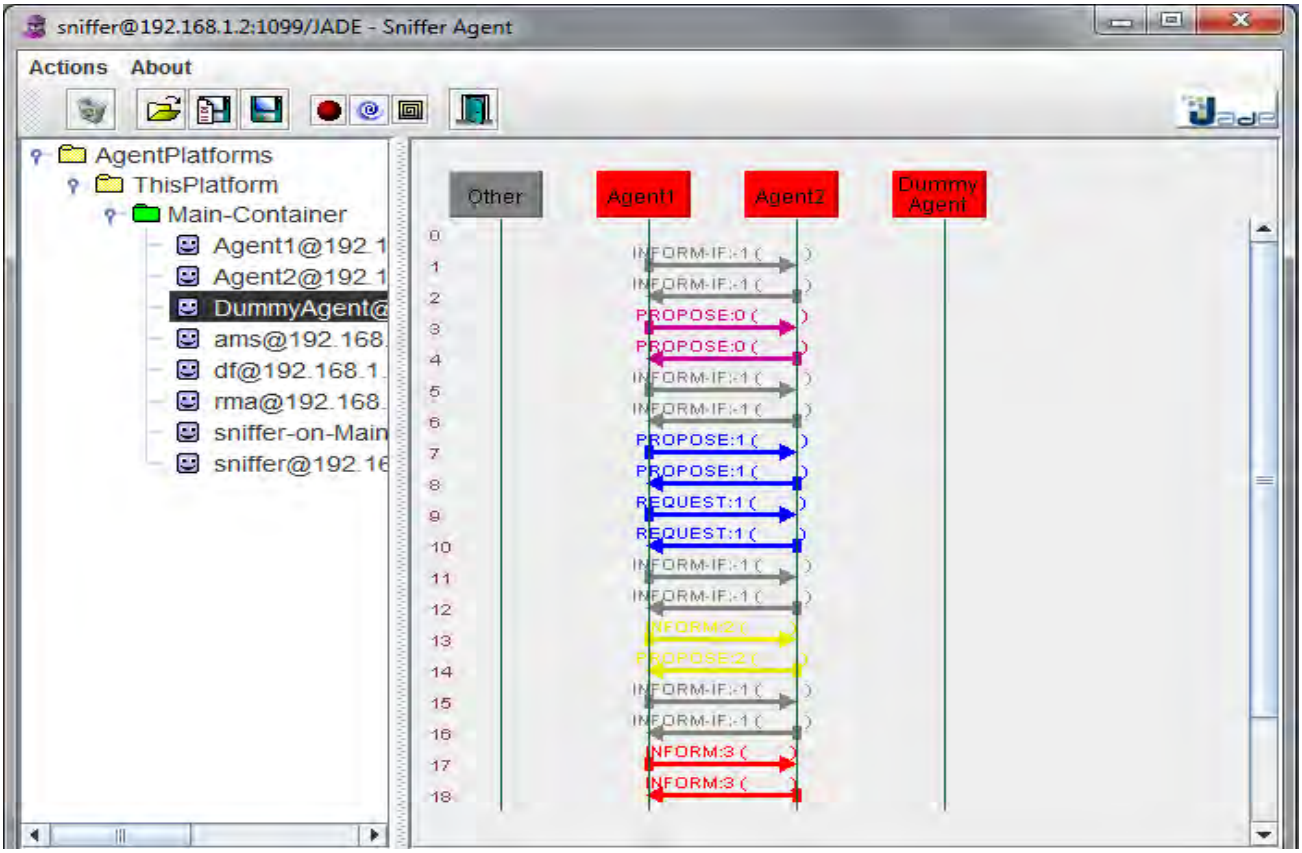


```
Launching the rma agent...
Launching the sniffer agent...
Platform is ready!
Launching agents...
DummyAgent launched!
Agent1 launched!
Agent2 launched!
Agents launched...
Press a key to start the agents!

Starting agents...
Agents started...

Agent2 has no neighbours.

Agent1 has no neighbours.
```

```
About Consumer1
--------------------
Agent1: My distance from Consumer1 is 5
Agent2: My distance from Consumer1 is 12
Agent1: I will serve Consumer1
        My capacity is now 2 and the consumer's requirements are now 0
        My current position is: 90
Agent2: You will serve Consumer1


About Consumer2
--------------------
Agent1: My distance from Consumer2 is 8
Agent2: My distance from Consumer2 is 8
Agent1: We have the same distance from Consumer2, so let's exchange our capacities to decide
Agent1: My capacity is 2
Agent2: My capacity is 8
Agent1: You will serve Consumer2
Agent2: I will serve Consumer2.
        My capacity is now 7 and the consumer's requirements are now 0
        My current position is: 21
```

```
About Consumer3
--------------------
Agent1: My capacity (2) is less than the Consumer3's resource requirements (4).
Agent2: My distance from Consumer3 is 11
Agent2: I will serve Consumer3.
        My capacity is now 3 and the consumer's requirements are now 0
        My current position is: 68


About Consumer4
--------------------
Agent1: My capacity (2) is less than the Consumer4's resource requirements (4).
Agent2: My capacity (3) is less than the Consumer4's resource requirements (4).


After the phase of negotiation:
--------------------------------
The consumers that will remain unserved forever are:
Consumer4
```

Εικόνα 4.26 Η διαδικασία εξυπηρέτησης των καταναλωτών



Εικόνα 4.27 Η ανταλλαγή μηνυμάτων μεταξύ των πρακτόρων για τη λήψη απόφασης

i \ j	0	1	2	3	4	5	6	7	8	9
0										
1		C4								
2		C2								
3										
4										
5										
6									C3 A2	
7										
8										
9	C1 A1									

Εικόνα 4.28 Τελική κατάσταση του περιβάλλοντος

Ai, i=1,2 : οι δύο πράκτορες

Cj, j=1,2,3,4 : οι τέσσερις καταναλωτές

(πράσινο: έχει εξυπηρετηθεί, κόκκινο: δεν έχει εξυπηρετηθεί)

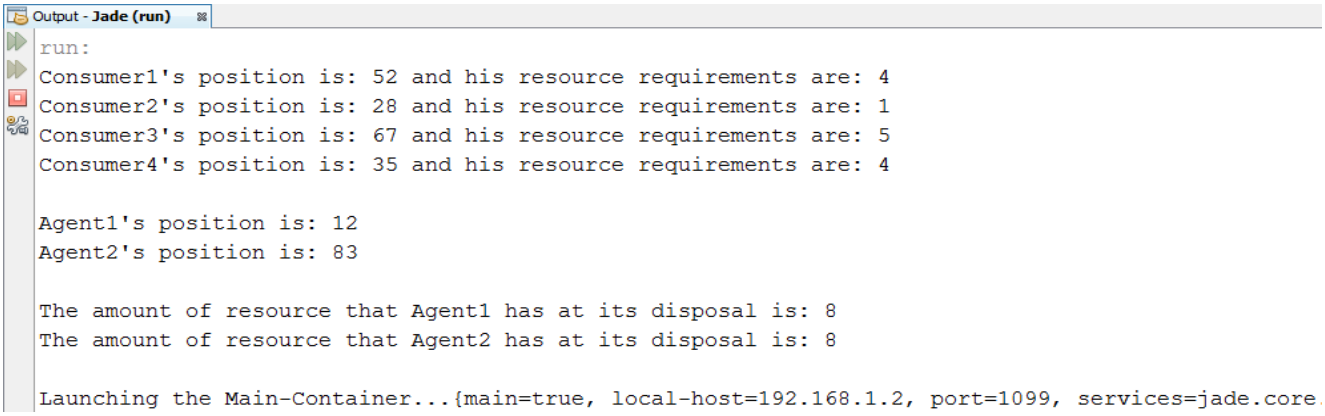
Υπολογισμός θέσης πράκτορα ή καταναλωτή: i*10 + j

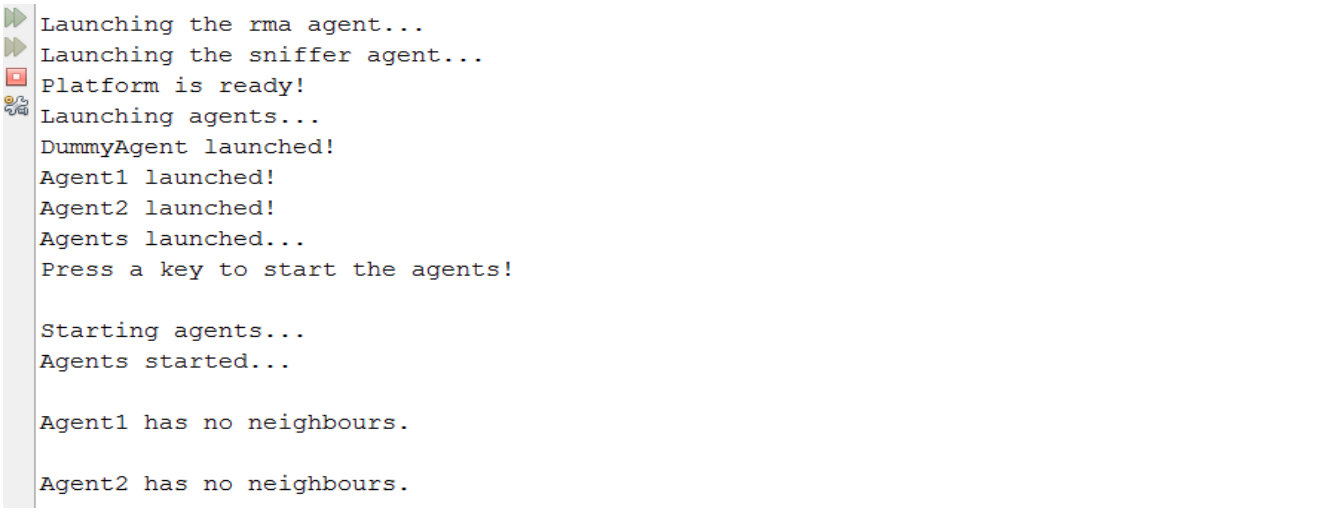
Σενάριο 5:

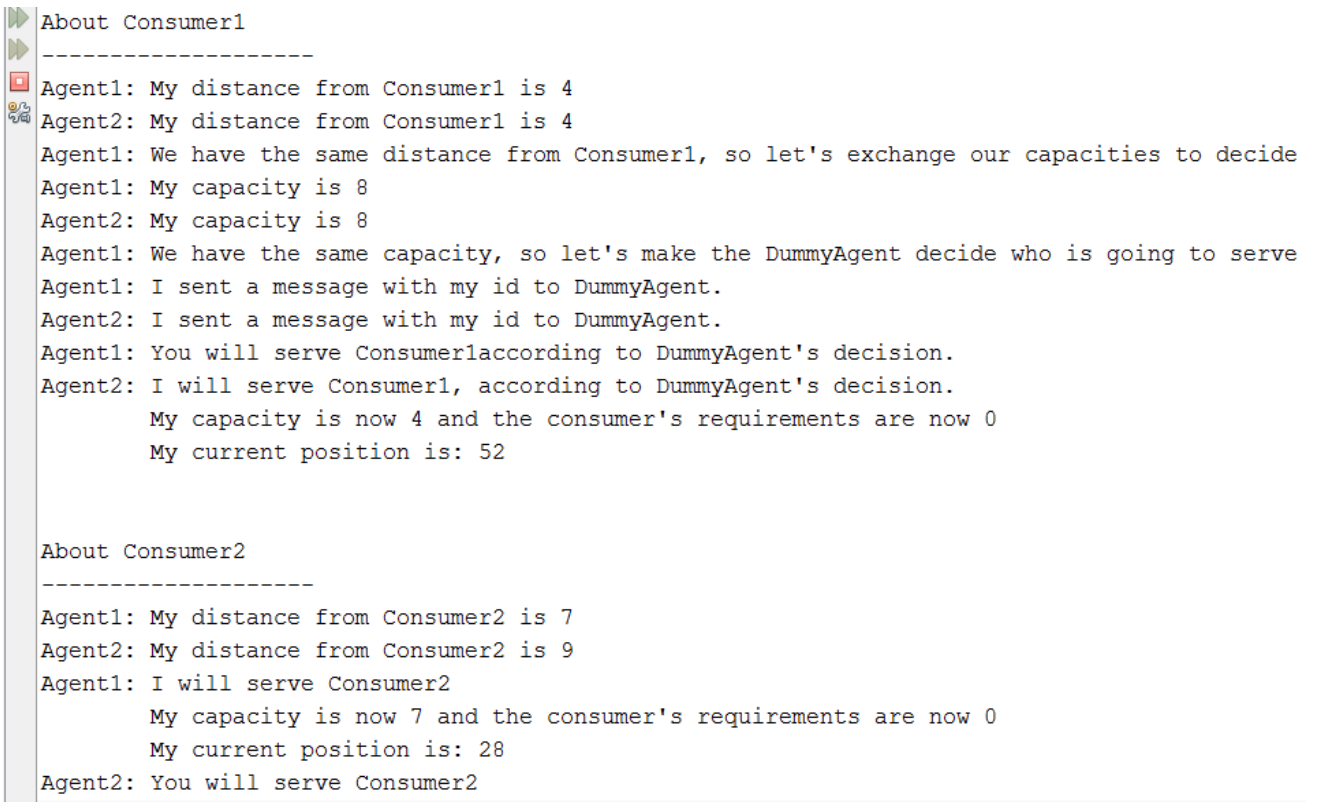
Σε αυτό το σενάριο εξετάζεται τι συμβαίνει εάν, κατά τη $2^{η}$ φάση του αλγορίθμου, ο έλεγχος του πρωτεύοντος αλλά και του δευτερεύοντος κριτηρίου διαπραγμάτευσης μεταξύ των πρακτόρων δεν μπορεί να οδηγήσει στη λήψη απόφασης για το ποιος θα εξυπηρετήσει ένα συγκεκριμένο καταναλωτή. Στο παράδειγμα που μελετάται, όπως φαίνεται και στην *Εικόνα 4.30*, προκύπτει ότι οι δύο πράκτορες ισαπέχουν από τον Consumer1, οπότε δεν μπορούν να καταλήξουν σε απόφαση βάσει της απόστασης (πρωτεύον κριτήριο). Έτσι, ακολουθεί ένας καινούριος γύρος ανταλλαγής μηνυμάτων στον οποίο ελέγχουν την ποσότητα πόρου που έχει ο καθένας στη διάθεσή του (δευτερεύον κριτήριο). Ούτε, όμως, και σε αυτή την περίπτωση μπορεί να βγει κάποιο συμπέρασμα, γι' αυτό αποφασίζουν ότι χρειάζονται τη βοήθεια του Dummy Agent. Οι δύο πράκτορες του στέλνουν ένα μήνυμα, αυτός αποφασίζει τυχαία ποιος θα αναλάβει την εξυπηρέτηση του συγκεκριμένου καταναλωτή, και τους απαντάει με στόχο να τους ενημερώσει για την απόφασή του. Για τους Consumer2, Consumer3 και Consumer4 η διαδικασία απόφασης δεν αντιμετωπίζει κανένα πρόβλημα. Στην *Εικόνα 4.31* απεικονίζονται αναλυτικά όλα τα μηνύματα που ανταλλάσσονται μεταξύ των δύο πρακτόρων κατά την εκτέλεση της $2^{ης}$ φάσης του αλγορίθμου, ενώ οι *Εικόνες 4.29* και *4.32,* παρουσιάζουν αντίστοιχα την αρχική και την τελική κατάσταση του περιβάλλοντος.

i \ j	0	1	2	3	4	5	6	7	8	9
0										
1			A1							
2									C2	
3						C4				
4										
5			C1							
6								C3		
7										
8				A2						
9										

Εικόνα 4.29 Αρχική κατάσταση του περιβάλλοντος
Ai, i=1,2 : οι δύο πράκτορες
Cj, j=1,2,3,4 : οι τέσσερις καταναλωτές
(πράσινο: έχει εξυπηρετηθεί, κόκκινο: δεν έχει εξυπηρετηθεί)
Υπολογισμός θέσης πράκτορα ή καταναλωτή: i*10 + j


Output - Jade (run)
run:
Consumer1's position is: 52 and his resource requirements are: 4
Consumer2's position is: 28 and his resource requirements are: 1
Consumer3's position is: 67 and his resource requirements are: 5
Consumer4's position is: 35 and his resource requirements are: 4
Agent1's position is: 12
Agent2's position is: 83
The amount of resource that Agent1 has at its disposal is: 8
The amount of resource that Agent2 has at its disposal is: 8
Launching the Main-Container...{main=true, local-host=192.168.1.2, port=1099, services=jade.core


Launching the rma agent...
Launching the sniffer agent...
Platform is ready!
Launching agents...
DummyAgent launched!
Agent1 launched!
Agent2 launched!
Agents launched...
Press a key to start the agents!
Starting agents...
Agents started...
Agent1 has no neighbours.
Agent2 has no neighbours.

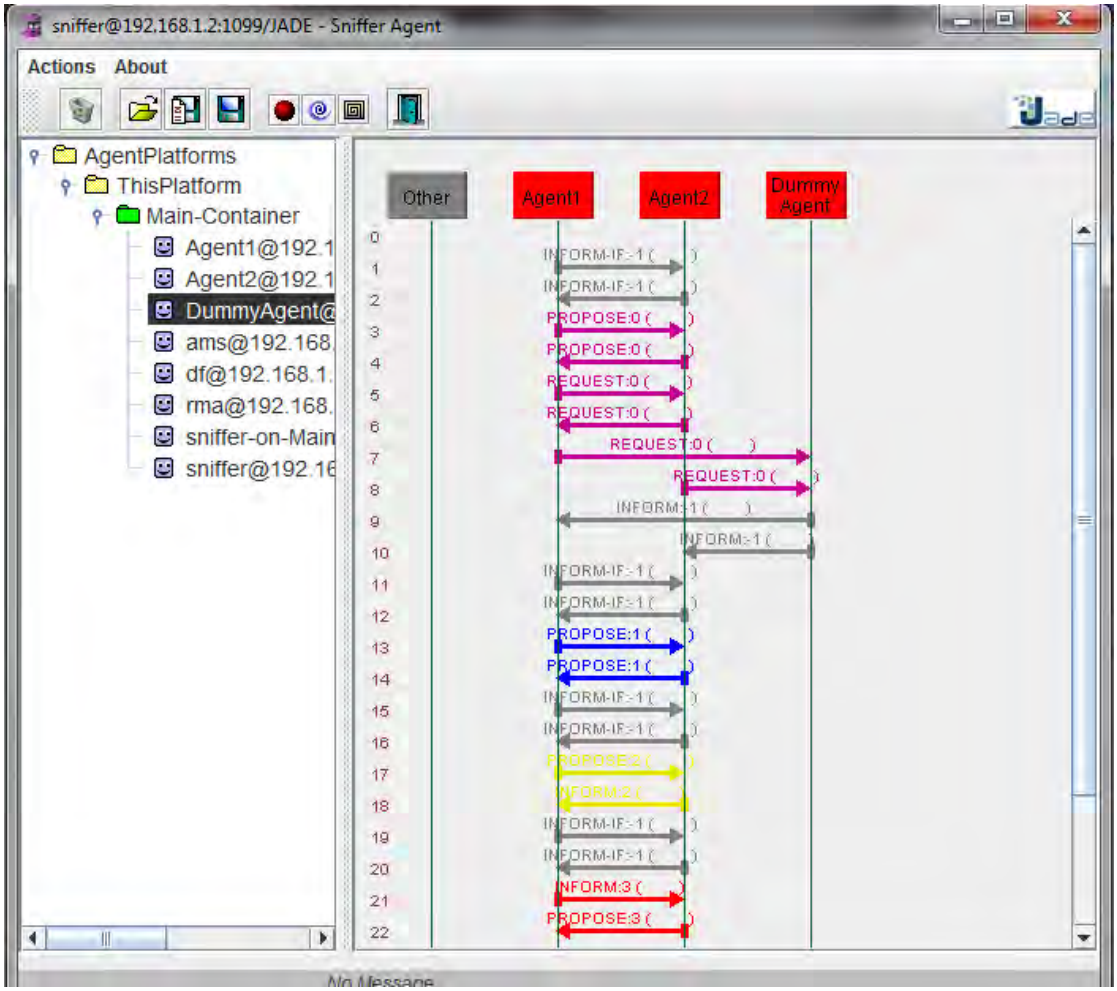

About Consumer1

Agent1: My distance from Consumer1 is 4
Agent2: My distance from Consumer1 is 4
Agent1: We have the same distance from Consumer1, so let's exchange our capacities to decide
Agent1: My capacity is 8
Agent2: My capacity is 8
Agent1: We have the same capacity, so let's make the DummyAgent decide who is going to serve
Agent1: I sent a message with my id to DummyAgent.
Agent2: I sent a message with my id to DummyAgent.
Agent1: You will serve Consumer1according to DummyAgent's decision.
Agent2: I will serve Consumer1, according to DummyAgent's decision.
My capacity is now 4 and the consumer's requirements are now 0
My current position is: 52
About Consumer2

Agent1: My distance from Consumer2 is 7
Agent2: My distance from Consumer2 is 9
Agent1: I will serve Consumer2
My capacity is now 7 and the consumer's requirements are now 0
My current position is: 28
Agent2: You will serve Consumer2

```
About Consumer3
--------------------
Agent1: My distance from Consumer3 is 5
Agent2: My capacity (4) is less than the Consumer3's resource requirements (5).
Agent1: I will serve Consumer3.
        My capacity is now 2 and the consumer's requirements are now 0
        My current position is: 67
Agent2: You will serve Consumer3


About Consumer4
--------------------
Agent1: My capacity (2) is less than the Consumer4's resource requirements (4).
Agent2: My distance from Consumer4 is 5
Agent2: I will serve Consumer4.
        My capacity is now 0 and the consumer's requirements are now 0
        My current position is: 35


After the phase of negotiation:
--------------------------------
There are no unserved consumers.
The agents managed to satisfy the needs of all of them.
```

Εικόνα 4.30 Η διαδικασία εξυπηρέτησης των καταναλωτών



Εικόνα 4.31 Η ανταλλαγή μηνυμάτων μεταξύ των πρακτόρων για τη λήψη απόφασης

i \ j	0	1	2	3	4	5	6	7	8	9
0										
1										
2									C2	
3						C4 A2				
4										
5			C1							
6								C3 A1		
7										
8										
9										

Εικόνα 4.32 Τελική κατάσταση του περιβάλλοντος

Ai, i=1,2 : οι δύο πράκτορες

Cj, j=1,2,3,4 : οι τέσσερις καταναλωτές

(πράσινο: έχει εξυπηρετηθεί, κόκκινο: δεν έχει εξυπηρετηθεί)

Υπολογισμός θέσης πράκτορα ή καταναλωτή: i*10 + j

5. Μελλοντικές επεκτάσεις

Έχοντας παρουσιάσει τη γενική σχεδίαση ενός αλγορίθμου για την επίλυση του προβλήματος της διαχείρισης πόρων από ένα πολυπρακτορικό σύστημα, καθώς και τα αποτελέσματα που προκύπτουν από εφαρμογή αυτού με συγκεκριμένα δεδομένα και υποθέσεις, αξίζει να σημειωθούν κάποιες τροποποιήσεις ή επεκτάσεις που μπορούν να γίνουν. Κάποιες από αυτές αφορούν διαφορετικές παραδοχές για το ίδιο το πρόβλημα και τη διατύπωσή του, ενώ άλλες σχετίζονται με τον τρόπο με τον οποίο αυτό μπορεί να μελετηθεί και να επιλυθεί. Σε κάθε περίπτωση, οι αλλαγές αυτές αυξάνουν την πολυπλοκότητα του προβλήματος και καθιστούν πιο χρονοβόρα και απαιτητική την επίλυσή του, ταυτόχρονα όμως αποτελούν κίνητρο για τη μελέτη του σε μεγαλύτερο βάθος και εμφανίζουν ιδιαίτερο ερευνητικό ενδιαφέρον.

5.1 Εισαγωγή παραμέτρων στο πρόβλημα

Το αρχικό πρόβλημα μπορεί να μελετηθεί από διαφορετικές σκοπιές, αλλάζοντας τις υποθέσεις που έχουν γίνει για τη συγκεκριμένη υλοποίηση. Αυτό γίνεται με την δημιουργία μίας σειράς σεναρίων, σε κάθε ένα από τα οποία αναιρείται μία ή περισσότερες από τις αρχικές παραδοχές που έχουν γίνει για τη μελέτη του προβλήματος στα πλαίσια της παρούσας εργασίας.

- **Εισαγωγή δυναμικότητας στο περιβάλλον**
  Η έννοια της δυναμικότητας μπορεί να επηρεάσει τον τρόπο με τον οποίο έχουν οριστεί τα διάφορα στοιχεία του περιβάλλοντος του προβλήματος που εξετάζεται. Αυτό σημαίνει ότι μπορεί να θεωρηθεί ότι:

  - Είναι δυναμικός, και όχι συγκεκριμένος και στατικός, ο αριθμός των καταναλωτών ή ο αριθμός των πρακτόρων ή και οι δύο μαζί. Αυτό ισοδυναμεί με αυξομείωση ενός ή και των δύο πληθυσμών κατά τη διάρκεια εφαρμογής του αλγορίθμου. Η δυσκολία εντοπίζεται στο γεγονός ότι ο καταναλωτής ή οποιοσδήποτε από τους πράκτορες που συμμετέχουν σε ένα γύρο διαπραγμάτευσης μπορούν να «εξαφανιστούν» ανά πάσα στιγμή από το περιβάλλον πριν τη λήξη του γύρου, όπως επίσης ότι μπορεί ανά πάσα στιγμή να εμφανιστούν πράκτορες οι οποίοι θέλουν να συμμετέχουν κι αυτοί στη διαπραγμάτευση (καθώς έχουν την απαιτούμενη ποσότητα πόρου για να ικανοποιήσουν τις ανάγκες του καταναλωτή). Σε μία τέτοια περίπτωση, ο κάθε πράκτορας θα πρέπει συνέχεια να ελέγχει εάν ο καταναλωτής εξακολουθεί να υπάρχει και να εντοπίζει και ποιοι πράκτορες υπάρχουν στο περιβάλλον, για να ξέρει σε ποιους θα στείλει μήνυμα με την προσφορά του (bid) για την εξυπηρέτηση του καταναλωτή.

  - Είναι δυναμική, και όχι καθορισμένη βάσει της θέσης του (αυτής που έχει κατά την έναρξη εκτέλεσης του αλγορίθμου), η ποσότητα του πόρου που

έχει ο κάθε πράκτορας στη διάθεσή του, η ανάγκη του κάθε καταναλωτή για τον πόρο ή και οι δύο ποσότητες μαζί. Η δυναμική μεταβολή αυτών των δύο ποσοτήτων μπορεί να υλοποιηθεί και να αντιπροσωπεύει την επιρροή που πιθανώς να έχουν διάφοροι εξωτερικοί παράγοντες σε αυτές, όπως θα γινόταν σε ένα αντίστοιχο πρόβλημα στον πραγματικό κόσμο. Αυτό σημαίνει ότι θα πρέπει να γίνονται συνεχείς έλεγχοι στην ποσότητα πόρου που ζητά ένας καταναλωτής και στην ποσότητα πόρου που διαθέτει ένας πράκτορας, γιατί ακόμα κι αν έχει γίνει διαπραγμάτευση και έχουν ληφθεί κάποιες αποφάσεις, όλα μπορούν να ανατραπούν.

- Είναι δυναμική η θέση του κάθε πράκτορα και του κάθε καταναλωτή. Ανάλογα με το πρόβλημα πραγματικού κόσμου που πιθανώς να αντιπροσωπεύει η υλοποίηση της παρούσας εργασίας, μπορεί να θεωρηθεί ότι είτε λόγω προγραμματιστικής επιλογής για τις μετακινούμενες οντότητες (πράκτορες, καταναλωτές) του περιβάλλοντος, είτε λόγω εξωτερικών παραγόντων, οι καταναλωτές και οι πράκτορες αλλάζουν ανά τακτά χρονικά διαστήματα τη θέση τους στο χώρο.

- Είναι δυναμική η ποσότητα του πόρου που υπάρχει σε κάθε θέση του περιβάλλοντος. Αυτό, αυτομάτως, αναιρεί και την υπόθεση της ισοκατανομής του πόρου. Είτε κάποια προγραμματιστική επιλογή είτε διάφοροι εξωτερικοί παράγοντες μπορούν ανά πάσα στιγμή να αυξάνουν ή να μειώνουν την ποσότητα του πόρου στο περιβάλλον και όχι με τον ίδιο τρόπο σε κάθε σημείο. Οπότε, αυτό θα έχει ως αποτέλεσμα κάποιες θέσεις στο περιβάλλον να έχουν μεγαλύτερη ποσότητα πόρου σε σχέση με άλλες και κατά συνέπεια θα επηρεάζεται και η ποσότητα του πόρου που αποδίδεται σε κάθε πράκτορα κατά την έναρξη εκτέλεσης του αλγορίθμου (αν θεωρηθεί ότι αυτή καθορίζεται αποκλειστικά βάσει της θέσης του, όπως έχει γίνει και στα πλαίσια της παρούσας εργασίας).

- **Συμμετοχή και επίδραση εξωτερικών παραγόντων**

  Κάθε πρόβλημα αποτελείται από κάποια στοιχεία τα οποία πρέπει να ληφθούν υπ' όψιν για την εύρεση ενός τρόπου επίλυσής του. Πολλές φορές, όμως, υπάρχουν και εξωτερικοί παράγοντες (στοιχεία που δεν ανήκουν στο άμεσο περιβάλλον του προβλήματος) που μπορούν να αυξάνουν ή να μειώνουν την πολυπλοκότητα του προβλήματος και κατά συνέπεια να παίζουν σημαντικό ρόλο στο σχεδιασμό της λύσης του. Το πρόβλημα της διαχείρισης πόρων μελετάται στην παρούσα εργασία χωρίς να λαμβάνονται υπ' όψιν εξωτερικές συνθήκες που πιθανώς να το επηρεάζουν. Οι μόνες μεταβολές που συμβαίνουν αφορούν στις οντότητες που υπάρχουν μέσα στο περιβάλλον του προβλήματος και αυτές εξετάζονται κατά τη διάρκεια εκτέλεσης του αλγορίθμου για την εύρεση λύσης. Αν όμως αυτή η γενική θεώρηση του προβλήματος, με τις συγκεκριμένες παραδοχές που έχουν γίνει, αναχθεί σε ένα συγκεκριμένο σενάριο του πραγματικού κόσμου, τότε θα υπάρχει ένα σύνολο από εξωτερικά γεγονότα που θα επηρεάζουν τη διαδικασία επίλυσης

του προβλήματος και τα οποία θα πρέπει να ληφθούν υπ' όψιν για μία ολοκληρωμένη από κάθε σκοπιά λύση.

- **Διαφορετικοί τύποι πόρων και ανάγκες καταναλωτών**

  Η μελέτη του προβλήματος στην παρούσα εργασία γίνεται με την υπόθεση ότι υπάρχει μόνο ένας τύπος πόρου. Στην γενικότερη περίπτωση, όμως, και σε ένα αντίστοιχο πρόβλημα στον πραγματικό κόσμο, υπάρχει η πιθανότητα στο περιβάλλον να υπάρχει ένα πλήθος από διαφορετικούς πόρους ο καθένας από τους οποίους είναι απαραίτητος για διαφορετικούς λόγους. Αντίστοιχα, οι καταναλωτές σε ένα τέτοιο σενάριο είναι πολύ πιθανόν να έχουν ανάγκη για ένα συνδυασμό από τους διαθέσιμους πόρους στο περιβάλλον τους και να χρειάζονται διαφορετική ποσότητα από τον καθένα, όπως επίσης και οι πράκτορες να έχουν στη διάθεσή τους μία ποικιλία από πόρους. Είναι φανερό ότι σε αυτή την περίπτωση θα πρέπει να γίνουν αρκετές αλλαγές στον αλγόριθμο επίλυσης, καθώς επίσης να λαμβάνονται υπ' όψιν περισσότερες παράμετροι κατά τη διάρκεια της διαπραγμάτευσης μεταξύ των πρακτόρων. Η ικανότητα ενός πράκτορα να εξυπηρετήσει ένα συγκεκριμένο καταναλωτή και κατά συνέπεια να μπορεί να υποβάλλει την πρότασή του σε ένα γύρο διαπραγμάτευσης, θα εξαρτάται από πολλά κριτήρια. Γι' αυτόν το λόγο ίσως θα πρέπει να σχηματιστεί ένα μαθηματικό μοντέλο το οποίο θα λαμβάνει υπ' όψιν του τις διάφορες παραμέτρους με διαφορετικό βάρος την κάθε μία, ώστε να προκύψει τελικά ένα αποτέλεσμα που θα αντιπροσωπεύει την ικανότητα εξυπηρέτησης του κάθε πράκτορα.

- **Απόδοση προτεραιοτήτων σε καταναλωτές ή/και ανάγκες**

  Υπάρχει περίπτωση, στα πλαίσια ενός προβλήματος, όπως αυτό που περιγράφεται σε αυτή την εργασία, να πρέπει να αποδοθούν προτεραιότητες, δηλαδή στοιχεία που φανερώνουν ποιος καταναλωτής έχει μεγαλύτερη ανάγκη για άμεση εξυπηρέτηση. Η προτεραιότητα που αντιστοιχεί σε έναν καταναλωτή μπορεί να είναι το αποτέλεσμα εκτίμησης των επιμέρους προτεραιοτήτων που αποδίδονται στις ανάγκες του για τα διάφορα είδη πόρων που υπάρχουν στο περιβάλλον του. Οι πράκτορες θα έχουν πλέον ως πρωτεύον κριτήριο για την εξυπηρέτηση που θα προσφέρουν το αν κάποιος καταναλωτής έχει μεγαλύτερη προτεραιότητα σε σχέση με κάποιον άλλο, ανεξάρτητα από το αν απέχει περισσότερο.

5.2 Τροποποιήσεις στον αλγόριθμο επίλυσης του προβλήματος

Εκτός από τις τροποποιήσεις που αναφέρθηκαν στην προηγούμενη ενότητα και πρέπει να γίνουν στον αλγόριθμο επίλυσης σε περίπτωση που αναιρεθούν ή τροποποιηθούν κάποιες από τις αρχικές παραδοχές για το πρόβλημα, μπορούν να γίνουν και κάποιες βελτιστοποιήσεις ή και επεκτάσεις στον ήδη υπάρχοντα αλγόριθμο.

- Όπως έχει αναφερθεί σε προηγούμενο κεφάλαιο, πρωτεύον κριτήριο της διαπραγμάτευσης μεταξύ των πρακτόρων είναι η απόσταση τους από κάθε καταναλωτή. Οπότε, κάθε φορά ελέγχεται αυτή η απόσταση και τελικά ο καταναλωτής εξυπηρετείται από τον πράκτορα που βρίσκεται πιο κοντά του. Ο τρόπος με τον οποίο τοποθετούνται οι καταναλωτές στις διάφορες λίστες (unservedConsumers, neighbours[i], $unservedByAgent_i$, i=0, 1), που χρησιμοποιούνται στην υλοποίηση της παρούσας εργασίας, έχει σχέση αποκλειστικά και μόνο με τη χρονική σειρά με την οποία δημιουργούνται (birth time). Αυτή είναι και η σειρά με την οποία εξετάζονται και από τους πράκτορες για να εξυπηρετηθούν. Αυτό που αξίζει να παρατηρηθεί σε αυτό το σημείο είναι ότι τελικά οι καταναλωτές θα ελεγχθούν και θα εξυπηρετηθούν λαμβάνοντας υπ' όψιν το κριτήριο της απόστασης μόνο σε σχέση με τους δύο πράκτορες και όχι και σε ατομικό επίπεδο για τον κάθε πράκτορα. Αυτό σημαίνει ότι ένας πράκτορας μπορεί να εξυπηρετήσει πρώτα έναν καταναλωτή επειδή είναι πιο κοντά του (σε σχέση με την απόσταση που έχει από τον άλλο πράκτορα), αλλά μόνο και μόνο επειδή είναι ο πρώτος που δημιουργήθηκε και άρα ο πρώτος που εξετάζεται και όχι επειδή είναι ο πιο κοντινός του σε σχέση με όλους τους καταναλωτές που περιμένουν εξυπηρέτηση από αυτόν. Θα μπορούσε, λοιπόν, να γίνει μία βαθύτερη μελέτη ώστε να βρεθεί ένας τρόπος με τον οποίο ο κάθε πράκτορας θα εξυπηρετεί κάθε φορά έναν καταναλωτή που όχι μόνο είναι σε μικρότερη απόσταση από αυτόν σε σχέση με τον άλλο πράκτορα, αλλά και σε μικρότερη απόσταση σε σχέση με όλους τους καταναλωτές που αποτελούν αρμοδιότητα του.

- Η πλατφόρμα JADE παρέχει τη δυνατότητα δημιουργίας κινητών πρακτόρων (mobile agents), οι οποίοι μπορούν να «μετακινηθούν» μέσα σε ένα δικτυακό περιβάλλον, προκειμένου να επιτύχουν τους στόχους τους. Στην ουσία, πρόκειται για διεργασίες (software processes), οι οποίες κατά τη διάρκεια της εκτέλεσης τους μεταφέρονται στους υπολογιστές που αποτελούν το δίκτυο – περιβάλλον.

  Υπάρχει περίπτωση το πρόβλημα της διαχείρισης πόρων να μην εντοπίζεται μόνο σε τοπικό επίπεδο, αλλά να έχει κατανεμημένο χαρακτήρα. Αυτό σημαίνει ότι το περιβάλλον του προβλήματος μπορεί να αντιστοιχεί σε μία JADE πλατφόρμα η οποία είναι κατανεμημένη μεταξύ ενός αριθμού από host (Jade distributed Agent Platform), με αποτέλεσμα όλοι οι καταναλωτές και οι πράκτορες να μη βρίσκονται σε ένα μόνο host. Στον κάθε host μπορεί να παρέχεται μόνο ένα συγκεκριμένο είδος πόρου και οι πράκτορες που βρίσκονται σε αυτόν να μπορούν να εξυπηρετήσουν μόνο ανάγκες που σχετίζονται με αυτόν τον πόρο, ωστόσο οι καταναλωτές μπορεί να ζητούν ποσότητες από τα είδη πόρων που υπάρχουν σε όλο το περιβάλλον, δηλαδή σε όλους τους hosts. Σε αυτό το σημείο, θα βοηθήσει η δυνατότητα των πρακτόρων να «μετακινούνται», ώστε να βρεθούν σε έναν άλλο host και να εξυπηρετήσουν την ανάγκη ενός καταναλωτή για τον πόρο που αυτοί έχουν στη διάθεσή τους. Σε ένα τέτοιο σενάριο, δεν τίθεται θέμα επικοινωνίας των πρακτόρων και ανταλλαγής μηνυμάτων πάνω από το δίκτυο. Θα επικοινωνούν μόνο μεταξύ τους οι πράκτορες που βρίσκονται στον ίδιο host, ώστε να αποφασίζουν ποιος είναι κάθε στιγμή ικανός να εξυπηρετήσει ένα συγκεκριμένο

καταναλωτή. Σε μια άλλη θεώρηση του ίδιου προβλήματος, είναι πιθανό να είναι αναγκαία η επικοινωνία ή ακόμα και η μεταφορά μεγάλου όγκου δεδομένων πάνω από το δίκτυο. Σε κάθε περίπτωση, όταν το πρόβλημα στηρίζεται σε μία κατανεμημένη JADE πλατφόρμα, υπάρχουν κάποιες δυσκολίες. Αυτές σχετίζονται κυρίως με το κόστος μεταφοράς των πρακτόρων από μία τοποθεσία σε μία άλλη, καθώς και με το γεγονός ότι ο απομακρυσμένος έλεγχος των κινητών πρακτόρων είναι πιο περίπλοκος και απαιτητικός.

Βιβλιογραφία

Bellifemine, F., Caire, G., Poggi, A. και Rimassa G. (2003) *JADE: A White Paper.* exp (in search of innovation), Volume 3, n.3

Bousquet, F. και Le Page, C. (2004) *Multi-agent simulations and ecosystem management: a review.* Ecological Modelling 176, 313-332, Science Direct

Callegari, D.A. και de Oliveira, F.M. (2000) *Applying Reinforcement Learning to Improve MCOE, an Intelligent Learning Environment for Ecology.* O.Cairo, L.E. Sucar, και F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 284-293

Clearwater, S. H., Hogg, T. και Huberman, B. (1992) *Cooperative Problem Solving.* Computation: The Micro and the Macro View, pp. 33-70, World Scientific

Deadman, P. (1999) *Modelling individual behaviour and group performance in an intelligent agent-based simulation of the tragedy of the commons.* Journal of Environmental Management 56, 159-172

Durfee, E. H., Lesser, V. R. και Corkill, D. D. (1989b) *Trends in cooperative distributed problem solving.* IEEE Transactions on Knowledge and Data Engineering, 1(1), 63-83

Durfee, E. H. και Montgomery, T. A. (1989) *Using MICE to Study Intelligent Dynamic Coordination.* In 2nd International Conference on Tools for Artificial Intelligence

Engelmore, R. και Morgan, T. (1988) *Blackboard Systems.* Addison-Wesley, Reading, MA

Findler, N.V. και Elder, G.D. (1995) *Multi-agent coordination and cooperation in a distributed dynamic environment with limited resources.* Artificial Intelligence in Engineering

FIPA: Foundation for Intelligent Physical Agents (1999) Specification part 2 – agent communication language. The text refers to the specification dated 16 April 1999 (http://www.fipa.org)

FIPA ACL Message Structure Specification:
http://www.fipa.org/specs/fipa00061/SC00061G.html

FIPA Communicative Act Library Specification
http://www.fipa.org/specs/fipa00037/SC00037J.html#_Toc26729708

Franklin, S. και Graesser, A. (1997) *Is it an agent, or just a program?* Intelligent Agents III, LNAI Vol. 1193, pp. 21-36, Springer, Berlin

Fischer, K., Chaib-draa, B., Muller, J. P., Pischel, M. και Gerber, C. (1999) *A Simulation Approach Based on Negotiation and Cooperation Between Agents: A Case Study.* IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews, Vol. 29, No. 4

Golfarelli, M., Maio, D. και Rizzi, S. (1997) *Introducing Swap-Based Negotiations in the Contract Net Protocol.* In International Joint Conference on Artificial Intelligence

Jianshi, Z., Zhongjing, W. και Wenbin, W. (2003) *Theory and model of water resources complex adaptive allocation system.* Journal of Geographical Sciences 13(1), 112-122

Kim, K. και Park, J. (2009) *A Survey of Applications of Artificial Intelligence Algorithms in Eco-environmental Modelling.* Environ. Eng. Res. Vol. 14, No. 2, pp. 102-110, Korean Society of Environmental Engineers

Kraus, S. (1997) *Negotiation and Cooperation in Multi-Agent Environments.* In Artificial Intelligence, 94(1-2), pp. 79-98

Lesser, V., Atighetchi, M., Benyo, B. και Horling, B. (1998) *A Multi-Agent System for Intelligent Environmental Control.* In Proceedings of the 3rd International Conference on Autonomous Agents

Luck, M., McBurney, P. και Priest, C. (2003) *Agent Technology: Enabling Next Generation Computing, A Roadmap for Agent Based Computing.* European Network of Excellence for Agent-based Computing, pp. 41-46

NetBeans 7.0 and 7.1 UML plugin
http://michelelepri.blogspot.gr/2012/03/netbeans-70-and-71-uml-plugin.html

Rady, H. A. (2011) *Multi-Agent System for Negotiation in a Collaborative Supply Chain Management.* International Journal of Video & Image Processing and Network Secutiry IJVIPNS-IJENS Vol. 11, No. 05

Rosenchein, J.S. και Zlotkin, G. (1994) *Rules of Encounter Designing Conventions for Automated Negotiation among Computers.* MIT Press, Cambridge, MA

Russel, S. και Norvig, P. (1995) *Artificial Intelligence: A Modern Approach.* Prentice Hall, Englewood Cliffs, NJ

Smith, R. G. (1977) *The CONTRACT NET: a formalism for the control of distributed problem solving.* In Proceedings of the 5th International Joint Conference on Artificial Intelligence (IJCAI-77), Cambridge, MA

Smith, R. G. (1980a) *The contract net protocol.* IEEE Transactions on Computers, C29(12)

Smith, R. G. (1980b) *A Framework for Distributed Problem Solving.* UMI Research Press

Smith, R. G. και Davis, R. (1980) *Frameworks for cooperation in distributed problem solving.* IEEE Transactions on Systems, Man and Cybernetics, 11(1)

Villa et al. (2009) *ARIES (Artificial Intelligence for Ecosystem Services): a new tool for ecosystem services assessment, planning and valuation, BioEcon 2009*

Wooldridge, M. και Jennings, N. R. (1995), *Intelligent Agents: Theory and Practice.* The Knowledge Engineering Review, 10(2), pp. 115-152

Wooldrigde, M. και Jennings, N. R. (1999) *The Cooperative Problem-Solving Process.* J. Logic Computat., Vol. 9, No. 4, pp. 563-592, Oxford University Press

Wooldridge, M. (2002) *An introduction to multi-agent systems.* John Wiley & Sons