ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ
ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μοντέλο αξιολόγησης προστασίας προσωπικών δεδομένων μέσω της χρήσης δεικτών ποσοτικοποίησης της ιδιωτικότητας

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Νικόλαος Σ. Σαραντόπουλος

Αθήνα, Σεπτέμβριος 2009

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ
ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μοντέλο αξιολόγησης προστασίας προσωπικών δεδομένων μέσω της χρήσης δεικτών ποσοτικοποίησης της ιδιωτικότητας

A model for evaluating the level of protection of personal information through the use of privacy guantification indicators

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Νικόλαος Σ. Σαραντόπουλος

Επιβλέπων : Μ. Ε. Θεολόγου
Καθηγητής Ε.Μ.Π.

Αθήνα, Σεπτέμβριος 2009

.....

Νικόλαος Σ. Σαραντόπουλος

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

Περίληψη

Αντικείμενο της παρούσας διπλωματικής εργασίας είναι η μελέτη των εξατομικευμένων, προσωποποιημένων υπηρεσιών που βασίζονται στην θέση και την ταυτότητα των χρηστών για να παρέχουν την απαιτούμενη διαφοροποίηση. Η παροχή εξατομικευμένων υπηρεσιών συνεπάγεται την αποστολή εκ μέρους του χρήστη δεδομένων που αφορούν την θέση και την ταυτότητά του. Όσο ακριβέστερα είναι τα δεδομένα που αποστέλλονται, τόσο καλύτερα προσαρμοσμένη είναι η υπηρεσία στις ανάγκες του χρήστη. Από την άλλη, όσο μεγαλύτερη η αοριστία των δεδομένων που αποστέλλει ο χρήστης, τόσο καλύτερα προστατευμένη είναι η ιδιωτικότητά του. Αυτή τη σχέση μεταξύ αποτελεσματικότητας υπηρεσίας και προστασίας της ιδιωτικότητας προσπαθούμε να μοντελοποιήσουμε και να μελετήσουμε. Προτείνονται δείκτες ποσοτικοποίησης της ιδιωτικότητας και της αποτελεσματικότητας τους οποίους χρησιμοποιούμε για να εξάγουμε συμπεράσματα για τη μεταξύ τους σχέση, μέσω του προσομοιωτή Siafu. Με τη χρήση προσομοίωσης εξετάστηκε η επίδραση της γεωγραφικής κατανομής των προορισμών στη σχέση ιδιωτικότητας – αποτελεσματικότητας, καθώς και η επίδραση που έχει η κατηγοριοποίηση των προτιμήσεων των χρηστών μέσω των δέντρων προτιμήσεων που χρησιμοποιούν οι υπηρεσίες. Τα αποτελέσματα στα οποία καταλήξαμε μπορούν στη συνέχεια να χρησιμοποιηθούν για την προτυποποίηση και σχεδιασμό κινητών προσωποποιημένων εφαρμογών με ισχυρή προστασία της ιδιωτικότητας των χρηστών. Στο πρώτο κεφάλαιο γίνεται αναφορά στις βασικές έννοιες των συστημάτων εξατομικευμένων υπηρεσιών. Στο δεύτερο ορίζονται οι δείκτες ιδιωτικότητας και αποτελεσματικότητας υπηρεσίας που θα χρησιμοποιηθούν στα επόμενα. Στο τρίτο κεφάλαιο παρουσιάζεται το σενάριο της προσομοίωσης με το οποίο μελετήσαμε τις σχέσεις ιδιωτικότητας-αποτελεσματικότητας υπηρεσίας. Στο τέταρτο κεφάλαιο αναλύονται οι βασικές κλάσεις του κώδικα της προσομοίωσης, ενώ στο πέμπτο παρουσιάζονται τα τελικά αποτελέσματα και συμπεράσματα.

Λέξεις – κλειδιά

Κινητές υπηρεσίες, εξατομίκευση (personalization), επίγνωση κατάστασης (context awareness), υπηρεσίες βασισμένες στη θέση (location-based services), ιδιωτικότητα (privacy), ποιότητα υπηρεσίας, αποτελεσματικότητα υπηρεσίας, εντροπία, υπηρεσίες επίγνωσης περιβάλλοντος (context-aware services)

Abstract

The subject of this diploma thesis is the study of personalized, context-aware services that are based on the location and identity of the user to provide the required personalization. Providing personalized services involve sending from the user data on the location and identity. The more accurate the data sent, the better the service is tailored to the needs of the user. On the other hand, the greater the indeterminacy of the data sent by the user, the better the protection of his privacy is. This relationship between effectiveness of service and privacy is that we are trying to model and study. Quantifying indicators of privacy and efficiency are proposed and used in order to draw conclusions about the relationship between them through the simulator Siafu. Using simulation we examined the impact of the geographical distribution of destinations in respect to the privacy – effectiveness relationship and the effect of the categorization of preferences through the preferences-trees used by the services. The results we obtained can be used for prototyping and designing personalized mobile applications with strong privacy protection. The first chapter refers to the basic concepts behind personalized mobile services. In the second, we introduce the privacy and efficiency indicators that we are going to use. The third chapter presents the scenario of the simulation to be used. In the fourth chapter we provide the documentation of the main classes of the simulation code, while in the fifth we present the final results and conclusions.

Keywords

mobile services, personalization, context awareness, location-based services (LBS), privacy, quality of service (QOS), entropy, context-aware services

Θα ήθελα να ευχαριστήσω τον Καθηγητή κ. Μ.Ε.Θεολόγου για την καθοδήγηση και τις συμβουλές του, καθώς και τον Επιστημονικό Συνεργάτη του Τομέα Επικοινωνιών, Ηλεκτρονικής και Συστημάτων Πληροφορικής Δρα Χ.Πατρικάκη για την πολύτιμη βοήθεια, υποστήριξη και συνεργασία του.

ΠΕΡΙΕΧΟΜΕΝΑ

1. ΕΙΣΑΓΩΓΙΚΕΣ ΈΝΝΟΙΕΣ..... 10
1.1. Διάχυτος υπολογισμός (Ubiquitous computing)..... 10
1.2. Περιβάλλον (Context)..... 10
1.3. Επίγνωση περιβάλλοντος (Context awareness)..... 12
1.4. Επίγνωση θέσης (Location awareness)..... 13
1.5. Υπηρεσίες βασισμένες στη θέση (Location-Based Services)..... 14
1.5.1. Αρχιτεκτονική LBS..... 15
1.5.2. Μέθοδοι εντοπισμού θέσης και ακρίβεια..... 18

2. ΙΔΙΩΤΙΚΟΤΗΤΑ ΠΡΟΣΩΠΙΚΩΝ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΙΚΟΤΗΤΑ ΧΡΗΣΗΣ ΕΞΑΤΟΜΙΚΕΥΜΕΝΩΝ ΥΠΗΡΕΣΙΩΝ..... 21
2.1. Γενικά..... 21
2.2. Είδη πληροφοριών που αποστέλλονται σε μία προσωποποιημένη υπηρεσία..... 21
2.3. Μοντέλα για την προστασία της ιδιωτικότητας..... 22
2.3.1. Τεχνικές θολώματος (blurring)..... 22
2.3.2. Μοντέλο κ-ανωνυμίας (k-anonymity)..... 23
2.3.3. Χρήση επιπέδων αφαίρεσης..... 26
2.4. Ποσοτικοποίηση ιδιωτικότητας και αποτελεσματικότητας..... 26
2.4.1. Εντροπία εντοπισμού θέσης..... 27
2.4.2. Εντροπία προσωπικών προτιμήσεων..... 28
2.4.3 Αποτελεσματικότητα υπηρεσίας..... 29

3. ΣΕΝΑΡΙΟ ΠΡΟΣΟΜΟΙΩΣΗΣ..... 32
3.1. Γενικά..... 32
3.2. Η πλατφόρμα PLASMA..... 32
3.3. Χρήση υπηρεσίας και παραδείγματα υπηρεσιών..... 33
3.4. Σενάριο προσομοίωσης..... 35
3.4.1. Η υπηρεσία της προσομοίωσης..... 35
3.4.2. Οι προορισμοί (Places) της προσομοίωσης..... 40
3.4.3. Οι χρήστες της υπηρεσίας..... 41

4. ΥΛΟΠΟΙΗΣΗ ΠΡΟΣΟΜΟΙΩΣΗΣ ΜΕ ΤΟΝ ΠΡΟΣΟΜΟΙΩΤΗ SIAFU..... 45
4.1. Εισαγωγή..... 45

4.2. Προσομοίωση……..……..……46
4.2.1. Μοντέλο Χρήστη (Agent Model)…….……47
4.2.2. Μοντέλο Κόσμου (World Model)..... 48
4.2.3. Μοντέλο Πλαισίου (Context Model).....48
4.3. Κλάσεις του Siafu..... 50
4.3.1. Agent Class……..... 50
4.3.2. Place Class……..….... 52
4.3.3. AgentTypeCharacteristics Class……...… 55
4.3.4. PlaceTypeCharacteristics Class…….……...… 57
4.3.5. AgentModel Class……..…..58

5. ΑΠΟΤΕΛΕΣΜΑΤΑ ΠΡΟΣΟΜΟΙΩΣΗΣ – ΣΥΜΠΕΡΑΣΜΑΤΑ……..… 60
5.1. Γενικά.....60
5.2. Ανεξαρτησία ιδιωτικότητας προτιμήσεων-αποτελεσματικότητας θέσης και ανεξαρτησία ιδιωτικότητας θέσης-αποτελεσματικότητας προτιμήσεων……..…62
5.3. Ιδιωτικότητα-Αποτελεσματικότητα προτιμήσεων.....67
5.3.1. Αποτελέσματα προσομοίωσης για εκτενές δέντρο προτιμήσεων……..…67
5.3.2. Αποτελέσματα προσομοίωσης για απλούστερο δέντρο προτιμήσεων……68
5.4. Ιδιωτικότητα-Αποτελεσματικότηα θέσης 70
5.4.1. Κατανομή Gauss ανά περιοχές 70
5.4.2. Ομοιόμορφη κατανομή ανά περιοχές……...71
5.4.3. Ομοιόμορφη κατανομή παντού…….…… 72
5.5. Σύνοψη αποτελεσμάτων-Συμπεράσματα ……..… 73

ΠΑΡΑΡΤΗΜΑ : Κώδικας.....74

ΒΙΒΛΙΟΓΡΑΦΙΑ.....114

1.ΕΙΣΑΓΩΓΙΚΕΣ ΕΝΝΟΙΕΣ

1.1. Διάχυτος Υπολογισμός

Ο διάχυτος υπολογισμός (ubiquitous computing) είναι το μοντέλο επικοινωνίας ανθρώπου-υπολογιστή (human-computer interaction) στο οποίο η επεξεργασία των πληροφοριών έχει ενσωματωθεί πλήρως στην καθημερινή ζωή [1]. Σε αντίθεση με το μοντέλο των επιτραπέζιων υπολογιστών (desktop computers), όπου συνειδητά ο χρήστης χρησιμοποιεί μία και μόνο συσκευή για έναν συγκεκριμένο σκοπό, στο μοντέλο του ubiquitous computing ο χρήστης "χρησιμοποιεί" ταυτόχρονα πολλές υπολογιστικές συσκευές και συστήματα κατά την διάρκεια των καθημερινών του δραστηριοτήτων, χωρίς να είναι απαραίτητο να το συνειδητοποιεί πάντοτε. Η κύρια ιδέα και το όραμα πίσω από το ubiquitous computing είναι η ύπαρξη μικρών, φθηνών δικτυωμένων υπολογιστικών συσκευών κατανεμημένων σε όλα τα επίπεδα της καθημερινής ζωής. Σύμφωνα με την παραπάνω ιδέα ο μέσος καταναλωτής θα έχει λοιπόν στην κατοχή του εκατοντάδες διασυνδεόμενες υπολογιστικές συσκευές. Είναι εύκολο να γίνει αντιληπτό πως τα ήδη υπάρχοντα μοντέλα επικοινωνίας ανθρώπου-υπολογιστή είναι ανεπαρκή για τις ανάγκες διαχείρισης των πολυάριθμων αυτών συσκευών γι' αυτό αναπτύσσονται μέθοδοι για την αυτοματοποίηση των περισσότερων εργασιών και γενικότερα για τη βελτίωση της επικοινωνίας ανθρώπου-υπολογιστή (human-computer interaction). Βασικός στόχος των νέων αυτών μεθόδων είναι η μείωση στο ελάχιστο του νοητικού φόρτου που θα επωμίζονται οι χρήστες αφού πολλές εργασίες θα προβλέπονται και θα εκτελούνται αυτόματα. Για την επίτευξη των στόχων αυτών, οι υπολογιστές πρέπει να συλλέγουν και να εφαρμόζουν γνώση σχετικά με το εννοιολογικό πλαίσιο του χρήστη στον πραγματικό κόσμο.

1.2. Περιβάλλον (Context)

Η έννοια του context (πλαίσιο, περιβάλλον) είναι ιδιαίτερα σημαντική στην πληροφορική και στις επικοινωνίες. Παρόλα αυτά, είναι ένας σαφής, επακριβής και πλήρης ορισμός του όρου αυτού είναι εξαιρετικά δύσκολο να δωθεί. Σύμφωνα με τους Dey και Abowd [2], [3], ως context ορίζεται οποιαδήποτε πληροφορία που χρησιμοποιείται για να χαρακτηρίσει την κατάσταση μιας οντότητας. Ως οντότητα ορίζεται ένα πρόσωπο, μια τοποθεσία ή ένα αντικείμενο που θεωρείται σχετικό με την αλληλεπίδραση μεταξύ του χρήστη και μιας εφαρμογής. Αντίστοιχα οντότητα μπορεί να θεωρηθεί και ο ίδιος ο χρήστης ή η εφαρμογή. Οι Dey και Abowd [2], [3] προτείνουν τέσσερις κύριους τύπους context για να χαρακτηρίσουν μία συγκεκριμένη κατάσταση μία οντότητας:

- ***Ταυτότητα (Identity)*:** η ταυτότητα της οντότητας
- ***Τοποθεσία (Location)*:** η γεωγραφική τοποθεσία της
- ***Δραστηριότητα (Activity)*:** η δραστηριότητα που εκτελεί η οντότητα
- ***Χρόνος (Time)*:** η χρονική περίοδος στην οποία η οντότητα εκτελεί μια δραστηριότητα

Όλοι οι άλλοι τύποι πληροφορίας περιβάλλοντος ορίζονται ως δευτερεύοντες γιατί συχνά για να αποκτήσουν νόημα απαιτούν συσχέτιση με ένα ή περισσότερα μέρη του κύριου εννοιολογικού περιβάλλοντος.

Στη συνέχεια παρουσιάζεται μία ομαδοποίηση διαφορετικών τύπων context με ειδική αναφορά στις κινητές υπηρεσίες που στηρίζονται σε χάρτες:

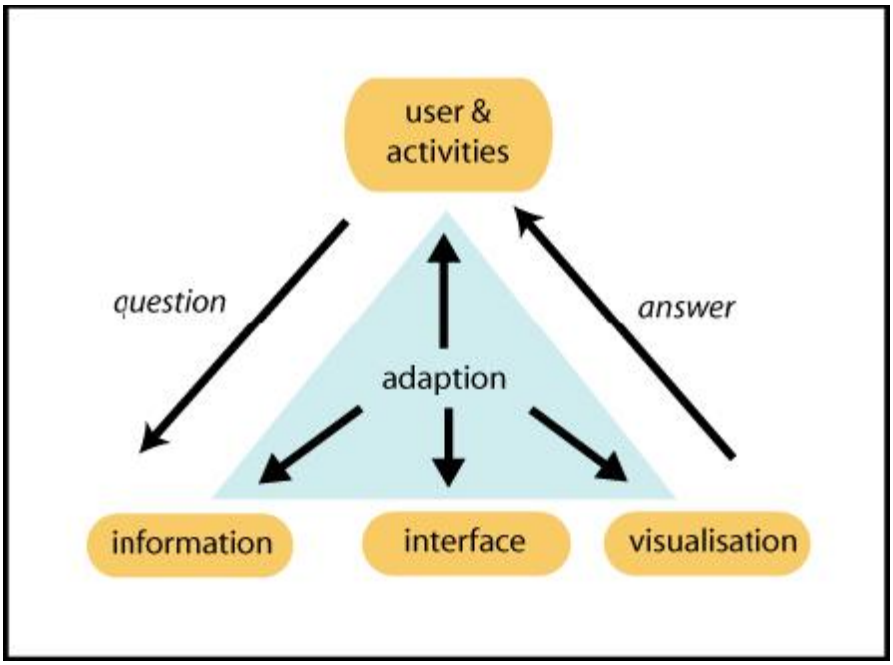
- ***Κινητός Χρήστης (Mobile map user)*:** Η ταυτότητα του χρήστη είναι σημαντική ώστε να επιτρέψει στην υπηρεσία να έχει πρόσβαση σε πληροφορίες σχετικές με θέματα όπως η ηλικία και το φύλο του χρήστη, οι προσωπικές του προτιμήσεις, οι φίλοι και οι συνάδελφοι του.
- ***Θέση (Location)*:** Η τοποθεσία είναι το στοιχείο του context που λαμβάνεται πιο συχνά υπόψη. Επιτρέπει σε πληροφορίες και υπηρεσίες να προσδιοριστούν ανάλογα με τη συγκεκριμένη θέση (localization).
- ***Χρόνος (Time)*:** Ο χρόνος μπορεί να αναφέρεται σε συγκεκριμένες στιγμές της ημέρας ή σε μεγαλύτερα χρονικά διαστήματα όπως πρωί, απόγευμα ή βράδυ, μέρα της εβδομάδας, του μήνα ή εποχή του χρόνου.
- ***Προσανατολισμός (Orientation)*:** Ο προσανατολισμός του χρήστη είναι σημαντικός ώστε να καθοριστεί η κατεύθυνση του χρήστη και έτσι τι βρίσκεται μπροστά του και τι πίσω.
- ***Ιστορικό πλοήγησης (Navigation history)*:** Το ιστορικό πλοήγησης επιτρέπει στους χρήστες να βλέπουν πού έχουν πάει, τι έχουν κάνει και τι έχουν δει. Είναι χρήσιμο στην πλοήγηση ώστε να κατευθύνει το χρήστη καθώς κινείται και να του επιτρέψει να επιστρέψει πίσω αν χαθεί.
- ***Σκοπός χρήσης (Purpose of use)*:** Ο σκοπός χρήσης ορίζεται από ενέργειες, στόχους, εργασίες και ρόλους των χρηστών. Διαφορετικοί τύποι χρήσεως απαιτούν διαφορετικούς τύπους πληροφορίας, τύπους παρουσίασης (π.χ. χάρτη, κείμενο ή ήχο), τύπους αλληλεπίδρασης.
- ***Κοινωνική κατάσταση (Social and cultural situation)*:** Η κοινωνική κατάσταση ενός χρήστη χαρακτηρίζεται από τη γειτνίαση με τους άλλους, τις κοινωνικές σχέσεις, τις συλλογικές εργασίες.
- ***Φυσικό περιβάλλον (Physical Surroundings)*:** Το φυσικό περιβάλλον περιλαμβάνει στοιχεία όπως το επίπεδο φωτισμού, ο περιβάλλων θόρυβος, κ.λπ.
- ***Ιδιότητες συστήματος (System Properties)*:** Αυτό σχετίζεται με το τι σύστημα χρησιμοποιεί ο χρήστης. Τι τύπο συσκευής και ποιες οι δυνατότητές της (π.χ. touch screen, έγχρωμη ή ασπρόμαυρη οθόνη κλπ.), αν έχουν πρόσβαση συνεχώς στο Internet ή είναι διακοπτόμενη, τι εύρος ζώνης (bandwidth) έχει η σύνδεση, τι ποιότητα έχουν οι πληροφορίες εντοπισμού (positioning information).

Τα συστήματα που αλλάζουν δυναμικά τη συμπεριφορά τους εξαιτίας του context χαρακτηρίζονται με διάφορες ονομασίες: reactive, responsive, situated, context-sensitive και environment directed [4], ωστόσο ο όρος adaptive (προσαρμοστικός) έχει επικρατήσει στην κινητή χαρτογραφία [5], [6]. Η προσαρμογή μπορεί να λάβει χώρα σε τέσσερα επίπεδα [5]:

1. ***Επίπεδο πληροφορίας (Information level)*** – Η προσαρμογή γίνεται στο

περιεχόμενο της πληροφορίας. Για παράδειγμα, φιλτράρισμα πληροφοριών ανάλογα με την ώρα της ημέρας.

2. ***Επίπεδο τεχνολογίας (Technology level)*** – Η πληροφορία κωδικοποιείται ανάλογα με τα τεχνικά χαρακτηριστικά της κάθε συσκευής.
3. ***Επίπεδο διεπαφής (User interface level)*** – Η προσαρμογή γίνεται στην διεπαφή. Για παράδειγμα, αυτόματη προσαρμογή του χάρτη καθώς ο χρήστης κινείται
4. ***Επίπεδο παρουσίασης (Presentation level)*** – Η προσαρμογή γίνεται στον τρόπο εμφάνισης της πληροφορίας. Για παράδειγμα, το αποτέλεσμα που βρίσκεται πιο κοντά στις επιλογές του χρήστη τονίζεται με πιο έντονα χρώματα ενώ τα υπόλοιπα με πιο αχνά.



Σχήμα 1.1: Επίπεδα προσαρμοστικότητας κατά Reichenbacher [6]

1.3. Επίγνωση περιβάλλοντος (Context awareness)

Η σημερινή κινητή υπολογιστική (mobile computing) και η διεισδυτική υπολογιστική (pervasive computing) μάς επιτρέπουν να κινηθούμε πέρα από αυτά τα δεσμευτικά πεδία των υπολογιστών προσανατολισμένων σε σενάρια χρήσης στο σπίτι και στο γραφείο [7]. Η ενσωμάτωση των υπολογιστών στην καθημερινή μας ζωή συχνά αναφέρεται ως pervasive computing. Η διαφορά του pervasive computing σε σχέση με το ubiquitous computing είναι ότι το pervasive computing έχει πιο στατική δομή ενώ το ubiquitous computing υποστηρίζει κινητούς hosts (mobile hosts) και κινητό κώδικα (mobile code) [8]. Η παρατήρηση των νέων αυτών τάσεων οδήγησε στις αρχές της δεκαετίας του 1990 τον Weiser [9] να κάνει ορισμένες προβλέψεις σχετικά με την επικοινωνία ανθρώπου-υπολογιστή τον 21ο αιώνα και να εφεύρει τον όρο ubiquitous computing [9]. Ο όρος αυτός περιγράφει έναν κόσμο όπου οι υπολογιστές ενσωματώνονται στη δομή της καθημερινής ζωής σε τέτοιο σημείο ώστε να μην μπορούν να διακριθούν από αυτή [7].

Σε αντίθεση με το πολύ περιορισμένο σύνολο επικοινωνιακών διαύλων των παραδοσιακών υπολογιστικών συστημάτων (σύνολο που αναφέρεται βέβαια σε συσκευές όπως η οθόνη, το ποντίκι ή το πληκτρολόγιο) [7], οι άνθρωποι χρησιμοποιούν τις πέντε αισθήσεις τους για την αλληλεπίδρασή τους με το περιβάλλον καθώς επίσης μοιράζονται μια κοινή πλούσια γλώσσα και έχουν μια γενικότερη αντίληψη για τον κόσμο που τους περιβάλλει. Η υπολογιστική με επίγνωση πλαισίου ή περιβάλλοντος (context-aware computing) [10] προσθέτει στους υπολογιστές αισθητήρες (sensors) και εκκινητήρες (actuators) ώστε να τους βοηθήσει στην καλύτερη κατανόηση του φυσικού περιβάλλοντος και την αλληλεπίδραση με αυτό. Τέτοια context-aware συστήματα συλλέγουν δεδομένα από τους αισθητήρες, τα οποία και χρησιμοποιούν για να δημιουργήσουν ένα μοντέλο του περιβάλλοντος, στη συνέχεια χρησιμοποιούν το μοντέλο αυτό για να παρέχουν πιο χρήσιμες και διαισθητικές (intuitive) υπηρεσίες στους χρήστες. Συνεπώς, ένα αποτελεσματικό context-aware σύστημα θα πρέπει να είναι πιο εύχρηστο από ένα παραδοσιακό πληροφοριακό σύστημα. Η συλλογή και η ερμηνεία των δεδομένων πλαισίου (contextual data) σχετικά με μια συγκεκριμένη υπηρεσία είναι απαραίτητό να είναι αυτοματοποιημένη. Χωρίς μια τέτοια αυτοματοποίηση ο χρήστης θα έπρεπε να παρέχει τις πληροφορίες συνεχώς, κάτι που θέλουμε να αποφύγουμε, καθώς ο συνδυασμός mobile και pervasive computing έχει συχνά ως αποτέλεσμα ο χρήστης να βρίσκεται σε ένα συνεχώς μεταβαλλόμενο πλαίσιο (user context) [7]. Είναι λοιπόν εμφανές πως το context-aware computing είναι ένα βήμα προς τον τελικό στόχο που είναι το ubiquitous computing.

Πολλές έρευνες στο χώρο του ubiquitous computing ασχολούνται με στην βελτίωση της επικοινωνίας ανθρώπου-υπολογιστή και συγκεκριμένα στον "προληπτικό υπολογισμό" (proactive computing) [11] ο οποίος στοχεύει στον περιορισμό της επικοινωνία ανθρώπου-υπολογιστή δίνοντας στον χρήστη έναν πιο εποπτικό ρόλο. Το proactive computing προσπαθεί να επεκτείνει το πεδίο της εφαρμογής (application domain) και να μειώσει την άμεση ανθρώπινη ανάμιξη συνδέοντας τους υπολογιστές άμεσα με τον φυσικό κόσμο με sensors και actuators. Τα δεδομένα υφίστανται επεξεργασία σε πραγματικό χρόνο, μειώνοντας την καθυστέρηση που προκύπτει από την αναμονή επιβεβαίωσης από το χρήστη. Τα συστήματα αυτά προσπαθούν να προβλέπουν τις ανάγκες του χρήστη χρησιμοποιώντας στατιστικά μοντέλα για την αντιμετώπιση της αβεβαιότητας.

1.4. Επίγνωση θέσης (Location awareness)

Η επίγνωση θέση (location awareness) είναι μία εκ των τεσσάρων κύριων ειδών context η οποία έγινε απαραίτητη πρόσφατα σε αντίθεση με την γνώση της ώρας (time) και της ταυτότητας (identity) για τις οποίες υπάρχουν καλά ορισμένες εφαρμογές και μέθοδοι στους υπολογιστές. Η χρήση λοιπόν υπολογιστών οι οποίοι διαθέτουν γνώση για το εννοιολογικό πλαίσιο του χρήστη σημαίνει πως πολύ περισσότερες προσωπικές πληροφορίες του χρήστη θα συλλέγονται, θα ταξινομούνται και θα διανέμονται από τους υπολογιστές αυτούς. Εδώ έγκειται και το ιδιαίτερο πρόβλημα της ισορροπίας μεταξύ των αναγκών του context-aware computing με την ανάγλη του χρήστη να διατηρεί τον έλεγχο της διανομής των προσωπικών του πληροφοριών.

Η πληροφορία αναφορικά με την τοποθεσία (location based information) είναι ένα από τα πιο σημαντικά δεδομένα του context στο ubiquitous computing και

χρησιμοποιείται από πλήθος εφαρμογών. Σημαντικό είναι το ζήτημα προστασίας ευαίσθητων προσωπικών δεδομένων από το γεγονός πως τα location systems κάνουν αυτόματη και συνεχή ανίχνευση της θέσεως του χρήστη. Αυτό δεν συνεπάγεται αυτομάτως πως ο χρήστης επιθυμεί απαραίτητα την αποτροπή κάθε πρόσβασης σε πληροφορίες που αφορούν τη θέση του καθώς κάποιες εφαρμογές μπορούν με τη χρήση αυτής της πληροφορίας να παρέχουν χρήσιμες πληροφορίες στον χρήστη. Ο χρήστης όμως επιθυμεί να έχει ο ίδιος τον έλεγχο των πληροφοριών αυτών. Με γνώμονα τα παραπάνω έχει υπάρξει αρκετή έρευνα στην προστασία της ιδιωτικότητας της θέσης (location privacy) για το ubiquitous computing και η οποία και έχει επικεντρωθεί στον ορισμό μηχανισμών που θα επιτρέπουν στους χρήστες να ελέγχουν οι ίδιοι πότε, ποιός και σε ποιο βαθμό θα έχει πρόσβαση στις πληροφορίες που αφορούν τη θέση τους.

1.5. Υπηρεσίες βασισμένες στη θέση (Location-Based Services)

Οι υπηρεσίες που βασίζονται στη θέση (Location-Based Services – LBS) είναι υπηρεσίες που χρησιμοποιούν και γίνονται διαθέσιμες μέσω ασύρματων μέσων μετάδοσης και παρέχουν πληροφορίες που σχετίζονται ή εξαρτώνται από την θέση του χρήστη. Οι χρήστες που χρησιμοποιούν κινητές συσκευές δεν χρειάζεται πλέον να παρέχουν οι ίδιοι πληροφορίες σχετικά με την θέση τους ώστε να χρησιμοποιήσουν location-based services καθώς η συλλογή αυτών των πληροφοριών γίνεται αυτόματα.

Οι LBS μπορούν να κατηγοριοποιηθούν στις παρακάτω τέσσερις μεγάλες κατηγορίες:

- ***Επιχείρηση προς Επιχείρηση (Business to Business)*** – Ως παράδειγμα υπηρεσιών αυτής της κατηγορίας μπορούμε να αναφέρουμε τις υπηρεσίες Fleet Management δηλαδή υπηρεσίες διαχείρισης εταιρικού στόλου οχημάτων (παρακολούθηση σε πραγματικό χρόνο, δρομολόγηση κλπ.).
- ***Επιχείρηση προς Πελάτη (Business to Consumer)*** – Ως παράδειγμα υπηρεσιών αυτής της κατηγορίας μπορούμε να αναφέρουμε τις υπηρεσίες προώθησης διαφημίσεων που βασίζονται στη θέση όπου ένας χρήστης λαμβάνει διαφημίσεις σχετικές με την θέση του.
- ***Πελάτης προς Επιχείρηση (Consumer to Business)*** – Ως παράδειγμα υπηρεσιών αυτής της κατηγορίας μπορούμε να αναφέρουμε τις υπηρεσίες Finder & Router (εύρεση κοντινότερου σημείου ενδιαφέροντος π.χ. εστιατορίου ή βενζινάδικου και πρόταση βέλτιστης διαδρομής προς αυτό).
- ***Πελάτης προς Πελάτη (Consumer to Consumer)*** – Ως παράδειγμα υπηρεσιών αυτής της κατηγορίας μπορούμε να αναφέρουμε τις υπηρεσίες εύρεσης φίλων όπου ο χρήστης ειδοποιείται αν κάποιος φίλος του βρίσκεται σε ακτίνα κάποιων μέτρων (με την προϋπόθεση ο φίλος του να έχει επιτρέψει την ανίχνευση του από το χρήστη).

Οι LBS είναι υπηρεσίες προσβάσιμες από κινητές συσκευές μέσω κινητού δικτύου (mobile network) και εκμεταλλεύονται την ικανότητα να χρησιμοποιούν τη θέση της κινητής συσκευής.

Τα γεωγραφικά συστήματα πληροφοριών (ή συστήματα γεωγραφικών πληροφοριών) (Geographic Information Systems – GIS) έχουν κάποια κοινά χαρακτηριστικά με τις LBS. Τέτοια κοινά χαρακτηριστικά είναι η διαχείριση δεδομένων με αναφορά σε θέση και μεθόδους ανάλυσης του χώρου (spatial analysis functions) που δίνουν απαντήσεις σε ερωτήματα όπως:

- Πού βρίσκομαι;
- Τι βρίσκεται κοντά;
- Πώς μπορώ να φτάσω;

Ωστόσο οι LBS και τα GIS έχουν διαφορετική προέλευση και απευθύνονται σε διαφορετικές ομάδες χρηστών [12]. Τα GIS αναπτύχθηκαν κατά την διάρκεια προηγούμενων δεκαετιών πάνω σε επαγγελματικές γεωγραφικές εφαρμογές. Αντίθετα, οι LBS αναπτύχθηκαν αρκετά πρόσφατα μαζί με την εξέλιξη των κινητών υπηρεσιών. Σε ό,τι αφορά τους χρήστες στους οποίους απευθύνονται, τα GIS μπορούν να θεωρηθούν πιο επαγγελματικά συστήματα που απευθύνονται σε έμπειρους χρήστες και γι' αυτό προσφέρουν μεγάλο εύρος λειτουργιών. Αντίθετα οι LBS αναπτύχθηκαν σαν συγκεκριμένες υπηρεσίες για μη εξειδικευμένους χρήστες. Επίσης, οι LBS έχουν να αντιμετωπίσουν τους περιορισμούς των κινητών υπολογιστικών συσκευών, όπως είναι η χαμηλή υπολογιστική ισχύ, οι μικρές οθόνες και η μικρή σε διάρκεια ζωής μπαταρία των κινητών συσκευών [6].

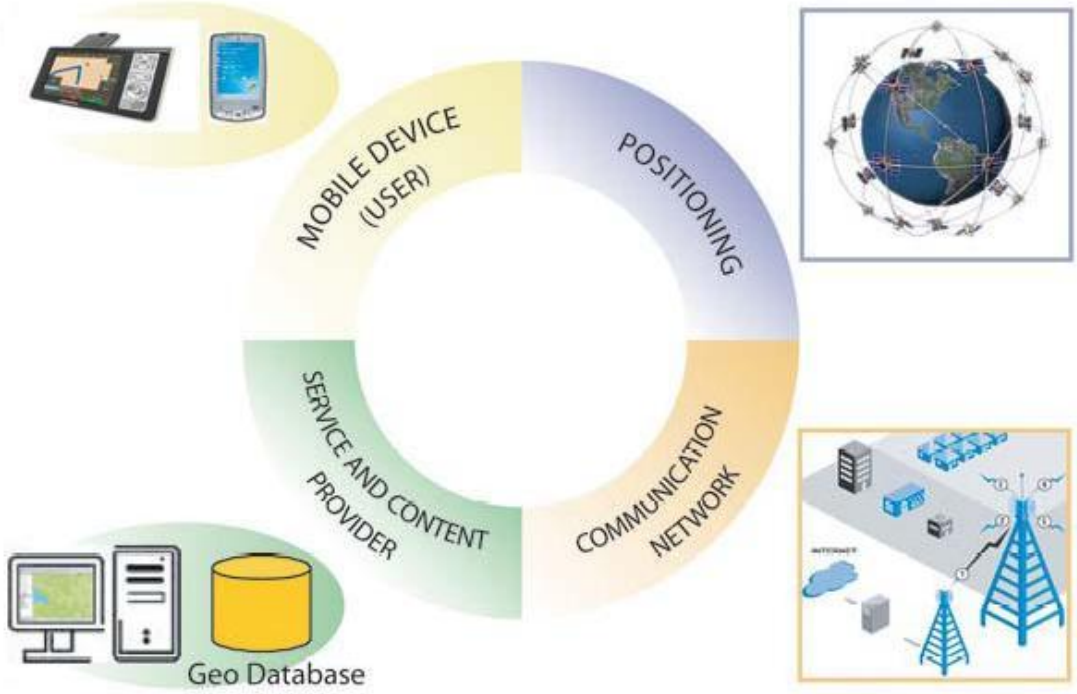
1.5.1. Αρχιτεκτονική LBS

Τα βασικά μέρη μιας LBS είναι τα ακόλουθα:

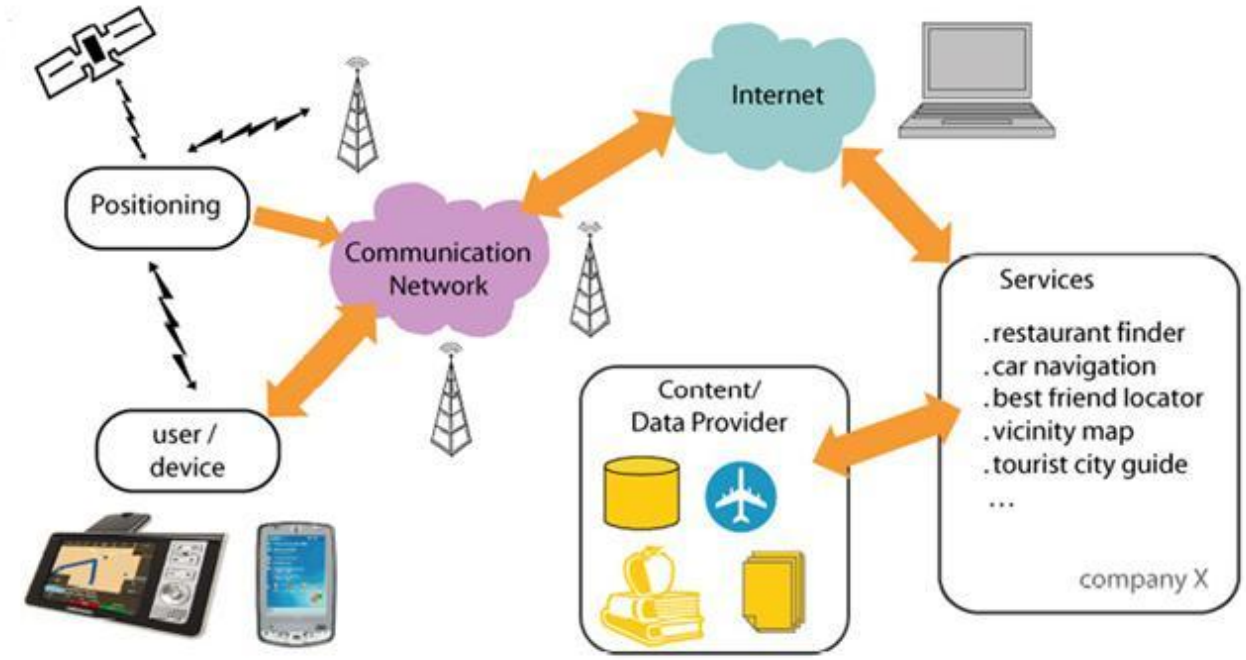
- ***Κινητές συσκευές (Mobile Devices)*:** Οι συσκευές αυτές είναι ένα εργαλείο για το χρήστη ώστε να ζητάει τις πληροφορίες που επιθυμεί. Τα αποτελέσματα μπορούν να δοθούν είτε ηχητικά, ή με κείμενο ή εικόνες. Τέτοιες συσκευές είναι οι φορητοί υπολογιστές, τα PDAs, τα smartphones, τα κινητά τηλέφωνα ή ακόμα και μία μονάδα πλοήγησης ενός αυτοκινήτου.
- ***Δίκτυο επικοινωνίας (Communication Network)*:** Το δίκτυο είναι αυτό που μεταφέρει τα δεδομένα του χρήστη και τα αιτήματα υπηρεσίας προς τον πάροχο της υπηρεσίας (service provider) και εν συνεχεία επιστρέφει τις ζητούμενες πληροφορίες πίσω στον χρήστη.
- ***Τμήμα Προσδιορισμού Θέσης (Positioning Component)*:** Η θέση του χρήστη μπορεί να υπολογιστεί είτε κάνοντας χρήση του κινητού δικτύου επικοινωνίας είτε χρησιμοποιώντας το Παγκόσμιο Σύστημα Προσδιορισμού Θέσης (Global Positioning System – GPS). Για την ανίχνευση της θέσης του χρήστη μέσα σε εσωτερικού χώρους υπάρχουν και άλλες μέθοδοι, ενώ σε περιπτώσεις που η θέση δεν προσδιοριστεί αυτόματα μπορεί να δοθεί από τον ίδιο τον χρήστη.
- ***Πάροχος υπηρεσίας (Service Provider)*:** Ο πάροχος της υπηρεσίας είναι υπεύθυνος για την λήψη και επεξεργασία του αιτήματος υπηρεσίας και προσφέρει έναν αριθμό διαφορετικών υπηρεσιών στο χρήστη, όπως για

παράδειγμα, τον υπολογισμό της θέσης, την εύρεση μιας διαδρομής, την αναζήτηση συγκεκριμένων πληροφοριών για αντικείμενα που ενδιαφέρουν το χρήστη κ.λπ.

- ***Πάροχος δεδομένων και περιεχομένου (Data and Content Provider)*:** Οι πάροχοι υπηρεσιών δεν μπορούν απαραίτητα να παρέχουν όλες τις πληροφορίες που μπορούν να ζητηθούν από τους χρήστες, συνεπώς μπορεί και αυτοί να ζητήσουν τις πληροφορίες αυτές από έναν τρίτο πάροχο (όπως για παράδειγμα εταιρείες παροχής χαρτών, εταιρείες πληροφόρησης για την κίνηση στους δρόμους, κλπ.) [6].



Σχήμα 1.2: Τα βασικά μέρη μιας LBS: Χρήστης, Δίκτυο Επικοινωνίας, Πάροχος Υπηρεσίας Εντοπισμού Θέσης, Πάροχος Περιεχομένου [6]



Σχήμα 1.3: Βασικά τμήματα μιας LBS και ροή πληροφορίας [6]

Οι location based υπηρεσίες μπορούν γενικά να διαχωριστούν σε δύο είδη ανάλογα με το αν η πληροφορία μεταφέρεται μετά από αλληλεπίδραση με το χρήστη ή όχι. Έτσι έχουμε:

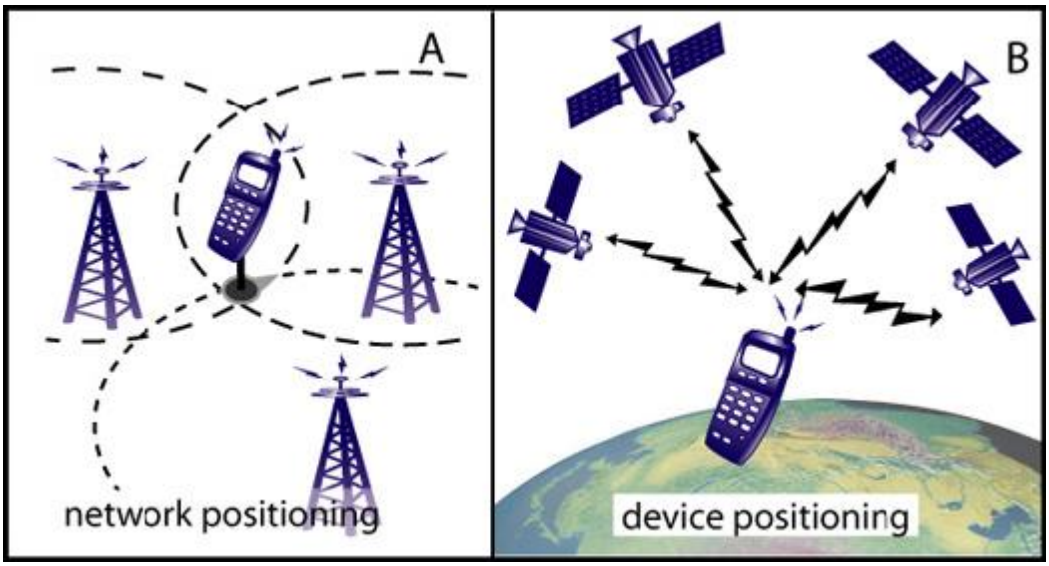
- Push Services: Πρόκειται για υπηρεσίες που μεταφέρουν την πληροφορία που ζητείται άμεσα από τον χρήστη.
- Pull Services: Πρόκειται για υπηρείες που μεταφέρουν πληροφορία, χωρίς να ζητηθεί ή με το να ζητηθεί έμμεσα από το χρήστη. Τέτοιες υπηρεσίες ενεργοποιούνται είτε από ένα γεγονός όπως για παράδειγμα αν ο χρήστης βρεθεί σε μία περιοχή ή όταν ένα συγκεκριμένο χρονικό διάστημα έχει παρέλθει [6].

Η βασική ιδέα πίσω από τις LBS είναι να απαντήσει σε ερωτήσεις όπως: «Πού βρίσκομαι;», «Πού είναι οι φίλοι μου;», «Τι βρίσκεται κοντά μου;». Όταν οι χρήστες βρεθούν σε ένα περιβάλλον ακόμη κι αυτό είναι σχετικά άγνωστο, η συμπεριφορά τους και οι ανάγκες τους είναι αρκετά προβλέψιμες είτε βρίσκονται στην χώρα τους είτε στο εξωτερικό, είτε περπατούν, είτε οδηγούν ένα όχημα. Οι χρήστες έχουν ανάγκη να βρουν κάτι να φάνε, ίσως να βρουν ένα φαρμακείο, ένα ATM ανάληψης χρημάτων, μία αφετηρία ταξί, κ.λπ. Όταν κάποιος βρεθεί στο εξωτερικό υπάρχουν επιπλέον ανάγκες όπως το να βρει κανείς τοπικά τουριστικά αξιοθέατα, να βρει ένα ξενοδοχείο ή ένα ανταλλακτήριο. Καθώς κάποιος οδηγεί, ενδέχεται να υπάρχουν άλλες ανάγκες όπως για παράδειγμα βοήθεια στην εύρεση μιας διαδρομής σε μία άγνωστη πόλη, λεπτομέρειες για οδική βοήθεια ή πληροφορίες για την κίνηση.

1.5.2. Μέθοδοι εντοπισμού θέσης και ακρίβεια

Οι μέθοδοι εντοπισμού θέσης μπορούν να ομαδοποιηθούν σε 3 ομάδες ως εξής:

Η πρώτη ομάδα ονομάζεται network-based positioning. Η θέση ενός χρήστη παρακολουθείται και επαληθεύεται με τη χρήση ενός δικτύου σταθμών βάσης (base station network). Εδώ η κινητή συσκευή στέλνει σήματα προς το δίκτυο ή δέχεται από αυτό. Η δεύτερη ομάδα ονομάζεται terminal-based positioning. Εδώ ο υπολογισμός της θέσης γίνεται από την ίδια τη συσκευή του χρήστη μέσω σημάτων που δέχεται από τους σταθμούς βάσης. Το Global Positioning System (GPS) είναι και το πιο γνωστό παράδειγμα ενός terminal-based positioning systems. Οι δορυφόροι GPS είναι οι σταθμοί βάσης για το σύστημα GPS. Η τρίτη ομάδα τεχνικών εντοπισμού που μπορούμε να διακρίνουμε είναι αυτή η οποία προκύπτει από το συνδυασμό network-based positioning και terminal-based positioning συστημάτων.



Σχήμα 1.4: Network-based positioning device-based positioning μέθοδοι εντοπισμού[6]

Οι βασικές αρχές για τον υπολογισμό της θέσης του χρήστη είναι οι εξής:

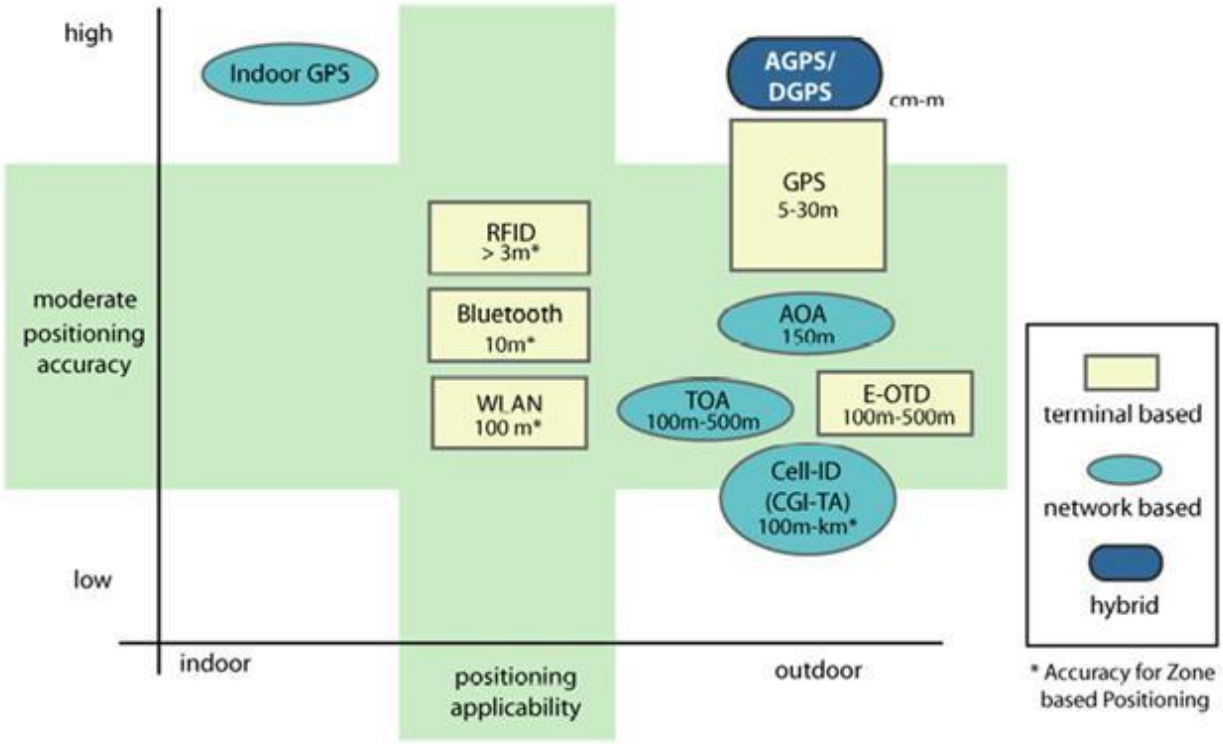
- Η θέση των σταθμών βάσης είναι γνωστή
- Οι πληροφορίες από ένα σήμα μετασχηματίζονται σε αποστάσεις
- Ο υπολογισμός της θέσης του χρήστη γίνεται με τη χρήση των αποστάσεων που υπολογίσθηκαν στους σταθμούς βάσεων.

Οι βασικές και πιο συχνές τεχνικές που χρησιμοποιούνται για τον εντοπισμό θέσης είναι οι εξής:

- **Cell of origin (COO), location signature, location beacons:** Το

αναγνωριστικό (id) της κυψέλης είναι συνήθως το αναγνωριστικό του πλησιέστερου σταθμού βάσης, π.χ. μια κεραία κινητής τηλεφωνίας. Με αυτή τη τεχνική, ως θέση του χρήστη νοείται ένας ορισμένος κύκλος ή κυψέλη γύρω από τον σταθμό βάσεως. Οι σταθμοί μετάδοσης (beacons) (π.χ. με υπέρυθρες ή RFID) χρησιμοποιούνται κυρίως σε εσωτερικούς χώρους. Τα beacons έχουν ένα αναγνωριστικό ή μεταδίδουν την ακριβή τους τοποθεσία στη κινητή συσκευή που πλησιάζει.

- **Time of Arrival (TOA):** Καθώς τα ηλεκτρομαγνητικά σήματα διαδίδονται με την ταχύτητα του φωτός, γνωρίζοντας την ταχύτητα και την διαφορά χρόνου μεταξύ αποστολής και λήψης, μπορεί να υπολογιστεί η απόσταση. Η ταχύτητα του φωτός είναι περίπου 300.000km/s και έτσι ο χρόνος εκτέλεσης είναι πολύ μικρός και για αυτό απαιτούνται χρονόμετρα ακριβείας. Η ίδια αρχή μπορεί να χρησιμοποιηθεί για πιο αργά σήματα όπως τα υπερηχητικά.
- **Time Difference of Arrival (TDOA), Enhanced Observed Time Difference (E-OTD):** Και αυτές οι τεχνικές υπολογίζουν την απόσταση μετρώντας το χρόνο εκτέλεσης, αλλά χρησιμοποιούν τη διαφορά χρόνου μεταξύ των σημάτων από τρεις (συνήθως) διαφορετικούς σταθμούς βάσεως. Έτσι, έχοντας σήματα από διαφορετικούς γειτονικούς σταθμούς βάσεως, η θέση μπορεί να υπολογιστεί με χρήση triangulation. Στην περίπτωση του TDOA, ο υπολογισμός της θέσης γίνεται από τον πάροχο του δικτύου, ενώ στην περίπτωση του E-ODT γίνεται από την κινητή συσκευή.
- **Angle of Arrival (AOA), Direction of Arrival (DOA):** Χρησιμοποιώντας κεραίες με συγκεκριμένα χαρακτηριστικά, μπορεί να ανιχνευθεί η γωνία άφιξης (angle of arrival) στην κινητή συσκευή. Επειδή για μία κινητή συσκευή αυτό δεν είναι ικανοποιητικά ακριβές, μια εναλλακτική είναι πολλοί σταθμοί βάσης που έχουν περισσότερες της μίας κεραίες (συνήθως 2-4) που διαιρούν τον περιγεγραμμένο κύκλο των σταθμών βάσεων σε τμήματα των 90, 120 ή 180 μοιρών.



Σχήμα 1.5: Μέθοδοι εντοπισμού θέσης, ακρίβεια και εφαρμογή (AGPS: Assisted GPS, AOA: Angle of Arrival, TOA: Time of Arrival, E-OTD: Enhanced Observed Time Difference) [6]

Οι δύο πιο διαδεδομένες τεχνικές εντοπισμού είναι το GPS και η επαλήθευση θέσης κάνοντας χρήση Cell-ID από τον πλησιέστερο πομποδέκτη σταθμού βάσης. Με την πρώτη από τις δύο να έχει καλύτερη ακρίβεια και να είναι εύκολα προσβάσιμη από όλους.

2.Ιδιωτικότητα προσωπικών δεδομένων και αποτελεσματικότητα χρήσης εξατομικευμένων υπηρεσιών

2.1 Γενικά

Η έννοια της εξατομικευμένης – προσωποποιημένης υπηρεσίας συνεπάγεται τη χρήση, από την εκάστοτε υπηρεσία, προσωπικών πληροφοριών. Οι πληροφορίες αυτές σχετίζονται με την ταυτότητα και την κατάσταση του χρήστη. Χωρίς την κοινοποίηση, εκ μέρους του χρήστη, τέτοιων πληροφοριών είναι αδύνατη η παροχή οποιουδήποτε είδους προσωποποιημένης υπηρεσίας. Από την άλλη πλευρά εγείρονται σοβαρά θέματα και προβληματισμοί σχετικά με την ασφάλεια της ιδιωτικότητας των χρηστών, δεδομένου ότι οι πληροφορίες που παρέχονται στην υπηρεσία μπορούν να χρησιμοποιηθούν με ποικίλους τρόπους, θεμιτούς και αθέμιτους.
Παραδείγματα παραβίασης της ιδιωτικότητας υπάρχουν πολλά. Απαντώνται κυρίως σε χώρες όπου η τεχνολογία είναι ένα βήμα μπροστά και τέτοιου είδους υπηρεσίες παρέχονται οργανωμένα και καθημερινά. Έτσι συναντάμε περιπτώσεις παρακολούθησης χρηστών, εκμετάλλευσης των προσωπικών τους δεδομένων για εμπορικούς σκοπούς και καταγραφής της ταυτότητας και των προτιμήσεών τους.
Ένας κόσμος όπως αυτός που περιγράφεται στο βιβλίο του Όργουελ «1984» [13] ίσως δεν θα απέχει και πολύ από την πραγματικότητα που θα βιώνουμε όλοι μας σε μερικά χρόνια. Για τον λόγο αυτό απαιτείται προσεκτικός σχεδιασμός τέτοιου είδους υπηρεσιών και τροπή προς την κατεύθυνση της προστασίας της ιδιωτικότητας των χρηστών.

2.2 Είδη πληροφοριών που αποστέλλονται σε μία προσωποποιημένη υπηρεσία

Οι πληροφορίες που θα πρέπει να στείλει ο χρήστης σε μια υπηρεσία για να μπορέσει εκείνη να του παρέχει προσωποποιημένη αντιμετώπιση θα πρέπει να είναι τέτοιες που να τον διαχωρίζουν από τους υπόλοιπους χρήστες της υπηρεσίας ή να τον κατατάσσουν σε κάποια ομάδα χρηστών. Θα μπορούσαμε να πούμε ότι για να λειτουργήσει μια τέτοια υπηρεσία θα πρέπει να γνωρίζει κυρίως πληροφορίες σχετικές με:

i. **Ταυτότητα χρήστη (user identity):** Σ’ αυτήν την κατηγορία περιλαμβάνονται στοιχεία όπως το φύλο, η ηλικία, το επάγγελμα, οι προτιμήσεις σχετικά με κάποιον τομέα της ζωής (φαγητό, διασκέδαση, διακοπές), τόπος διαμονής.
ii. **Τοποθεσία (location):** Η γεωγραφική θέση στην οποία βρίσκεται ο χρήστης όταν στέλνει το αίτημα του και τις πληροφορίες που τον αφορούν προς την υπηρεσία. Σε συγκεκριμένες περιπτώσεις μπορεί η γεωγραφική θέση που αποστέλλεται να είναι η περιοχή που σκοπεύει ο χρήστης να επισκεφτεί ή να ζητήσει πληροφορίες σχετικά με αυτήν.
iii. **Χρόνος (time):** Αναφέρεται συνήθως στην συγκεκριμένη χρονική στιγμή κατά την οποία γίνεται το ερώτημα στην υπηρεσία. Η έννοια του χρόνου αναφέρεται είτε σε χρονική στιγμή στη διάρκεια της μέρας, είτε σε κάποιο

ευρύτερο χρονικό διάστημα της μέρας (πρωί, μεσημέρι, βράδυ), είτε σε ακόμη ευρύτερα χρονικά διαστήματα (ολόκληρη μέρα, μήνας, έτος). Σε συγκεκριμένες περιπτώσεις ενδέχεται να αποστέλλεται ένα μελλοντικό χρονικό σημείο στο οποίο ο χρήστης σκοπεύει να εξασκήσει κάποια δραστηριότητα.

Σε ένα υποθετικό παράδειγμα, όπου ο χρήστης επικοινωνεί με μια υπηρεσία που έχει τη δυνατότητα να προτείνει κάποιο χώρο διασκέδασης, ο χρήστης θα έπρεπε να στείλει τα εξής ενδεικτικά στοιχεία για να μπορέσει η υπηρεσία να βρει το βέλτιστο προορισμό για το συγκεκριμένο χρήστη:

i. **Στοιχεία ταυτότητας / προτιμήσεων:**
   - Φύλο
   - Ηλικία
   - Προτιμήσεις σχετικά με διασκέδαση (διασκεδάζει σε μπαρ ή καφετέριες)
   - Επίπεδο τιμών που επιθυμεί να πληρώσει
   - Να επιτρέπεται το κάπνισμα στο συγκεκριμένο χώρο

ii. **Τοποθεσία**
   - Η τοποθεσία που βρίσκεται ο χρήστης ή
   - η τοποθεσία γύρω από την οποία επιθυμεί να ευρεθούν χώροι διασκέδασης

iii. **Χρόνος**
   - Η χρονική στιγμή ή η χρονική περίοδος κατά την οποία γίνεται το ερώτημα προς την υπηρεσία ή
   - η χρονική στιγμή στην οποία επιθυμεί ο χρήστης να διασκεδάσει (π.χ. είναι πρωί και ζητούνται προορισμοί για το βράδυ)

Από τα παραπάνω στοιχεία που παραλαμβάνει η υπηρεσία, η ταυτότητα, οι προτιμήσεις του χρήστη και η τοποθεσία κατά πρώτον λόγο, αλλά και ο χρόνος κατά δεύτερο μπορούν να θεωρηθούν ως προσωπικά δεδομένα. Επομένως θα ήταν καλό για την ασφάλεια της ιδιωτικότητας του, ο χρήστης να μην παρέχει αφειδώς τα προσωπικά του δεδομένα σε τρίτους που δεν γνωρίζει το πώς θα τα χρησιμοποιήσουν.

2.3 Μοντέλα για την προστασία της ιδιωτικότητας

Στη συνέχεια περιγράφονται κάποια μοντέλα, τα οποία έχουν προταθεί για την προστασία της ιδιωτικότητας προσωπικών δεδομένων:

2.3.1 Τεχνικές θολώματος (blurring)

Διάφορες τεχνικές «θολώματος» (blurring) ή εισαγωγής «θορύβου» έχουν προταθεί για την προστασία της ιδιωτικότητας των χρηστών. Οι περισσότερες εξ' αυτών επικεντρώνονται στην προστασία της ιδιωτικότητας γεωγραφικής θέσης (location

privacy)[14], αλλά μπορούν να χρησιμοποιηθούν με κάποιες αλλαγές και για τη βελτίωση της ιδιωτικότητας ταυτότητας/προτιμήσεων (identity-preferences privacy).

Στο [15] προτείνεται από τους Kido, Yanagisawa και Satoh μια ιδιαίτερη υλοποίηση μιας τέτοιας τεχνικής, σύμφωνα με την οποία το κινητό τερματικό του χρήστη μεταδίδει στον πάροχο της υπηρεσίας εκτός από την πραγματική θέση του χρήστη και άλλες ψευδείς ('dummy') θέσεις. Ο πάροχος με την σειρά του πρέπει να επιστρέψει μια απάντηση για κάθε αίτημα που έχει αποσταλεί, χωρίς να γνωρίζει ποιο από τα αιτήματα περιέχει την πραγματική θέση του χρήστη. Τέλος, στο κινητό τερματικό φιλτράρονται οι απαντήσεις και κρατείται μόνο εκείνη που αντιστοιχεί στην πραγματική θέση.

Παρόλο που τέτοιου είδους τεχνικές φαίνεται να αποδίδουν καλά σε θέματα ασφάλειας και ιδιωτικότητας, εντούτοις υποφέρουν σε θέματα επίδοσης. Το τελευταίο είναι απόρροια του γεγονότος ότι οι υπηρεσίες προσφέρονται μέσω ασύρματων συνδέσεων και κινητών συσκευών, που επικοινωνούν με εξυπηρετητές που πρέπει να προσπελάσουν τεράστιες βάσεις δεδομένων. Το εύρος ζώνης τέτοιου είδους συνδέσεων είναι ακόμη αισθητά μικρότερο από αντίστοιχες σταθερής πρόσβασης και ως εκ τούτου δεν υπάρχει η πολυτέλεια για μετάδοση και αναμετάδοση άχρηστων ουσιαστικά πληροφοριών χωρίς την εισαγωγή μεγάλων καθυστερήσεων. Ακόμη η απαίτηση για μεγάλο αριθμό επιπρόσθετων δεδομένων αυξάνει σημαντικά το κόστος, ενώ εισάγει ακόμη μεγαλύτερες καθυστερήσεις στις προσπελάσεις στις βάσεις των εξυπηρετητών του παρόχου της υπηρεσίας.

Ένα ακόμα πρόβλημα των εν λόγω τεχνικών είναι ότι οι αλγόριθμοι υλοποίησής τους θα πρέπει να είναι τέτοιοι ώστε να μην επιτρέπουν την αποκάλυψη της θέσης του χρήστη, με χρήση μεθόδων στατιστικής ανάλυσης. Για παράδειγμα, έστω ότι ο χρήστης Α βρίσκεται στην περιοχή Β και χρειάζεται να χρησιμοποιήσει μια υπηρεσία. Τότε σύμφωνα με το μοντέλο που αναλύσαμε θα σταλεί στον πάροχο ένα πλήθος αιτημάτων με το πεδίο της περιοχής αλλαγμένο σε καθένα από αυτά. Έτσι σε κάποιο θα εμφανίζεται η πραγματική περιοχή Β, ενώ στα υπόλοιπα θα έχουμε ψευδή στοιχεία, έστω τις περιοχές Γ και Δ. Αν ο ίδιος χρήστης θελήσει σε μικρό χρονικό διάστημα από το πρώτο του ερώτημα να χρησιμοποιήσει την ίδια υπηρεσία από την ίδια περιοχή, τότε τα αιτήματα του θα περιλαμβάνουν τις περιοχές Β και άλλες δύο τις οποίες θα πρέπει να επιλέξει ο αλγόριθμος. Στην περίπτωση αυτή όμως εξετάζοντας την τομή των περιοχών που δηλώθηκαν στο πρώτο ερώτημα με τις αντίστοιχες του δεύτερου περιορίζουμε τις πιθανές περιοχές της ακριβής θέσης του χρήστη ή ακόμα και την προσδιορίζουμε πλήρως.

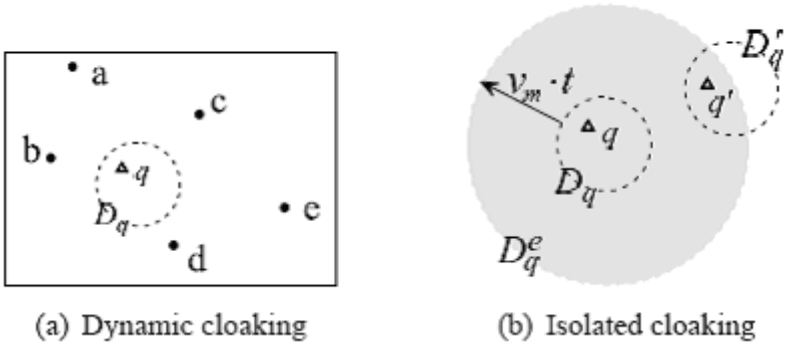
2.3.2 Μοντέλο κ-ανωνυμίας (k-anonymity)

Μια διαφορετική προσέγγιση στο πρόβλημα της προστασίας της ιδιωτικότητας γεωγραφικής θέσης είναι η απόκρυψη της ακριβούς θέσης του χρήστη μπλέκοντάς την και αναμιγνύοντάς την με τις αντίστοιχες πληροφορίες θέσης μιας ευρύτερης ομάδας χρηστών. Η ιδέα είναι να «χαθεί» η ακριβής θέση του χρήστη μέσα στην πληθώρα των πληροφοριών θέσης, όπως ακριβώς περνά απαρατήρητος ένας άνθρωπος μέσα σε ένα μεγάλο πλήθος. Η παραπάνω κατάσταση είναι μια μορφή ανωνυμίας, ένας ορισμός της οποίας δίνεται από τους Pfitzmann και Koehntopp στο [16], όπου η ανωνυμία προσδιορίζεται ως εκείνη η κατάσταση στην οποία το

υποκείμενο δεν μπορεί να ταυτοποιηθεί σε ένα σύνολο υποκειμένων, το σύνολο ανωνυμίας, δηλαδή όταν ένας τρίτος δεν μπορεί να ξεχωρίσει μεταξύ τους τα υποκείμενα που ανήκουν στο σύνολο ανωνυμίας (anonymity set) . Ο ορισμός της κ-ανωνυμίας προκύπτει φυσικά από τον γενικότερο ορισμό της ανωνυμίας. Έτσι θα ονομάζουμε κ-ανωνυμία την επίτευξη ανωνυμίας, όπου ο πληθάριθμος του συνόλου ανωνυμίας είναι κ.

Στο [17], ο Mokbel, Chow και Aref ακολουθούν το μοντέλο k-ανωνυμίας, σε μια απόπειρα να δημιουργήσουν ένα πλαίσιο, το οποίο θα ανταποκρίνεται στις ανάγκες για ιδιωτικότητα (privacy) και ποιότητα (quality) σε ερωτήματα προς βάσεις δεδομένων σχετικά με location based υπηρεσίες. Το μοντέλο τους βασίζεται στη χρήση ενός location anonymizer («ανωνυμοποιητή θέσης») και ενός privacy-aware query processor («επεξεργαστή ερωτημάτων με αντίληψη ιδιωτικότητας») για την απόκρυψη πληροφοριών σχετικών με τη θέση του χρήστη. Το μοντέλο αυτό είναι γνωστό ως Casper και μια σύντομη περιγραφή του γίνεται στη συνέχεια.

Το προτεινόμενο μοντέλο αποτελείται από δύο κύρια συστατικά, τον *location anonymizer* και τον *privacy-aware query processor*. Ο *location anonymizer* εντάσσει το χρήστη σε μια ευρύτερη περιοχή (cloaked spatial region) καθιστώντας ασαφή την ακριβή θέση του χρήστη ανάλογα με τις απαιτήσεις ασφάλειας(εξασφάλιση privacy) του χρήστη. Ο privacy-aware query processor είναι ενσωματωμένος μέσα στον κεντρικό εξυπηρετητή της υηρεσίας (location-based database server) ώστε να παρέχει πληροφορίες σχετικά με τις ευρύτερες περιοχές ένταξης χρηστών (spatial cloaked regions) αντί με την ακριβή τους θέση.
Οι χρήστες προσδιορίζουν το ικανοποιητικό γι' αυτούς επίπεδο ασφάλειας μέσω ενός προσδιορισμένου από το χρήστη privacy profile. Ένα user privacy profile περιλαμβάνει δύο παραμέτρους: k και $A_{min.}$, όπου το k υποδηλώνει ότι ο κινητός χρήστης θέλει να είναι *k-ανώνυμος*, δηλαδή η ταυτότητά του να εντάσσεται σε ένα πλήθος k χρηστών, ώστε ο προσδιορισμός της ταυτότητάς του να γίνεται με πιθανότητα *1/k* , ενώ από την άλλη η παράμετρος A_{min} προσδιορίζει ότι ο χρήστης θα τοποθετεί την τοποθεσία του σε μια ευρύτερη περιοχή επιφάνειας τουλάχιστον A_{min}. Όσο μεγαλύτερες οι τιμές των k και A_{min}, τόσο αυστηρότερες οι απαιτήσεις ιδιωτικότητας.



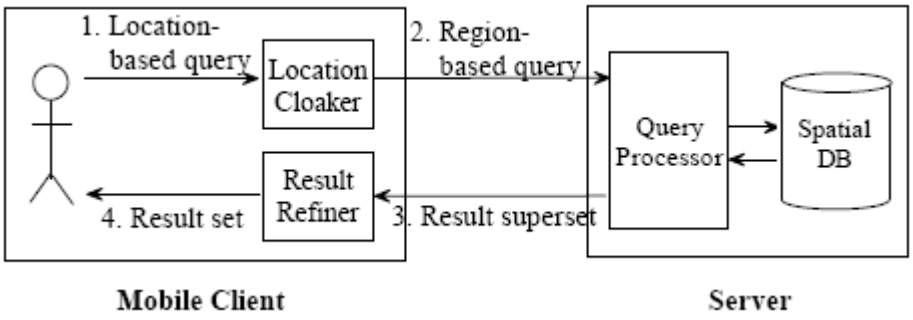
Εικόνα 2.1: Δυναμικός Υπολογισμός της απόκρυψης θέσης(Dynamic location cloaking) [18]

Στην εικόνα 2.1 (a) [18]βλέπουμε τον υπολογισμό της περιοχής απόκρυψης θέσης (spatial cloak region) στην περίπτωση που προσδιορίζεται από την παράμετρο k όπου

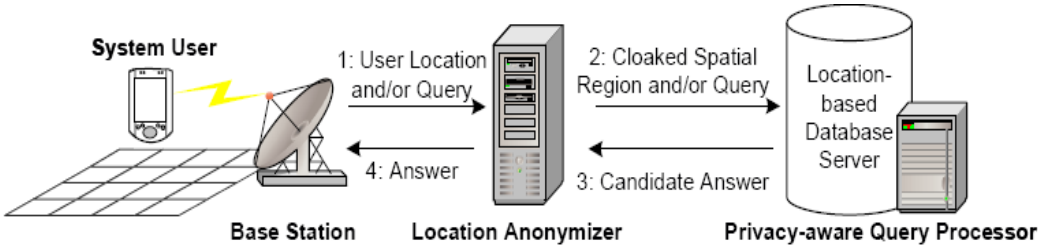
στην υπολογισμένη περιοχή υπάρχουν οι χρήστες a,b,c,d,e . Αυτή θα είναι η περιοχή στην οποία θα ενταχθεί ο χρήστης και θα σταλεί ως παράμετρος του request. Μέσα σε αυτή την περιοχή, η πιθανότητα επιτυχής ταυτοποίησης του χρήστη που έκανε το ερώτημα είναι *1/5*. Στην εικόνα (b) βλέπουμε την υπολογισμένη περιοχή που προκύπτει από την παράμετρο A_{min}.
Ο υπολογισμός της spatial cloak με τον πρώτο τρόπο καλείται *dynamic spatial cloaking* επειδή εξαρτάται κάθε φορά από την κατανομή των χρηστών και ως εκ τούτου το μέγεθος της περιοχής που θα καλύπτει *k-1* χρήστες γύρω από τον χρήστη που κάνει το ερώτημα (άρα σύνολο *k* μαζί με το χρήστη του ερωτήματος), θα είναι μεταβλητό. Αντίθετα στην περίπτωση (b) η επιφάνεια της περιοχής ορίζεται από το χρήστη,είναι σταθερή και καλείται *static spatial cloak.*
Εφόσον αποκρύπτεται η πραγματική ταυτότητα του χρήστη, απαιτείται ένας location anonymizer ο οποίος αντιστοιχεί σε ένα έμπιστο τρίτο μέλος(trusted third party) λειτουργώντας σαν ένα middle layer μεταξύ του mobile user και του location-based database server με σκοπό:

1) να λαμβάνει την ακριβή τοποθεσία από τον mobile user μαζί με το privacy profile του κάθε χρήστη
2) με βάση το Privacy profile του χρήστη να καθιστά ασαφή την ακριβή του θέση εντάσσοντάς τον σε μια ευρύτερη περιοχή, το μέγεθος της οποίας αναφέρεται στις απαιτήσεις ιδιωτικότητας του χρήστη
3) να αλλάζει το request του χρήστη ενημερώνοντας το πεδίο που ορίζει το location με τη νέα ευρύτερη περιοχή και επιπλέον να παρέχει ένα ψευδώνυμο στο χρήστη ώστε να αποκρύπτονται τα πραγματικά του στοιχεία και στη συνέχεια να προωθεί το επεξεργασμένο ερώτημα με τη νέα ασαφή θέση του χρήστη(cloaked spatial area) στον location-based database server.
4) Ακολούθως να στέλνει την απάντηση του location-based database server στον χρήστη. Σημειώνουμε ότι ο location-based database server δεν θα μπορούσε να στείλει άμεσα την απάντηση στο χρήστη γιατί δεν γνωρίζει την ακριβή του ταυτότητα, αλλά μόνο το ψευδώνυμο που έχει προμηθεύσει ο anonymizer.

Στην εικόνα παρατηρούμε ότι η διασφάλιση του location-privacy(Εικόνα β) των χρηστών ικανοποιείται με την εισαγωγή ενός επιπλέον layer infrastruction στο κλασσικό client-server provisioning model(Εικόνα α).



Εικόνα 2.2: Αρχιτεκτονική χωρίς διασφάλιση ιδιωτικότητας θέσης [18]



Εικόνα 2.3: Αρχιτεκτονική με ενσωματωμένο το στάδιο διασφάλισης ανωνυμίας (Location Anonymizer) [19]

2.3.3 Χρήση επιπέδων αφαίρεσης

Οι μέθοδοι αυτοί βασίζονται στην γνωστή από τη Θεωρία της Πληροφορίας έννοια της Εντροπίας [20]. Όσο μεγαλύτερη είναι η τιμή της εντροπίας, τόσο μικρότερη η ποσότητα και η ποιότητα της πληροφορίας που μεταδίδεται και επομένως τόσο μεγαλύτερη και η προστασία της ιδιωτικότητας που απολαμβάνουν οι χρήστες. Η αύξηση της εντροπίας επιτυγχάνεται με τη χρήση επιπέδων αφαίρεσης / αοριστίας για την αναφορά των δεδομένων από τους χρήστες στην υπηρεσία.
Από την άλλη μεριά η ποιότητα της υπηρεσίας εξαρτάται σε πολύ μεγάλο βαθμό από την ακρίβεια των πληροφοριών που λαμβάνει από τους χρήστες. Επομένως βλέπουμε ότι αυξανομένης της προστασίας της ιδιωτικότητας, η αποτελεσματικότητα και ως εκ τούτου και η χρησιμότητα μιας υπηρεσίας μειώνεται. Το ζητούμενο εδώ είναι να βρεθεί το επίπεδο εκείνο αφαίρεσης που θα συνδυάζει τον απαιτούμενο βαθμό προστασίας με το επιθυμητό επίπεδο ποιότητας υπηρεσιών.
Από την παραπάνω περιγραφή συμπεραίνουμε ότι είναι απαραίτητη η ύπαρξη μιας οντότητας η οποία θα παραλαμβάνει την ακριβή πληροφορία από τον χρήστη και θα επιλέγει το κατάλληλο επίπεδο αοριστίας έτσι ώστε να ικανοποιούνται οι απαιτήσεις προστασίας ιδιωτικότητας του εν λόγω χρήστη, αλλά και οι απαιτήσεις ποιότητας υπηρεσίας. Η οντότητα αυτή μπορεί να είναι ένας εξυπηρετητής ανωνυμίας (anonymizing server) ή ακόμη να είναι το ίδιο το κινητό τερματικό που θα προσφέρει την εν λόγω υπηρεσία στους χρήστες.
Στην περίπτωση χρήσης anonymizing server έχουμε το μειονέκτημα των επιπρόσθετων καθυστερήσεων που συνεπάγονται οι επιπλέον μεταδόσεις δεδομένων, όπως επίσης και τον κίνδυνο αστοχίας του ή ακόμα και την διενέργεια επίθεσης με σκοπό την υποκλοπή των πληροφοριών των χρηστών. Στην αντίθετη περίπτωση, της χρήσης δηλαδή των ίδιων των κινητών τερματικών στο ρόλο του anonymizer, αποφεύγονται οι παραπάνω κίνδυνοι, ενώ το υπολογιστικό κόστος που συνεπάγεται μια τέτοια σχεδίαση είναι μικρό για τα σύγχρονα PDA και τα έξυπνα κινητά τηλέφωνα.

2.4 Ποσοτικοποίηση Ιδιωτικότητας και Αποτελεσματικότητας

Για να μελετήσουμε τη σχέση ιδιωτικότητας – αποτελεσματικότητας υπηρεσίας θα πρέπει να ποσοτικοποιήσουμε τα δύο αυτά μεγέθη και να εξάγουμε την μεταξύ τους σχέση. Όσον αφορά την ιδιωτικότητα, ένα μοντέλο ποσοτικοποίησής της προτείνεται στο [21], στο οποίο γίνεται εφαρμογή της έννοιας της εντροπίας που ανέπτυξε ο Shannon στο [22] για να μοντελοποιήσει την αβεβαιότητα μιας πηγής πληροφορίας. Διακρίνονται και ορίζονται δύο είδη εντροπίας:

- **Εντροπία εντοπισμού θέσης**
- **Εντροπία προσωπικών προτιμήσεων**

Τα δύο αυτά είδη εντροπίας συνδυάζονται μεταξύ τους για να μας δώσουν ένα τελικό αριθμό, μέσω του οποίου θα μελετηθούν οι κίνδυνοι για την προστασία της ιδιωτικότητας.

2.4.1 Εντροπία εντοπισμού θέσης

Για την λειτουργία υπηρεσιών βασισμένων στη θέση είναι απαραίτητη η χρήση ενός συστήματος χαρτών, παραδείγματα τέτοιων συστημάτων είναι το Windows Live Local (a.k.a. MSN Virtual Earth) [23], το Google Maps [24], το Yahoo! Maps [25], κ.α. Τα περισσότερα τέτοια συστήματα κάνουν χρήση της Μερκατορικής προβολής (Mercator projection) [26]. Ένα από τα πιο σημαντικά χαρακτηριστικά των συστημάτων χαρτών είναι η υποστήριξη διαφορετικών επιπέδων λεπτομέρειας. Κάποια από αυτά τα συστήματα, όπως τα Virtual Earth [27] και Google Maps [24], είναι οργανωμένα σε ιεραρχική δομή με τη χρήση tiles («πλακίδια» ή τετραγωνίδια) και σε αυτά υποστηρίζεται ένας συγκεκριμένος αριθμός επιπέδων ακρίβειας (συνήθως 20-23). Σε κάθε ένα από αυτά τα επίπεδα, ο χάρτης χωρίζεται σε tiles τα οποία έχουν συγκεκριμένο μέγεθος σε pixels, και κάθε tile έχει δοθεί ένα αναγνωριστικό (ID). Για παράδειγμα έστω ότι στο επίπεδο 1, υπάρχουν 4 tiles, με IDs 0, 1, 2, 3. Στο επόμενο επίπεδο, η περιοχή του χάρτη που αντιστοιχεί σε κάθε tile διασπάται επίσης σε tiles (και πάλι συνήθως 4), επιστρέφοντας στο παράδειγμά μας, το tile 2 του επιπέδου 1 χωρίζεται στο επίπεδο 2 στα tiles με IDs: 20, 21, 22, 23. Η παραπάνω διαδικασία συνεχίζεται με τον ίδιο τρόπο σε όλα τα επίπεδα, έχοντας ως αποτέλεσμα τη δημιουργία μιας ιεραρχικής δομής (στο παράδειγμά μας, μιας ιεραρχικής δενδρικής δομής τύπου quad-tree). Σε αυτή τη δομή η ακρίβεια της θέσης αυξάνεται (και αντίστοιχα, η ιδιωτικότητα μειώνεται), όσο το επίπεδο λεπτομέρειας γίνεται μεγαλύτερο. Για παράδειγμα, εάν ένας χρήστης βρίσκεται σε μια περιοχή που αναπαρίσταται από το tile που έχει ID 1023312202121312 (στο 16ο επίπεδο λεπτομέρειας), τότε ως θέση του θα μπορούσε να αναφέρει εναλλακτικά το tile με ID 102331220212131 (στο 15ο επίπεδο) ή αυτό με ID 10233122021213 (στο 14ο επίπεδο), με την ακρίβεια να μειώνεται κάθε φορά. Η μόνη προϋπόθεση της μεθοδολογίας αυτής είναι η ιεραρχική δομή οργάνωσης του χώρου (π.χ. ένα region quad-tree [28], [29]) ενώ παραμένει ανεξάρτητη από το γεωγραφικό σύστημα αναπαράστασης.
Θα εφαρμόσουμε τώρα το μοντέλο εντροπίας του Shannon, σύμφωνα με το οποίο η εντροπία H ορίζεται ως εξής:

$$\boldsymbol{H} = -K \sum_{j=1}^{M} p_j \, log\left(p_j\right) \qquad (1)$$

στην περίπτωση ενός συστήματος γεωγραφικής θέσης (geographical location system) βασισμένου σε κάποια ιεραρχική δομή. Θα υποθέσουμε ότι για τον εντοπισμό της θέσης του χρήστη, θα χρησιμοποιηθεί το Global Positioning system (GPS), αφού αυτό είναι το ακριβέστερο σύστημα εντοπισμού παγκοσμίως. Ας συμβολίσουμε με *sGPS* το εμβαδό της περιοχής που αναφέρει το GPS ως θέση του χρήστη (ιδανικά αυτό ισούται με 1 m2 περίπου, αφού τόση είναι η επιφάνεια που καταλαμβάνει ένας άνθρωπος [30], όμως εφόσον εδώ εμπλέκεται το ζήτημα της ακρίβειας του GPS, η τιμή του θα είναι μεγαλύτερη – πιο συγκεκριμένα θα συζητηθεί παρακάτω). Ας

συμβολίσουμε με A το πραγματικό εμβαδό που αντιστοιχεί στην περιοχή του χάρτη την οποία ο χρήστης αναφέρει σε ένα LBS επεξεργαστή ερωτημάτων ως θέση του, δηλαδή στην προαναφερθείσα περίπτωση το πραγματικό εμβαδό που αντιστοιχεί σε ένα tile κάποιου επιπέδου λεπτομέρειας. Στην περίπτωση αυτή, το M ισούται με τον αριθμό των διακριτών πιθανών θέσεων, οι οποίες μπορούν να αντιστοιχούν στην πραγματική ακριβή θέση του χρήστη στην περιοχή. Επομένως,

$$M = \frac{A}{S_{GPS}} \qquad (2)$$

Κατά συνέπεια, p_j είναι η πιθανότητα κάθε μια από τις πιθανές ακριβείς θέσεις στην αναφερόμενη περιοχή να συνιστά την πραγματική ακριβή θέση του χρήστη και θα ισούται επομένως με:

$$p_j = \frac{S_{GPS}}{A} \qquad (3)$$

Από τις σχέσεις (1), (2), (3), προκύπτει ότι η εντροπία εντοπισμού θέσης δίνεται από τον τύπο:

$$\boldsymbol{H} = -\sum_{j=1}^{A/S_{GPS}} \frac{S_{GPS}}{A} \log\left(\frac{S_{GPS}}{A}\right)$$

Η τιμή του A εξαρτάται από το επίπεδο λεπτομέρειας (με άλλα λόγια, το επίπεδο ακρίβειας). Συνεπώς, η τιμή της εντροπίας H μεταβάλλεται όταν πλακίδια(tiles) διαφορετικών επιπέδων ακρίβειας στην ιεραρχική δομή (π.χ. κόμβοι διαφορετικού βάθους στο quad-tree) επιλέγονται από το χρήστη για να προσδιορίσουν τη θέση του.

2.4.2 Εντροπία προσωπικών προτιμήσεων

Μια αποτελεσματική και ποιοτική εξατομικευμένη υπηρεσία είναι συνήθως αποτέλεσμα της γνωστοποίησης στον server των προσωπικών προτιμήσεων του χρήστη. Οι προτιμήσεις του χρήστη μπορεί να περιλαμβάνουν παραμέτρους για πληθώρα ενδιαφερόντων που ο ίδιος μπορεί να έχει. Για παράδειγμα, το είδος ταινιών που προτιμά να παρακολουθεί ο χρήστης είναι απαραίτητη πληροφορία για μια υπηρεσία η οποία έχει ως σκοπό να κάνει προτάσεις στο χρήστη όταν αυτός δηλώσει ότι θέλει να πάει στον κινηματογράφο. Άλλα χαρακτηριστικά παραδείγματα είναι το αγαπημένο είδος μουσικής και η προτίμηση του χρήστη σε κάποια κουζίνα (π.χ. γαλλική, κινέζικη,μεσογειακή κ.λπ.). Οι προτιμήσεις είναι επίσης ένας τομέας που μπορεί να περιγραφεί είτε πιο συγκεκριμένα, οδηγώντας σε πιο ακριβή και «καλύτερης ποιότητας» αποτελέσματα, θυσιάζοντας την ιδιωτικότητα, είτε πιο αφηρημένα, εξασφαλίζοντας έτσι μεγαλύτερη ιδιωτικότητα, μειώνοντας όμως την ακρίβεια και την ποιότητα των αποτελεσμάτων. Επιστρέφοντας στο παράδειγμα του κινηματογράφου, ένας χρήστης μπορεί να δηλώσει πολύ συγκεκριμένα την προτίμησή του σε μαύρες κωμωδίες ή, πολύ πιο αόριστα, σε κωμωδίες.
Η εφαρμογή του μοντέλου εντροπίας κατά Shannon είναι απλή και προκύπτει από τη χρήση της αναδρομικής ιδιότητας ως ακολούθως: Υποθέτουμε ότι οι προτιμήσεις των χρηστών έχουν ομαδοποιηθεί και καταγραφεί σε ένα δέντρο προτιμήσεων, στο οποίο

κάθε κόμβος παιδί είναι ένα υποσύνολο του κόμβου πατέρα. Σαν παράδειγμα αναφέρουμε ότι αν κόμβος πατέρας είναι η κατηγορία ταινιών που καλείται θρίλερ, τότε σαν κόμβους παιδιά θα μπορούσαμε να έχουμε τα ψυχολογικά θρίλερ, τα θρίλερ φαντασίας κλπ. Η εντροπία **H** μιας κατηγορίας που βρίσκεται σε φύλλο του δέντρου ορίζεται ίση με **H**=0. Για οποιαδήποτε άλλη κατηγορία που βρίσκεται σε κόμβο του δέντρου η εντροπία υπολογίζεται βάσει των εντροπιών των κατηγοριών παιδιών αναδρομικά με την σχέση:

$$\mathbf{H}_{\kappa} = \mathbf{H}(p_1,p_{2,\ldots},p_n) + p_1\,\mathbf{H}_1^{k-1} + p_2\,\mathbf{H}_2^{k-1} + \ldots + p_n\,\mathbf{H}_n^{k-1} \quad (4)$$

,όπου $\mathbf{H}_{\kappa}$ η εντροπία μιας κατηγορίας που βρίσκεται σε κόμβο κ επιπέδου από κάποιο φύλλο, n το πλήθος των κόμβων παιδιών του εν λόγω κόμβου, $\mathbf{H}_i^{k-1}$ η εντροπία του i-στου από τους κόμβους παιδιά που βρίσκονται στο κ-1 επίπεδο και p_i η δεσμευμένη πιθανότητα ο χρήστης να έχει επιλέξει την κατηγορία που βρίσκεται στο i-στο κόμβο παιδί, δεδομένου ότι έχει επιλέξει κατηγορία που βρίσκεται σε κάποιον από τους κόμβους παιδιά. Προφανώς ισχύει :

$$p_1 + p_2 + \ldots + p_n = 1. \quad (5)$$

Τέλος, η συνάρτηση $H(p_1,p_{2,\ldots},p_n)$ είναι η συνάρτηση υπολογισμού εντροπίας του Shannon:

$$\mathbf{H}(p_1,p_{2,\ldots},p_n) = -K(p_1\log p_1 + \ldots + p_n\log p_n) \quad (6)$$

, όπου K μια αυθαίρετη σταθερά κλίμακας. Βλέπουμε επομένως ότι καθώς ανεβαίνουμε επίπεδα από τα φύλλα προς τους κόμβους, η εντροπία αυξάνει, επομένως ο χρήστης στέλνει λιγότερη πληροφορία στην υπηρεσία και είναι καλύτερα προστατευμένος.

2.4.3 Αποτελεσματικότητα υπηρεσίας

Αφού ορίσαμε την έννοια της εντροπίας και ποσοτικοποιήσαμε την ιδιωτικότητα ενός χρήστη, το επόμενο βήμα είναι να κάνουμε το ίδιο και για την έννοια της αποτελεσματικότητας μιας υπηρεσίας. Ας θεωρήσουμε μια υποθετική υπηρεσία με τη βοήθεια της οποίας θα εξάγουμε χρήσιμα συμπεράσματα και θα μαθηματικοποιήσουμε τον όρο «αποτελεσματικότητα χρήσης».
Υποθέτουμε ότι υπάρχει μια υπηρεσία, στην οποία ο χρήστης παρέχει την γεωγραφική του θέση και τις προσωπικές του προτιμήσεις σχετικά με ταινίες, με κάποιο επίπεδο αοριστίας και τα δύο, και η υπηρεσία του επιστρέφει μια λίστα από κινηματογράφους οι οποίοι συμβαδίζουν με τις προτιμήσεις του και παράλληλα βρίσκονται κοντά του. Είναι προφανές από όσα έχουν ειπωθεί ότι ο χρήστης θα πάρει την καλύτερη ποιότητα αποτελεσμάτων όταν δώσει στην υπηρεσία την ακριβή του θέση και τις ακριβείς του προτιμήσεις. Είναι επίσης προφανές ότι η ποιότητα των αποτελεσμάτων θα είναι φθίνουσα συνάρτηση της ιδιωτικότητας, δηλαδή όσο αυξάνει η προστασία της ιδιωτικότητας, τόσο θα απομακρυνόμαστε από την μέγιστη αποτελεσματικότητα που μπορεί να προσφέρει η υπηρεσία.
Αν θεωρήσουμε ως επίπεδο αναφοράς τα αποτελέσματα που προκύπτουν με χρήση ακριβών πληροφοριών εκ μέρους του χρήστη, τότε για να προσδιορίσουμε ένα οποιοδήποτε άλλο επίπεδο αποτελεσματικότητας αρκεί να ορίσουμε μια νόρμα στον

χώρο των αποτελεσμάτων και να υπολογίζουμε κάθε φορά την απόσταση των δεδομένων αποτελεσμάτων από τα αποτελέσματα της πλήρους αποτελεσματικότητας. Δεδομένου ότι για διαφορετικά ερωτήματα με το ίδιο μέτρο ιδιωτικότητας ενδέχεται η αποτελεσματικότητα που υπολογίζουμε να είναι διαφορετική, για να βρούμε μια χρήσιμη τιμή για την αποτελεσματικότητα θα πρέπει να πάρουμε τον στατιστικό μέσο όρο των διαφόρων ερωτημάτων.
Στο παράδειγμα με τους κινηματογράφους, έστω ότι ο χρήστης στέλνει την ακριβή του γεωγραφική θέση και την προτίμησή του για ταινίες Western. Η υπηρεσία προφανώς θα επιστρέψει μια λίστα με κινηματογράφους στους οποίους παίζονται Western και βρίσκονται κοντά στην θέση του χρήστη. Αν ο χρήστης θέλει να προστατεύσει την ιδιωτικότητά του, τότε μπορεί αντί της ακριβούς θέσης του να στείλει μια ευρύτερη περιοχή στην οποία βρίσκεται και αντί για Western να ζητήσει μια ευρύτερη κατηγορία ταινιών όπως Action – Adventure. Στην περίπτωση αυτή η λίστα που θα επιστραφεί θα είναι ένα υπερσύνολο της προηγούμενης, αφού θα περιλαμβάνει όλους τους κινηματογράφους που παίζουν ταινίες μιας ευρύτερης κατηγορίας. Μια πρώτη προσέγγιση απόστασης λοιπόν στον χώρο των αποτελεσμάτων είναι ο λόγος των πληθάριθμων των λιστών των δύο περιπτώσεων.

Λόγος Πληθάριθμων Λιστών: Είναι φανερό ότι ο δείκτης αποτελεσματικότητας που μόλις ορίσαμε δεν επηρεάζεται από το επίπεδο αοριστίας θέσης παρά μόνο από το επίπεδο αφαίρεσης των προτιμήσεων του χρήστη. Αυτό συμβαίνει διότι με αλλαγή στο επίπεδο αοριστίας της θέσης το μόνο που αλλάζει είναι οι σχετικές θέσεις των προορισμών στη λίστα, οι οποίες εξαρτώνται από το ποιος προορισμός βρίσκεται εγγύτερα στην αναφερθείσα θέση του χρήστη, ενώ το πλήθος των προορισμών που επιστρέφονται παραμένει σταθερό. Επομένως γίνεται σαφές ότι πρόκειται για έναν δείκτη που μετράει την αποτελεσματικότητα ως προς τις προτιμήσεις του χρήστη.

Ας θεωρήσουμε τώρα την ίδια υπηρεσία, αλλά με διαφορετική πολιτική στα επιστρεφόμενα αποτελέσματα. Έστω λοιπόν ότι αντί για όλους τους κινηματογράφους που ανταποκρίνονται στα κριτήρια που θέτει ο χρήστης, επιστρέφεται ένας συγκεκριμένος αριθμός από αυτούς με βάση το ποιοι βρίσκονται κοντινότερα στις προτιμήσεις του χρήστη. Ο αριθμός αυτός δεν θα πρέπει να παίζει κανέναν ρόλο στον μαθηματικό ορισμό της αποτελεσματικότητας. Με βάση τα ανωτέρω καταλήγουμε σε δυο ακόμη δείκτες αποτελεσματικότητας που μπορούν να χρησιμοποιηθούν.

Αποτελεσματικότητα θέσης: Ο δείκτης αποτελεσματικότητας αυτός προκύπτει από σύγκριση των στοιχείων των δύο λιστών ένα προς ένα. Η πρώτη λίστα περιέχει τα αποτελέσματα που επιστρέφει η υπηρεσία σε κάποιο αίτημα του χρήστη όταν τα δεδομένα έχουν αποσταλεί σε κάποιο επίπεδο αοριστίας, ενώ η δεύτερη τα αποτελέσματα όταν τα δεδομένα έχουν αποσταλεί με πλήρη ακρίβεια. Ο μέσος όρος των αποστάσεων μεταξύ των αντιστοίχων στοιχείων (1^o με 1^o,2^o με 2^o,κ.ο.κ) αποτελεί τον δείκτη που ονομάζουμε δείκτη αποτελεσματικότητας ως προς την θέση.

Αποτελεσματικότητα προτιμήσεων: Με παρόμοια σύγκριση των στοιχείων των δύο λιστών όπως ανωτέρω προκύπτει και ο δείκτης αποτελεσματικότητας προτιμήσεων. Η μόνη διαφορά είναι ότι εδώ μιλάμε για αποστάσεις όχι πλέον γεωγραφικές, αλλά στον χώρο των προτιμήσεων. Η απόσταση αυτή ορίζεται μέσω της κατηγοριοποίησης των προτιμήσεων σε ένα ιεραρχικό δέντρο προτιμήσεων, όπου εξειδικεύονται καθώς προχωρούμε προς τα φύλλα οι προτιμήσεις. Δυο κόμβοι του δέντρου οι οποίοι έχουν

κοινό πατέρα δεχόμαστε ότι έχουν απόσταση μεταξύ τους ίση με 1. Αν ο κοινός πρόγονος βρίσκεται δύο επίπεδα ψηλότερα, τότε η απόσταση τους είναι ίση με 2. Όμοια ορίζονται και οι υπόλοιπες αποστάσεις με βάση τα επίπεδα που χωρίζουν τους δύο κόμβους από τον κοινό τους πρόγονο.

Ένας συνολικός δείκτης αποτελεσματικότητας μπορεί να προκύψει από τον συνδυασμό των τριών προηγουμένων δεικτών, εισάγοντας κατάλληλα βάρη για την επίδραση του καθενός δείκτη. Τα βάρη δεν καθορίζονται μονοσήμαντα, αλλά επιλέγονται αυθαίρετα με βάση το τι θεωρείται σημαντικότερο σε μια δεδομένη υπηρεσία. Επομένως από υπηρεσία σε υπηρεσία τα βάρη αλλάζουν και ως εκ τούτου και η επίδραση των τριών δεικτών στην εξαγωγή του δείκτη Ολικής Αποτελεσματικότητας.

3.Σενάριο Προσομοίωσης

3.1 Γενικά

Στην προσπάθειά μας να ανακαλύψουμε τις σχέσεις που διέπουν την προστασία της ιδιωτικότητας και την αποτελεσματικότητα της χρήσης εξατομικευμένων υπηρεσιών, θα αναπτύξουμε ένα σενάριο χρήσης μιας τέτοιας υπηρεσίας. Το σενάριο αυτό θα το προσομοιώσουμε στον υπολογιστή και θα το μελετήσουμε περαιτέρω με τη βοήθεια του λογισμικού Siafu. Πρόκειται για ελεύθερο λογισμικό, ανοικτού κώδικα κατάλληλο για προσομοιώσεις προσωποποιημένων – εξατομικευμένων υπηρεσιών.
Η προσομοίωση αυτή θα μας δώσει ποσοτικές και ποιοτικές πληροφορίες για την ασφάλεια ως προς την ιδιωτικότητα που απολαμβάνουν οι χρήστες, καθώς και για την αποτελεσματικότητα της χρήσης μιας υπηρεσίας μέσω της ευχαρίστησης που αυτή προσφέρει στους χρήστες. Με τη χρήση της προσομοίωσης είναι δυνατή η διόρθωση προβλημάτων ασφαλείας που πιθανόν αντιμετωπίζει μια υπηρεσία , αλλά και η βελτιστοποίηση της απόδοσης της, προτού αυτή δοθεί προς χρήση στους τελικούς χρήστες – πελάτες της υπηρεσίας.
Το σενάριό μας θα βασιστεί στην πλατφόρμα PLASMA (Personalized, Location Aware Services over Mobile Architectures) και στην θεώρηση διαφόρων κατηγοριών χρηστών με διαφορετικά χαρακτηριστικά η καθεμία. Το τελικό αποτέλεσμα προσδοκούμε να είναι η εξαγωγή της έκφρασης της αποτελεσματικότητας που απολαμβάνουν οι χρήστες συναρτήσει της ασφάλειας της ιδιωτικότητάς τους.

3.2 Η πλατφόρμα PLASMA

Η ανάπτυξη και ο σχεδιασμός της πλατφόρμας PLASMA βρίσκεται σε εξέλιξη. Πρόκειται για μια προσπάθεια σχεδίασης και υλοποίησης ενός πλαισίου υπηρεσιών για κινητά τερματικά (PDA, έξυπνα κινητά τηλέφωνα), το οποία να είναι σε θέση να προσφέρει προσωποποιημένες – εξατομικευμένες υπηρεσίες (personalized services) με επίγνωση κατάστασης (context awareness). Ιδιαίτερη έμφαση δίνεται στην επίγνωση θέσης (Location awareness) και στην επίγνωση των προτιμήσεων και της ταυτότητας των χρηστών (preferences – identity awareness).
Η ιδιαιτερότητα του PLASMA έγκειται στο γεγονός ότι οι υπηρεσίες που προσφέρει δεν παρέχονται με συμβατικό τρόπο, παρά επιτρέπεται στον χρήστη να σχεδιάσει και να δημιουργήσει ο ίδιος μια υπηρεσία. Πρόκειται λοιπόν για μία πλήρως ανοικτή αρχιτεκτονική, όπου ο καθένας μπορεί να δοκιμάσει την δική του υπηρεσία. Απόρροια του γεγονότος αυτού είναι η πολύ μεγάλη ευελιξία και η επεκτασιμότητα της πλατφόρμας.
Η αρχιτεκτονική του PLASMA στηρίζεται στην ανάπτυξη service components, κομματιών κώδικα εξειδικευμένα για να επιτελούν και να προσφέρουν την υπηρεσία, τα οποία αποθηκεύονται σε έναν κεντρικό εξυπηρετητή. Τα κινητά τερματικά, μέσω Internet (Wi-Fi, ADSM, WAP) αποκτούν πρόσβαση στον εξυπηρετητή και απολαμβάνουν τις υπηρεσίες . Οι λειτουργίες που επιτελούνται στα service components και η επεξεργασία των πληροφοριών σ' αυτά περιλαμβάνουν τα ακόλουθα:

i. Παραλαβή του αιτήματος του κινητού τερματικού συνοδευόμενο από στοιχεία της ταυτότητας του αιτούντος, της θέσης του, του χρόνου στον οποίο

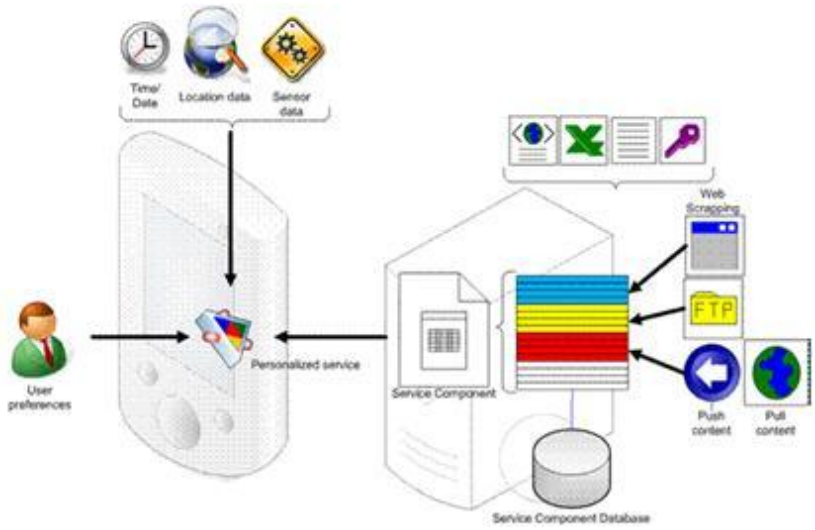
αναφέρεται το αίτημα και πιθανώς και άλλα δεδομένα που προέρχονται από αισθητήρες που βρίσκονται ενσωματωμένοι στο τερματικό.

ii. Επεξεργασία των δεδομένων που βρίσκονται και φυλάσσονται στον εξυπηρετητή για να γίνει δυνατή η ικανοποίηση του αιτήματος και η αποστολή της απάντησης του ερωτήματος.

iii. Εξατομίκευση των πληροφοριών με βάση τις ρυθμίσεις – προτιμήσεις του χρήστη, τις συνθήκες (context), τη θέση (location) , τον χρόνο (time) και τα δεδομένα των αισθητήρων (sensor data).

Γίνεται αντιληπτό από τα παραπάνω ότι η πλατφόρμα μπορεί να χωριστεί σε δύο επιμέρους τμήματα. Το μέρος i) των κινητών τερματικών και των χρηστών και ii) το μέρος του εξυπηρετητή, ο οποίος περιέχει τα service components και φυλάσσει τις βάσεις δεδομένων στις οποίες βρίσκονται αποθηκευμένα τα δεδομένα και οι πληροφορίες για τα σημεία ενδιαφέροντος (Points Of Interest).

Η αρχιτεκτονική του συστήματος είναι αυτή που παρουσιάζεται στο παρακάτω σχήμα:



3.3 Χρήση υπηρεσίας και παραδείγματα υπηρεσιών

Για να μπορέσει ένα κινητό τερματικό να χρησιμοποιήσει μια υπηρεσία που παρέχεται από το PLASMA, θα πρέπει αρχικά η εφαρμογή που τρέχει σ ‘αυτό να συνδεθεί στον κεντρικό εξυπηρετητή και να αναγνωρίσει όλα τα διαθέσιμα service components που υπάρχουν σ ‘αυτόν. Στην συνέχεια γίνεται η αρχική παραμετροποίηση με βάση τις πληροφορίες που έχει μαζέψει και στέλνει η φορητή συσκευή.

Οι πληροφορίες αυτές στέλνονται στο κατάλληλο επίπεδο αφαίρεσης ώστε να παρέχεται η προστασία των προσωπικών δεδομένων του χρήστη, αλλά και η επιζητούμενη αποτελεσματικότητα της υπηρεσίας. Το «κατάλληλο» επίπεδο αφαίρεσης δεν είναι κάτι που μπορεί να οριστεί μονοσήμαντα, αφού ο όρος

«προστασία προσωπικών δεδομένων» ποσοτικοποιείται διαφορετικά από χρήστη σε χρήστη και από κατάσταση σε κατάσταση. Συνεπώς δίνεται η δυνατότητα στον εκάστοτε χρήστη να επιλέξει ο ίδιος τον βαθμό αφαίρεσης και ως εκ τούτου και τον βαθμό προστασίας της ιδιωτικότητας του.
Μετά την αρχικοποίηση μιας υπηρεσίας, παρέχεται στον χρήστη η επιλογή, αν φυσικά υποστηρίζεται από την εν λόγω υπηρεσία, να χρησιμοποιήσει την υπηρεσία on demand, δηλαδή όταν και όποτε θελήσει ο ίδιος ή να την αφήσει να τρέχει συνεχώς στο παρασκήνιο, αφήνοντας το κινητό τερματικό να καθορίζει αυτόματα την παροχή πληροφορίας από και προς αυτό. Αν λοιπόν υποθέσουμε ότι η υπηρεσία μας είναι τέτοια ώστε μας παρέχει πληροφορίες για τον τοπικό καιρό, χρήση on demand θα σήμαινε ότι παραμένει ανενεργή μέχρι ο χρήστης να ζητήσει πρόβλεψη και τότε μόνον στέλνει τα δεδομένα της πρόβλεψης. Στην αντίθετη περίπτωση, η υπηρεσία καλείται συνεχώς από το κινητό τερματικό, οι προβλέψεις φτάνουν συνέχεια και εμφανίζονται στην οθόνη του τερματικού.
Στην συνέχεια παρουσιάζουμε παραδείγματα υπηρεσιών που θα μπορούσαν ή έχουν υλοποιηθεί στην πλατφόρμα PLASMA.

Υπηρεσία εύρεσης εστιατορίου

Ας υποθέσουμε ότι κάποιος χρήστης βρίσκεται σε μια άγνωστη προς αυτόν περιοχή και αναζητεί ένα εστιατόριο για να γευματίσει. Η μεγάλη ποικιλία σε τύπους εστιατορίου, σε επίπεδο τιμών, αλλά και το γεγονός ότι δεν γνωρίζει την περιοχή τον οδηγούν στην αναζήτηση βοήθειας μέσω της υπηρεσίας που παρέχεται από το PLASMA.
Το κινητό του τερματικό συνδέεται με τον κεντρικό εξυπηρετητή της υπηρεσίας και αφού αποσταλούν οι απαραίτητες πληροφορίες, όπως είδος εστιατορίου (ταβέρνα, κινέζικο, ελληνική κουζίνα), επίπεδο τιμών (φτηνό, ακριβό, πολυτελείας), περιοχή αναζήτησης, ο εξυπηρετητής επιστρέφει μια λίστα με εστιατόρια που ανταποκρίνονται καλύτερα στις προτιμήσεις του χρήστη και παράλληλα βρίσκονται κοντά του.
Το επίπεδο της λεπτομέρειας στις παρεχόμενες στον εξυπηρετητή πληροφορίες καθορίζεται από τον ίδιο τον χρήστη, ανάλογα με το πόσο τον ενδιαφέρει η προστασία των ιδιωτικών του δεδομένων και με το βαθμό της αποτελεσματικότητας που προσδοκά να έχει η υπηρεσία.

Υπηρεσία εμφάνισης σημείων ενδιαφέροντος πάνω σε χάρτη

Η μορφή της υπηρεσίας αυτής μοιάζει πολύ με αντίστοιχες που συναντάμε στα σύγχρονα συστήματα πλοήγησης με GPS. Η κύρια διαφορά των δύο υπηρεσιών είναι ότι στα συστήματα πλοήγησης ο χάρτης και τα σημεία ενδιαφέροντος βρίσκονται αποθηκευμένα στην συσκευή GPS και ανανεώνονται από τους κατασκευαστές ανά χρονικά διαστήματα αρκετά μεγάλα, ενώ στην εκδοχή του PLASMA τα σημεία ενδιαφέροντος αναζητούνται δυναμικά από τον εξυπηρετητή κάθε φορά που χρειάζονται. Στην δεύτερη περίπτωση η ανανέωση και η προσθήκη νέων σημείων ενδιαφέροντος γίνεται άμεσα προσθέτοντας απλώς κάποιο καινούριο σημείο στις βάσεις δεδομένων του εξυπηρετητή.
Σε μια τέτοια υπηρεσία ο χρήστης μπορεί να ζητά από τον εξυπηρετητή συγκεκριμένη κατηγορία σημείων ενδιαφέροντος τα οποία και θα εμφανίζονται στον χάρτη καθώς ο χρήστης θα περνά από εκεί. Η απαιτούμενη πληροφορία που θα πρέπει να παρέχεται στο σύστημα για κάτι τέτοιο είναι η θέση του κινητού

τερματικού, καθώς και το είδος των σημείων ενδιαφέροντος που μας ενδιαφέρουν. Είναι προφανές ότι αυτή η υπηρεσία είναι δυνατόν να παρέχεται on demand ή να τρέχει συνεχώς στο παρασκήνιο.

Υπηρεσία επιλογής κινηματογράφου

Ένα ακόμα παράδειγμα προσωποποιημένης – εξατομικευμένης υπηρεσίας αποτελεί η υπηρεσία επιλογής κινηματογράφου με βάση συγκεκριμένα κριτήρια. Τα κριτήρια αυτά θα μπορούσαν να είναι το είδος της ταινίας που προβάλλεται στον κινηματογράφο, η τοποθεσία στην οποία βρίσκεται, η ύπαρξη ή όχι οργανωμένου χώρου στάθμευσης, η ύπαρξη παιδότοπου ή ακόμα και η τιμή στην οποία προσφέρονται οι εν λόγω υπηρεσίες.
Οι πληροφορίες αυτές παρέχονται στον εξυπηρετητή με κάποιο επίπεδο αφαίρεσης, το οποίο καθορίζεται από τον χρήστη σε συνάρτηση με το επίπεδο ιδιωτικότητας που επιθυμεί.
Το είδος αυτό της υπηρεσίας είναι εκείνο που θα χρησιμοποιήσουμε στην προσομοίωσή μας για να ελέγξουμε το πόσο επηρεάζεται η αποτελεσματικότητα της υπηρεσίας με την αύξηση της ιδιωτικότητας και το αντίστροφο.
Άλλα παραδείγματα υπηρεσιών είναι τα ακόλουθα: υπηρεσία εύρεσης διανυκτερευόντων φαρμακείων, εύρεση βενζινάδικων, κάδων ανακύκλωσης, καφετεριών.

3.4 Σενάριο Προσομοίωσης

Για να μπορέσουμε να προσομοιώσουμε στον υπολογιστή ένα φαινόμενο, θα πρέπει να ξεχωρίσουμε τα ουσιώδη στοιχεία, τα οποία είναι ιδιάζουσας σημασίας, από τις λεπτομέρειες του φαινομένου. Στην περίπτωσή μας, για διευκόλυνση, θα χωρίσουμε το προς μελέτη σενάριο σε τρία επιμέρους τμήματα, τα οποία και θα περιγράψουμε αναλυτικά παρακάτω. Τα τρία αυτά τμήματα είναι τα εξής:

i. η υπηρεσία που θα εξετάσουμε
ii. ο κόσμος της προσομοίωσης
iii. οι χρήστες και η συμπεριφορά τους σε σχέση με τα υπόλοιπα στοιχεία της προσομοίωσης

Η δομή αυτή επιλέχτηκε με κριτήριο την καλύτερη αντιστοίχησή της με την ήδη υπάρχουσα δομή του προσομοιωτή Siafu, την οποία και θα αναλύσουμε σε επόμενη ενότητα.

3.4.1 Η υπηρεσία της προσομοίωσης

Το σημαντικότερο κομμάτι της προσομοίωσης αποτελεί ασφαλώς η υπηρεσία της οποίας την ασφάλεια και την αποτελεσματικότητα θα εξετάσουμε. Επίσης είναι εκείνη που θα καθορίσει σε μεγάλο βαθμό ποια στοιχεία του πραγματικού κόσμου θα πρέπει να συμπεριλάβουμε στην προσομοίωσή μας και ποια είναι ασφαλές και συνετό να παραλείψουμε.
Η υπηρεσία του σεναρίου μας είναι μια υπηρεσία εύρεσης ενός προορισμού για διασκέδαση, τέτοιου ώστε να ταιριάζει με τις ιδιαίτερες προτιμήσεις κάθε χρήστη, ενώ παράλληλα να βρίσκεται και κοντά σε αυτόν. Τέτοιου είδους υπηρεσίες απαιτούν

την παροχή δεδομένων προσωπικών προτιμήσεων στην υπηρεσία, άρα αντιμετωπίζουν ζητήματα προστασίας της ιδιωτικότητας, ενώ σε περίπτωση αυξημένης προστασίας και υψηλού επιπέδου αφαίρεσης, τα προβλήματα ιδιωτικότητας μετατρέπονται σε προβλήματα αποτελεσματικότητας.
Σύμφωνα με την αρχιτεκτονική της πλατφόρμας PLASMA, η υπηρεσία μας μπορεί να διαιρεθεί σε δύο μέρη. Το πρώτο αποτελείται από τον κεντρικό εξυπηρετητή της υπηρεσίας, ενώ το δεύτερο είναι το σύνολο των κινητών τερματικών που χρησιμοποιούν την υπηρεσία. Μια περιγραφή των δύο μερών δίνεται στην συνέχεια.

Κεντρικός εξυπηρετητής: Ο εξυπηρετητής της υπηρεσίας αποτελείται από δύο τμήματα. Το πρώτο τμήμα είναι οι βάσεις δεδομένων που περιέχουν τις πληροφορίες των οντοτήτων που ενδιαφέρουν. Στην περίπτωσή μας οι οντότητες για τις οποίες ενδιαφερόμαστε είναι χώροι διασκέδασης. Το δεύτερο τμήμα είναι το λογισμικό που βρίσκεται στον εξυπηρετητή και ο ρόλος του είναι να παραλαμβάνει τα αιτήματα των χρηστών και αφού τα επεξεργαστεί και συμβουλευτεί τις βάσεις δεδομένων, να επιστρέφει μια λίστα χώρων διασκέδασης που να ταιριάζει στις εκάστοτε προτιμήσεις του χρήστη. Τα δύο τμήματα αναλύονται στα επόμενα.

Βάση Δεδομένων: Όπως αναφέρθηκε, στη βάση δεδομένων φυλάσσονται οι πληροφορίες που αφορούν τους χώρους διασκέδασης. Οι χώροι διασκέδασης που περιλαμβάνονται, μπορούν να ανήκουν σε μία από τις παρακάτω κατηγορίες:

- Καφετέριες
- Μπαρ
- Κινηματογράφοι
- Ταβέρνες
- Κρεπερί
- Internet Cafe

Για κάθε μία από τις κατηγορίες αυτές κρατούνται στοιχεία σχετικά με την λειτουργία και τα χαρακτηριστικά του κάθε καταστήματος. Αναλυτικότερα υπάρχουν δεδομένα για:

i. Γεωγραφική Θέση: η θέση στην οποία βρίσκεται το κατάστημα
ii. Ωράριο Λειτουργίας :οι ώρες στις οποίες ανοίγει και κλείνει το κάθε κατάστημα. Σχετίζεται άμεσα με τον χρόνο υποβολής του αιτήματος του χρήστη, ώστε να μην προτείνονται προορισμοί που είναι εκτός λειτουργίας την χρονική στιγμή του αιτήματος.
iii. Επίπεδο Ποιότητας Υπηρεσιών :σε κλίμακα από 1 έως 5 αξιολογείται το επίπεδο υπηρεσιών του κάθε καταστήματος. Χρησιμοποιείται για να συσχετίσει τα καταστήματα με τις προτιμήσεις των χρηστών.
iv. Επίπεδο Τιμών :σε κλίμακα από 1 έως 5 αξιολογείται το επίπεδο υπηρεσιών του κάθε καταστήματος. Χρησιμοποιείται για να συσχετίσει τα καταστήματα με τις προτιμήσεις των χρηστών.
v. Ειδικά για την κατηγορία των κινηματογράφων, κρατείται επίσης το είδος της ταινίας που προβάλλεται.

Επεξεργασία αιτημάτων: Η μορφή και οι πληροφορίες που περιλαμβάνουν τα αιτήματα των χρηστών παρουσιάζονται στη συνέχεια. Για να καταστεί δυνατή η επεξεργασία ενός αιτήματος, αυτό θα πρέπει να περιέχει τις παρακάτω πληροφορίες.

i. Είδος ή είδη χώρων διασκέδασης που αναζητούνται από το χρήστη.
ii. Γεωγραφική Θέση (ακριβής ή σε κάποιο επίπεδο αοριστίας) του χρήστη
iii. Χρόνος κατά τον οποίο θέλει ο χρήστης να διασκεδάσει. Μπορεί να συμπίπτει με το χρόνο υποβολής του αιτήματος, μπορεί να αναφέρεται και σε άλλη χρονική στιγμή.
iv. Ειδικά για την κατηγορία των κινηματογράφων, δίνεται η δυνατότητα στον χρήστη να προσδιορίσει την προτίμησή του σε κάποιο είδος ταινίας. Η προτίμηση αυτή μπορεί να είναι είτε ακριβής (αναφέρεται στην υπηρεσία όλο το μονοπάτι στο δέντρο προτιμήσεων) είτε σε κάποιο επίπεδο αοριστίας (αναφέρεται το μονοπάτι έως κάποιον κόμβο και δεν φτάνουμε σε φύλλο).
v. Μέγιστη απόσταση από το σημείο στο οποίο βρίσκεται ο χρήστης. Αν κάποιος χώρος ταιριάζει με τις προτιμήσεις ου χρήστη, αλλά βρίσκεται μακρύτερα από την δηλωθείσα μέγιστη απόσταση, τότε ο προορισμός αυτός αποκλείεται και δεν βρίσκεται στη λίστα αποτελεσμάτων που επιστρέφεται.
vi. Μέγιστο πλήθος επιστρεφόμενων προορισμών. Το μέγιστο πλήθος στοιχείων της λίστας αποτελεσμάτων που πρέπει να επιστρέψει η υπηρεσία. Στην περίπτωση που υπάρχουν παραπάνω προορισμοί που ικανοποιούν τις προτιμήσεις του χρήστη, κρατούνται εκείνοι που είναι περισσότερο ταιριαστοί.

Αφού λοιπόν αφιχθεί ένα αίτημα στον κεντρικό εξυπηρετητή και περιέχει όλες τις παραπάνω πληροφορίες, τότε γίνεται η επεξεργασία του για να κατασκευασθεί η λίστα των αποτελεσμάτων. Από το σύνολο των δυνατών προορισμών επιλέγονται εκείνοι που 1)ανήκουν στην κατηγορία ή κατηγορίες που έχει προσδιορίσει ο χρήστης, 2)βρίσκονται σε απόσταση μικρότερη από τη μέγιστη επιθυμητή από το σημείο που βρίσκεται (ή έχει αναφέρει ότι βρίσκεται ο χρήστης), 3)είναι ανοικτά κατά τον χρόνο που επιθυμεί να τα επισκεφτεί ο χρήστης και 4)ειδικά για την κατηγορία των κινηματογράφων, προβάλλουν το είδος της ταινίας που επιθυμεί ο χρήστης. Τέλος, αν το πλήθος των προορισμών που ικανοποιούν τα παραπάνω κριτήρια υπερβαίνει το μέγιστο πλήθος προορισμών που απαιτείται να επιστραφούν, τότε κρατούνται εκείνοι που βρίσκονται κοντύτερα στις προτιμήσεις του χρήστη.
Για να γίνει το τελευταίο, χρειαζόμαστε μια σχέση διάταξης μεταξύ των προορισμών. Αυτή τη σχέση, για όλες της κατηγορίες πλην των κινηματογράφων, μας τη δίνει η απόσταση του προορισμού από τη θέση του χρήστη. Μ’ αυτόν τον τρόπο ψηλότερα στη λίστα βρίσκεται ο κοντινότερος προς τον χρήστη προορισμός. Όσον αφορά την κατηγορία των κινηματογράφων, τα κριτήρια που μπορούν να μας δώσουν κατάταξη είναι:

i. Η απόσταση από τη θέση του χρήστη και
ii. Η απόσταση πάνω στο δέντρο της κατηγορίας ταινιών που προβάλλεται στον κινηματογράφο από την δηλωθείσα κατηγορία προτίμησης του χρήστη.

Για την υλοποίηση της υπηρεσίας μας, επιλέγουμε να γίνεται διάταξη πρώτα κατά «απόσταση προτιμήσεων» και στη συνέχεια κατά «απόσταση γεωγραφικής θέσης». Με αυτόν τον τρόπο τοποθετούνται ψηλότερα στη λίστα οι κινηματογράφοι που βρίσκονται κοντινότερα στις προτιμήσεις του χρήστη, ενώ ανάμεσα σε δύο κινηματογράφους με ίδια «απόσταση προτιμήσεων» πρώτος τοποθετείται ακείνος με τη μικρότερη «απόσταση θέσης».
Αφού επιλεγούν οι πιθανοί προορισμοί και γίνει η διάταξή τους σύμφωνα με τα παραπάνω, ενσωματώνονται στα αποτελέσματα τα στοιχεία για την ποιότητα και το κόστος των υπηρεσιών που προσφέρει ο κάθε προορισμός. Με αυτόν τον τρόπο δίνεται η δυνατότητα στον χρήστη που θα παραλάβει τα αποτελέσματα να τα κατατάξει περαιτέρω. Αυτή η δεύτερη επεξεργασία συμβαίνει τοπικά, για λόγους προστασίας ιδιωτικότητας, στο κινητό τερματικό σύμφωνα με το τοπικό προφίλ του χρήστη και θα αναφερθούμε σε αυτήν στη συνέχεια.

Κινητά τερματικά: Το δεύτερο κομμάτι της αρχιτεκτονικής της υπηρεσίας αποτελείται από το σύνολο των κινητών τερματικών που ζητούν υπηρεσίες από τον κεντρικό εξυπηρετητή.
Ο ρόλος των κινητών τερματικών στην διάρθρωση της υπηρεσίας μας είναι τριπλός. Χρησιμοποιούνται για να επιτελούν τους παρακάτω σκοπούς:

i. Υπολογισμός του επιπέδου αφαίρεσης / αοριστίας στο οποίο οι προτιμήσεις και η θέση του χρήστη θα προσαρμοστούν προτού σταλούν στον κεντρικό εξυπηρετητή.
ii. Επικοινωνία κινητού τερματικού – εξυπηρετητή. Αποστολή των δεδομένων και των προτιμήσεων του χρήστη και παραλαβή των αποτελεσμάτων του αιτήματος.
iii. Μετά την επιστροφή των αποτελεσμάτων πραγματοποιούν το τελικό φιλτράρισμα της λίστας με βάση το τοπικό προφίλ του χρήστη και εμφανίζουν τα αποτελέσματα στον χρήστη ή ακόμη και επιλέγουν τον βέλτιστο για τον χρήστη προορισμό.

Υπολογισμός επιπέδου αφαίρεσης:

Πριν την αποστολή του αιτήματος στον κεντρικό εξυπηρετητή, δίνεται η δυνατότητα στον χρήστη να καθορίσει το επίπεδο προστασίας της ιδιωτικότητας που επιθυμεί. Αυτό επιτυγχάνεται διαλέγοντας ο χρήστης ένα από τα 5 επίπεδα προστασίας ιδιωτικότητας για θέση και το ίδιο γίνεται και για την προστασία ιδιωτικότητας των προτιμήσεών του.
Για την περίπτωση της ιδιωτικότητας γεωγραφικής θέσης, ορίζεται ως επίπεδο 1 η αναφορά της ακριβούς θέσης του χρήστη στην υπηρεσία. Όσο ο αριθμός του επιπέδου αυξάνει, άλλο τόσο αυξάνει και το επίπεδο αοριστίας των δεδομένων που αποστέλλονται στην υπηρεσία. Το μοντέλο που ακολουθείται είναι εκείνο που περιγράφηκε στην παράγραφο 2.4.1, όπου όμως το πλήθος των επιπέδων λεπτομέρειας για την περίπτωσή μας είναι 5 και παρουσιάζονται στη συνέχεια.

- Επίπεδο 1: Αναφέρεται στην υπηρεσία ως θέση του χρήστη η ακριβής / πραγματική του θέση

- Επίπεδο 2: Η πόλη μας χωρίζεται σε 64 ίσες ορθογώνιες παραλληλόγραμμες περιοχές και ως θέση του χρήστη αναφέρεται το κέντρο του παραλληλογράμμου στο οποίο βρίσκεται ο χρήστης.
- Επίπεδο 3: Η πόλη μας χωρίζεται σε 8 ίσες ορθογώνιες παραλληλόγραμμες περιοχές και ως θέση του χρήστη αναφέρεται το κέντρο του παραλληλογράμμου στο οποίο βρίσκεται ο χρήστης.
- Επίπεδο 4: Η πόλη μας χωρίζεται σε 4 ίσες ορθογώνιες παραλληλόγραμμες περιοχές και ως θέση του χρήστη αναφέρεται το κέντρο του παραλληλογράμμου στο οποίο βρίσκεται ο χρήστης.
- Επίπεδο 5: Αποτελεί το μέγιστο επίπεδο προστασίας της ιδιωτικότητας. Ως θέση του χρήστη αναφέρεται στην υπηρεσία το κέντρο της πόλης ανεξάρτητα από την πραγματική θέση του χρήστη.

Παρόμοια προσέγγιση ακολουθείται και για την αναφορά των προτιμήσεων του χρήστη στην περίπτωση της επιλογής είδους ταινίας. Το πλήθος των επιπέδων λεπτομέρειας είναι και σε αυτήν την περίπτωση 5, ενώ το επίπεδο 1 εξακολουθεί και εδώ να συμβολίζει την πλήρη ακρίβεια στην αναφορά των προτιμήσεων. Ως πλήρη ακρίβεια στην περίπτωση των προτιμήσεων, ορίζουμε την αναφορά ολόκληρου του μονοπατιού από την ρίζα ως το φύλλο με την κατηγορία που επιθυμεί ο χρήστης στο xml που ορίζονται οι κατηγορίες των ταινιών. Άνοδος επιπέδου κατά ν σημαίνει αντίστοιχα αναφορά του μονοπατιού μέχρι τον κόμβο που βρίσκεται ν επίπεδα πάνω από το φύλλο με την ακριβή προτίμηση του χρήστη.

Επικοινωνία Κινητού Τερματικού – Κεντρικού Εξυπηρετητή

Η επικοινωνία των κινητών τερματικών με τον εξυπηρετητή της υπηρεσίας γίνεται μέσω internet. Για τις ανάγκες της προσομοίωσής μας θα υποθέσουμε ότι υπάρχει ασύρματη πρόσβαση στο internet σε κάθε περιοχή της πόλης μας. Αυτό ακόμη δεν είναι εφικτό, αλλά με τις νέες τεχνολογίες internet over gsm που θα χρησιμοποιούν τις κεραίες κινητής τηλεφωνίας για μεταφορά δεδομένων και ασύρματη πρόσβαση στο διαδίκτυο, κάτι τέτοιο θα είναι πολύ σύντομα πραγματικότητα. Επίσης, δεδομένου ότι σκοπός μας είναι να μελετήσουμε προβλήματα ιδιωτικότητας και όχι επίδοση συτημάτων, υποθέτουμε ότι η υπηρεσία μας διαθέτει επαρκές εύρος ζώνης ώστε να εξυπηρετεί οποιονδήποτε αριθμό χρηστών θελήσουμε να συμπεριλάβουμε στην προσομοίωσή μας.

Τελικό Φιλτράρισμα – Επιλογή Βέλτιστου Προορισμού

Μετά την επιστροφή από την υπηρεσία της λίστας των προορισμών που προτείνονται στον χρήστη, υπάρχει η δυνατότητα περαιτέρω επεξεργασίας της στο κινητό τερματικό. Η επεξεργασία αυτή βασίζεται στο προφίλ του χρήστη που κρατείται τοπικά στην κινητή συσκευή. Το τοπικό αυτό προφίλ δεν ανακοινώνεται προς τα έξω και έτσι δεν δημιουργούνται θέματα ασφαλείας.

Στην προσομοίωσή μας το τοπικό προφίλ αποτελείται ενδεικτικά από 3 παραμέτρους.

- Βάρος που δίνει ο χρήστης στο πόσο κοντά βρίσκεται ο προτεινόμενος προορισμός.

- Βάρος που δίνει ο χρήστης στην ποιότητα των υπηρεσιών που προσφέρονται.
- Βάρος που δίνει ο χρήστης στο κόστος υπηρεσιών του προτεινόμενου προορισμού.

Οι 3 αυτές παράμετροι χρησιμοποιούνται στη συνέχεια για να κατατάξουν τους προτεινόμενους προορισμούς. Η τρόπος με τον οποίο γίνεται η κατάταξη αυτή περιγράφεται στη συνέχεια.
Για κάθε επιστρεφόμενο από την υπηρεσία προτεινόμενο προορισμό υπολογίζεται ένας αριθμός που δείχνει το πόσο ταιριάζει αυτός ο προορισμός με το τοπικό προφίλ του χρήστη. Πρόκειται επομένως για μια διαδικασία αθροίσματος με βάρη. Ο τύπος που χρησιμοποιούμε είναι ο ακόλουθος:
w = (βάρος απόστασης)*(απόσταση χρήστη-προορισμού) + (βάρος ποιότητας)*(ποιότητα υπηρεσιών προορισμού) +(βάρος κόστους)*(κόστος υπηρεσιών προορισμού)
Στην συνέχεια τα αποτελέσματα εμφανίζονται στην οθόνη κατά φθίνουσα σειρά w. Ο προορισμός με το μέγιστο w προτείνεται ως βέλτιστος. Φυσικά υπάρχει η δυνατότητα ο χρήστης να παρακάμψει αυτή τη διαδικασία και να διαλέξει μόνος του ον προορισμό που τον ικανοποιεί σύμφωνα με τις ιδιαίτερες περιστάσεις στις οποίες ενδεχομένως να βρίσκεται.

3.4.2 Οι Προορισμοί(Places) της προσομοίωσης

Η πόλη που θα προσομοιώσουμε διαθέτει ένα μεγάλο πλήθος προορισμών τους οποίους μπορούν να επισκεφτούν οι χρήστες. Φυσικά, σε μία προσομοίωση δεν μπορούν να συμπεριληφθούν όλοι οι δυνατοί προορισμοί παρά μόνο όσοι έχουν σχέση με την υπηρεσία μας. Παρακάτω παραθέτουμε τις κατηγορίες των προορισμών που προσομοιώνονται, καθώς και εκείνα τα χαρακτηριστικά τους που μοντελοποιούναι και είναι ουσιώδη για την διενέργεια της προσομοίωσης.
Οι τύποι των προορισμών(Places) που χρησιμοποιούνται είναι:

- Καφετέριες
- Internet Cafe
- Κινηματογράφοι
- Κρεπερί
- Μπαρ
- Ταβέρνες
- Σπίτια Χρηστών
- Χώροι Εργασίας Χρηστών

Για να μπορέσουμε να χρησιμοποιήσουμε τις παραπάνω κατηγορίες, θα πρέπει για κάθε μια κατηγορία να ορίσουμε κάποιες παραμέτρους διαθεσιμότητας, καθώς και τον τρόπο κατανομής τους στην ευρύτερη περιοχή της πόλης.

Παράμετροι Διαθεσιμότητας

Για κάθε κατηγορία προορισμών ορίζουμε τις παρακάτω παραμέτρους διαθεσιμότητας:

- Ώρα ανοίγματος: Η ονομαστική ώρα έναρξης του ωραρίου λειτουργίας.
- Ώρα κλεισίματος: Η ονομαστική ώρα λήξης του ωραρίου λειτουργίας.
- Διαστήματα Υψηλής (Μέσης, Χαμηλής) Κίνησης: Τα χρονικά διαστήματα κατά τα οποία η συγκεκριμένη κατηγορία προορισμών αντιμετωπίζει υψηλή (μέση, χαμηλή) επισκεψιμότητα και κίνηση.
- Πιθανότητα μη διαθεσιμότητας σε περίοδο Υψηλής (Μέσης, Χαμηλής) Κίνησης: Η πιθανότητα ο προορισμός να είναι πλήρως κατειλημμένος σε περίοδο Υψηλής (Μέσης, Χαμηλής) Κίνησης .
- Χρόνος μη διαθεσιμότητας σε περίπτωση μη διαθεσιμότητας σε περίοδο Υψηλής (Μέσης, Χαμηλής) Κίνησης: Ο μέσος χρόνος που ο προορισμός παραμένει μη διαθέσιμος σε περίπτωση μη διαθεσιμότητάς του.

Τα παραπάνω χαρακτηριστικά είναι στην πραγματικότητα τυχαίες μεταβλητές γύρω από μια μέση τιμή. Οι παραπάνω ορισθείσες παράμετροι είναι οι μέσες τιμές των τυχαίων αυτών μεταβλητών.
Για τις κατηγορίες των σπιτιών και των χώρων εργασίας, τα παραπάνω χαρακτηριστικά δεν έχουν νόημα, αφού προφανώς αυτοί οι προορισμοί είναι πάντοτε διαθέσιμοι προς χρήση.
Κατανομή των Προορισμών
Ίσως το σημαντικότερο χαρακτηριστικό μιας κατηγορίας προορισμών είναι ο τρόπος με τον οποίο κατανέμονται αυτοί οι προορισμοί στην ευρύτερη περιοχή μας. Η κατανομή αυτή έχει σημαντική επιρροή στα αποτελέσματα της προσομοίωσης μας όσον αφορά την ιδιωτικότητα των χρηστών και την αποτελεσματικότητα της υπηρεσίας μας.
Είναι προφανές ότι οι διαφορετικές κατηγορίες προορισμών θα έχουν και διαφορετικές κατανομές, όπως ακριβώς συμβαίνει και στην πραγματικότητα. Για τις ανάγκες της προσομοίωσης μας θα χρησιμοποιηθεί η κατανομή που ονομάζεται “Small Worlds”. Σ’ αυτήν την κατανομή, ορίζουμε ένα πλήθος περιοχών για κάθε κατηγορία και σε κάθε περιοχή υποθέτουμε ότι η κατανομή των προορισμών είναι Γκαουσιανή. Έτσι ορίζεται ένα πλήθος «μικρών κόσμων», στους οποίους η κατανομή είναι πυκνότερη στο κέντρο της περιοχής και αραιώνει καθώς κινούμαστε προς το εξωτερικό της.
Το πρόγραμμα της προσομοίωσης μας δίνει επίσης την δυνατότητα να χρησιμοποιήσουμε διαφορετικές κατανομές, αν αυτές ταιριάζουν καλύτερα σε κάποιο συγκεκριμένο σενάριο ή προσομοίωση. Έτσι θα μπορούσαμε να χρησιμοποιήσουμε ομοιόμορφη κατανομή σε όλη την έκταση της πόλης ή κατά περιοχές ή ακόμα και κατανομή δακτυλίου όπου η πυκνότητα των προορισμών αυξάνει από το κέντρο καθώς κινούμαστε προς το εξωτερικό της περιοχής.

3.4.3 Οι Χρήστες της υπηρεσίας

Το σημαντικότερο πρόβλημα σε μια προσομοίωση είναι να μοντελοποιηθεί με ακρίβεια η συμπεριφορά των ανθρώπων που την επηρεάζουν. Αυτό συμβαίνει γιατί ο κάθε άνθρωπος αντιδρά διαφορετικά στα διάφορα ερεθίσματα που δέχεται και πολλές φορές η συμπεριφορά του εξαρτάται από εξαιρετικά μεγάλο αριθμό παραγόντων που είναι αδύνατο να προβλεφθούν, πόσο μάλλον να προσομοιωθούν. Η προσέγγιση που ακολουθούμε είναι να εισάγουμε ένα μοντέλο συμπεριφοράς των χρηστών της

υπηρεσίας μας, το οποίο να είναι όσο το δυνατόν πλησιέστερα στην πραγματική συμπεριφορά των χρηστών.
Το πρώτο βήμα για τη δημιουργία ενός μοντέλου είναι να ξεχωρίσουμε τα ουσιώδη χαρακτηριστικά ενός χρήστη, τα οποία και θα πρέπει να οριστούν και προσομοιωθούν. Για την δική μας περίπτωση ορίζουμε τις παρακάτω παραμέτρους για κάθε χρήστη:

i. hasJob :Η παράμετρος αυτή παίρνει τιμή true για τους εργαζόμενους και τους μαθητές – φοιτητές, οι οποίοι πρέπει κάποια συγκεκριμένη ώρα της ημέρας να βρίσκονται στη δουλειά τους ή στο σχολείο. Για τους υπόλοιπους χρήστες της υπηρεσίας έχει τιμή false.
ii. Work Start Hour – Work Start Minute (Ώρα έναρξης εργασίας): Η ονομαστική ώρα στην οποία ξεκινά την εργασία του.
iii. Work Start Blur : Η ακριβής ώρα έναρξης της εργασίας είναι μια τυχαία μεταβλητή με μέση τιμή την ονομαστική ώρα έναρξης. Η μεταβλητή Work Start Blur ορίζει ένα χρονικό διάστημα γύρω από την ονομαστική ώρα έναρξης, στο οποίο κατανέμεται ομοιόμορφα η πραγματική ώρα έναρξης της εργασίας.
iv. Work Duration (Διάρκεια εργασίας): Η ονομαστική διάρκεια της εργασίας του.
v. Work Duration Blur: Ορίζει ένα χρονικό διάστημα γύρω από την Work Duration στο οποίο κατανέμεται ομοιόμορφα η τυχαία μεταβλητή της πραγματικής διάρκειας εργασίας.
vi. Location Privacy – Preferences Privacy : Οι προτιμήσεις του χρήστη σχετικά με την προστασία της ιδιωτικότητάς του.
vii. Βάρος Κόστους Υπηρεσίας (PriceWeightPref) : Η σημασία που δίνει ο χρήστης στο κόστος των υπηρεσιών που προσφέρει ο προτεινόμενος προορισμός.
viii. Βάρος Ποιότητας Υπηρεσίας (QualityWeightPref) : Η σημασία που δίνει ο χρήστης στην ποιότητα των υπηρεσιών που προσφέρει ο προτεινόμενος προορισμός.
ix. Βάρος Εγγύτητας Προορισμού (distWeightPref) : Η σημασία που δίνει ο χρήστης στο πόσο κοντά βρίσκεται ο προτεινόμενος προορισμός.
x. Ασχολίες – Δραστηριότητες Χρήστη (Occupations) : Οι ασχολίες με τις οποίες αρέσκεται να διασκεδάζει ο χρήστης.

Για κάθε ασχολία του χρήστη ορίζουμε κάποια επιπλέον χαρακτηριστικά, τα οποία φαίνονται παρακάτω:

i. Τόπος που λαμβάνει χώρα η ασχολία: Με ποιον ή ποιους από τους πιθανούς προορισμούς συνδέεται η συγκεκριμένη ασχολία.
ii. Μέση Διάρκεια Ασχολίας
iii. Duration Blur : Το χρονικό διάστημα γύρω από τη μέση διάρκεια στο οποίο κατανέμεται ομοιόμορφα η τυχαία μεταβλητή της πραγματικής διάρκειας της ασχολίας

iv. Μέση τιμή ώρας έναρξης της ασχολίας (centerHour – centerMinutes) και variance ώρας έναρξης: Υποθέτουμε ότι η ώρα έναρξης της ασχολίας ακολουθεί κατανομή Gauss με παραμέτρους σ (μέση τιμή ώρας έναρξης) και μ(μεταβλητότητα).

Για την δημιουργία ενός όσο το δυνατόν ευέλικτου και προσαρμόσιμου μοντέλου, χωρίζουμε τον συνολικό πληθυσμό των χρηστών σε ομάδες με παραπλήσια χαρακτηριστικά (μαθητές, συνταξιούχους, εργαζόμενους, νοικοκυρές κλπ). Υποθέτουμε δηλαδή ότι δύο άνθρωποι της ίδιας ομάδας έχουν χοντρικά την ίδια συμπεριφορά με μικρές μεταβολές.

Συμπεριφορά Χρηστών

Αφού ορίσαμε τα χαρακτηριστικά των χρηστών, επόμενο βήμα είναι να ορίσουμε την συμπεριφορά τους σε διάφορες καταστάσεις. Η γενική συμπεριφορά των χρηστών θα είναι η ίδια και το μόνο που θα τους διαφοροποιεί θα είναι τα ιδιαίτερα χαρακτηριστικά τους που θα αλλάζουν από χρήστη σε χρήστη.
Ξεκινάμε υποθέτοντας ότι κάποιος χρήστης θέλει κάποια χρονική στιγμή μέσα στην ημέρα να διασκεδάσει. Ο τύπος διασκέδασης που θα επιλέξει εξαρτάται από τις προσωπικές του προτιμήσεις, καθώς και από την ώρα της ημέρας. Με βάση τα δύο παραπάνω επιλέγεται ένας τύπος χώρου διασκέδασης.
Επόμενο ερώτημα που απασχολεί τον χρήστη είναι το ποιος από όλους τους χώρους διασκέδασης του τύπου που έχει επιλέξει ταιριάζει καλύτερα σε κείνον, ενώ παράλληλα βρίσκεται και κοντά του. Οι επιλογές του χρήστη είναι οι εξής: είτε επιλέγει από μόνος του κάποιον γνωστό προορισμό, είτε ζητά πληροφορίες από την υπηρεσία μας. Το ποια από τις δύο επιλογές επιλέγει είναι θέμα του ίδιου του χρήστη. Έχουμε λοιπόν δύο ενδεχόμενα, με την πιθανότητα του καθενός να εξαρτάται από τα χαρακτηριστικά του εν λόγω χρήστη.
Αν ο χρήστης επιλέξει από μόνος του προορισμό, τότε κατευθύνεται προς τα εκεί. Από την άλλη αν επιλέξει να συμβουλευτεί την υπηρεσία, θα πρέπει να στείλει τα δεδομένα που περιγράφτηκαν στην παράγραφο…. Επίσης, επιλέγει και τον βαθμό προστασίας ιδιωτικότητας θέσης και προτιμήσεων που θέλει να έχει. Ο τελευταίος εξαρτάται από τα χαρακτηριστικά του χρήστη και θεωρείται σταθερός για όλη την διάρκεια της προσομοίωσης. Αφού αποσταλούν τα δεδομένα, λαμβάνεται από τον χρήστη η λίστα των προτεινόμενων από την υπηρεσία προορισμών ταξινομημένη κατά φθίνουσα σειρά ταιριάσματος με τις προτιμήσεις του χρήστη. Μετά την λήψη των αποτελεσμάτων από το κινητό τερματικό, υπάρχει η δυνατότητα δεύτερης επεξεργασίας από τη συσκευή με βάση το τοπικό προφίλ του χρήστη και εύρεση του βέλτιστου προορισμού ή επιλογή από τον χρήστη ενός από τους προτεινόμενους χώρους διασκέδασης. Μετά την τελική επιλογή προορισμού, ο χρήστης κατευθύνεται προς τον προορισμό αυτό. Αν για κάποιο λόγο, η υπηρεσία αδυνατεί να βρει προορισμούς που να ταιριάζουν με τον χρήστη, τότε ο χρήστης είτε επιλέγει από μόνος του, είτε επιλέγει να περιπλανηθεί για λίγο και να ξαναπροσπαθήσει αργότερα.
Φτάνοντας στον επιλεγμένο προορισμό, δεν είναι βέβαιο ότι ο χρήστης θα εξυπηρετηθεί. Ο κάθε προορισμός έχει για κάθε ώρα της ημέρας ένα επίπεδο κίνησης και μια συγκεκριμένη πιθανότητα να μην είναι διαθέσιμος. Εξαίρεση αποτελούν τα σπίτια και οι χώροι εργασίας των χρηστών που είναι πάντοτε διαθέσιμοι. Στην περίπτωση που ο προορισμός δεν είναι διαθέσιμος, ο χρήστης έχει δύο επιλογές. Είτε περιμένει κάποιο χρονικό διάστημα, το οποίο είναι διαφορετικό για κάθε χρήστη και καθορίζεται από τα χαρακτηριστικά του, με την ελπίδα σ’ αυτήν την χρονική περίοδο

να ξαναγίνει διαθέσιμος ο προορισμός, είτε επιλέγει να μετακινηθεί αλλού. Ο νέος προορισμός επιλέγεται είτε από τον χρήστη, είτε ζητώντας βοήθεια από την υπηρεσία. Αν παρόλο που επιλέξει να περιμένει ο χώρος δεν γίνει ξανά διαθέσιμος στο χρονικό διάστημα της αναμονής, τότε πάλι επιλέγει να κινηθεί σε νέο χώρο. Οι πιθανότητες μη διαθεσιμότητας, καθώς και οι αντίστοιχοι χρόνοι μη διαθεσιμότητας, εξαρτώνται από την ώρα της ημέρας και από την κατανομή της κίνησης για κάθε τύπο προορισμού και αποτελούν χαρακτηριστικά του συγκεκριμένου τύπου. Ακόμη όμως και διαθέσιμος να είναι ο προορισμός υπάρχει πάντα η πιθανότητα να συμβεί κάτι αναπάντεχο και ο χρήστης να χρειαστεί να αποχωρήσει από το χώρο. Κάτι τέτοιο μπορεί να είναι μια ανεπιθύμητη συνάντηση, ένα επείγον τηλεφώνημα ή ακόμη μια ξαφνική αδιαθεσία. Η πιθανότητα αυτή είναι μικρή, πλην όμως υπολογίζεται και λαμβάνεται υπόψη στην προσομοίωσή μας.

Στην πιθανότερη των περιπτώσεων, ο χρήστης βρίσκει τον προορισμό διαθέσιμο και ξεκινά την ασχολία του. Η κατανομή του χρόνου που αφιερώνει ένας χρήστης για μια συγκεκριμένη ασχολία εξαρτάται από τα χαρακτηριστικά του χρήστη και από το είδος της ασχολίας. Σαν παράδειγμα μπορούμε να αναφέρουμε ότι διαφορετικό χρόνο για καφέ αφιερώνει ένας φοιτητής από έναν εργαζόμενο και επίσης διαφορετικός χρόνος ξοδεύεται σε έναν κινηματογράφο απ' ότι σε μια κρεπερί. Αφού παρέλθει ο χρόνος της ασχολίας ο χρήστης είναι σε θέση να διαλέξει τον νέο του προορισμό.

Ανάμεσα στους πιθανούς προορισμούς των χρηστών βρίσκονται και τα σπίτια τους και οι χώροι εργασίας τους. Κάθε χρήστης αντιστοιχίζεται στην αρχή της προσομοίωσης με κάποιο από τα διαθέσιμα σπίτια της προσομοίωσης, ενώ το ίδιο γίνεται και για τους χρήστες που διαθέτουν εργασία για τους χώρους εργασίας. Οι χρήστες γυρίζουν στα σπίτια τους για ασχολίες όπως η ξεκούραση και ο ύπνος, ενώ συγκεκριμένες ώρες της ημέρας πρέπει να βρίσκονται στους χώρους εργασίας. Δεδομένου ότι η παρουσία στους χώρους εργασίας είναι υποχρεωτική, κάθε χρήστης αφήνει ό,τι κάνει και κατευθύνεται προς τα κει όταν η ώρα την ημέρας το επιβάλλει.

Το μοντέλο που χρησιμοποιείται για την συμπεριφορά των χρηστών λαμβάνει υπόψη σε ποια κατηγορία ανήκει ο χρήστης για να καθορίσει το επίπεδο αφαίρεσης / αοριστίας στο οποίο θα σταλούν τα δεδομένα στην υπηρεσία. Για τις ανάγκες της προσομοίωσης όμως έχουμε την δυνατότητα να ορίσουμε εμείς το επίπεδο λεπτομέρειας στο οποίο θα αναφέρονται οι προτιμήσεις και οι θέσεις όλων των χρηστών, ώστε να βγάλουμε συμπεράσματα για το πώς επηρεάζει το επίπεδο προστασίας ιδιωτικότητας την αποτελεσματικότητα της υπηρεσίας.

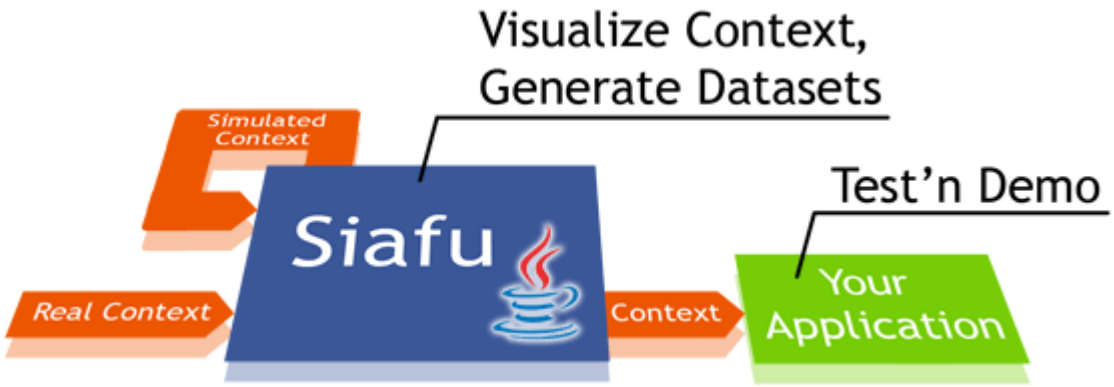
4. Υλοποίηση της προσομοίωσης με τον προσομοιωτή Siafu

4.1. Εισαγωγή

Η δοκιμή και αξιολόγηση των mobile context-aware εφαρμογών και υπηρεσιών καθίσταται ιδιαίτερα δύσκολη λόγω της πολυπλοκότητας που συνοδεύει τη συλλογή και επεξεργασία contextual data. Επίσης, η αξιολόγηση αλγορίθμων μηχανικής μάθησης (machine learning) σε περιβάλλον context-aware εφαρμογών γίνεται ακόμη πιο δύσκολη λόγω της έλλειψης τυποποιημένων συνόλων δεδομένων και εργαλείων προσομοίωσης.
Τα εργαστήρια της NEC ανέπτυξαν έναν γενικό προσομοιωτή επικεντρωμένο στους προαναφερθείς σκοπούς [15]. Ο προσομοιωτής αυτός έχει αποδειχτεί ένα καλό εργαλείο επίδειξης κινητών υπηρεσιών και εφαρμογών που απευθύνονται σε ομάδες χρηστών. Επίσης, έχει αναπτυχθεί έτσι ώστε να είναι ιδιαίτερα προσαρμόσιμος και μπορεί να κάνει την απεικόνιση πληροφοριών context καθώς και μεμονωμένων οντοτήτων, όχι μόνο μέσω ενός διαδραστικού γραφικού περιβάλλοντος, αλλά και μέσω αρχείων .csv (Comma Separated Values). Με στόχο την υποστήριξη μεγάλου εύρους καθηκόντων και σεναρίων οι πληροφορίες χωρίζονται σε 3 κύριες πηγές:

- Η συμπεριφορά των agents.
- Το προσομοιωμένο σενάριο.
- Το περιβάλλον των context-variables.

Για την υλοποίηση του προσομοιωτή έχει χρησιμοποιηθεί η γλώσσα προγραμματισμού Java.



Εικόνα 4.1: Προσομοιωτής αναγνώρισης περιβάλλοντος Siafu (Context Simulator Siafu)

Η αποτελεσματική δοκιμή mobile context-aware εφαρμογών και υπηρεσιών είναι προς το παρόν ένα δύσκολο έργο. Η ετερογενής φύση των data sources και των αισθητήρων είναι ένας εκ των κύριων λόγων καθώς καθιστά δύσκολη τη συγκέντρωση και την επεξεργασία των context variables. Επίσης, η συλλογή μεγάλης ποσότητας δεδομένων δεν είναι δυνατή λόγω της έλλειψης άμεσα διαθέσιμου context infrastructure, ενώ η συλλογή από αισθητήρες μετρήσεων οι οποίες δεν έχουν επεξεργαστεί σπάνια βοηθάει το χρήστη.
Αντίθετα θα πρέπει να αντλήσουμε υψηλότερου επίπεδου συμπεράσματα από τα δεδομένα. Για παράδειγμα, θα πρέπει να χρησιμοποιήσουμε τις GPS μετρήσεις ώστε

να ανακαλύψουμε προορισμούς που έχουν νόημα για το χρήστη. Ωστόσο, η δυσκολία της αξιολόγησης καθιστά την εξαγωγή τέτοιων συμπερασμάτων πραγματική πρόκληση.
Τα παραπάνω προβλήματα δεν παρατηρούνται μόνο σε context-aware εφαρμογές αλλά εμφανίζονται σε όλα τα περίπλοκα συστήματα. Γι' αυτό το σκοπό οι ερευνητές αυτών των πεδίων δημιουργούν εργαλεία προσομοίωσης με σκοπό να μοντελοποιήσουν το πρόβλημα.

Παρ' όλ' αυτά, η ιδέα της χρησιμοποίησης προσομοίωσης σε context-aware computing είναι σχετικά νέα. Μία πρόσφατη ερευνητική κατεύθυνση είναι η προσομοίωση low level sensors. Για παράδειγμα στην αναφορά [17] μπορούμε να δούμε την προσομοίωση μετρήσεων αισθητήρων σε έναν "έξυπνο" χώρο. Άλλη προσέγγιση είναι ο συνδυασμός προσομοίωσης εικονικής πραγματικότητας μαζί με προσομοίωση φυσικών αισθητήρων. Τέτοια παραδείγματα είναι οι προσομοιωτές QuakeSim [18], [19] και το UbiWISE [20]. Η τρίτη κατεύθυνση είναι η χρησιμοποίηση multi-agent προσομοίωσης. Παράδειγμα αυτής της προσέγγισης είναι ο Generic Location Event Simulator [21] ο οποίος δημιουργεί context-dependent events, τα οποία μπορούν να χρησιμοποιηθούν για τον έλεγχο της λειτουργικότητας ενός middleware. Ο προσομοιωτής Siafu έχει 2 κύριες διαφορές από προηγούμενες εργασίες. Κατά πρώτο λόγο, έχει υιοθετήσει καλά τεκμηριωμένες τεχνικές από το πεδίο της προσομοίωσης. Ενώ κατά δεύτερον, έχει σχεδιαστεί έτσι ώστε να είναι αρκετά γενικός και συνεπώς εφαρμόσιμος σε μεγάλο εύρος σεναρίων.

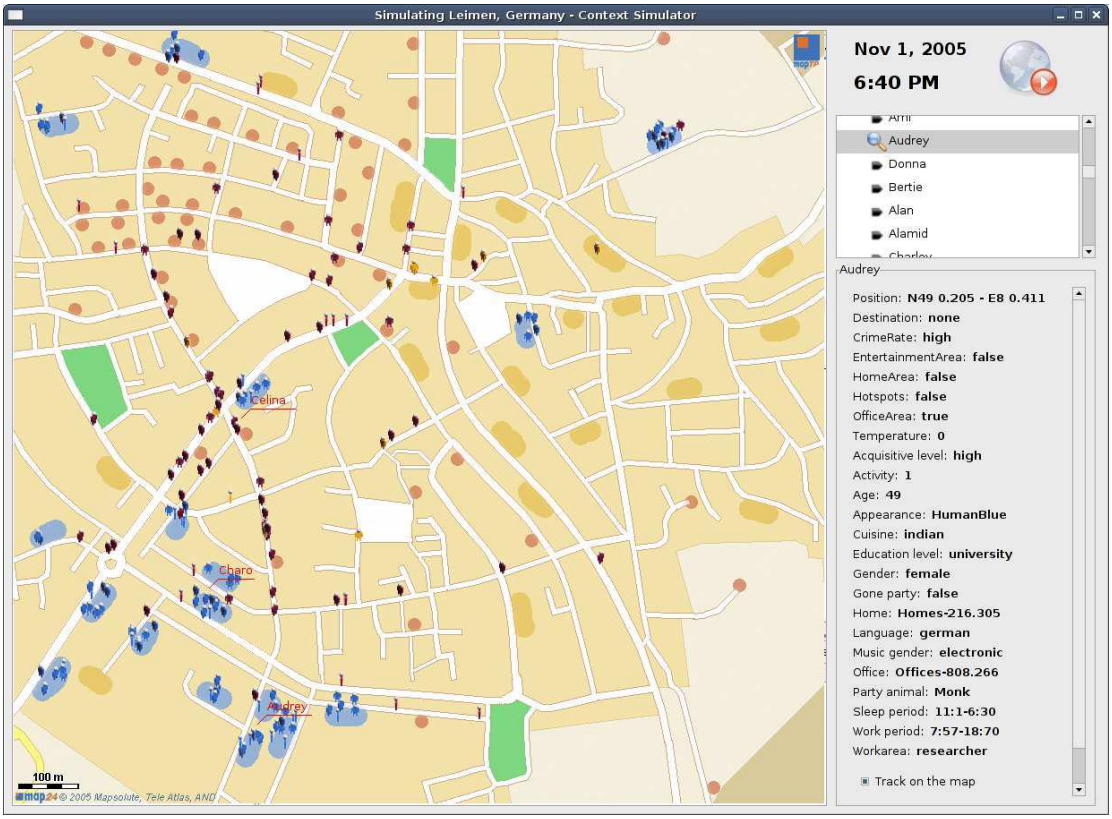
4.2 Προσομοίωση

Για να καταστήσουν τον Siafu έναν ευέλικτο προσομοιωτή, οι κατασκευαστές του προχώρησαν στον διαχωρισμό των πηγών πληροφορίας. Ο διαχωρισμός αυτός έγινε ως εξής:

- Η συμπεριφορά των agents εκφράζεται στο ξεχωριστό agent model.
- Το περιβάλλον (το οποίο και περιλαμβάνει πιθανά τυχαία γεγονότα) εκφράζεται στο world model.
- Τα context data προσομοιώνονται στο πρόγραμμα από το context model.

Τα τρία αυτά μέρη αναλύονται περιληπτικά στις επόμενες ενότητες.

Εικόνα 4.2: Οι πληροφορίες κατάστασης (context information) για τον χρήστη Alan [15]



Εικόνα 4.3: Παράδειγμα οθόνης προσομοίωσης επίγνωσης περιβάλλοντος [15]

Στην εικόνα 2 μπορούμε να δούμε τον προσομοιωτή καθώς και το information panel που οπτικοποιεί τις πληροφορίες context.

4.2.1. Μοντέλο Χρήστη (Agent Model)

Όπως αναφέραμε η πρώτη πηγή πληροφοριών είναι το agent model που αποτελεί την λογική αποφάσεων του χρήστη. Δοθέντος του τρέχοντος περιβάλλοντος (context) και της κατάστασης των οντοτήτων, όπως για παράδειγμα των places και των ανθρώπων γύρω του, το agent model αποφασίζει τι θα πρέπει να κάνει ο agent. Το περιβάλλον ενός agent παρέχεται από το context model, ενώ οι πληροφορίες σχετικά με την κατάσταση των γύρω οντοτήτων δίνονται από το world model.
Βασική ιδιότητα του agent model είναι η αλλαγή των χαρακτηριστικών ενός agent. Για παράδειγμα μπορεί να μετατρέψει τη δραστηριότητα του agent από sleeping σε working ή να αλλάξει τη διάθεσή του από χαρούμενο σε λυπημένο. Άλλη μία ιδιότητα του agent model είναι η αλλαγή του προορισμού ενός agent, έτσι, ενώ η κίνηση και η δρομολόγηση του agent γίνεται αυτόματα από τον προσομοιωτή, η απόφαση του πού θα πάει βαραίνει το agent model.
Η αλλαγή της συμπεριφοράς ενός agent από τον σχεδιαστή της προσομοίωσης γίνεται με την ανάθεση ενός διαφορετικού agent model στον agent. Επειδή η δημιουργία πολύπλοκων προσομοιώσεων με μεγάλο αριθμό χρηστών θα απαιτούσε την δημιουργία χρηστών πάντα από την αρχή, επιτρέπουμε τη χρήση template η οποία και απλοποιεί αρκετά την διαδικασία αυτή. Ένα τέτοιο template είναι ένα στερεότυπο μοντέλο χρήση, σε αυτό το agent model προσδιορίζεται σε υψηλό επίπεδο αφαίρεσης. Τα templates αυτά μπορούν να μοντελοποιηθούν σαν μηχανές κατάστασης με την αλλαγή κατάστασης να πυροδοτείται απο αλλαγές στο περιβάλλον, είτε προαιρετικά,

από τυχαίους παράγοντες

4.2.2. Μοντέλο Κόσμου (World Model)

Η επόμενη πηγή πληροφοριών είναι το world model το οποίο αποτελείται από τρία μέρη: πρώτον το περιβάλλον που θέλουμε να προσομοιώσουμε, δεύτερον οι προορισμοί ενδιαφέροντος και τέλος τα global events. Μπορούμε για παράδειγμα να χρησιμοποιήσουμε ένα αστικό σενάριο, σε αυτό το περιβάλλον είναι η πόλη που προσομοιώνουμε, οι προορισμοί θα μπορούσαν να είναι γραφεία, διαμερίσματα και χώροι διασκέδασης, ενώ τέλος, τα global events θα μπορούσαν να περιλαμβάνουν φεστιβάλ ή περιόδους διακοπών.
Το world model προσδιορίζεται από ένα σύνολο image files τα οποία απαιτούνται ακόμα και αν το γραφικό περιβάλλον δεν χρησιμοποιείται. Το βασικό σύνολο των image files αποτελείται από ένα background map και από έναν alpha map. Ο alpha map ορίζει τις περιοχές που είναι προσβάσιμες, δηλαδή τα σημεία που δεν είναι τοίχοι. Κατά την αρχή της προσομοίωσης ο προσομοιωτής χρησιμοποιεί τον alpha map με τον οποίο προϋπολογίζει το gradient map που χρησιμοποιείται για την εύρεση του μονοπατιού. Τα gradient maps αποθηκεύονται στο δίσκο και ξαναυπολογίζονται μόνο εάν το περιβάλλον της προσομοίωσης αλλάξει. Τα περισσότερα από τα places, όπως για παραδειγμα τα γραφεία και τα διαμερίσματα, είναι παρόντα σε όλη τη διάρκεια της προσομοίωσης. Παρ' όλ' αυτά κάποια places, όπως για παράδειγμα τα meeting places ή pickup locations, δημιουργούνται για ένα προσωρινό μόνο χρονικό διάστημα και εξαφανίζονται όταν δε χρειάζονται πλέον.
Οι ιδιότητες των places ορίζονται από key-value pairs τα οποία μπορούν να τροποποιηθούν σε κάθε επανάληψη. Για παράδειγμα, ένα nightclub μπορεί να προγραμματίσει ένα special party ή ένα εστιατόριο μπορεί να ορίσει ειδικές προσφορές. Επίσης, οι agents έχουν τη δυνατότητα να δημιουργήσουν νέα places ή να αφαιρέσουν κάποια από τα ήδη υπάρχοντα. Οι ενέργειες αυτές συμβαίνουν με ντετερμινιστικό τρόπο, έτσι όταν ικανοποιούνται οι συνθήκες που έχουν ορισθεί στο agent model, τότε ένα νέο place θα δημιουργείται ή ένα παλιό θα αφαιρείται. Για παράδειγμα όταν το agent model υποδηλώνει ότι θα πρέπει να προγραμματισθεί ένα meeting, τότε θα δημιουργείται ένα meeting place και το meeting place θα αφαιρείται από την προσομοίωση μόλις το meeting τελειώσει.
Όπως και στα agent models, έτσι και στο world model, το Siafu παρέχει τη δυνατότητα χρήσης templates από χαρακτηριστικά κοινά για όλα τα places, αλλά και χαρακτηριστικά ειδικά για κάθε τύπο place, στοχεύοντας στην απλοποίηση της διαδικασίας δημιουργίας ενός world model. Επιπρόσθετα το template ορίζει πώς και πότε τα places θα αλλάξουν τις ιδιότητές τους (properties). Για παράδειγμα template μπορούμε να δούμε τα nightclub και τα γραφεία των εργαζόμενων μιας εταιρίας. Για ένα nightclub οι αρχικές ιδιότητες ορίζουν το είδος μουσικής που παίζει, ενώ το event model ορίζει πώς και πότε το place προγραμματίζει special events. Από την άλλη μεριά, για ένα γραφείο χρειάζεται μόνο να ορίσουμε τη θέση του γραφείου στο κτίριο της εταιρείας. Τα places που ακολουθούν ένα template μπορούν να πάρουν υπόσταση από image maps τα οποία υποδηλώνουν το που θα βρίσκεται το κάθε place. Εν συνεχεία ο προσομοιωτής δημιουργεί τα places στις τοποθεσίες που ορίζονται από το image map και θέτει τις ιδιότητες του κάθε place στις default τιμές τους.

4.2.3. Μοντέλο Πλαισίου (Context Model)

Όπως είδαμε η τελευταία πηγή πληροφοριών είναι το context model, το οποίο διαχειρίζεται τις context variables που χρησιμοποιούνται στην προσομοίωση. Για

κάθε context variable, το context model προσδιορίζει τις πιθανές τιμές της μεταβλητής, το μοντέλο που χρησιμοποιείται για την προσομοίωση των τιμών και δυνητικά πώς οι τιμές αυτές κατανέμονται χωρικά πάνω στο περιβάλλον. Για παράδειγμα μπορούμε να θεωρήσουμε ένα context model το οποίο περιέχει δύο μεταβλητές θερμοκρασία και WiFi hotspot coverage. Για τη θερμοκρασία, το context model θα μπορούσε να προσδιορίζει τη μέση θερμοκρασία σε συγκεκριμένες περιοχές καθώς επίσης και μια κατανομή των τιμών. Από την άλλη, για τα WiFi hotspots το context model θα μπορούσε να συμπεριλάβει τις θέσεις των base stations, την ισχύ σήματος των base stations και την απόσταση από τα base stations. Εναλλακτικά, η τιμή μιας context variable μπορεί να διαβαστεί από εξωτερικούς αισθητήρες.

Το Siafu υποστηρίζει annotated bitmaps για την εύκολη εισαγωγή, αρχικοποίηση και ενημέρωση των context variables,. Για τη δημιουργία μιας νέας context variable, ο σχεδιαστής της προσομοίωσης χρειάζεται να παρέχει στον προσομοιωτή ένα overlay image map και ένα calibration file. Αυτά χρησιμοποιούνται από τον προσομοιωτή για τον υπολογισμό μιας αντιστοίχησης που συσχετίζει pixel values με συγκεκριμένες τιμές της context variable σε κάθε σημείο του προσομοιωμένου περιβάλλοντος. Μόλις γίνει αυτός ο υπολογισμός η αντιστοίχηση αποθηκεύεται σε ένα αρχείο. Για ένα παράδειγμα μπορούμε να θεωρήσουμε την προσομοίωση της θερμοκρασίας μιας πόλης. Εδώ ο σχεδιαστής της θα μπορούσε αρχικά να παράγει ένα fractal cloud image. Εν συνεχεία πρέπει η πυκνότητα των τιμών της εικόνας να προσαρμοστεί ώστε στις περιοχές που απεικονίζουν εσωτερικό κτηρίων να υπάρχει ομοιόμορφη θερμοκρασία. Μετά, ο προσομοιωτής υπολογίζει την αντιστοιχία μεταξύ περιοχής και θερμοκρασίας και μόλις ο προσομοιωτής φορτωθεί, η έξοδος περιλαμβάνει και την θερμοκρασία στις θέσεις των agents. Στην εικόνα 3 βλέπουμε τα overlays της θερμοκρασίας και της WiFi hotspot coverage μετά από υπέρθεση με το χάρτη της προσομοίωσης.

(a) Hotspot range (b) Temperature

Εικόνα 4.5: Μεταβλητές πλαισίου (context variables)

4.3 Κλάσεις του Siafu

Μια λεπτομερής ανάλυση των κλάσεων του Siafu μπορεί να βρεθεί στο [37]. Στην παρούσα ενότητα θα παρουσιάσουμε μόνο εκείνες τις κλάσεις που έχουν τροποποιηθεί ή συμπληρωθεί για τις ανάγκες της προσομοίωσής μας.

4.3.1 Agent Class

Τα αντικείμενα της κλάσης Agent αναπαριστούν τους χρήστες της υπηρεσίας στην προσομοίωσή μας. Στα επόμενα αναλύονται οι μεταβλητές, καθώς και οι μέθοδοι της κλάσης.

Μεταβλητές:

world: Το αντικείμενο World της προσομοίωσης. Όλοι οι agents θα πρέπει να ανήκουν στον ίδιο κόσμο γι' αυτό η μεταβλητή αυτή ορίζεται ως static μεταβλητή.

onAuto: Μεταβλητή τύπου boolean η τιμή της οποίας καθορίζει την κατάσταση του agent. Αν έχει την τιμή true, τότε ο χρήστης συμπεριφέρεται και κινείται σύμφωνα με το προγραμματισμένο μοντέλο συμπεριφοράς που περιγράφεται στην κλάση AgentModel, ενώ αν είναι false, τότε καθορίζουμε εμείς από το περιβάλλον της προσομοίωσης το που και το πώς θα κινηθεί.

name: Μεταβλητή τύπου String,η οποία περιέχει το όνομα του χρήστη.

dir: Μεταβλητή τύπου integer με δυνατές τιμές από 0-7 που αντιστοιχούν στις πιθανές κατευθύνσεις του agent. Το 0 αντιστοιχεί στο Νότο, ενώ οι υπόλοιπες κατευθύνσεις αντιστοιχούν σε στροφή κατά 45^{o} αντί-ωρολογιακά μέχρι το 7 που αντιστοιχεί σε Νότιο-Δυτική διεύθυνση.

speed: Μεταβλητή τύπου integer που παριστάνει την ταχύτητα του agent εκφρασμένη σε pixels/βήμα προσομοίωσης.

pos: Ένα αντικείμενο τύπου Position που εκφράζει τη θέση του agent πάνω στο χάρτη τη συγκεκριμένη χρονική στιγμή.

destination: Ένα αντικείμενο τύπου Place που εκφράζει τον τρέχοντα προορισμό προς τον οποίο κατευθύνεται ο agent.

atDestination: Μεταβλητή τύπου boolean που γίνεται true όταν ο agent φτάσει στον προορισμό που δηλώνει η μεταβλητή **destination.** Σε αντίθετη περίπτωση έχει την τιμή false.

Dest: Ένα αντικείμενο τύπου Place που περιέχει τον χώρο στον οποίο βρίσκεται ο agent την συγκεκριμένη χρονική στιγμή. Αν είναι στον δρόμο, τότε η τιμή της μεταβλητής γίνεται null.

willWait: Μεταβλητή τύπου Boolean που καθορίζει το αν ο agent περιμένει σε περίπτωση που κάποιος προορισμός είναι μη διαθέσιμος ή αναζητά κάποιον άλλο προορισμό.

avoidDest: Μεταβλητή τύπου Place, η οποία περιέχει κάποιον προορισμό που ο agent πρέπει να αποφύγει να διαλέξει. Χρησιμοποιείται για προορισμούς που ήταν μη διαθέσιμοι όταν ο χρήστης τους επισκέφτηκε και επομένως πρέπει να τους αποφύγει στην νέα αναζήτηση προορισμού που θα κάνει.

preferencesArray[]: Ένας πίνακας με στοιχεία τύπου Node ο οποίος περιέχει την προτίμηση του χρήστη για την κατηγορία ταινιών που θέλει. Η προτίμηση αποθηκεύεται σαν μια αλληλουχία κόμβων δέντρου, από την ρίζα του δέντρου μέχρι το φύλλο που περιέχει την ακριβή προτίμηση του χρήστη.

image: Μεταβλητή τύπου String που παρέχει μια αναφορά στο αρχείο που περιέχει το sprite με το οποίο θα εμφανίζεται ο agent πάνω στο χάρτη.

previousImage: Μεταβλητή τύπου String που περιέχει το όνομα του προηγούμενου image του χρήστη. Χρησιμοποιείται σε περιπτώσεις που θέλουμε ο agent να απεικονίζεται με δύο διαφορετικά sprites στον χάρτη ανάλογα με τις επικρατούσες συνθήκες. Για παράδειγμα θα μπορούσαμε να απεικονίζουμε διαφορετικά τον χρήστη πεζό και διαφορετικά όταν μετακινείται με κάποιο μέσο μεταφοράς.

visible: Μια μεταβλητή τύπου Boolean που καθορίζει το αν ο agent θα εμφανίζεται στον χάρτη της προσομοίωσης. Αν είναι true τότε ο χρήστης καθίσταται ορατός, ενώ για την τιμή false ο agent δεν εμφανίζεται παρόλο που συνεχίζει κανονικά τις δραστηριότητές του.

Τα χαρακτηριστικά του agent τα οποία είναι ιδιαίτερα για κάθε προσομοίωση και διαφέρουν από προσομοίωση σε προσομοίωση ανάλογα με το φαινόμενο που μελετάμε αποθηκεύονται στη μεταβλητή **info.**

info: Μια μεταβλητή τύπου SortedMap<String, Publishable> με την οποία μπορούμε να αποθηκεύουμε χαρακτηριστικά γνωρίσματα των agents. Το όνομα κάθε χαρακτηριστικού είναι μια μεταβλητή τύπου String, ενώ η τιμή του μπορεί να είναι κάθε αντικείμενο που υλοποιεί το Interface Publishable.

infoFieldsLocked: Μια static μεταβλητή τύπου boolean η οποία γίνεται true μόλις επιστρέψουμε από τη μέθοδο createAgents() της κλάσης World. Δηλώνει ότι δεν μπορούμε να εισάγουμε νέα χαρακτηριστικά στους agents μετά την αρχικοποίησή τους.

Μέθοδοι:

Οι μέθοδοι της κλάσης Agent είναι στην πλειοψηφία τους μέθοδοι πλοήγησης του χρήστη και μέθοδοι αλλαγής και προσπέλασης των μεταβλητών υπόστασης του χρήστη (get – set). Οι κυριότερες από αυτές παρουσιάζονται παρακάτω.

moveTowardsPlace():Δέχεται ως όρισμα ένα Place το οποίο αποτελεί τον προορισμό του agent. Χρησιμοποιεί το σύστημα πλοήγησης του Siafu που περιγράφεται στην κλάση Gradient και σε κάθε θέση του χρήστη αποφασίζει την κατεύθυνση κίνησης του .

moveTowardsDestination(): Καλεί την **moveTowardsPlace()** με όρισμα την μεταβλητή **destination** του εκάστοτε agent.

wander(): Η μέθοδος αυτή κάνει τον agent να περιπλανιέται τυχαία. Δεν κατευθύνεται προς ένα συγκεκριμένο προορισμό, αλλά γίνεται τυχαία επιλογή της κατεύθυνσης σε κάθε βήμα του χρήστη. Η ταχύτητα της περιπλάνησης είναι 1 pixel/βήμα, ενώ ο agent αλλάζει κατεύθυνση με πιθανότητα 1/soberness, όπου soberness είναι μια παράμετρος που δηλώνει τη νηφαλιότητα του χρήστη. Όσο μεγαλύτερη είναι η παράμετρος αυτή, τόσο πιο ευθύγραμμα θα κινείται ο agent όταν περιπλανιέται. Όταν ο agent πέσει σε τοίχο αναγκαστικά αλλάζει κατεύθυνση. Η επιλογή της νέας κατεύθυνσης γίνεται με διαδοχική στροφή κατά 135^{o} έως ότου στη νέα κατεύθυνση ο agent να μπορεί να κινηθεί και το επόμενό του βήμα να μην τον ρίχνει σε τοίχο.

wanderAround(): Παρόμοια μέθοδος με τη μέθοδο **wander()** με την διαφορά ότι εδώ περιορίζουμε τον agent να κινείται σε μια περιοχή με συγκεκριμένο κέντρο και ακτίνα. Σαν παραμέτρους περνάμε το κέντρο της περιοχής που είναι τύπου Place, την ακτίνα της περιοχής σε pixel, καθώς και την «νηφαλιότητα» στην κίνηση του χρήστη με την παράμετρο soberness. Ο χρήστης περιπλανιέται όπως στη μέθοδο **wander()** με τη διαφορά ότι όταν φτάσει εκτός ορίου περιοχής, αντί να βγει έξω από την περιοχή, καλείται η μέθοδος **moveTowardsPlace()** με τιμή παραμέτρου το κέντρο της περιοχής, οπότε και ο χρήστης κατευθύνεται προς τον προορισμό που βρίσκεται στο κέντρο.

4.3.2 Place Class

Τα Places ορίζουν ένα προορισμό στον κόσμο της προσομοίωσης. Αντίθετα με τα αντικείμενα της κλάσης Position που απλά ορίζουν τις συντεταγμένες ενός σημείου πάνω στο χάρτη της προσομοίωσης, τα places μπορούν να έχουν τις δικά τους χαρακτηριστικά. Τα χαρακτηριστικά αυτά μπορεί να είναι κοινά για όλα τα places, ή να διαφέρουν ανάλογα με τον τύπο του place. Επίσης υπάρχουν μεγέθη που διαθέτουν τα places σε κάθε προσομοίωση και άλλα που προστίθενται στα places ειδικά για τις ανάγκες μιας προσομοίωσης. Τα standard χαρακτηριστικά παριστάνονται με μεταβλητές κλάσης και υπόστασης ενώ για τα extra υπάρχει μια μεταβλητή υπόστασης TreeMap<String, Publishable> info στην οποία μπορούμε να αντιστοιχήσουμε για κάθε ένα χαρακτηριστικό, την τιμή του χαρακτηριστικού για αυτό το αντικείμενο Place.

Μεταβλητές

***world*:** ένα αντικείμενο τύπου World που αντιστοιχεί στον κόσμο της προσομοίωσης. Όλα τα αντικείμενα της κλάσης Place θα πρέπει να ανήκουν στον ίδιο κόσμο γι' αυτό η μεταβλητή αυτή δηλώνεται static. Όλοι οι constructor της κλάσης Place παίρνουν σαν όρισμα μια παράμετρο World και ελέγχουν αν η παράμετρος αυτή έχει πάντα την ίδια τιμή. Η αρχικοποίηση της static world γίνεται με τη μέθοδο `initialize()` που καλείται μία μόνο φορά ως πρώτη εντολή της μεθόδου `createPlaces()` της κλάσης World. Η πρώτη εντολή κάθε constructor είναι η κλήση της μεθόδου `basicChecks()` που ελέγχει εάν η παράμετρος world του constructor έχει την ίδια τιμή με την static variable *world*. Σε αντίθετη περίπτωση, η `basicChecks()` ρίχνει μια `RuntimeException` με μηνυμα "`All your places must belong to the same world`" και το πρόγραμμα «κρεμάει».

gradients: για κάθε place που δημιουργούμε, υπολογίζουμε τις αποστάσεις οποιουδήποτε σημείου του χάρτη από αυτό και τις αποθηκεύουμε σε ένα αντικείμενο Gradient(βλέπε κλάση Gradient). Για οικονομία στη μνήμη, διατηρούμε ένα μέρος μόνο των συνολικών υπολογισμένων gradient στη μνήμη και επαναφέροντας το ζητούμενο gradient μόνο όταν χρειάζεται ακριβώς όπως και στη μνήμη cache. Η μεταβλητή αυτή είναι τύπου PersistedCachedMap (ειδικότερα υπόσταση της SiafuGradientCache που κληρονομεί από την PersistedCachedMap).

Όλες οι μεταβλητές που ακολουθούν είναι μεταβλητές υπόστασης.

type: ένα String που δηλώνει τον τύπο του place π.χ. Bar, Restaurant, cinema, Café κ.τ.λ .

pos: η θέση του place πάνω στο χάρτη του κόσμου μας. Η μεταβλητή αυτή είναι τύπου Position.

name: μια μεταβλητή τύπου String που δηλώνει το όνομα του place. Σε περίπτωση που δεν δηλώνεται στον constructor το όνομα του place προκύπτει από τον τύπο του place + " – " + θέση Position του place, δηλαδή για ένα place τύπου Bar στη θέση (256,75) το όνομα του place θα είναι: "Bar-256,75".

info: μια μεταβλητή τύπου TreeMap<String, Publishable> με την οποία μπορούμε να προσθέτουμε χαρακτηριστικά σε κάθε place. Αντίθετα με τους agents στα places μπορούμε να προσθέτουμε χαρακτηριστικά οποτεδήποτε θέλουμε και επιπλέον κάθε place μπορεί να έχει διαφορετικό σύνολο χαρακτηριστικών. Αυτό συμβαίνει γιατί τα χαρακτηριστικά των agents Αντίθετα στον Agent ενημερώνονται στο αρχείο εξόδου της κλάσης CSVPrinter οπότε για να είναι στοιχισμένα τα στοιχεία, θα πρέπει όλοι οι agents να έχουν το ίδιο σύνολο ιδιοτήτων εξ' ου και η ανάγκη για infoLocking δηλαδή απαγορεύεται να προσθέσουμε ιδιότητες στους agents μετά την έξοδο από τη μέθοδο createAgents() της κλάσης AgentModel. Αντίθετα, δεν καταγράφονται οι ιδιότητες των places στο αρχείο εξόδου οπότε δε δημιουργούνται προβλήματα από την πρόσθεση νέων χαρακτηριστικών οποτεδήποτε μέσα στην προσομοίωση.

temporaryGradient: μεταβλητή τύπου Gradient. Χρησιμοποιείται για να αποθηκεύσει το Gradient ενός temporary place. Ένα προσωρινό place είναι ένα place που θα το χρειαστούμε μία φορά στην προσομοίωση οπότε δεν υπάρχει ανάγκη να διατηρήσουμε το αντικείμενο Gradient. Τα temporary places τα χρειαζόμαστε όταν βρισκόμαστε σε ένα σημείο και θέλουμε να μεταφερθούμε σε ένα προορισμό για τον οποίο ξέρουμε ότι στο μέλλον δε θα χρειαστεί να κινηθούμε ξανά προς τα εκεί, άρα δε χρειάζεται να διατηρήσουμε στο δίσκο το αρχείο με το persisted αντικείμενο Gradient που αντιστοιχεί σε αυτό το place. Για να μην συλλέξει ο garbage collector της Java ένα place, το Siafu προσθέτει μια αναφορά για αυτό το place στη static μεταβλητή *gradients*. Για να προσθέσουμε το place στο *gradients* θέτουμε την παράμετρο `Position relevantPosition` του constructor

```
Place(final String type, final Position pos, final World world,final String name, final Position relevantPosition)
```

ίση με `null`. Αντίθετα για να δημιουργήσουμε ένα προσωρινό place περνάμε στον constructor του Place

```
Place(final String type, final Position pos, final World world,final String name, final Position relevantPosition)
```

την παράμετρο `Position relevantPosition`, τη θέση Position που βρίσκεται ο agent τώρα. Έτσι ο υπολογισμός του Gradient δεν γίνεται για κάθε σημείο του χάρτη αλλά σταματά μόλις υπολογίσουμε τη διαδρομή που συνδέει τη θέση `relevantPosition` με τον προσωρινό προορισμό.

Χρήσιμες μέθοδοι της κλάσης Place

set(String key, Publishable value): με τη μέθοδο αυτή αντιστοιχούμε στο χαρακτηριστικό με όνομα key την τιμή value.

get(String key): με τη μέθοδο αυτή ανακτούμε την τιμή του χαρακτηριστικού με όνομα key.

getInfoKeys(): επιστρέφει το keyset των χαρακτηριστικών του συγκεκριμένου place.

getInfoValues(): επιστρέφει ένα σύνολο με τις τιμές των χαρακτηριστικών του συγκεκριμένου place.

pointFrom(): Μια συνάρτηση δρομολόγησης σε περίπτωση που προορισμός ενός agent είναι το συγκεκριμένο place. Παίρνει σαν παράμετρο το σημείο στο οποίο βρίσκεται ο agent και επιστρέφει την κατεύθυνση κίνησης που θα πρέπει να κινηθεί ο agent στο επόμενο βήμα ώστε η διαδρομή του να είναι βέλτιστη. Η μέθοδος αυτή είναι overloaded. Μπορούμε να δώσουμε ή όχι σαν επιπλέον παράμετρο την τωρινή κατεύθυνση του χρήστης ώστε η δρομολόγηση να γίνει με όσο το δυνατόν πιο ευθύγραμμη κίνηση. Δηλαδή αν ανάμεσα στις βέλτιστες κατευθύνσεις περιέχεται και η τωρινή κατεύθυνση του χρήστη, τότε επιστρέφεται αυτή.

distanceFrom(): δέχεται σαν παράμετρο ένα Position και επιστρέφει το μήκος της βέλτιστης διαδρομής από το Position αυτό μέχρι το place. Η απόσταση που επιστρέφεται υπολογίζεται θέτοντας κόστος 10 για δύο pixel που είναι γειτονικά οριζόντια ή κάθετα και κόστος 14 για γειτονικά pixel που γειτονεύουν διαγώνια.

Οι μέθοδοι της κλάσης Place συμπληρώνονται από setters – getters για τις υπόλοιπες μεταβλητές υπόστασης.

4.3.3 AgentTypeCharacteristics Class

Η παραμετροποίηση της προσομοίωσης γίνεται μέσω ενός αρχείου xml το οποίο ονομάζεται config.xml. Σε αυτό το αρχείο ορίζονται όλες οι παράμετροι που αφορούν τη συμπεριφορά των χρηστών. Για να μην χρειάζεται ο κώδικας να ανατρέχει συνεχώς στο εν λόγω αρχείο xml και καθυστερεί σημαντικά στην εκτέλεση της προσομοίωσης, διαβάζουμε στην αρχή της προσομοίωσης το xml και κατασκευάζουμε ένα αντικείμενο της κλάσης AgentTypeCharacteristics για κάθε κατηγορία χρηστών που ορίζεται στο xml. Το αντικείμενο αυτό περιέχει όλες τις απαραίτητες πληροφορίες για την κατηγορία χρηστών για την οποία κατασκευάστηκε. Η δομή της κλάσης είναι απλή. Αποτελείται από μια σειρά μεταβλητών στις οποίες αποθηκεύονται τα χαρακτηριστικά των χρηστών και τις αντίστοιχες μεθόδους προσπέλασης get και set για κάθε μία μεταβλητή.

Μεταβλητές

name: μια μεταβλητή τύπου String, στην οποία αποθηκεύεται το όνομα της κατηγορίας των χρηστών τα χαρακτηριστικά των οποίων αποθηκεύονται στο αντικείμενο AgentTypeCharacteristics (π.χ Students, Grandpa).

amountOfAgents: μεταβλητή τύπου int που παριστάνει το πλήθος των χρηστών που ανήκουν στην κατηγορία .

HOME,OFFICE: ένα αντικείμενο τύπου Place το οποίο παριστάνει το σπίτι ή το γραφείο στο οποίο διαμένουν ή δουλεύουν όλοι οι χρήστες που ανήκουν στην ομάδα. Δεν χρησιμοποιείται στην προσομοίωσή μας δεδομένου ότι θεωρούμε ότι ο κάθε χρήστης έχει το δικό του σπίτι και το δικό του χώρο εργασίας.

image: μεταβλητή τύπου String η οποία παρέχει μια αναφορά στο αρχείο που περιέχει την εικόνα με την οποία θα εμφανίζονται οι χρήστες της κατηγορίας.

hasJob: μια boolean μεταβλητή που δείχνει αν η κατηγορία χρηστών είναι εργαζόμενοι ή όχι.

workStart: μεταβλητή τύπου EasyTime, η οποία παριστάνει την ονομαστική ώρα στην οποία ξεκινούν δουλειά οι χρήστες της κατηγορίας.

workStartBlur: μεταβλητή τύπου int στην οποία αποθηκεύεται το χρονικό διάστημα γύρω από την ώρα workStart στο οποίο κατανέμεται με ομοιόμορφη κατανομή η πραγματική ώρα έναρξης της εργασίας των χρηστών.

workDuration: μεταβλητή τύπου int, η οποία εκφράζει την ονομαστική διάρκεια της εργασίας των χρηστών.

workDurationBlur: μεταβλητή τύπου int στην οποία αποθηκεύεται το διάστημα γύρω από την χρονική διάρκεια workDuration στο οποίο κατανέμεται με ομοιόμορφη κατανομή η πραγματική διάρκεια της εργασίας των χρηστών.

locationPrivacy: μεταβλητή τύπου int, η οποία εκφράζει το επίπεδο αφαίρεσης/αοριστίας στο οποίο αποστέλλουν τη θέση τους στην υπηρεσία οι χρήστες που ανήκουν στην κατηγορία

preferencesPrivacy:μεταβλητή τύπου int, η οποία εκφράζει το επίπεδο αφαίρεσης/αοριστίας στο οποίο αποστέλλουν τις προτιμήσεις τους στην υπηρεσία οι χρήστες που ανήκουν στην κατηγορία

priceWeightPref:μεταβλητή τύπου int που εκφράζει την σημασία, το βάρος, που δίνουν οι χρήστες της κατηγορίας στις τιμές ενός προορισμού. Χρησιμοποιείται κατά τη διάρκεια της επεξεργασίας των αποτελεσμάτων που επιστράφηκαν από την υπηρεσία τοπικά στο κινητό τερματικό.

priceWeightPrefBlur: μεταβλητή τύπου int που παριστάνει το διάστημα γύρω από το βάρος ως προς τις τιμές των προορισμών, μέσα στο οποίο κατανέμεται ομοιόμορφα το πραγματικό βάρος για κάθε χρήστη ξεχωριστά. Έτσι κάθε χρήστης της κατηγορίας διαφέρει από έναν άλλο, όμως γενικά οι χρήστες κατανέμονται γύρω από το priceWeightPref.

distWeightPref, distWeightPrefBlur: όμοια με τα παραπάνω priceWeightPref και priceWeightPrefBlur για την απόσταση από τον προορισμό.

qualityWeightPref, qualityWeightPrefBlur: όμοια με τα παραπάνω για την ποιότητα υπηρεσίας ενός προορισμού.

musicTypeWeightPref, musicTypeWeightPrefBlur: όμοια με τα παραπάνω για το είδος μουσικής που παίζεται στον προορισμό.

hasCar: μια μεταβλητή τύπου boolean που μας δείχνει αν οι χρήστες της κατηγορίας διαθέτουν αυτοκίνητο.

MAX_DESTINATION: μεταβλητή τύπου int,δείχνει την μέγιστη απόσταση που είναι διατεθειμένοι οι χρήστες να μετακινηθούν μέχρι τον προορισμό.

MAX_DISTANCE_ON_FOOT: μεταβλητή τύπου int, δείχνει την μέγιστη απόσταση που είναι διατεθειμένοι οι χρήστες να περπατήσουν μέχρι τον προορισμό.

MAX_WAITING_INTERVAL: μεταβλητή τύπου int, δείχνει το μέγιστο χρονικό διάστημα που περιμένουν οι χρήστες σε περίπτωση που φτάνοντας σε έναν προορισμό τον βρουν μη διαθέσιμο. Μετά τη λήξη αυτού του χρονικού διαστήματος οι χρήστες μετακινούνται σε έναν νέο προορισμό.

Μέθοδοι

printProperties(): Εκτυπώνει όλα τα χαρακτηριστικά της κατηγορίας χρηστών

Όλες οι υπόλοιπες μέθοδοι είναι τύπου get-set και χρησιμοποιούνται για την προσπέλαση των μεταβλητών που αναφέρθηκαν.

4.3.4 PlaceTypeCharacteristics Class

Αντίστοιχη της **AgentTypeCharacteristics** είναι η **PlaceTypeCharacteristics** για την κατηγορία των Places. Σε αυτήν αποθηκεύονται οι παράμετροι κάθε κατηγορίας προορισμών που ορίζεται στο **config.xml**.

Μεταβλητές

name:μεταβλητή τύπου String, το όνομα της κατηγορίας προορισμών τα χαρακτηριστικά των οποίων αποθηκεύονται στο αντικείμενο PlaceTypeCharacteristics (π.χ Café, Bar).

open:μεταβλητή τύπου EasyTime που αναπαριστά την ονομαστική ώρα έναρξης λειτουργίας του προορισμού.

openBlur: μεταβλητή τύπου int, η οποία εκφράζει ένα χρονικό διάστημα γύρω από την ώρα open στο οποίο κατανέμεται με ομοιόμορφη κατανομή η πραγματική ώρα έναρξης λειτουργίας ενός προορισμού.

close:μια μεταβλητή τύπου EasyTime που αναπαριστά την ονομαστική ώρα λήξης λειτουργίας του προορισμού.

closeBlur: μεταβλητή τύπου int, η οποία εκφράζει ένα χρονικό διάστημα γύρω από την ώρα close στο οποίο κατανέμεται με ομοιόμορφη κατανομή η πραγματική ώρα λήξης λειτουργίας του προορισμού.

unavailabilityTime:μεταβλητή τύπου HashMap στην οποία αποθηκεύονται οι χρόνοι μη διαθεσιμότητας για κάθε μία από τις τρεις δυνατές περιόδους λειτουργίας (χαμηλής, μέσης και υψηλής κίνησης).

unavailability:μεταβλητή τύπου HashMap στην οποία αποθηκεύονται οι πιθανότητες μη διαθεσιμότητας για τον προορισμό για κάθε μία από τις τρεις δυνατές περιόδους λειτουργίας (χαμηλής, μέσης, υψηλής κίνησης).

trafficStates:μεταβλητή τύπου TreeMap στην οποία αποθηκεύονται οι ώρες της ημέρας κατά τις οποίες ο προορισμός έχει χαμηλή, μέση ή υψηλή κίνηση.

Οι μέθοδοι είναι όλες μέθοδοι προσπέλασης τύπου get-set και χρησιμοποιούνται για την αρχικοποίηση του αντικειμένου από το xml και για την ανάκτηση των δεδομένων κατά τη διάρκεια της προσομοίωσης.

4.3.5 AgentModel Class

Η κλάση **AgentModel** είναι η κλάση στην οποία περιγράφεται το μοντέλο και οι κανόνες συμπεριφοράς των χρηστών της υπηρεσίας. Σε αυτήν την παράγραφο θα περιγράψουμε το μοντέλο ακολουθώντας τον κώδικα της κλάσης AgentModel. Μετά την αρχικοποίηση της κλάσης από τον constructor καλείται η μέθοδος **doIteratio**n**()** η οποία εκτελείται συνεχώς κατά τη διάρκεια της προσομοίωσης.

doIteration() : Σε αυτή τη μέθοδο αυξάνεται κατά ένα βήμα η ώρα της προσομοίωσης και για κάθε χρήστη που χειρίζεται η προσομοίωση καλείται η μέθοδος **handlePerson().** Ορισμένοι χρήστες μπορεί να είναι στον έλεγχο του χειριστή της προσομοίωσης. Η συμπεριφορά αυτών των χρηστών καθορίζεται εξολοκλήρου από τον χειριστή. Στην περίπτωσή μας όλοι οι χρήστες ελέγχονται μέσω της μεθόδου **handlePerson().**

handlePerson(): Η μέθοδος αυτή ελέγχει καταρχάς την ώρα της προσομοίωσης και το αν ο agent πρέπει να βρίσκεται στη δουλειά του. Αν αυτό συμβαίνει, τότε αλλάζει την μεταβλητή ACTIVITY του αντικειμένου Agent που χειρίζεται και τη θέτει ίση με GOING_TO_WORK, ώστε ο agent να πάει στην δουλειά του. Αν δεν συμβαίνει κάτι τέτοιο, τότε ελέγχει το πεδίο ACTIVITY του αντικειμένου Agent και ανάλογα με την τιμή του προβαίνει στις ακόλουθες ενέργειες:

περίπτωση CHOOSE_NEXT_DESTINATION: Στην περίπτωση αυτή ο χρήστης επιθυμεί να επιλέξει νέο προορισμό. Αυτό μπορεί να γίνει με δύο τρόπους. Είτε αποφασίζει μόνος του, είτε χρησιμοποιεί την υπηρεσία της προσομοίωσης. Με βάση τα χαρακτηριστικά του agent υπολογίζονται οι πιθανότητες των δυο αυτών περιπτώσεων και με τη βοήθεια μιας random συνάρτησης επιλέγεται τυχαία η μία από τις δύο και καλείται η μέθοδος **chooseByYourself**() ή η **makeRequest()** αντίστοιχα. Οι συναρτήσεις αυτές επιστρέφουν έναν προορισμό, ο οποίος τίθεται στο πεδίο destination του αντικειμένου agent έτσι ώστε ο agent να κατευθυνθεί σε αυτόν τον προορισμό. Σε περίπτωση που η **makeRequest()** δεν επιστρέψει αποτέλεσμα λόγω αδυναμίας εύρεσης κατάλληλου προορισμού, τότε ο agent επιλέγει να περιπλανηθεί για λίγο και να ξανακάνει το ερώτημά του στην υπηρεσία ή να διαλέξει μόνος του τον προορισμό του.

περίπτωση GOING_TO_WORK: Στην περίπτωση αυτή ο χρήστης κατευθύνεται από τον κώδικα του SIAFU αυτόματα προς τον τόπο εργασίας του. Ο κώδικάς μας δεν χρειάζεται να κάνει κάτι για αυτό, παρά μόνο όταν ο χρήστης φτάσει στον προορισμό του. Αν ο χρήστης έχει ήδη φτάσει, τότε αλλάζει το πεδίο ACTIVITY του agent από GOING_TO_WORK σε WORKING, υπολογίζεται μέσω των πεδίων workDuration και workDurationBlur η ακριβής ώρα που σχολάει ο χρήστης και ενημερώνεται το πεδίο WORK_END του agent.

περίπτωση WORKING: Σε αυτήν την περίπτωση δεν γίνεται τίποτα μέχρι ο χρόνος της προσομοίωσης να ξεπεράσει την ώρα που αναφέρεται στο πεδίο WORK_END του agent. Αν συμβεί αυτό, τότε με βάση τα χαρακτηριστικά του agentType workStart και workStartBlur υπολογίζεται η ώρα έναρξης της εργασίας του agent την

επόμενη μέρα και αποθηκεύεται στο πεδίο WORK_START του agent. Τέλος, η κατάσταση του agent μέσω της μεταβλητής ACTIVITY αλλάζει σε CHOOSE_NEXT_DESTINATION.

περίπτωση GOING_TO_DESTINATION: Σε αυτήν την περίπτωση ο χρήστης κατευθύνεται προς τον προορισμό του. Αν δεν έχει φτάσει, τότε δεν γίνεται τίποτα. Αν έχει φτάσει τότε αλλάζει η κατάστασή του στο πεδίο ACTIVITY και γίνεται ARRIVE_AT_DESTINATION.

περίπτωση ARRIVE_AT_DESTINATION: Όταν ο χρήστης έχει φτάσει στον προορισμό του ελέγχεται η διαθεσιμότητα του προορισμού μέσω των πιθανοτήτων μη διαθεσιμότητας που ορίζονται στο PlaceTypeCharacteristics για τον κάθε τύπο προορισμού. Αν ο προορισμός είναι διαθέσιμος, τότε η κατάσταση του agent αλλάζει και το πεδίο ACTIVITY του agent γίνεται SET_OCCUPATION_TIME. Αν ο προορισμός την ώρα που φτάνει ο χρήστης είναι μη διαθέσιμος, τότε ο χρήστης μπορεί είτε να περιμένει, οπότε το ACTIVITY αλλάζει σε WAITING, είτε να φύγει και να προσπαθήσει να βρει έναν νέο προορισμό, οπότε το ACTIVITY αλλάζει σε CHOOSE_NEXT_DESTINATION. Η απόφαση αυτή βασίζεται στις πιθανότητες που ορίζονται στο config.xml και αφορούν αυτήν την περίπτωση.

περίπτωση WAITING: Στην περίπτωση αυτή ο χρήστης βρίσκεται σε αναμονή. Αν ο προορισμός ξαναγίνει διαθέσιμος, τότε αλλάζει η κατάστασή του σε SET_OCCUPATION_TIME. Υπάρχει όμως και η περίπτωση ο χρόνος αναμονής να ξεπεράσει το μέγιστο χρόνο αναμονής που ορίζεται στο αντικείμενο Agent και επομένως ο χρήστης δεν μπορεί να περιμένει άλλο, οπότε επιλέγει να φύγει και να αναζητήσει καινούριο προορισμό. Η κατάστασή του στην τελευταία περίπτωση γίνεται CHOOSE_NEXT_DESTINATION.

περίπτωση SET_OCCUPATION_TIME: Ορίζεται ο χρόνος που μπορεί ο χρήστης να διαθέσει στην συγκεκριμένη ασχολία μέσω της μεθόδου **setNextEvent(),** η οποία λαμβάνει υπόψη της τα χαρακτηριστικά της ασχολίας όπως ορίζονται στο **config.xml**, την ώρα που πρέπει ο χρήστης να βρίσκεται στην δουλειά του, καθώς και το ωράριο του καταστήματος στο οποίο βρίσκεται ο χρήστης. Μετά από αυτά αλλάζει η κατάσταση του χρήστη σε BEING_OCCUPIED.

περίπτωση BEING_OCCUPIED:Ο χρήστης βρίσκεται στον επιθυμητό προορισμό και απασχολείται μέχρι την ώρα που έχει υπολογιστεί μέσω της μεθόδου **setNextEvent**(). Αν η ώρα αυτή δεν έχει φτάσει, ο χρήστης παραμένει στον προορισμό και δεν συμβαίνει τίποτα. Αν όμως η ώρα περάσει, τότε ο χρήστης πρέπει να φύγει. Διακρίνονται οι περιπτώσεις: i) έχει φτάσει η ώρα για δουλειά, οπότε η κατάσταση του χρήστη αλλάζει σε GOING_TO_WORK, είτε ii) τελειώνει η ώρα που έχει αποφασίσει να διαθέσει στην δραστηριότητα αυτή, οπότε αποφασίζει να φύγει, να αναζητήσει νέο προορισμό και η κατάστασή του αλλάζει σε CHOOSE_NEXT_DESTINATION.

Οι παραπάνω περιπτώσεις αποτελούν τις δυνατές καταστάσεις στις οποίες μπορεί να βρεθεί ένας χρήστης της υπηρεσίας. Ανάλογα την περίπτωση περιγράφηκαν οι αντιδράσεις του, καθώς και οι μεταβάσεις του από κατάσταση σε κατάσταση.

5.Αποτελέσματα Προσομοίωσης – Συμπεράσματα

5.1 Γενικά

Για την εξαγωγή συμπερασμάτων σχετικά με την σχέση προστασίας ιδιωτικότητος και αποτελεσματικότητας της υπηρεσίας τρέξαμε την προσομοίωση με διαφορετικές κάθε φορά παραμέτρους εισόδου. Οι παράμετροι που επηρεάζουν την ιδιωτικότητα των χρηστών, καθώς και την αποτελεσματικότητα μιας υπηρεσίας και που εξετάσθηκαν είναι οι παρακάτω:

i. Προτιμήσεις προστασίας ιδιωτικότητος χρηστών
ii. Γεωγραφική κατανομή των δυνατών προορισμών στην πόλη
iii. Δέντρο προτιμήσεων που χρησιμοποιεί η υπηρεσία

Κάθε φορά κρατούσαμε σταθερούς τους δύο από τους τρεις αυτούς παράγοντες μεταβάλλοντας μόνο τον τρίτο. Με αυτόν τον τρόπο καταφέραμε να απομονώσουμε και να μελετήσουμε την επίδραση καθεμιάς παραμέτρου στην σχέση προστασίας ιδιωτικότητος-αποτελεσματικότητας υπηρεσίας.

Προτιμήσεις προστασίας ιδιωτικότητας χρηστών: Η προστασία της ιδιωτικότητας μπορεί να διαιρεθεί σε δύο μέρη. Το πρώτο μέρος είναι η προστασία ιδιωτικότητας θέσης, ενώ το δεύτερο η προστασία ιδιωτικότητας των προτιμήσεων του χρήστη. Η πρώτη αναφέρεται στο πόσο καλά αποκρύπτουμε την πραγματική θέση του χρήστη από την υπηρεσία, ενώ η δεύτερη τις προτιμήσεις.Η προστασία αυτή επιτυγχάνεται και για τα δύο αυτά μέρη με την χρήση επιπέδων αφαίρεσης/αοριστίας. Για την μεν ιδιωτικότητα θέσης, τα δυνατά επίπεδα είναι 5 και είναι αυτά που περιγράφονται στην παράγραφο …… Όσον αφορά την ιδιωτικότητα προτιμήσεων, τα δυνατά επίπεδα εξαρτώνται από το δέντρο προτιμήσεων που χρησιμοποιείται στην προσομοίωση και συγκεκριμένα από το βάθος-πλήθος επιπέδων του εκάστοτε δέντρου. Στην περίπτωσή μας χρησιμοποιήθηκαν δύο δέντρα προτιμήσεων για ταινίες των οποίων το βάθος ήταν 4 επίπεδα. Ως εκ τούτου και τα επίπεδα αφαίρεσης/αοριστίας θα είναι 4. Σύμφωνα με τα όσα έχουν ειπωθεί στα προηγούμενα αναμένουμε μείωση στην αποτελεσματικότητα της υπηρεσίας όσο αυξάνεται η προστασία της ιδιωτικότητας και αντίστροφα μείωση στην προστασία της ιδιωτικότητας όσο αυξάνεται η αποτελεσματικότητα-επίδοση της υπηρεσίας μας. Ο κώδικας της προσομοίωσης που έχουμε σχεδιάσει μας επιτρέπει να εισάγουμε αν θέλουμε διαφορετικές ρυθμίσεις ιδιωτικότητας για κάθε κατηγορία χρηστών που έχει οριστεί. Έτσι θα μπορούσαμε να έχουμε χρήστες οι οποίοι να ενδιαφέρονται περισσότερο ή λιγότερο για την προστασία της ιδιωτικότητας τους ή την αποτελεσματικότητα που τους προσφέρει η υπηρεσία από κάποιους άλλους. Στην περίπτωση μας όμως είναι αναγκαίο για την μελέτη μας να θεωρήσουμε ότι όλοι οι χρήστες της υπηρεσίας έχουν τις ίδιες ρυθμίσεις ιδιωτικότητας. Έτσι σε κάθε τρέξιμο οι χρήστες διατηρούν όλα τα χαρακτηριστικά που τους κάνουν ξεχωριστούς και που αναλύθηκαν στο Κεφάλαιο 3 πλην των ρυθμίσεων ιδιωτικότητας που τίθενται ίδιες για όλους.

Γεωγραφική κατανομή των δυνατών προορισμών στην πόλη: Ένας σημαντικός παράγοντας που καθορίζει σε μεγάλο βαθμό την σχέση ιδιωτικότητας-

αποτελεσματικότητας είναι η κατανομή των προορισμών στην πόλη, δηλαδή το πού συγκεντρώνονται οι προορισμοί, η πυκνότητά τους στις εν λόγω περιοχές καθώς και οι μεταξύ τους αποστάσεις. Ο προσομοιωτής Siafu μας δίνει την δυνατότητα να χρησιμοποιήσουμε αρκετές διαφορετικές κατανομές για τους διάφορους προορισμούς των χρηστών. Η γεωγραφική κατανομή μιας κατηγορίας προορισμών διαφέρει από την κατανομή μιας άλλης. Σαν παράδειγμα μπορούμε να θεωρήσουμε την κατανομή των καφετεριών σε μια πόλη, οι οποίες είναι συνήθως συγκεντρωμένες με μεγάλη πυκνότητα γύρω από συγκεκριμένες περιοχές , ενώ αντίθετα για την κατηγορία των κινηματογράφων μπορούμε να πούμε ότι ακολουθούν διαφορετική κατανομή σε κάθε πόλη , η οποία μπορεί σε κάποιες περιπτώσεις να θεωρηθεί ακόμη και ομοιόμορφη. Για να προσεγγίσουμε όσο το δυνατόν καλύτερα την πραγματικότητα, οι κατανομές που χρησιμοποιήθηκαν στις προσομοιώσεις μας ήταν οι ακόλουθες:

i. Κατανομή Gauss ανά περιοχές (μοντέλο Small World): Στην κατανομή αυτή θεωρούμε ότι οι προορισμοί μας εμφανίζονται σε συγκεκριμένες μόνο περιοχές και όχι οπουδήποτε στην πόλη. Η πυκνότητα των προορισμών είναι μεγαλύτερη στο κέντρο της περιοχής και φθίνει όσο απομακρυνόμαστε από αυτό ακολουθώντας την Κανονική κατανομή Gauss (μ=0,σ=1). Οι περιοχές με την σειρά τους επιλέγονται τυχαία από τον κώδικα της προσομοίωσης και υποτέθηκε ότι ακολουθούν ομοιόμορφη κατανομή σε ολόκληρη την πόλη.
ii. Ομοιόμορφη Κατανομή κατά περιοχές: Θεωρούμε ότι οι προορισμοί εμφανίζονται σε συγκεκριμένες μόνο περιοχές και όχι οπουδήποτε στην πόλη. Η πυκνότητα των προορισμών είναι παντού η ίδια μέσα στην περιοχή. Οι θέσεις των περιοχών επιλέγονται από την προσομοίωση όπως και στην προηγούμενη περίπτωση και κατανέμονται ομοιόμορφα σε ολόκληρη την πόλη.
iii. Ομοιόμορφη Κατανομή σε ολόκληρη την πόλη: Στην περίπτωση αυτή οι προορισμοί μας μπορούν να βρίσκονται οπουδήποτε στην πόλη. Η πυκνότητά τους είναι η ίδια σε οποιοδήποτε σημείο της πόλης.

Ανάλογα με το είδος των προορισμών και την συγκεκριμένη πόλη που θέλουμε να προσομοιώσουμε, κάποια από τις παραπάνω κατανομές είναι η καταλληλότερη και πλησιέστερη στην πραγματικότητα. Υπάρχει ακόμη η δυνατότητα να καθορίσουμε εμείς τις ακριβείς θέσεις όλων των προορισμών και με αυτόν τον τρόπο να προσομοιώσουμε μια συγκεκριμένη υπηρεσία σε μια συγκεκριμένη πόλη. Στην περίπτωσή μας οι θέσεις των προορισμών επιλέχθηκαν τυχαία από τον κώδικα της προσομοίωσης, ακολουθώντας όμως κάθε φορά κάποια από τις κατανομές που προαναφέρθηκαν. Όσον αφορά την ιδιωτικότητα των χρηστών και την αποτελεσματικότητα της υπηρεσίας, η κατανομή των προορισμών στην πόλη παίζει σπουδαίο ρόλο. Σαν παράδειγμα αναφέρουμε έναν χρήστη που ψάχνει ένα είδος προορισμού που κατανέμεται κατά Gauss σε 2 περιοχές. Αυτομάτως, η υπηρεσία μπορεί να υπολογίσει με πολύ καλές πιθανότητες την μελλοντική του θέση, η οποία δεν μπορεί να διαφέρει κατά πολύ από τα κέντρα των δύο περιοχών. Το αντίθετο συμβαίνει όσο οι κατανομές απλώνουν στον χώρο. Έτσι μπορούμε διαισθητικά να συμπεράνουμε ότι η ομοιόμορφη κατανομή σε ολόκληρη την πόλη προστατεύει την ιδιωτικότητα των χρηστών καλύτερα από την Gauss κατά περιοχές. Το αντίθετο αναμένουμε να συμβαίνει όσον αφορά την αποτελεσματικότητα της υπηρεσίας που μπορούμε να πετύχουμε δηλώνοντας την θέση μας με κάποιο επίπεδο αφαίρεσης.

Αυτό συμβαίνει διότι με μικρή μόνο αλλαγή στην αναφορά της θέσης του χρήστη, οι προορισμοί που επιστρέφονται θα αλλάζουν εντελώς σε σχέση με εκείνους της πλήρους ακρίβειας(ακριβούς θέσης). Το τελευταίο είναι απόρροια του γεγονότος της μη συγκέντρωσης των προορισμών σε μια περιοχή. Δεδομένης της σημασίας της γεωγραφικής κατανομής των προορισμών στον καθορισμό της σχέσης ιδιωτικότητας-αποτελεσματικότητας, η καθεμιά θα εξεταστεί ξεχωριστά σε μια σειρά προσομοιώσεων. Θα υποθέσουμε δηλαδή ότι όλοι οι προορισμοί ακολουθούν μια συγκεκριμένη κατανομή από τις τρεις που αναφέρθηκαν και θα μελετήσουμε τα ιδιαίτερα χαρακτηριστικά που προσδίδει η καθεμιά σε αυτή τη σχέση.

Δέντρο Προτιμήσεων: Η υπηρεσία που εξετάστηκε αφορά την επιλογή κινηματογράφων από τους χρήστες. Οι προτιμήσεις των χρηστών όσον αφορά τις ταινίες που προβάλλονται στους κινηματογράφους κατηγοριοποιήθηκαν με τη βοήθεια δύο δέντρων προτιμήσεων. Το πρώτο από τα δύο είναι ένα εξαντλητικό δέντρο με πολλές κατηγορίες ταινιών, αρκετά εξειδικευμένο και λεπτομερές, ενώ το δεύτερο είναι απλούστερο με λιγότερες και μικρότερες διακλαδώσεις. Τα επίπεδα και των δύο δέντρων, τα οποία παρουσιάζονται στα σχήματα 1 και 2, είναι τέσσερα. Θα δούμε ότι ανάλογα με την δομή και την πολυπλοκότητα του κάθε δέντρου προκύπτουν και διαφορετικά αποτελέσματα για την ιδιωτικότητα των χρηστών και την αποτελεσματικότητα της υπηρεσίας μας.

5.2 Ανεξαρτησία ιδιωτικότητας προτιμήσεων – αποτελεσματικότητας θέσης και ανεξαρτησία ιδιωτικότητας θέσης – αποτελεσματικότητας προτιμήσεων.

Με τον όρο ιδιωτικότητα θέσης(προτιμήσεων) αναφερόμαστε στο πόσο καλά προστατεύεται η πραγματική θέση(προτιμήσεις) του χρήστη από την υπηρεσία, ενώ με τον όρο αποτελεσματικότητα θέσης(προτιμήσεων) αναφερόμαστε στο πόσο κοντά γεωγραφικά (πάνω στο δέντρο προτιμήσεων) βρίσκονται τα αποτελέσματα που επιστρέφει η υπηρεσία σε σχέση με αυτά της πλήρους ακριβείας, όταν παρέχουμε δηλαδή στην υπηρεσία την πραγματική θέση και προτίμησή μας. Στην παρούσα ενότητα θα αποδείξουμε ότι υπάρχει μια σχέση ανεξαρτησίας μεταξύ ιδιωτικότητας προτιμήσεων και αποτελεσματικότητας θέσης , όπως επίσης και μεταξύ ιδιωτικότητας θέσης και αποτελεσματικότητας προτιμήσεων. Για να το πετύχουμε αυτό θα πρέπει να εξετάσουμε κατ’ αρχάς τον τρόπο με τον οποίο λειτουργεί η υπηρεσία μας και τον αλγόριθμο με τον οποίο επεξεργάζεται τα δεδομένα και επιστρέφει αποτελέσματα.
Υποθέτουμε ότι κάποιος χρήστης ζητά βοήθεια από την υπηρεσία μας για την εύρεση ενός προορισμού που τον ικανοποιεί. Για να μπορέσει η υπηρεσία να επιστρέψει αποτελέσματα θα πρέπει να διαθέτει την προτίμηση του χρήστη και την θέση του σε κάποιο επίπεδο αφαίρεσης/αοριστίας. Αφού αποσταλούν τα δεδομένα στην υπηρεσία γίνεται η εξής επεξεργασία: Από το σύνολο των δυνατών προορισμών διαλέγονται εκείνοι μόνο που ικανοποιούν τα κριτήρια προτιμήσεων του χρήστη και κατόπιν ταξινομούνται με βάση την απόστασή τους από την δηλωθείσα θέση του χρήστη. Για να βρούμε πόσο αποτελεσματική είναι αυτή η διαδικασία σε σχέση με την πλήρη ακρίβεια, θα πρέπει να συγκρίνουμε τα αποτελέσματα που επιστρέφονται με τις δεδομένες ρυθμίσεις προστασίας ιδιωτικότητας με εκείνα που επιστρέφονται χωρίς προστασία ιδιωτικότητας, δηλαδή με την παροχή στην υπηρεσία ακριβών δεδομένων θέσης και προτιμήσεων.

Για την αποτελεσματικότητα προτιμήσεων θα ορίσουμε δύο δείκτες – μέτρα. Ο πρώτος είναι ο λόγος του πλήθους των επιστρεφόμενων προορισμών στα εκάστοτε επίπεδα αφαίρεσης προς το πλήθος των επιστρεφόμενων προορισμών με πλήρη ακρίβεια. Ο παραπάνω λόγος θα είναι πάντοτε μεγαλύτερος ή ίσος του 1, με την τιμή 1 να αντιστοιχεί στην μέγιστη αποτελεσματικότητα, η οποία επιτυγχάνεται όταν δεν χρησιμοποιούμε αφαίρεση στα δεδομένα. Το παραπάνω συμβαίνει διότι όταν στέλνουμε τις προτιμήσεις σε μεγαλύτερο επίπεδο αοριστίας οι προτιμήσεις γίνονται πιο γενικές και έτσι περισσότεροι προορισμοί τις ικανοποιούν. Στο μεγαλύτερο επίπεδο αφαίρεσης τελικά θα επιστρέφονται όλοι οι προορισμοί. Ο δεύτερος δείκτης που χρησιμοποιούμε είναι η μέση απόσταση των επιστρεφόμενων προορισμών από την πραγματική προτίμηση του χρήστη στο δέντρο προτιμήσεων. Για τον υπολογισμό θεωρούμε ότι δύο προτιμήσεις που έχουν κοινό πατέρα έχουν απόσταση 1, κοινό παππού απόσταση 2 και συνεχίζουμε με τον ίδιο τρόπο για όλους τους κόμβους-πρόγονους. Η μέγιστη απόσταση δύο προορισμών πάνω στα δέντρα προτιμήσεων που χρησιμοποιούμε είναι ίση με 3, αφού τα δέντρα διαθέτουν τέσσερα επίπεδα. Βλέπουμε λοιπόν ότι καθώς οι δείκτες που ορίσαμε αυξάνονται, η αποτελεσματικότητα – επίδοση της υπηρεσίας μας μειώνεται.

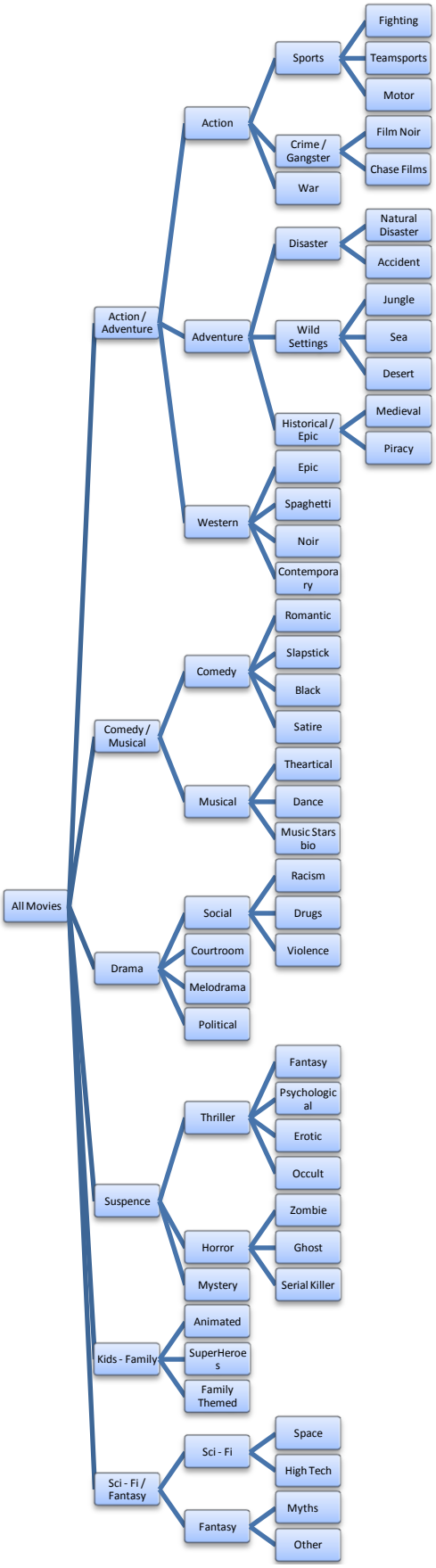
Υποθέτουμε τώρα ότι κρατώντας σταθερό το επίπεδο αφαίρεσης προτιμήσεων, αλλάζουμε το επίπεδο αφαίρεσης θέσης. Ουσιαστικά αυτό που θα συμβεί θα είναι να αποσταλεί στην υπηρεσία διαφορετική θέση για τον χρήστη και να αλλάξει η σειρά με την οποία θα ταξινομηθούν οι επιστρεφόμενοι προορισμοί. Το σύνολο των επιστρεφόμενων προορισμών θα παραμείνει το ίδιο και το μόνο που θα αλλάξει θα είναι η σχετική τους θέση στη λίστα. Από την παραπάνω ανάλυση μπορούμε να συνάγουμε ότι κανένας από τους δείκτες αποτελεσματικότητας προτιμήσεων δεν εξαρτάται από την θέση στη λίστα των προορισμών, επομένως κανείς από τους δύο δεν θα μεταβληθεί από την αλλαγή στο επίπεδο αοριστίας θέσης. Άρα λόγω του τρόπου με τον οποίο έχουμε ορίσει την αποτελεσματικότητα προτιμήσεων βλέπουμε ότι είναι ανεξάρτητη από το επίπεδο αοριστίας θέσης και συνεπώς και από την προστασία ιδιωτικότητας θέσης που απολαμβάνει ο χρήστης.

Αντίστοιχα, θα πρέπει να ορίσουμε έναν δείκτη για την αποτελεσματικότητα θέσης. Αν προσπαθήσουμε να ορίσουμε ως αποτελεσματικότητα θέσης την μέση απόσταση των επιστρεφόμενων προορισμών από την πραγματική θέση του χρήστη, γρήγορα θα καταλήξουμε σε αδιέξοδο. Αυτό συμβαίνει διότι αλλάζοντας το επίπεδο αφαίρεσης θέσης το σύνολο των επιστρεφόμενων προορισμών παραμένει το ίδιο, με διαφορετική ταξινόμηση και η μέση απόσταση από την πραγματική θέση του χρήστη δεν αλλάζει. Η αποτελεσματικότητα της υπηρεσίας ως προς τη θέση προφανώς μειώνεται, όμως αυτό δεν αποτυπώνεται στον παραπάνω δείκτη. Βλέπουμε επομένως ότι θα πρέπει να οριστεί διαφορετικά η αποτελεσματικότητα ως προς την θέση. Ο ορισμός που δίνουμε εδώ βασίζεται στην σύγκριση των αποτελεσμάτων σε κάποιο επίπεδο αοριστίας με εκείνα του επιπέδου πλήρους ακρίβειας. Ως αποτελεσματικότητα θέσης λοιπόν ορίζουμε τον μέσο όρο των αποστάσεων μεταξύ των προορισμών που βρίσκονται σε αντίστοιχες θέσεις κάθε λίστας. Δηλαδή παίρνουμε την απόσταση του $1^{ου}$ της $1^{ης}$ λίστας(με αοριστία ως προς την θέση) από τον $1^{ο}$ της $2^{ης}$ (πλήρους ακρίβειας), προσθέτουμε την απόσταση του $2^{ου}$ της $1^{η}$ λίστας από τον $2^{ο}$ της $2^{ης}$,συνεχίζουμε για όλους τους άλλους προορισμούς και διαιρούμε με το πλήθος των προορισμών. Το πλήθος των προορισμών των δύο λιστών θα είναι το ίδιο αφού αυτό καθορίζεται μόνο από το επίπεδο αοριστίας/αφαίρεσης προτιμήσεων το οποίο παραμένει το ίδιο και στις δυο περιπτώσεις.

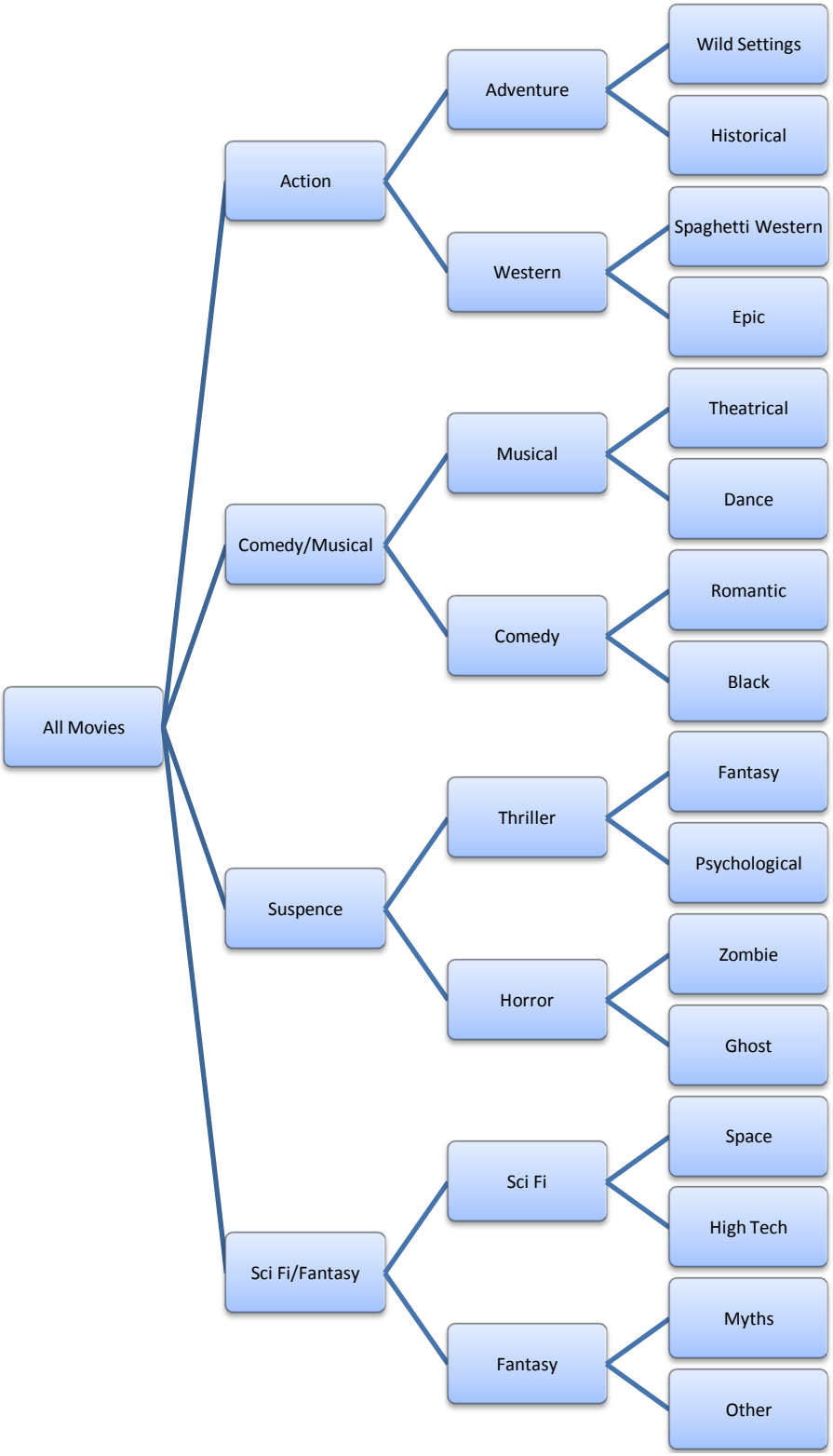
Αν κάνουμε και εδώ την υπόθεση ότι διατηρούμε σταθερό το επίπεδο αοριστίας θέσης και αλλάζουμε μόνο το επίπεδο αοριστίας προτιμήσεων, τότε όπως είναι

φυσικό το πλήθος των στοιχείων των λιστών που θα έχουμε να συγκρίνουμε θα αλλάξει. Οι προορισμοί που θα προστεθούν ή θα αφαιρεθούν όμως στις δύο λίστες θα έχουν την ίδια γεωγραφική κατανομή με αυτούς που υπήρχαν πριν την αλλαγή στο επίπεδο αοριστίας προτιμήσεων. Επομένως, αφού όπως είναι φανερό το μέτρο της αποτελεσματικότητας θέσης εξαρτάται από την κατανομή των προορισμών στην πόλη, το μέτρο αυτό δεν θα αλλάξει με αλλαγή του επιπέδου αφαίρεσης προτιμήσεων. Βλέπουμε λοιπόν ότι και η αποτελεσματικότητα θέσης είναι ανεξάρτητη της προστασίας ιδιωτικότητας προτιμήσεων. Το παραπάνω θεωρητικό αποτέλεσμα επιβεβαιώνεται από τα αποτελέσματα της προσομοίωσης τα οποία παρουσιάζονται στη συνέχεια.

Σύμφωνα με τα παραπάνω η σχέση ιδιωτικότητας – αποτελεσματικότητας μπορεί να μελετηθεί ανεξάρτητα για τη θέση και την αποτελεσματικότητα. Αν ορίσουμε όμως διαφορετικά την έννοια της αποτελεσματικότητας υπηρεσίας, τότε οι σχέσεις των δύο αυτών μεγεθών ίσως να μην είναι τόσο απλές. Εμείς θα περιορισθούμε στους ορισμούς που παρουσιάσαμε σε αυτήν την ενότητα.



Σχήμα 1. Το πρώτο δέντρο προτιμήσεων



Σχήμα 2. Το δεύτερο δέντρο προτιμήσεων

5.3 Ιδιωτικότητα-Αποτελεσματικότητα Προτιμήσεων

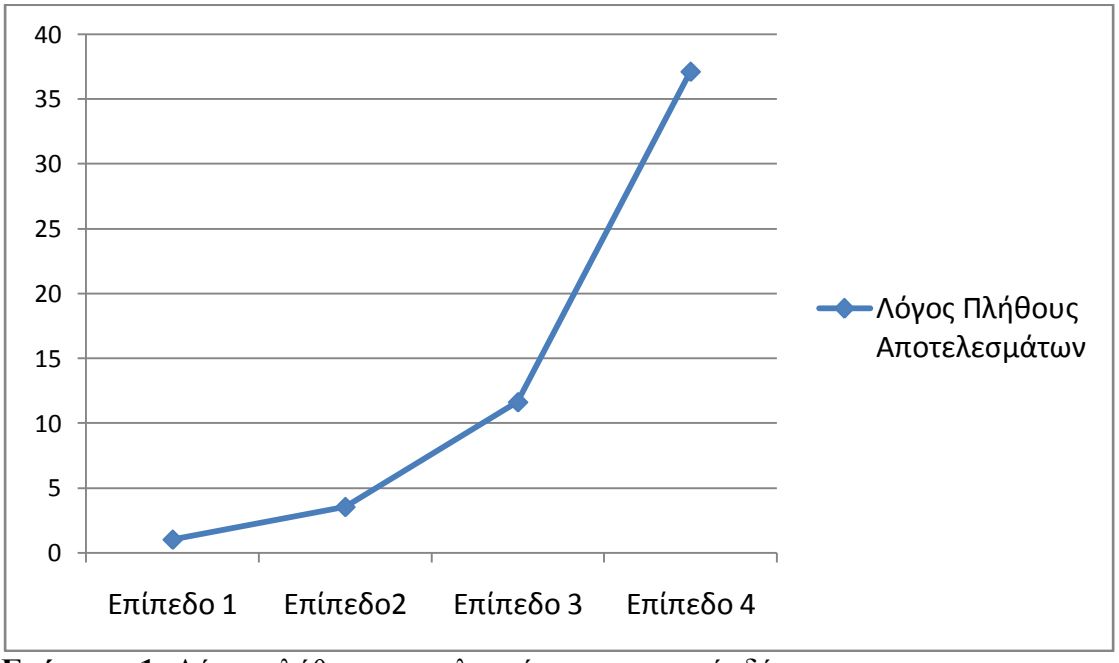
Στην παράγραφο αυτή θα παρουσιάσουμε τα αποτελέσματα της προσομοίωσής μας για την σχέση μεταξύ ιδιωτικότητας και αποτελεσματικότητας προτιμήσεων. Όπως αποδείξαμε στην προηγούμενη παράγραφο η σχέση αυτή εξαρτάται από τις ρυθμίσεις προστασίας ιδιωτικότητας (4 επίπεδα) των χρηστών και από το δέντρο προτιμήσεων που χρησιμοποιεί η υπηρεσία, ενώ είναι ανεξάρτητη από την γεωγραφική κατανομή των προορισμών στην έκταση της πόλης. Έγιναν δυο σειρές προσομοιώσεων, μία για κάθε δέντρο, ενώ σε κάθε σειρά πραγματοποιήθηκαν τέσσερα σύνολα προσομοιώσεων, ένα για κάθε επίπεδο αοριστίας προτιμήσεων των χρηστών. Τα αποτελέσματα των προσομοιώσεων ακολουθούν στη συνέχεια.

5.3.1 Αποτελέσματα προσομοίωσης για εκτενές δέντρο προτιμήσεων

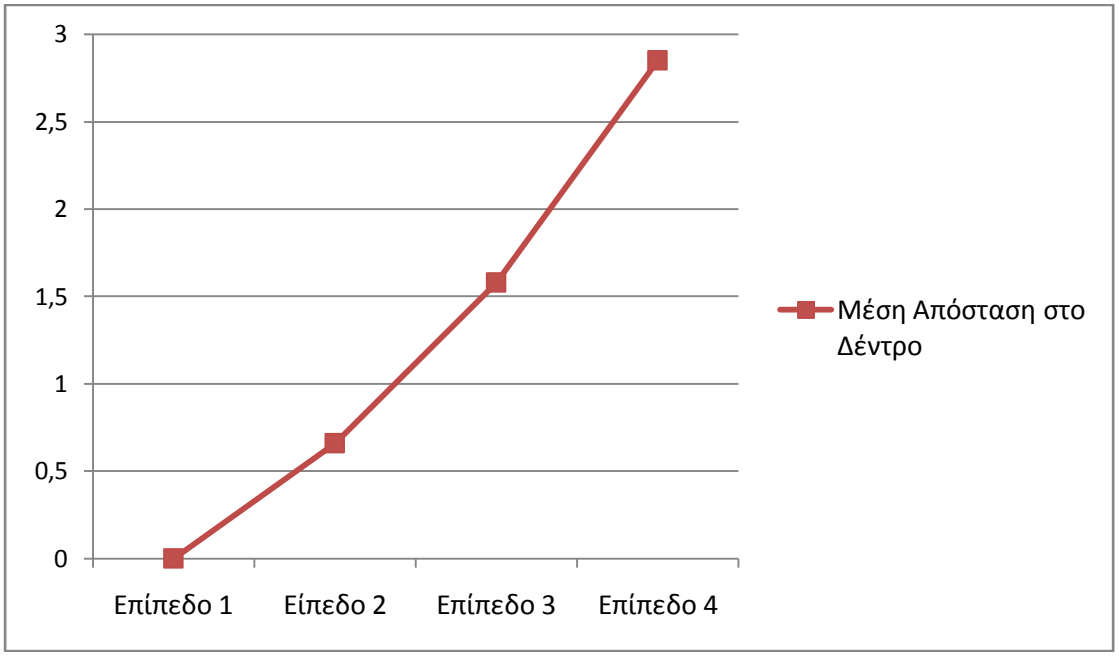
Τα αποτελέσματα της προσομοίωσης χρησιμοποιώντας ως δέντρο προτιμήσεων αυτό του σχήματος 1 παρουσιάζονται στον πίνακα και τα γραφήματα που ακολουθούν. Τα επίπεδα 1 έως 4 αναφέρονται στις ρυθμίσεις προστασίας ιδιωτικότητας προτιμήσεων των χρηστών, όπου το επίπεδο 1 είναι αυτό της πλήρους ακρίβειας.

	Επίπεδο 1	Επίπεδο 2	Επίπεδο 3	Επίπεδο 4
Λόγος πλήθους αποτελεσμάτων	1	3.52	11.6	37.1
Μέση απόσταση στο δέντρο	0	0.66	1.58	2.85

Πίνακας 1. Αριθμητικά αποτελέσματα προσομοίωσης για εκτενές δέντρο προτιμήσεων



Γράφημα 1: Λόγος πλήθους αποτελεσμάτων για εκτενές δέντρο



Γράφημα 2. Μέση απόσταση στο δέντρο για εκτενές δέντρο προτιμήσεων

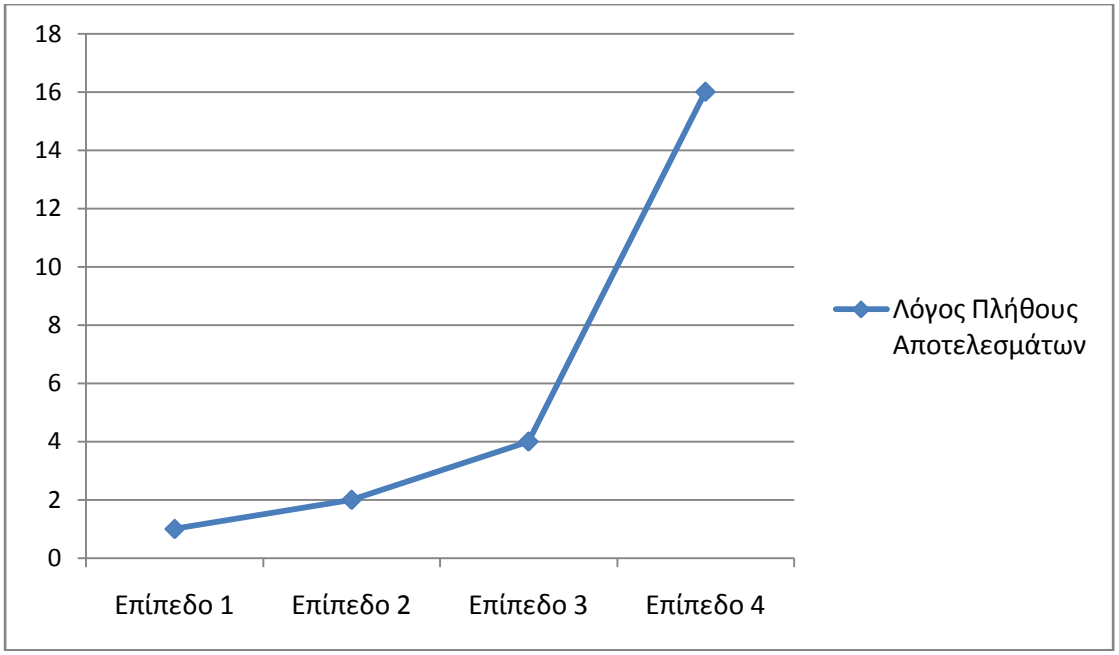
Παρατηρούμε ότι και οι δύο δείκτες που έχουμε ορίσει για την αποτελεσματικότητα υπηρεσίας αυξάνουν καθώς αυτή μειώνεται. Όπως αναμενόταν, καθώς το επίπεδο αοριστίας αυξάνεται, μειώνεται η αποτελεσματικότητα της υπηρεσίας και αυτό αποτυπώνεται στα γραφήματα 1 και 2 σαν δυο αύξουσες συναρτήσεις. Η αύξηση αυτή δεν είναι γραμμική, αλλά και στις δυο περιπτώσεις προσεγγίζει την εκθετική συνάρτηση. Αυτό όπως θα δούμε στη συνέχεια δεν συμβαίνει μόνο σε αυτή την περίπτωση, αλλά για όλες όσες εξετάζουμε. Οι καμπύλες των αποτελεσμάτων στρέφουν τα κοίλα προς τα πάνω, δηλαδή έχουν διαρκώς αυξανόμενο ρυθμό ανόδου, που συνεπάγεται ραγδαία πτώση της αποτελεσματικότητας από ένα σημείο και μετά. Στην περίπτωσή μας μπορούμε να δούμε ότι μετά το $3^{ο}$ επίπεδο αοριστίας προτιμήσεων η πτώση στην αποτελεσματικότητα είναι πολύ μεγάλη.

5.3.2 Αποτελέσματα προσομοίωσης για απλούστερο δέντρο προτιμήσεων

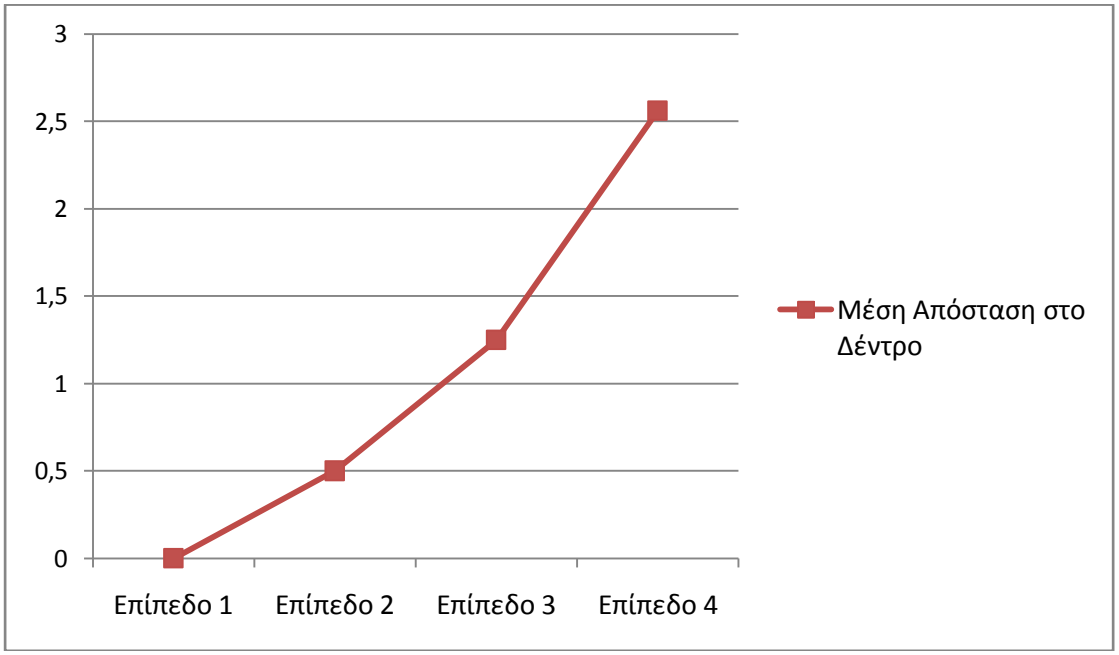
Τα αποτελέσματα των προσομοιώσεων για το δεύτερο δέντρο είναι τα ακόλουθα:

	Επίπεδο 1	Επίπεδο 2	Επίπεδο 3	Επίπεδο 4
Λόγος πλήθους αποτελεσμάτων	1	2	4	16
Μέση απόσταση στο δέντρο	0	0.5	1.25	2.56

Πίνακας 2. Αριθμητικά αποτελέσματα προσομοίωσης για το απλούστερο δέντρο προτιμήσεων



Γράφημα 3. Λόγος πλήθους αποτελεσμάτων για απλούστερο δέντρο προτιμήσεων



Γράφημα 4. Μέση απόσταση στο δέντρο για απλούστερο δέντρο προτιμήσεων

Ισχύουν και εδώ τα όσα ειπώθηκαν στην παράγραφο 5.3.1. Η μορφή των καμπυλών και τα γενικά συμπεράσματα παραμένουν τα ίδια. Συγκρίνοντας όμως τα αποτελέσματα που πήραμε με το απλούστερο δέντρο με αυτά του εκτενούς βλέπουμε ότι παρόλο που η μέγιστη αποτελεσματικότητα που επιτυγχάνεται με τη χρήση εκτενούς δέντρου είναι καλύτερη, η μείωση της αποτελεσματικότητας είναι μεγαλύτερη αναλογικά για το εκτενές παρά για το απλούστερο δέντρο. Έτσι για πλήρη ακρίβεια η ποιότητα της υπηρεσίας είναι καλύτερη για εκτενές δέντρο, αλλά καθώς το επίπεδο αοριστίας ανεβαίνει η ποιότητα των δυο υπηρεσιών επηρεάζεται διαφορετικά, με εκείνη του εκτενούς να επηρεάζεται αναλογικά περισσότερο.

5.4 Ιδιωτικότητα-Αποτελεσματικότητα Θέσης

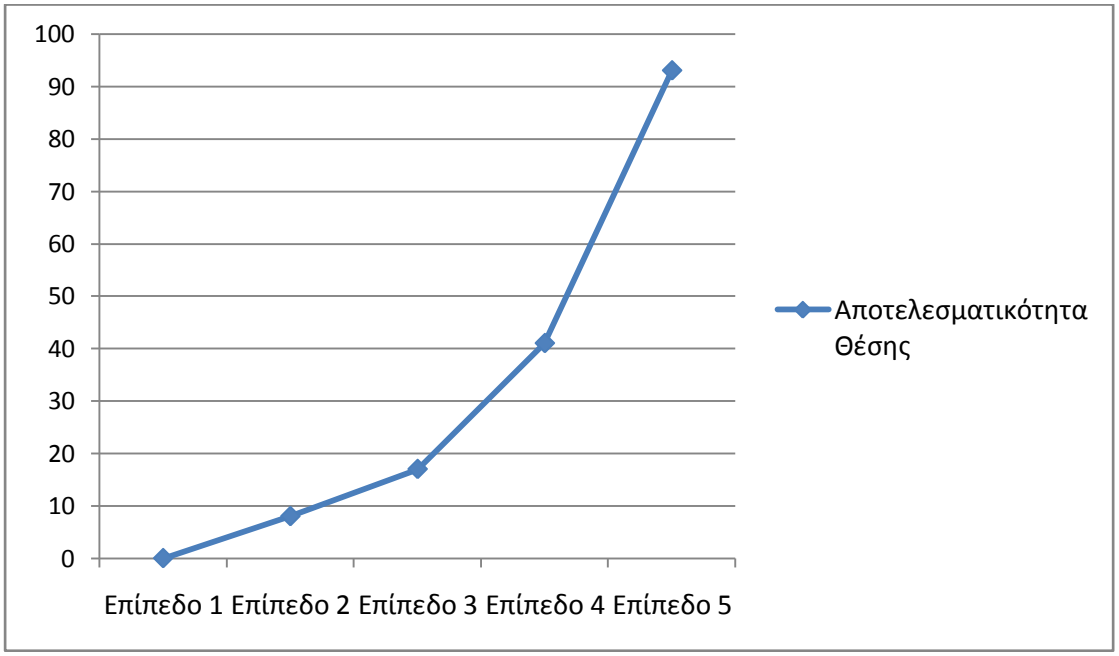
Στην συνέχεια παρουσιάζονται τα αποτελέσματα των προσομοιώσεων για την σχέση ιδιωτικότητας – αποτελεσματικότητας θέσης. Οι παράγοντες που επηρεάζουν αυτή τη σχέση είναι η γεωγραφική κατανομή και το επίπεδο αοριστίας θέσης των χρηστών, ενώ είναι ανεξάρτητη από το δέντρο προτιμήσεων που χρησιμοποιείται καθώς και από το επίπεδο αοριστίας προτιμήσεων όπως αποδείξαμε στην παράγραφο …επίσης βλέπουμε την επίδραση της γεωγραφικής κατανομής των προορισμών στην σχέση αυτή. Το επίπεδο 1 αναφέρεται στην ακριβή θέση του χρήστη, ενώ όσο αυξάνει το επίπεδο αυξάνεται και το επίπεδο αφαίρεσης/αοριστίας.

5.4.1 Κατανομή Gauss ανά περιοχές

Στην περίπτωση αυτή οι προορισμοί κατανέμονται Γκαουσιανά σε περιοχές. Τα αποτελέσματα παρουσιάζονται στη συνέχεια, με το επίπεδο 1 να δηλώνει την ακριβή θέση και το επίπεδο αοριστίας να αυξάνει για τα επόμενα επίπεδα.

	Επίπεδο 1	Επίπεδο 2	Επίπεδο 3	Επίπεδο 4	Επίπεδο 5
Αποτελεσματικότητα Θέσης	0	8	17	41	93

Πίνακας 3. Αριθμητικά αποτελέσματα προσομοίωσης για προορισμούς που κατανέμονται Γκαουσιανά ανά περιοχές



Γράφημα 5. Δείκτης αποτελεσματικότητας θέσης ως προς επίπεδο προστασίας ιδιωτικότητας θέσης για Γκαουσιανή ανά περιοχές κατανομή των προορισμών

Βλέπουμε ότι ο δείκτης αποτελεσματικότητας αυξάνει με την αύξηση του επιπέδου αοριστίας, όπως αναμενόταν. Σύμφωνα με τον ορισμό του δείκτη

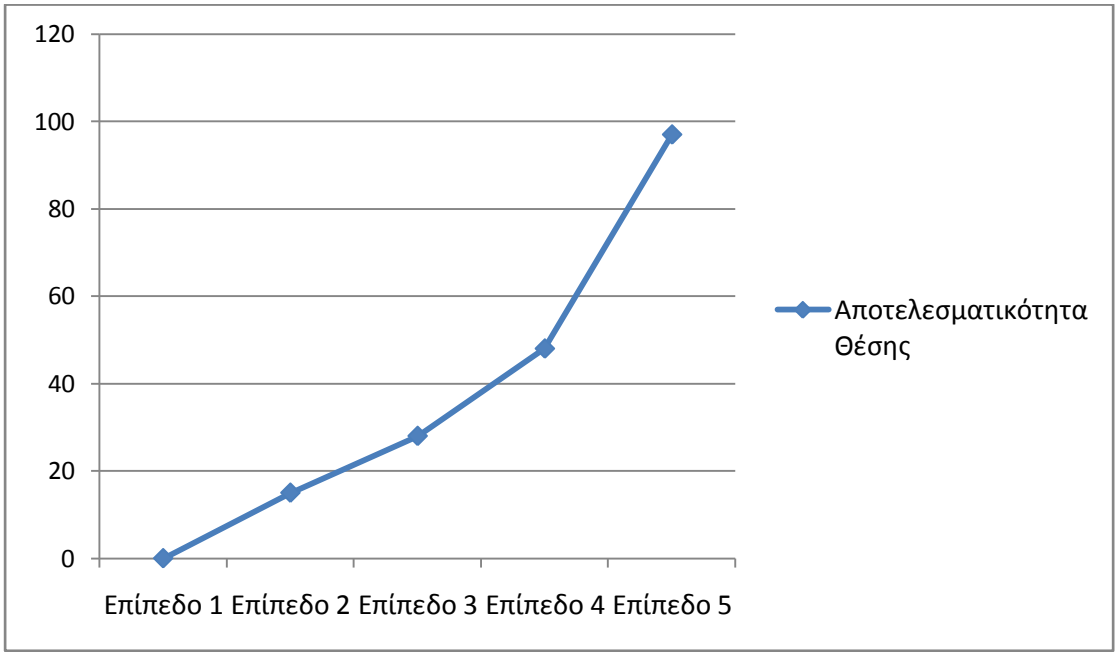
αποτελεσματικότητας αύξησή του σημαίνει μείωση στην αποτελεσματικότητα που επιτυγχάνεται από την υπηρεσία. Παρατηρούμε ότι η μορφή της καμπύλης δεν είναι γραμμική, αλλά προσεγγίζει κάποιας μορφής εκθετική συνάρτηση. Θα μπορούσαμε να πούμε ότι μέχρι το επίπεδο 3 η γραφική παράσταση είναι γραμμική. Όπως θα δούμε η εν λόγω μορφή είναι κοινή και για τις υπόλοιπες κατανομές και «γραμμικοποιείται» εντελώς όσο η κατανομή προσεγγίζει την ομοιόμορφη κατανομή παντού. Θεωρητικά σε μια κατανομή ομοιόμορφη που εκτείνεται ως το άπειρο, θα αναμέναμε ως αποτέλεσμα μια ευθεία γραμμή.

5.4.2 Ομοιόμορφη κατανομή ανά περιοχές

Στην περίπτωση αυτή οι προορισμοί κατανέμονται ομοιόμορφα σε περιοχές. Τα αποτελέσματα παρουσιάζονται στη συνέχεια, με το επίπεδο 1 να δηλώνει την ακριβή θέση και το επίπεδο αοριστίας να αυξάνει για τα επόμενα επίπεδα.

	Επίπεδο 1	Επίπεδο 2	Επίπεδο 3	Επίπεδο 4	Επίπεδο 5
Αποτελεσματικότητα Θέσης	0	15	28	48	97

Πίνακας 4. Αριθμητικά αποτελέσματα προσομοίωσης για προορισμούς που κατανέμονται ομοιόμορφα ανά περιοχές



Γράφημα 6. Δείκτης αποτελεσματικότητας θέσης ως προς επίπεδο προστασίας ιδιωτικότητας θέσης για ομοιόμορφη ανά περιοχές κατανομή των προορισμών

Ισχύουν και εδώ τα όσα ειπώθηκαν στην παράγραφο 5.4.1. Η μορφή της καμπύλης παραμένει ίδια. Συγκρίνοντας τα αποτελέσματα με εκείνα της παραγράφου 5.4.1 μπορούμε να πούμε ότι η ομοιόμορφη κατανομή ανά περιοχές δίνει παραπλήσια αποτελέσματα με την Γκαουσιανή ανά περιοχές όντας λίγο χειρότερη σε αποτελεσματικότητα. Αυτό συμβαίνει διότι, όπως θα δούμε και στην περίπτωση της

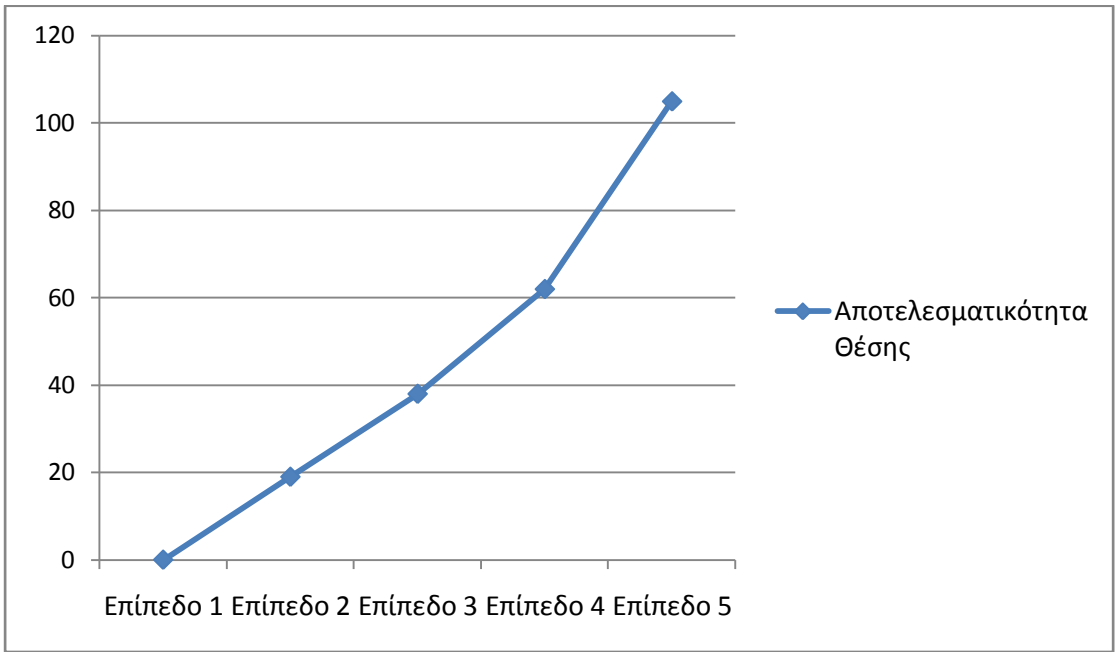
ομοιόμορφης κατανομής παντού, όσο περισσότερο συγκεντρωμένοι είναι οι προορισμοί σε ένα μέρος, τόσο καλύτερη η αποτελεσματικότητα θέσης που επιτυγχάνει η υπηρεσία. Για οποιαδήποτε θέση αναφοράς χρήστη, η υπηρεσία προτείνει προορισμούς που βρίσκονται κοντά σε αυτούς της πλήρους ακρίβειας, αφού όλοι οι προορισμοί είναι συγκεντρωμένοι. Δεδομένου ότι η ομοιόμορφη ανά περιοχές κατανομή είναι λίγο πιο απλωμένη από την Γκαουσιανή, τα αποτελέσματα που πήραμε δικαιολογούνται πλήρως. Η καμπύλη που παίρνουμε είναι γραμμική μέχρι το επίπεδο 4, περισσότερο δηλαδή από εκείνη της Γκαουσιανής κατανομής, όπως και αναμενόταν.

5.4.3 Ομοιόμορφη Κατανομή Παντού

Στην περίπτωση αυτή οι προορισμοί κατανέμονται ομοιόμορφα σε ολόκληρη την πόλη. Τα αποτελέσματα παρουσιάζονται στη συνέχεια, με το επίπεδο 1 να δηλώνει την ακριβή θέση και το επίπεδο αοριστίας να αυξάνει για τα επόμενα επίπεδα.

	Επίπεδο 1	Επίπεδο 2	Επίπεδο 3	Επίπεδο 4	Επίπεδο 5
Αποτελεσματικότητα Θέσης	0	19	38	62	111

Πίνακας 5. Αριθμητικά αποτελέσματα προσομοίωσης για προορισμούς που κατανέμονται ομοιόμορφα ανά περιοχές



Γράφημα 7. Δείκτης αποτελεσματικότητας θέσης ως προς επίπεδο προστασίας ιδιωτικότητας θέσης για ομοιόμορφη παντού κατανομή των προορισμών

Τα συμπεράσματα των παραγράφων 5.4.1 και 5.4.2 επεκτείνονται και στην περίπτωση της ομοιόμορφης κατανομής σε ολόκληρη την πόλη. Δεδομένου ότι αυτή η κατανομή είναι πολύ πιο απλωμένη από τις προηγούμενες, παρατηρείται μειωμένη αποτελεσματικότητα και σχεδόν γραμμική καμπύλη. Τα αποτελέσματα αυτά μας δείχνουν ότι για μια τέτοια κατανομή, η αποτελεσματικότητα θέσης μιας υπηρεσίας

επηρεάζεται περισσότερο σε σχέση με τις συγκεντρωμένες κατανομές των παραγράφων 5.4.1 και 5.4.2. Από την άλλη πλευρά η κατανομή αυτή προστατεύει καλύτερα την ιδιωτικότητα θέσης των χρηστών, αφού ναι μεν η παρούσα θέση αναφέρεται με την ίδια αοριστία, ο επόμενος προορισμός όμως είναι δύσκολο να προβλεφθεί με ακρίβεια, αφού η κατανομή του είναι απλωμένη σε ολόκληρη την πόλη.

5.5 Σύνοψη Αποτελεσμάτων - Συμπεράσματα

Για την μελέτη της σχέσης προστασίας ιδιωτικότητας-αποτελεσματικότητας υπηρεσίας διακρίναμε δύο είδη ιδιωτικότητας και αποτελεσματικότητας. Το πρώτο είδος αφορά την θέση, ενώ το δεύτερο τις προτιμήσεις. Αποδείξαμε ότι η αποτελεσματικότητα ενός είδους σχετίζεται μόνο με την προστασία ιδιωτικότητας του ιδίου είδους, ενώ είναι ανεξάρτητη από αυτήν του άλλου. Έτσι μελετήσαμε τις σχέσεις ιδιωτικότητα θέσης-αποτελεσματικότητα θέσης και ιδιωτικότητα προτιμήσεων-αποτελεσματικότητα προτιμήσεων.
Αυξανομένης της προστασίας ιδιωτικότητας θέσης, δηλαδή αναφέροντας ο χρήστης την θέση του σε υψηλό επίπεδο αοριστίας, το επίπεδο της αποτελεσματικότητας μειώνεται, δηλαδή οι προορισμοί που επιστρέφονται βρίσκονται μακρύτερα σε σχέση με αυτούς της πλήρους ακρίβειας και αντίστροφα. Ο κυριότερος παράγοντας που επηρεάζει αυτή τη σχέση είναι η κατανομή των πιθανών προορισμών των χρηστών στην πόλη. Εξετάστηκαν τρεις κατανομές οι οποίες σε σειρά φθίνουσας αποτελεσματικότητας για δεδομένη προστασία ιδιωτικότητας ήταν: Γκαουσιανή κατά περιοχές, ομοιόμορφη κατά περιοχές και ομοιόμορφη σε όλη την πόλη. Η κατάσταση αντιστρέφεται όσον αφορά την προστασία ιδιωτικότητας θέσης που προσφέρει η κάθε κατανομή με την ομοιόμορφη παντού να είναι η πιο ασφαλής για τους χρήστες και οι ομοιόμορφη κατά περιοχές και Γκαουσιανή να ακολουθούν με αυτή τη σειρά. Καταλήγουμε λοιπόν στο γενικό συμπέρασμα ότι όσο πιο απλωμένη στο χώρο είναι μια κατανομή, τόσο περισσότερη προστασία ιδιωτικότητας προσφέρει, αλλά χάνει σε αποτελεσματικότητα υπηρεσίας.
Όσον αφορά τη σχέση ιδιωτικότητας-αποτελεσματικότητας προτιμήσεων, ο παράγων που εξετάστηκε ήταν η μορφή του δέντρου προτιμήσεων που χρησιμοποιεί η υπηρεσία. Για εκτενές δέντρο συμπεραίνουμε ότι παρόλο που η αποτελεσματικότητα-ποιότητα υπηρεσίας για πλήρη ακρίβεια είναι ανώτερη εκείνης για απλούστερο δέντρο, εντούτοις καθώς το επίπεδο αοριστίας προτιμήσεων αυξάνει, η αποτελεσματικότητα της υπηρεσίας μειώνεται ταχύτερα απ’ ότι η αντίστοιχη για το απλούστερο δέντρο. Έτσι, το εκτενές δέντρο προσφέρει αποτελεσματικότητα υπηρεσίας εκεί που η προστασία ιδιωτικότητας δεν μας ενδιαφέρει τόσο, ενώ το απλούστερο συνίσταται σε περιπτώσεις όπου η προστασία των προσωπικών δεδομένων των χρηστών είναι το σπουδαιότερο.
Πέρα από την θεωρητική μελέτη της σχέσεως ιδιωτικότητας-αποτελεσματικότητας ο κώδικας της προσομοίωσης μπορεί να χρησιμοποιηθεί και για την αξιολόγηση μιας πραγματικής υπηρεσίας, η οποία προσφέρεται σε κάποια πόλη. Οι προορισμοί σε αυτήν την περίπτωση δεν κατανέμονται τυχαία ακολουθώντας κάποια κατανομή από τις διαθέσιμες της προσομοίωσης, αλλά οι θέσεις τους είναι οι πραγματικές και συγκεκριμένες. Η συμπεριφορά των χρηστών μπορεί να παραμετροποιηθεί βάσει των στοιχείων που διαθέτει ο πάροχος και να επιλεγεί ένα κατάλληλο δέντρο προτιμήσεων έτσι ώστε να καλύπτονται οι προδιαγραφές προστασίας ιδιωτικότητας και αποτελεσματικότητας που έχουν τεθεί.

Παράρτημα:Κώδικας

Ακολουθεί ο κώδικας που έχει γραφτεί και τροποποιηθεί στην κατεύθυνση της υλοποίησης της προσομοίωσης. Ο κώδικας γράφτηκε σε γλώσσα Java σε περιβάλλον Eclipse. Ο παρατιθέμενος κώδικας καλύπτει τις ακόλουθες βασικές κλάσεις:
Agent Class, Place Class, AgentTypeCharacteristics Class, PlaceTypeCharacteristics Class και AgentModel Class.

Agent.java

```
public class Agent implements Trackable {

	private static final int WANDER_TURN = 3;
	private static final int DEFAULT_SOBERNESS = 5;
	private static final int POSSIBLE_DIRECTIONS = 8;
	private static World world;
	private static boolean infoFieldsLocked = false;
	private static final SortedSet<String> INFO_FIELDS =
			new TreeSet<String>();
	private static final Random RAND = new Random();
	private Boolean onAuto = true;
	private String name;
	private int dir;
	private int speed = 1;
	private Position pos;
	private boolean atDestination;
	private Place destination;
	private String image;
	private String previousImage;
	private SortedMap<String, Publishable> info;
	private boolean visible = true;
	private Place nikosDest;
	private boolean willWait;
	private Place avoidDest;
	public Node[] pathArray;

	public Agent(final String name, final Position start,
			final String image, final World world) {
		basicChecks(world);
		this.name = name;
		this.info = new TreeMap<String, Publishable>();
		this.dir = 0;
		this.image = image;
		this.previousImage = image;
		this.pos = start;
		this.destination = null;
		this.atDestination = true;

	}

	private void basicChecks(final World thisAgentsWorld) {
		if (Agent.world == null) {
			throw new InitializationRequiredException(
					"You need to initialize the agents.");
		}
		if (Agent.world != thisAgentsWorld) {
```

```
                  throw new RuntimeException(
                              "All your users must belong to the same
world");
            }
      }

      public Agent(final Position start, final String image,
                  final World world) {
            this(SequentialNamer.getNextName(), start, image, world);
      }

      public static void lockInfoFields() {
            infoFieldsLocked = true;
      }

      public boolean isAtDestination() {
            return atDestination;
      }

      public void setAtDestination(boolean atDestination){
            this.atDestination = atDestination;
      }

      public String toString() {
            return getName();
      }

      public void setName(final String name) {
            this.name = name;
      }

      public String getName() {
            return name;
      }

      public String getImage() {
            return image;
      }

      public void setImage(final String image) {
            if (!world.getAvailableSprites().contains(image)) {
                  throw new UnexistingSpriteException(image);
            }
            this.previousImage = this.image;
            this.image = image;
      }

      public void setPreviousImage() {
            this.image = this.previousImage;
      }

      public Place getDestination() {
            return destination;
      }

      public Place getNikosDest(){
            return nikosDest;
      }

      public Place getAvoidDest(){
            return avoidDest;
```

```
	}

	public boolean getWillWait(){
		return willWait;
	}

	public void setNikosDest(Place dest){
		this.nikosDest = dest;
	}

	public void setAvoidDest(Place dest){
		this.avoidDest = dest;
	}

	public void setWillWait(boolean wait){
		this.willWait = wait;
	}

	public int getSpeed() {
		return speed;
	}

	public void setSpeed(final int speed) {
		this.speed = speed;
	}

	public void setDestination(final Place destination) {
		if (this.destination != null
				&& this.destination.equals(destination)) {
			return;
		} else {
			atDestination = false;
			this.destination = destination;
		}
	}

	public int getDir() {
		return dir;
	}

	public void setDir(final int newDir) {
		dir = newDir % POSSIBLE_DIRECTIONS;
		if (dir < 0) {
			dir += POSSIBLE_DIRECTIONS;
		}
	}

	public Position getPos() {
		return pos;
	}

	public void setPos(final Position pos) {
		this.pos = pos;
	}

	public Publishable set(final String key, final Publishable
value) {
		if (infoFieldsLocked && !info.containsKey(key)) {
			throw new InfoFieldsLockedException(key);
		}
		INFO_FIELDS.add(key);
```

```
            return info.put(key, value);
      }

      public static Set<String> getInfoKeys() {
            return new TreeSet<String>(INFO_FIELDS);
      }

      public Collection<Publishable> getInfoValues() {
            return info.values();
      }

      public Publishable get(final String key) {
            if (!info.containsKey(key)) {
                  throw new InfoUndefinedException(key);
            }

            return info.get(key);
      }

      public void turn(final int turn) {
            setDir(dir + turn);
      }

      public void moveTowardsDestination() {
            if (isAtDestination()) {
                  return;
            } else {
                  for (int i = 0; i < speed; i++) {
                        moveTowardsPlace(destination);
                        if (pos.equals(destination.getPos())) {
                              destination = null;
                              atDestination = true;
                              break;
                        }
                  }
            }
      }

      public void moveInDirection(final int moveDir)
                  throws PositionUnreachableException {
            if (moveDir == -1) {
                  return;
            }
            pos = pos.calculateMove(moveDir);
            this.dir = moveDir;
      }

      public void moveTowardsPlace(final Place place) {
            try {
                  moveInDirection(place.pointFrom(pos, dir));
            } catch (PositionUnreachableException e) {
                  System.err.println("Agent '" + this + "' can't
reach '" + place + "' at '" + pos + "'");
            }
      }

      public void wanderAround(final Place place, final int radius,
                  final int soberness) {
            if (place.distanceFrom(pos) > radius) {
                  moveTowardsPlace(place);
            } else {
```

```
                wander(soberness);
            }
        }

        public void wanderAround(final Place place, final int radius) {
            wanderAround(place, radius, DEFAULT_SOBERNESS);
        }

        public void wander() {
            wander(DEFAULT_SOBERNESS);
        }

        public void wander(final int soberness) {
            boolean stuck = true;
            int searchDir =
                        (RAND.nextInt(2) == 1) ? (-WANDER_TURN) :
WANDER_TURN;
            int tries = 0;
            while (stuck && (tries < POSSIBLE_DIRECTIONS)) {
                Position target = null;
                try {
                    target = pos.calculateMove(dir);
                    pos = target;
                    stuck = false;
                } catch (PositionUnreachableException e) {
                    turn(searchDir);
                    tries++;
                }
            }
            if (tries == POSSIBLE_DIRECTIONS) {
                System.err.println("My name's " + name
                            + " and you've got me stuck in a "
                            + "one pixel wide room! I have
rights!");
            }
            if (RAND.nextInt(soberness) == 0) {
                if (RAND.nextInt(2) == 1) {
                    turn(1);
                } else {
                    turn(-1);
                }
            }
        }

        public FlatData getContext(final String ctxName)
                throws UnknownContextException {
            if (info.containsKey(ctxName)) {
                return info.get(ctxName).flatten();
            } else if (world.getOverlays().containsKey(ctxName)) {
                return
world.getOverlays().get(ctxName).getValue(pos)
                            .flatten();
            } else if (ctxName.equals("Time")) {
                return new Text("" +
world.getTime().getTimeInMillis())
                            .flatten();
            } else if (ctxName.equals("Name")) {
                return new Text(name).flatten();
            } else if (ctxName.equals("Position")) {
                return pos.flatten();
            } else if (ctxName.equals("atDestination")) {
```

```
                  return new Text(new
Boolean(atDestination).toString())
                              .flatten();
            } else if (ctxName.equals("Destination")) {
                  return destination.flatten();
            } else {
                  throw new UnknownContextException(ctxName);
            }
      }

      public synchronized void returnControl() {
            onAuto = true;
      }

      public synchronized void getControl() {
            setVisible(true);
            onAuto = false;
      }

      public synchronized boolean isOnAuto() {
            return onAuto;
      }

      public boolean isVisible() {
            return visible;
      }

      public void setVisible(final boolean visible) {
            this.visible = visible;
      }

      public static void initialize(final World agentsWorld) {
            world = agentsWorld;
            infoFieldsLocked = false;
      }
}
```

Place.java

```
public class Place implements Trackable, Publishable, Overlayable {

      private static PersistentCachedMap gradients;
      private static World world;
      private String name;
      private String type;
      private Position pos;
      private Gradient temporaryGradient;
      private boolean visible = true;
      public Node[] pathArray;
      public int pathArrayLength;
      private SortedMap<String, Publishable> info;

      public static void initialize(final World newWorld) {
            Place.world = newWorld;
            gradients = null;
            gradients = new
SiafuGradientCache(Controller.DEFAULT_GRADIENT_PATH,
```

```
                                    world.getWorldName(),
World.getCacheSize(),World.shouldPrefillCache());
      }

      public Place(final String type, final Position pos, final World
world,final Position relevantPosition) {
            this(type, pos, world, type + "-" + pos,
relevantPosition);
      }

      public Place(final String type, final Position pos, final World
world) {
            this(type, pos, world, type + "-" + pos, null);
      }

      public Place(final String type, final Position pos, final World
world,final String name) {
            this(type, pos, world, name, null);
      }

      public Place(final String type, final Position pos, final World
world,final String name, final Position relevantPosition) {
            basicChecks(world);
            this.info = new TreeMap<String, Publishable>();
            this.type = type;
            this.pos = pos;
            this.name = name;
            world.addPlaceType(type);
            if (relevantPosition != null) {
                  temporaryGradient = new Gradient(pos, world,
relevantPosition);
            } else if (!gradients.containsKey(pos.toString())) {
                  gradients.put(pos, new Gradient(pos, world));
            }
            Random rand = new Random();
            String a = new Integer(rand.nextInt(5)+1).toString();
            this.set("price",new Text(a));
            a = new Integer(rand.nextInt(5)+1).toString();
            this.set("quality",new Text(a));
            System.out.println(type);
            if (type.equals("Cinema")){
                  try{
                      DocumentBuilderFactory docBuilderFactory =
DocumentBuilderFactory.newInstance();
                      DocumentBuilder docBuilder =
docBuilderFactory.newDocumentBuilder();
                      Document doc = docBuilder.parse (new
File("C:\\Users\\Nikos\\workspace\\tre.xml"));
                      Node currentNode = doc.getDocumentElement();
                      Node nodeArray[]=new Node[10];;
                      int k=0;
                      int l;
                      nodeArray[0]=currentNode;
                      while (currentNode.getChildNodes().getLength()-
2 > 0){
                     System.out.print(currentNode.getNodeName()+"---
");
                        l =
rand.nextInt((currentNode.getChildNodes().getLength()-1)/2);
                        currentNode =
currentNode.getChildNodes().item(2*l+1);
```

```
                        k=k+1;
                        nodeArray[k]=currentNode;

                      }
                      this.pathArray = nodeArray;
                      this.pathArrayLength = k;
                      System.out.println(currentNode.getNodeName()+"
leaf");
                      for (int i=0;i<=k;i=i+1){

      System.out.println(pathArray[i].getNodeName());
                      }
                      }
                      catch (SAXParseException err) {
                      System.out.println ("** Parsing error" + ",
line " + err.getLineNumber () + ", uri " + err.getSystemId ());
                        System.out.println(" " + err.getMessage ());
                        }
                        catch (SAXException e) {
                        Exception x = e.getException ();
                        ((x == null) ? e : x).printStackTrace ();
                        }
                        catch (Throwable t) {
                        t.printStackTrace ();
                        }
                  }
      }

      private void basicChecks(final World thisPlacesWorld) {
            if (gradients == null) {
                  throw new InitializationRequiredException(
                              "You need to initialize the Place
class");
            }
            if (Place.world != thisPlacesWorld) {
                  throw new RuntimeException(
                              "All your places must belong to the
same World.");
            }
      }

      public Place(final FlatData flatData) {
            final int nameField = 0;
            final int typeField = 1;
            final int latField = 2;
            final int lonField = 3;
            String data = flatData.getData();
            TypeUtils.check(this, data);
            String place = data.substring(data.indexOf(':') + 1);
            String[] part = place.split("#");
            this.name = part[nameField];
            this.type = part[typeField];
            double lat = new Double(part[latField]);
            double lon = new Double(part[lonField]);
            this.pos = new Position(lat, lon);
            this.info = new TreeMap<String, Publishable>();
            world.addPlaceType(this.type);
            if (!gradients.containsKey(pos.toString())) {
            gradients.put(pos, new Gradient(pos, world));
            }
```

```
	}

	public String toString() {
		return getName();
	}

	public String getName() {
		return name;
	}

	public void setName(final String name) {
		this.name = name;
	}

	public String getType() {
		return type;
	}

	public Position getPos() {
		return pos;
	}

	public Publishable set(final String key, final Publishable
value) {
		return info.put(key, value);
	}

	public Publishable get(final String key) {
		if (!info.containsKey(key)) {
			throw new InfoUndefinedException(key);
		}
		return info.get(key);
	}
	public Set<String> getInfoKeys() {
		return info.keySet();
	}

	public Collection<Publishable> getInfoValues() {
		return info.values();
	}

	public static void fillCache(final int size) {
		if (gradients == null) {
			throw new RuntimeException(
					"Instanciate the PersistentCachedMap
first.");
		} else {
			gradients.fillCache(size);
		}
	}

	public Gradient getGradient() {
		if (temporaryGradient != null) {
			return temporaryGradient;
		} else {
			return (Gradient) gradients.get(pos.toString());
		}
	}

	public int pointFrom(final Position targetPos) {
		return getGradient().pointFrom(targetPos);
```

```
	}

	public int pointFrom(final Position targetPos, final int
preferedDir) {
		return getGradient().pointFrom(targetPos, preferedDir);
	}

	public int distanceFrom(final Position targetPos) {
		return getGradient().distanceFrom(targetPos);
	}

	public FlatData flatten() {
		String data;
		double[] coords = pos.getCoordinates();
		data = this.getClass().getName() + ":";
		data += name + "#";
		data += type + "#";
		data += coords[0] + "#" + coords[1];
		return new FlatData(data);
	}

	public boolean isVisible() {
		return visible;
	}

	public void setVisible(final boolean visible) {
		this.visible = visible;
	}
}
```

AgentTypeCharacteristics.java

```
public class AgentTypeCharacteristics {

	private String name;
	private int amountOfAgents;
	private Place HOME;
	private Place OFFICE;
	private String image;
	private boolean hasJob;
	private EasyTime workStart;
	private int workStartBlur;
	private int workDuration;
	private int workDurationBlur;
	private int locationPrivacy;
	private int preferencesPrivacy;
	private int priceWeightPref;
	private int priceWeightPrefBlur;
	private int distWeightPref;
	private int distWeightPrefBlur;
	private int qualityWeightPref;
	private int qualityWeightPrefBlur;
	private int musicTypeWeightPref;
	private int musicTypeWeightPrefBlur;
	private Boolean hasCar;
	private int REFILL_INTERVAL;
	private int REFILL_BLUR;
```

```
	private float WILL_REQUEST_THRESHOLD;
	private float SATISFIED_THRESHOLD;
	private int MIN_PRIVACY;
	private float PRIVACY_VIOLATION_OVERCOMED;
	private int MAX_DESTINATION;
	private int MAX_DISTANCE_ON_FOOT;
	private int MAX_WAITING_INTERVAL;
	private final TreeMap<String, Occupation> occupations = new
TreeMap<String, Occupation>();

	public AgentTypeCharacteristics(){}

	public void printProperties(){
		System.out.println();
		System.out.println("Type Name:" + getName());
		System.out.println("Amount of Agents:" +
getAmountOfAgents());
		System.out.println("Image:" + getImage());
		System.out.println("Has Job:" + getHasJob());
		if(getHasJob()){
			System.out.println("WorkStartHour:" +
getWorkStart().getHour());
			System.out.println("WorkStartMinutes:" +
getWorkStart().getMinute());
			System.out.println("WorkStartBlur:" +
getWorkStartBlur());
			System.out.println("WorkDuration:" +
getWorkDuration());
			System.out.println("DurationBlur:" +
getWorkDurationBlur());
		}
		System.out.println("Has Car:" + getHasCar());
		if(getHasCar()){
			System.out.println("Refill interval:" +
getREFILL_INTERVAL());
			System.out.println("Refill blur:" +
getREFILL_BLUR());
		}
		System.out.println();
		System.out.println("Occupations");

	System.out.println("************************************");
		for(Occupation occupation: getOccupations().values()){
			System.out.println(occupation.toString());

		}
		System.out.println();
		System.out.println("----------");
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public int getAmountOfAgents() {
		return amountOfAgents;
	}
```

```
public void setAmountOfAgents(int amountOfAgents) {
	this.amountOfAgents = amountOfAgents;
}

public Place getHOME() {
	return HOME;
}

public void setHOME(Place home) {
	HOME = home;
}

public int getMAX_DESTINATION() {
	return MAX_DESTINATION;
}

public void setMAX_DESTINATION(int max_destination) {
	MAX_DESTINATION = max_destination;
}

public int getMAX_DISTANCE_ON_FOOT() {
	return MAX_DISTANCE_ON_FOOT;
}

public void setMAX_DISTANCE_ON_FOOT(int max_distance_on_foot) {
	MAX_DISTANCE_ON_FOOT = max_distance_on_foot;
}

public int getMAX_WAITING_INTERVAL() {
	return MAX_WAITING_INTERVAL;
}

public void setMAX_WAITING_INTERVAL(int max_waiting_interval) {
	MAX_WAITING_INTERVAL = max_waiting_interval;
}

public Place getOFFICE() {
	return OFFICE;
}

public void setOFFICE(Place office) {
	OFFICE = office;
}

public Boolean getHasCar() {
	return hasCar;
}

public void setHasCar(Boolean hasCar) {
	this.hasCar = hasCar;
}

public int getREFILL_BLUR() {
	return REFILL_BLUR;
}

public void setREFILL_BLUR(int refill_blur) {
	REFILL_BLUR = refill_blur;
}
```

```
	public int getREFILL_INTERVAL() {
		return REFILL_INTERVAL;
	}

	public void setREFILL_INTERVAL(int refill_interval) {
		REFILL_INTERVAL = refill_interval;
	}

	public float getSATISFIED_THRESHOLD() {
		return SATISFIED_THRESHOLD;
	}

	public void setSATISFIED_THRESHOLD(float satisfied_threshold) {
		SATISFIED_THRESHOLD = satisfied_threshold;
	}

	public float getWILL_REQUEST_THRESHOLD() {
		return WILL_REQUEST_THRESHOLD;
	}

	public void setWILL_REQUEST_THRESHOLD(float
will_request_threshold) {
		WILL_REQUEST_THRESHOLD = will_request_threshold;
	}

	public TreeMap<String,Occupation> getOccupations() {
		return occupations;
	}

	public String getImage() {
		return image;
	}

	public void setImage(String image) {
		this.image = image;
	}

	public boolean getHasJob() {
		return hasJob;
	}

	public void setHasJob(boolean hasJob) {
		this.hasJob = hasJob;
	}

	public int getWorkDuration() {
		return workDuration;
	}

	public void setWorkDuration(int workDuration) {
		this.workDuration = workDuration;
	}

	public int getWorkDurationBlur() {
		return workDurationBlur;
	}

	public void setWorkDurationBlur(int workDurationBlur) {
		this.workDurationBlur = workDurationBlur;
	}
```

```
	public int getWorkStartBlur() {
		return workStartBlur;
	}

	public void setWorkStartBlur(int workStartBlur) {
		this.workStartBlur = workStartBlur;
	}

	public EasyTime getWorkStart() {
		return workStart;
	}

	public void setWorkStart(EasyTime workStart) {
		this.workStart = workStart;
	}

	public void setLocationPrivacy(int locationPrivacy) {
		this.locationPrivacy = locationPrivacy;
	}

	public void setPreferencesPrivacy(int preferencesPrivacy) {
		this.preferencesPrivacy = preferencesPrivacy;
	}

	public void setPriceWeightPref(int priceWeightPref) {
		this.priceWeightPref = priceWeightPref;
	}

	public void setPriceWeightPrefBlur(int priceWeightPrefBlur) {
		this.priceWeightPrefBlur = priceWeightPrefBlur;
	}

	public void setDistWeightPref(int distWeightPref) {
		this.distWeightPref = distWeightPref;
	}

	public void setDistWeightPrefBlur(int distWeightPrefBlur) {
		this.distWeightPrefBlur = distWeightPrefBlur;
	}

	public void setMusicTypeWeightPref(int musicTypeWeightPref) {
		this.musicTypeWeightPref = musicTypeWeightPref;
	}

	public void setMusicTypeWeightPrefBlur(int
musicTypeWeightPrefBlur) {
		this.musicTypeWeightPrefBlur = musicTypeWeightPrefBlur;
	}

	public void setQualityWeightPref(int qualityWeightPref) {
		this.qualityWeightPref = qualityWeightPref;
	}

	public void setQualityWeightPrefBlur(int qualityWeightPrefBlur)
{
		this.qualityWeightPrefBlur = qualityWeightPrefBlur;
	}

	public int getLocationPrivacy() {
		return locationPrivacy;
	}
```

```
	public int getPreferencesPrivacy() {
		return preferencesPrivacy;
	}

	public int getPriceWeightPref() {
		return priceWeightPref;
	}

	public int getPriceWeightPrefBlur() {
		return priceWeightPrefBlur;
	}

	public int getDistWeightPref() {
		return distWeightPref;
	}

	public int getDistWeightPrefBlur() {
		return distWeightPrefBlur;
	}

	public int getMusicTypeWeightPref() {
		return musicTypeWeightPref;
	}

	public int getMusicTypeWeightPrefBlur() {
		return musicTypeWeightPrefBlur;
	}

	public int getQualityWeightPref() {
		return qualityWeightPref;
	}

	public int getQualityWeightPrefBlur() {
		return qualityWeightPrefBlur;
	}

}
```

PlaceTypeCharacteristics.java

```
public class PlaceTypeCharacteristics {

	private String name;
	private EasyTime open;
	private int openBlur;
	private EasyTime close
	private HashMap<String, Integer> unavailabilityTime =
		new HashMap<String, Integer>();
	private int timeBlur;
	private HashMap<String, Float> unavailability =
		new HashMap<String, Float>();
	private TreeMap<EasyTime, String> trafficStates =
		new TreeMap<EasyTime, String>();

	public void printProperties(){
		System.out.println();
		System.out.println("Name:" + getName());
```

```
            System.out.println("Open Hour:" + getOpen().getHour());
            System.out.println("Open Minutes:" +
getOpen().getMinute());
            System.out.println("Open Blur:" + getOpenBlur());
            System.out.println("Close Hour:" + getClose().getHour());
            System.out.println("Close Minutes:" +
getClose().getMinute());
            System.out.println("Close Blur:" + getCloseBlur());
            System.out.println();
            System.out.println("Traffic parameters");
            System.out.println();
            for(String traffic: getUnavailability().keySet()){
                  System.out.println(traffic + " Traffic
Unavailability:"
                              + getUnavailability().get(traffic));
                  System.out.println(traffic + " Unavailability
Time:"
                              +
getUnavailabilityTime().get(traffic));
                  System.out.println();
            }
            System.out.println();
            System.out.println("Traffic States");
            System.out.println();
            for(EasyTime time: getTrafficStates().keySet()){
                  System.out.println("Traffic Hour:" +
time.getHour());
                  System.out.println("Traffic Minutes:" +
time.getMinute());
                  System.out.println("Traffic:" +
getTrafficStates().get(time));
                  System.out.println();
            }
            System.out.println();
            System.out.println("-------");
      }

      public EasyTime getClose() {
            return close;
      }

      public void setClose(EasyTime close) {
            this.close = close;
      }

      public int getCloseBlur() {
            return closeBlur;
      }

      public void setCloseBlur(int closeBlur) {
            this.closeBlur = closeBlur;
      }

      public String getName() {
            return name;
      }

      public void setName(String name) {
            this.name = name;
      }
```

```
	public EasyTime getOpen() {
		return open;
	}

	public void setOpen(EasyTime open) {
		this.open = open;
	}

	public int getOpenBlur() {
		return openBlur;
	}

	public void setOpenBlur(int openBlur) {
		this.openBlur = openBlur;
	}

	public int getTimeBlur() {
		return timeBlur;
	}

	public void setTimeBlur(int timeBlur) {
		this.timeBlur = timeBlur;
	}

	public HashMap<String, Integer> getUnavailabilityTime() {
		return unavailabilityTime;
	}

	public TreeMap<EasyTime, String> getTrafficStates() {
		return trafficStates;
	}

	public HashMap<String, Float> getUnavailability() {
		return unavailability;
	}
}
```

AgentModel.java

```
public class AgentModel extends BaseAgentModel {

	private static final Random RAND = new Random();
	private static final int TOP_SPEED = 10;
	private EasyTime now;
	private static HashMap<String, AgentTypeCharacteristics>
agentTypes;
	private Configuration config;
	private boolean useSpatialCloak;
	private boolean useStaticSpatialCloak;
	private boolean useDynamicSpatialCloak;

	public AgentModel(final World world) {
		super(world);
		config = world.getSimulationConfig();
		agentTypes = setAgentTypes();
		for(AgentTypeCharacteristics characteristics:
agentTypes.values()){
			characteristics.printProperties();
		}
```

```
            setUseSpatialCloak();
            setUseStaticSpatialCloak();
            setUseDynamicSpatialCloak();
            placeConsistencyCheck();
      }

      private void placeConsistencyCheck(){
            HashSet<String> occupationsPlaceTypes = new
HashSet<String>();
            for(AgentTypeCharacteristics characteristics:
agentTypes.values()){
                  for(Occupation
occupation:characteristics.getOccupations().values()){
      occupationsPlaceTypes.add(occupation.getOccupationPlace());
                  }
            }
            System.out.println("********");
            System.out.println();
            System.out.println();
            System.out.println("OCCUPATION PLACE CONSISTENCY
CHECK:");
            System.out.println("------------");
            System.out.println("Occupation Place Types:" +
occupationsPlaceTypes.toString());
            System.out.println("World's Place Types:" +
world.getPlaceTypes().toString());
            System.out.println();
            System.out.println();
            System.out.println("**********");
            for(String userDefinedPlaceType: occupationsPlaceTypes){
      if(!world.getPlaceTypes().contains(userDefinedPlaceType)){
                        System.out.println("The "
+userDefinedPlaceType + " type of place "+ "does not exist on the
world. \nYou must set the occupation that uses " + "\nthese types of
places to use one of the world's place types: \n" +
world.getPlaceTypes().toString());
                        throw new RuntimeException("The " +
userDefinedPlaceType + " type of place \n" + "does not exist on the
world. \nYou must set the occupation that uses " + "these types of
places \nto use one of the worlds place types: " +
world.getPlaceTypes().toString());
                  }
            }
      }

      public ArrayList<Agent> createAgents() {
            ArrayList<Agent> people = new ArrayList<Agent>();
            for(AgentTypeCharacteristics characteristics:
agentTypes.values()){
                  int amountOfAgents =
characteristics.getAmountOfAgents();
                  String image = characteristics.getImage();
                  String agentType = characteristics.getName();

                  Position startPosition = new Position(0,0);
                  for(int i = 0; i < amountOfAgents; i++){
                        Agent newAgent = new Agent(startPosition,
image, world);
                        newAgent.setSpeed(8);
                        int privacy;
                        int mean;
```

```
                        int blur;
                        int pref;
                        String Pref;
                        Random rand = new Random();

                    mean = characteristics.getPriceWeightPref();
                    blur =
characteristics.getPriceWeightPrefBlur();
                    pref = mean + rand.nextInt(blur)-(blur/2);
                    Pref = new Integer(pref).toString();

                    newAgent.set("priceWeightPref",new Text(Pref));

                    mean = characteristics.getDistWeightPref();
                    blur = characteristics.getDistWeightPrefBlur();

                    pref = mean + rand.nextInt(blur)-(blur/2);
                    Pref = new Integer(pref).toString();

                    newAgent.set("distWeightPref",new Text(Pref));

                    mean = characteristics.getQualityWeightPref();
                    blur
=characteristics.getQualityWeightPrefBlur();
                    pref = mean + rand.nextInt(blur)-(blur/2);
                    Pref = new Integer(pref).toString();

                    newAgent.set("qualityWeightPref",new
Text(Pref));
                    mean =characteristics.getMusicTypeWeightPref();
                    blur =
characteristics.getMusicTypeWeightPrefBlur();
                    pref = mean + rand.nextInt(blur)-(blur/2);
                    Pref = new Integer(pref).toString();

                    newAgent.set("musicWeightPref",new Text(Pref));
                    privacy = characteristics.getLocationPrivacy();

                      newAgent.set("locationPrivacy",new Text(new
Integer(privacy).toString()));
                      privacy =
characteristics.getPreferencesPrivacy();
                      newAgent.set("preferencesPrivacy",new
Text(new Integer(privacy).toString()));
                      newAgent.set(TYPE, new Text(agentType));
                      newAgent.set(ACTIVITY,
Activity.CHOOSE_NEXT_DESTINATION);
                      newAgent.set(OCCUPATION, new Text("no
occupation cause simulation just started"));
                      newAgent.set(NEXT_EVENT, new Text("no next
event cause simulation just started"));
                      int startHour =
config.getInt("starttime.hour");
                      int startMinute =
config.getInt("starttime.minute");
                      newAgent.set(PREVIOUS_EVENT, new
EasyTime(startHour, startMinute));
                      setHome(newAgent);
      newAgent.setPos(((Place)newAgent.get(HOME)).getPos());
                      if(agentTypes.get(agentType).getHasJob()){
                            setOffice(newAgent);
```

```
                              EasyTime workStart = new
EasyTime(agentTypes.get(agentType).getWorkStart());

      workStart.blur(agentTypes.get(agentType).getWorkStartBlur());
                              newAgent.set(WORK_START, workStart);
                              int workDuration =
agentTypes.get(agentType).getWorkDuration();
                              int workDurationBlur=
agentTypes.get(agentType).getWorkDurationBlur();
                              EasyTime workEnd = new
EasyTime(workStart).shift(workDuration,0).blur(workDurationBlur);
                              newAgent.set(WORK_END, workEnd);
                        }
                        setMaxDestination(newAgent);
                        setMinSuggestions(newAgent);
                        setSpatialPerturbation(newAgent);
                        setStaticSpatialCloakRange(newAgent);
                        setMinNearAgents(newAgent);
                        setUnexpectedFactor(newAgent);
                        setWillRequestService(newAgent);
                        setWaitingTime(newAgent);
                        setWaitingBlur(newAgent);
                        setWillWander(newAgent);
                        setWanderDuration(newAgent);
                        setWillViolatePrivacy(newAgent);
                        people.add(newAgent);
                  }
            }
            for(Agent agent: people){
                  Iterator<String> key =
Agent.getInfoKeys().iterator();
                  Iterator<Publishable> value =
agent.getInfoValues().iterator();
                  while(key.hasNext()){
                        System.out.println(key.next() + ":" +
value.next().toString());
                  }
            }
            for(Place place: world.getPlaces()){
                  if(world.isAWall(place.getPos())){
                        throw new RuntimeException("I have created a
wall on position " + place.getPos());
                  }
            }
            System.out.println();
            System.out.println();
            return people;
      }

      public void doIteration(final Collection<Agent> agents) {

            Calendar time = world.getTime();
            now = new EasyTime(time.get(Calendar.HOUR_OF_DAY),
time.get(Calendar.MINUTE));
            for (Agent a : agents) {
                  if (!a.isOnAuto()) {
                        continue;                }
                  handlePerson(a);
            }
      }
```

```
      private HashMap<String, AgentTypeCharacteristics>
setAgentTypes(){
            HashMap<String,AgentTypeCharacteristics> agentTypes =
                  new HashMap<String, AgentTypeCharacteristics>();
                  Configuration config = world.getSimulationConfig();

            System.out.println();
            int amountOfAgentTypes = 0;
            if(config.containsKey("agentTypes.amountOfAgentTypes")){
                  amountOfAgentTypes =
config.getInt("agentTypes.amountOfAgentTypes");
                  System.out.println();
                  System.out.println("amountOfAgentsTypes:" +
amountOfAgentTypes);
                  System.out.println();
                  System.out.println();
            }else{
                  throw new NoSuchElementException("There is no
amountOfAgentTypes tag");
            }
            String typeName;
            int amountOfAgents;
            boolean hasJob;
            int workStartHour;
            int workStartMinutes;
            int workStartBlur;
            int workDuration;
            int workDurationBlur;
            int locationPrivacy;
            int preferencesPrivacy;
            int priceWeightPref;
            int priceWeightPrefBlur;
            int distWeightPref;
            int distWeightPrefBlur;
            int qualityWeightPref;
            int qualityWeightPrefBlur;
            int musicTypeWeightPref;
            int musicTypeWeightPrefBlur;
            boolean hasCar;
            int refill;
            int refillBlur;
            String image;
            int amountOfOccupations;
            String occupationName;
            String occupationPlace;
            int averageHoursDuration;
            int averageMinutesDuration;
            int durationBlur;
            int centerHour;
            int centerMinutes;
            double variance;
            int i = 0;
            while(config.getString("agentTypes.type(" + i + ").name")
!= null){
                  AgentTypeCharacteristics typeCharacteristics = new
AgentTypeCharacteristics();
                typeName = config.getString("agentTypes.type(" + i +
").name");
                  agentTypes.put(typeName, typeCharacteristics);
                typeCharacteristics.setName(typeName);
```

```
                    amountOfAgents = config.getInt("agentTypes.type(" +
i + ").amount");
      typeCharacteristics.setAmountOfAgents(amountOfAgents);

                    image = config.getString("agentTypes.type(" + i +
").image");
                    typeCharacteristics.setImage(image);

                    hasJob = config.getBoolean("agentTypes.type(" + i +
").hasJob");
                    typeCharacteristics.setHasJob(hasJob);

                    workStartHour = config.getInt("agentTypes.type(" +
i + ").workStartHour");
                    workStartMinutes = config.getInt("agentTypes.type("
+ i + ").workStartMinutes");
      typeCharacteristics.setWorkStart(new EasyTime(workStartHour,
workStartMinutes));
                    workStartBlur = config.getInt("agentTypes.type(" +
i + ").workStartBlur");

      typeCharacteristics.setWorkStartBlur(workStartBlur);
            workDuration = config.getInt("agentTypes.type(" + i +
").workDuration");
                    typeCharacteristics.setWorkDuration(workDuration);
                          workDurationBlur =
config.getInt("agentTypes.type(" + i + ").workDurationBlur");
      typeCharacteristics.setWorkDurationBlur(workDurationBlur);
                    locationPrivacy = config.getInt("agentTypes.type("
+ i + ").locationPrivacy");
      typeCharacteristics.setLocationPrivacy(locationPrivacy);
                    preferencesPrivacy =
config.getInt("agentTypes.type(" + i + ").preferencesPrivacy");
      typeCharacteristics.setPreferencesPrivacy(preferencesPrivacy);
                    priceWeightPref = config.getInt("agentTypes.type("
+ i + ").priceWeightPref");
      typeCharacteristics.setPriceWeightPref(priceWeightPref);
                    priceWeightPrefBlur =
config.getInt("agentTypes.type(" + i + ").priceWeightPrefBlur");
      typeCharacteristics.setPriceWeightPrefBlur(priceWeightPrefBlur)
;
                    distWeightPref = config.getInt("agentTypes.type(" +
i + ").distWeightPref");
      typeCharacteristics.setDistWeightPref(distWeightPref);
                    distWeightPrefBlur =
config.getInt("agentTypes.type(" + i + ").distWeightPrefBlur");
      typeCharacteristics.setDistWeightPrefBlur(distWeightPrefBlur);
                    qualityWeightPref =
config.getInt("agentTypes.type(" + i + ").qualityWeightPref");
      typeCharacteristics.setQualityWeightPref(qualityWeightPref);
                    qualityWeightPrefBlur =
config.getInt("agentTypes.type(" + i + ").qualityWeightPrefBlur");
      typeCharacteristics.setQualityWeightPrefBlur(qualityWeightPrefB
lur);
                    musicTypeWeightPref =
config.getInt("agentTypes.type(" + i + ").musicTypeWeightPref");
      typeCharacteristics.setMusicTypeWeightPref(musicTypeWeightPref)
;
                    musicTypeWeightPrefBlur =
config.getInt("agentTypes.type(" + i + ").musicTypeWeightPrefBlur");
```

```
      typeCharacteristics.setMusicTypeWeightPrefBlur(musicTypeWeightP
refBlur);
                  hasCar = config.getBoolean("agentTypes.type(" + i +
").hasCar");
                  typeCharacteristics.setHasCar(hasCar);
                  if(hasCar){
                      refill = config.getInt("agentTypes.type(" + i +
").refill");

      typeCharacteristics.setREFILL_INTERVAL(refill);
      refillBlur = config.getInt("agentTypes.type(" + i +
").refillBlur");

      typeCharacteristics.setREFILL_BLUR(refillBlur);
                  }
                  amountOfOccupations = 0;
                  if(config.containsKey("agentTypes.type(" + i
+").amountOfOccupations")){
                        amountOfOccupations =
config.getInt("agentTypes.type(" + i + ").amountOfOccupations");
                        System.out.println();
                        System.out.println("Given amount of
occupations for " +typeName + ":" + amountOfOccupations);
                  }else{
                        throw new NoSuchElementException("You must
provide the amount of occupations "+ "for the agent type " +
typeName);
                  }
                  int j = 0;
                  String occupationTag;
                  while(config.getString("agentTypes.type(" + i +
").occupation(" + j + ").place") != null){
                        occupationTag = "agentTypes.type(" + i +
").occupation(" + j + ")";
                        occupationName =
config.getString(occupationTag + ".name");
                        occupationPlace =
config.getString(occupationTag + ".place");
                        averageHoursDuration =
config.getInt(occupationTag + ".averageHoursDuration");
                        averageMinutesDuration =
config.getInt(occupationTag + ".averageMinutesDuration");
                        durationBlur = config.getInt(occupationTag +
".durationBlur");
                        centerHour = config.getInt(occupationTag +
".centerHour");
                        centerMinutes = config.getInt(occupationTag +
".centerMinutes");
                        variance = config.getDouble(occupationTag +
".variance");
                        EasyTime meanDuration = new
EasyTime(averageHoursDuration, averageMinutesDuration);
                        EasyTime center = new EasyTime(centerHour,
centerMinutes);

                        Occupation newOccupation;
                        if(config.containsKey(occupationTag +
".amplitude")){
                              double amplitude =
config.getDouble(occupationTag + ".amplitude");
```

```
                              newOccupation = new
Occupation(occupationName, occupationPlace,
                                          meanDuration, durationBlur,
center, variance, amplitude);
                        }else{
                              newOccupation = new
Occupation(occupationName, occupationPlace,
                                          meanDuration, durationBlur,
center, variance);
                        }
      typeCharacteristics.getOccupations().put(occupationName,newOccu
pation);
                  j++;
                  }
                  System.out.println("Occupations found for " +
typeName + ":" + j);
                  System.out.println();
                  if(j != amountOfOccupations){
                        throw new RuntimeException("The found amount
of occupations" + " for the agent type" + typeName + "does not" + "
comply with the amount defined in the " + "AgentTypes.type(" + i +
").amountOfOccupations tag of " + "the configuration file");
                  }
                  i++;
            }
            System.out.println();
            System.out.println("*********");
            System.out.println();
            System.out.println("Agent types found:" + i);
            System.out.println()
            System.out.println("*********");
            if(i != amountOfAgentTypes){
                  throw new RuntimeException("The found agent types
does not" + " comply with the amount defined in the " +
"agentTypes.amountOfAgentTypes field of the configuration file");
            }
            return agentTypes;
      }

      public void setMaxDestination(Agent a){
            int maxDestination = config.getInt("maxDestination");
            a.set(MAX_DESTINATION, new
IntegerNumber(maxDestination));
      }

      public void setMinSuggestions(Agent a){
            int minSuggestions = config.getInt("minSuggestions");
            a.set(MIN_SUGGESTIONS, new
IntegerNumber(minSuggestions));
      }

      public void setWillRequestService(Agent a){
            double willRequestService =
config.getDouble("willRequestService");
            a.set(WILL_REQUEST_SERVICE, new
FloatNumber(willRequestService));
      }

      public void setUnexpectedFactor(Agent a){
            float unexpectedFactor =
config.getFloat("unexpectedFactor");
```

```
            a.set(UNEXPECTED_FACTOR, new
FloatNumber(unexpectedFactor));
      }

      public void setWaitingTime(Agent a){
            int waitingTime = config.getInt("waitingTime");
            a.set(WAITING_TIME, new IntegerNumber(waitingTime));
      }

      public void setWillWander(Agent a){
            double willWander = config.getDouble("willWander");
            a.set(WILL_WANDER, new FloatNumber(willWander));
      }

      public void setWanderDuration(Agent a){
            int wanderDuration = config.getInt("wanderDuration");
            a.set(WANDER_DURATION, new
IntegerNumber(wanderDuration));
      }

      public void setWillViolatePrivacy(Agent a){
            double willViolatePrivacy =
config.getDouble("willViolatePrivacy");
            a.set(WILL_VIOLATE_PRIVACY, new
FloatNumber(willViolatePrivacy));
      }

      public void setWaitingBlur(Agent a){
            int waitingBlur = config.getInt("waitingBlur");
            a.set(WAITING_BLUR, new IntegerNumber(waitingBlur));
      }

      public void setHome(Agent a){
            a.set(HOME,world.getRandomPlaceOfType("Home"));
      }

      public void setOffice(Agent a){
            Place office = world.getRandomPlaceOfType("Office");
            a.set(OFFICE, office);
      }

      public void setSpatialPerturbation(Agent a){
            if(config.containsKey("minSpatialPerturbation")){
                  int minSpatialPerturbation =
config.getInt("minSpatialPerturbation");
                  a.set(MIN_SPATIAL_PERTURBATION, new
IntegerNumber(minSpatialPerturbation));
            }else{
                  a.set(MIN_SPATIAL_PERTURBATION, new
IntegerNumber(0));
            }
      }

      public void setStaticSpatialCloakRange(Agent a){
      if(config.containsKey("privacyAlgorithm.staticSpatialCloak.use"
)){
                  int cloakRange =
config.getInt("privacyAlgorithm.staticSpatialCloak.cloakRange");
                  a.set(STATIC_SPATIAL_CLOAK_RANGE, new
IntegerNumber(cloakRange));
            }else{
```

```
                  a.set(STATIC_SPATIAL_CLOAK_RANGE, new
IntegerNumber(0));
            }
      }

      public void setMinNearAgents(Agent a){
      if(config.containsKey("privacyAlgorithm.dynamicSpatialCloak.min
NearAgents")){
                  int minNearAgents =
config.getInt("privacyAlgorithm.dynamicSpatialCloak.minNearAgents");
                  a.set(MIN_NEAR_AGENTS, new
IntegerNumber(minNearAgents));
            }else{
                  a.set(MIN_NEAR_AGENTS, new IntegerNumber(0));
            }
      }

      private void setUseSpatialCloak(){
            useSpatialCloak =
config.getBoolean("privacyAlgorithm.useSpatialCloak");
      }

      private void setUseStaticSpatialCloak(){
      useStaticSpatialCloak =
config.getBoolean("privacyAlgorithm.staticSpatialCloak.use");
      }

      private void setUseDynamicSpatialCloak(){
            useDynamicSpatialCloak =
config.getBoolean("privacyAlgorithm.dynamicSpatialCloak.use");
      }

      private TreeMap<String, Double>
calculateOccupationProbabilities(String agentType, EasyTime now){

            TreeMap<String, Double> occupationProbabilities = new
TreeMap<String,Double>();
            for(String occupationName:
agentTypes.get(agentType).getOccupations().keySet()){
                  Occupation occupation =
agentTypes.get(agentType).getOccupations().get(occupationName);
                  double probability =
occupation.getOccupationProbability(now);
                  occupationProbabilities.put(occupationName,
probability);
                  System.out.println(occupationName + ":" +
probability);
            }
            double sum = 0d;
            for(double probability:occupationProbabilities.values()){
                  sum += probability;
            }
            System.out.println();
            System.out.println("Normalized probability");
            System.out.println("******************************");
            for(String
occupationName:occupationProbabilities.keySet()){
                  double normalizedProbability =
occupationProbabilities.get(occupationName)/sum;
                  occupationProbabilities.put(occupationName,
normalizedProbability);
```

```
                  System.out.println(occupationName + ":" +
normalizedProbability) ;
            }
            return occupationProbabilities;
      }

      private Position calculateStaticCloakCenter(Agent a){
            int cloakRange =
((IntegerNumber)a.get(STATIC_SPATIAL_CLOAK_RANGE)).getNumber();
            int agentRow = a.getPos().getRow();
            int agentCol = a.getPos().getCol();
            int centerRow = agentRow + RAND.nextInt(cloakRange) -
(cloakRange/2);
            int centerCol = agentCol + RAND.nextInt(cloakRange) -
(cloakRange/2);
            return new Position(centerRow,centerCol);
      }

      private Position calculateCenterOfRequest(Agent a){
      if(useStaticSpatialCloak){
                  return calculateStaticCloakCenter(a);
            }else{
                  return a.getPos();
            }
      }

      private ArrayList<Place> requestDestinations(Position center,
int maxDestination, ArrayList<String> placeTypes, Node[] preference,
int privacy,int length, int requiredDest ){
            ArrayList<Place> destinations = new ArrayList<Place>();
            ArrayList<Place> cinemas = new ArrayList<Place>();
            System.out.println(placeTypes.toString());
            boolean ok = false;
            int total =0;
            while(!ok){
            for(Place place: world.getPlaces()){
                  if(place.getPos().isNear(center, maxDestination) &&
placeTypes.contains(place.getType())){
                        System.out.println(length);

      if(place.getType().equals("Cinema") ){
                              System.out.println("cinema");

      System.out.println(preference[length].getNodeName());
                              System.out.println("elegxw gia
"+preference[length+1-privacy].getNodeName());

                              if (length+1-
privacy<=place.pathArrayLength){
                                    System.out.println("exei
"+place.pathArray[length+1-privacy].getNodeName().toString());
                               if(preference[length+1-
privacy].getNodeName().equals(place.pathArray[length+1-
privacy].getNodeName())){
                               System.out.println("----
"+place.toString());
                               System.out.println("cinema manager");
                               if (!destinations.contains(place)){
                           destinations.add(place);
                           cinemas.add(place);
System.out.println(distance(place.getPos(),center));
```

```
                              total =total +1;
                              System.out.println("totalcinema "+total
);}}
                                 }
                            }else {System.out.println("----
"+place.toString());
                               if (!destinations.contains(place)){
                                   destinations.add(place);
                                   System.out.println("totaltotal
"+total );}}
                  }
              }if (total<requiredDest){System.out.println("privacy
change");privacy = privacy +1;}else {ok = true;}}
           System.out.println(cinemas.toString());
           world.pause(true);
             return destinations;
      }
      private void filterRepliedDestinations(ArrayList<Place>
repliedDestinations, Agent a){}

      private Place chooseFromDestinations(ArrayList<Place>
destinations,
                  TreeMap<String,Double> occupationProbabilities,
Agent agent){

            Place finalDestination = null;

            if(destinations.isEmpty()){
                  System.out.println();
                  System.out.println("-chooseFromDestinations:The
array list of destinations to choose is empty."
                              + "I return null as the chosen
destination.");
                  System.out.println();
                  return null;
            }

            System.out.println();
            System.out.println("-chooseFromDestinations:Enter the
distribution function calculation");
            System.out.println("******************************");

            while(!occupationProbabilities.isEmpty()){
                  System.out.println();
                  String agentType =
((Text)agent.get(TYPE)).getText();
                  double probability = RAND.nextDouble();
                  System.out.println("Choose distribution function
probability:" + probability);
                  String chosenOccupation =
occupationProbabilities.lastKey();
                  double sum = 0d;
                  for(String occupationName:
occupationProbabilities.keySet()){
                        sum +=
occupationProbabilities.get(occupationName);
                        if(probability < sum){
                              chosenOccupation = occupationName;
                              System.out.println("Chosen occupation:"
+ occupationName);
```

```
                                    agent.set(OCCUPATION, new
Text(chosenOccupation));
                                    break;
                              }
                        }
                        occupationProbabilities.remove(chosenOccupation);
                        System.out.println("Remove chosen occupation:" +
chosenOccupation);
                        String typePlace =
agentTypes.get(agentType).getOccupations().

      get(chosenOccupation).getOccupationPlace();
                        ArrayList<Place> chosenOccupationPlaces = new
ArrayList<Place>();
                        for(Place place: destinations){
      if(place.getType().equalsIgnoreCase(typePlace)){
                              chosenOccupationPlaces.add(place);
                              }
                        }
                        finalDestination =
chooseFinalDestination(chosenOccupationPlaces, agent);
                        if(finalDestination != null){
                              System.out.println("-chooseFromDestinations-
return non null-finalDestination:"
                                          + finalDestination.getName());
                              System.out.println("------------------");
                              return finalDestination;
                        }else{
                              System.out.println("-chooseFromDestinations-
chooseFinalDestination returned null.");
                              double probabilityToWander =
getWillWander(agent);
                              double randomProbability = RAND.nextDouble();
                              if(randomProbability < probabilityToWander){
                                    System.out.println("-
chooseFromDestinations-last if-random probability:" +
randomProbability + " is less than " + "probabilityToWander:" +
probabilityToWander);
                                    System.out.println("Return null so as
to make a wander and retry after that wander.");
                                    System.out.println();
                                    System.out.println("-------");
                                    System.out.println();
                                    System.out.println();

      System.out.println("********************");
                                    return null;
                              }else{
                                    System.out.println("-
chooseFromDestinations-last else- random probability:" +
randomProbability + " is greater than " + "probabilityToWander:" +
probabilityToWander);
                                    System.out.println("Trying occupation
other than the originally chosen.");
                                    System.out.println();
                                    System.out.println("----------");
                              }
                        }
                  }
                  return null;
      }
```

```
	private double getWillWander(Agent agent){
		return ((FloatNumber)agent.get(WILL_WANDER)).getNumber();
	}

	private Place chooseFinalDestination(ArrayList<Place>
chosenOccupationPlaces, Agent agent){
		Place finalDestination = null;
		double probability = RAND.nextDouble();
		System.out.println("-chooseFinalDestination-probability
for overcome privacy:" + probability);
		if(privacyCheck(chosenOccupationPlaces, agent) ||
(probability <= willOvercomePrivacyViolation(agent))){
			System.out.println("-chooseFinalDestination method:
call bestDestination method");
			finalDestination =
bestDestination(chosenOccupationPlaces, agent);
		}
		return finalDestination;
	}

	private Place bestDestination(ArrayList<Place>
chosenOccupationPlaces, Agent agent){
		Place finalDestination = null;
		double priceWeight,qualityWeight,distanceWeight;
		priceWeight =
Integer.parseInt(agent.get("priceWeightPref").toString());
		qualityWeight =
Integer.parseInt(agent.get("qualityWeightPref").toString());
		distanceWeight =
Integer.parseInt(agent.get("distWeightPref").toString());
		int price,quality;
		double distance;
		double maxWeightedSum = 0;
		double weightedSum ;
		String placeType ;
		HashMap<String,PlaceTypeCharacteristics>
PlacesCharacteristics = WorldModel.placeTypes;
		PlaceTypeCharacteristics typeChar ;
		for(Place place: chosenOccupationPlaces){
			price =
Integer.parseInt(place.get("price").toString());
			quality =
Integer.parseInt(place.get("quality").toString());
			distance = distance(agent.getPos(),
place.getPos());
			weightedSum = priceWeight*price +
qualityWeight*quality + distanceWeight*(5-(distance/100));
			System.out.println("weightedSum "+weightedSum);
			placeType = place.getType();
			typeChar = PlacesCharacteristics.get(placeType);

			  if (weightedSum>maxWeightedSum ){

				System.out.println(place.getName());
				System.out.println(">"+maxWeightedSum);
				if(agent.getAvoidDest()==null ||
!agent.getAvoidDest().equals(place)){
				   if((!place.getType().equals("Bar") &&
!place.getType().equals("Cafe"))||now.isIn(new
TimePeriod(typeChar.getOpen(),typeChar.getClose()))){
```

```
                                    maxWeightedSum = weightedSum;
                                    finalDestination = place;}
                              else System.out.println("closed "+place);
                           }
                           else System.out.println("avoid that dest
"+place );
                     }
               }
               if(agent.getAvoidDest()!=null){
               System.out.println("avoid
"+agent.getAvoidDest().getName());}
               agent.setAvoidDest(null);
               if (finalDestination!=null){
                     System.out.println("suggestion
"+finalDestination.getName());
                  price =
Integer.parseInt(finalDestination.get("price").toString());
                  quality =
Integer.parseInt(finalDestination.get("quality").toString());
                  distance = distance(agent.getPos(),
finalDestination.getPos());
                  weightedSum = priceWeight*price +
qualityWeight*quality + distanceWeight*(5-(distance/100));
                  System.out.println("weightedSumfinal "+weightedSum);}
               return finalDestination;
               }

      private double willOvercomePrivacyViolation(Agent agent){

               double violatePrivacy =
((FloatNumber)agent.get(WILL_VIOLATE_PRIVACY)).getNumber();
               System.out.println("-willOvercomePrivacyViolation
method:" + violatePrivacy);
               return violatePrivacy;
      }

      private boolean privacyCheck(ArrayList<Place>
chosenOccupationPlaces, Agent agent){
               boolean isOK;
               int minSuggestions =
Integer.parseInt(agent.get(MIN_SUGGESTIONS).toString());
               if(minSuggestions <= chosenOccupationPlaces.size()){
               System.out.println("minimum Suggestions
"+minSuggestions+" <= "+"chosen Occupations
"+chosenOccupationPlaces.size()+" "+agent.getName());
               System.out.println("-privacyCheck method:PASS");
               isOK = true;
               }
               else {
                     System.out.println("minimum Suggestions
"+minSuggestions+" > "+"chosen Occupations
"+chosenOccupationPlaces.size()+" "+agent.getName());
                     System.out.println("-privacyCheck method:FAILED");
                     isOK = false;
               }
               return isOK;
      }

      public double distance(final Position posA, final Position
posB){
```

```
        double distanceI = Math.pow(posA.getRow() - posB.getRow(),
2);
        double distanceJ = Math.pow(posA.getCol() - posB.getCol(),
2);
        double distance = Math.sqrt(distanceI + distanceJ);
            return distance;
      }

      private Place makeRequest(Agent a){
            String agentType = ((Text)a.get(TYPE)).getText();

            Position pos = a.getPos();
            TreeMap<String, Double> occupationProbabilities =
                  calculateOccupationProbabilities(agentType,now);
            filterOccupationProbabilities(occupationProbabilities);
            ArrayList<String> destinationPlaceTypes = new
ArrayList<String>();
            for(String occupationName:
occupationProbabilities.keySet()){
                  String placeType =
agentTypes.get(agentType).getOccupations().get(occupationName).getOcc
upationPlace();
                  destinationPlaceTypes.add(placeType);
            }
            Position center = calculateCenterOfRequest(a);
            int maxDestination =
((IntegerNumber)a.get(MAX_DESTINATION)).getNumber();
            Random rand = new Random();
            Node nodeArray[]=new Node[10];
            int length =0;
            try{
                  DocumentBuilderFactory docBuilderFactory =
DocumentBuilderFactory.newInstance();
                DocumentBuilder docBuilder =
docBuilderFactory.newDocumentBuilder();
                Document doc = docBuilder.parse (new
File("C:\\Users\\Nikos\\workspace\\tre.xml"));
                Node currentNode = doc.getDocumentElement();

                int k=0;
                int l;
                nodeArray[0]=currentNode;
                while (currentNode.getChildNodes().getLength()-2 >
0){
                  System.out.print(currentNode.getNodeName()+"---");
                  l =
rand.nextInt((currentNode.getChildNodes().getLength()-1)/2);
                  currentNode =
currentNode.getChildNodes().item(2*l+1);
                  k=k+1;
                  nodeArray[k]=currentNode;
                }
                System.out.println(currentNode.getNodeName()+"
leaf");
                length=k;
                System.out.println("ok");
                }
                catch (SAXParseException err) {
                  System.out.println ("** Parsing error" + ", line "
+ err.getLineNumber () + ", uri " + err.getSystemId ());
                  System.out.println(" " + err.getMessage ());
```

```
                  }catch (SAXException e) {
                  Exception x = e.getException ();
                  ((x == null) ? e : x).printStackTrace ();
                  }catch (Throwable t) {
                  t.printStackTrace ();
                  }
                  int privacy = 1;
                  int required = 5;
            ArrayList<Place> repliedDestinations =
                  requestDestinations(center, maxDestination,
destinationPlaceTypes, nodeArray,privacy,length,required);
            filterRepliedDestinations(repliedDestinations, a);

            if(repliedDestinations.isEmpty()){
                  return null;
            }
            Place destination =
chooseFromDestinations(repliedDestinations,
occupationProbabilities,a);
            return destination;
      }

      private Place chooseByYourself(Agent a){
            String agentType = ((Text)a.get(TYPE)).getText();
            Position pos = a.getPos();
            TreeMap<String, Double> occupationProbabilities =
                  calculateOccupationProbabilities(agentType,now);
            filterOccupationProbabilities(occupationProbabilities);
            ArrayList<String> destinationPlaceTypes = new
ArrayList<String>();
            for(String occupationName:
occupationProbabilities.keySet()){
                  String placeType =
agentTypes.get(agentType).getOccupations().get(occupationName).getOcc
upationPlace();
                  destinationPlaceTypes.add(placeType);
            }
            ArrayList<Place> destinations = new ArrayList<Place>();
            double probability = RAND.nextDouble();
            System.out.println("-chooseByYouself:RAND probability for
probability distribution:" + probability);
            String chosenOccupation =
occupationProbabilities.lastKey();
            double sum = 0d;
            for(String occupationName:
occupationProbabilities.keySet()){
                  sum += occupationProbabilities.get(occupationName);
                  if(probability < sum){
                        chosenOccupation = occupationName;
                        System.out.println("-chooseByYourself:Chosen
occupation:" + occupationName);
                        a.set(OCCUPATION, new
Text(chosenOccupation));
                        break;
                  }
            }
            System.out.println("Chosen occupation:" +
chosenOccupation);
            String typePlace =
agentTypes.get(agentType).getOccupations().
                        get(chosenOccupation).getOccupationPlace();
```

```
            return world.getRandomPlaceOfType(typePlace);
      }
      private void handlePerson(final Agent a) {
            String agentType = ((Text)a.get(TYPE)).getText();
            String activity = ((Activity)a.get(ACTIVITY)).toString();
            String agentName = a.getName();
            if(now.isIn(new TimePeriod((EasyTime)a.get(WORK_START),
(EasyTime)a.get(WORK_END))) &&
      ((!activity.equalsIgnoreCase(Activity.WORKING.toString())) &&
      (!activity.equalsIgnoreCase(Activity.GOING_TO_WORK.toString()))
)){
                  System.out.println("The time now is " + now + " and
the work starts at " + (EasyTime)a.get(WORK_START));
                  System.out.println("Agent's "+agentName+"  activity
changes from whatever he did to GOING_TO_WORK");
                  a.set(ACTIVITY, Activity.GOING_TO_WORK);
                        a.setDestination((Place)a.get(OFFICE));
                  a.setNikosDest((Place)a.get(OFFICE));

                  System.out.println("The world has paused.");
                  System.out.println();
                  return ;
            }
            switch ((Activity) a.get(ACTIVITY)) {
            case CHOOSE_NEXT_DESTINATION:
                  System.out.println("Agent's "+agentName+"
activity:CHOOSE_NEXT_DESTINATION");
                  Place destination;
                  double willRequestDestination =
((FloatNumber)a.get(WILL_REQUEST_SERVICE)).getNumber();
                  double thisProbability = RAND.nextDouble();
                  if(thisProbability<willRequestDestination){
                        System.out.println("This Probability:" +
thisProbability + ":is smaller than" + " the willRequestDestination "
+ willRequestDestination +"\nso the agent " +agentName+ "won't
request the service, he will choose a place by himself");
                        destination = chooseByYourself(a);
                  }else{
                        System.out.println("This Probability:" +
thisProbability + ":is greater than" + " the willRequestDestination "
+ willRequestDestination + "\nso the agent"+agentName+ "will ask the
service for a destination.");
                        destination = makeRequest(a);
                  }
                  if(destination != null){
                        String occupation =
((Text)a.get(OCCUPATION)).getText();
                        if(occupation.equalsIgnoreCase("sleeping") ||
occupation.equalsIgnoreCase("resting")){
                              destination = (Place)a.get(HOME);
                        }
                        a.setDestination(destination);
                        a.setNikosDest(destination);
                        System.out.println("Agent's "+agentName+"
position now:" + a.getPos());
                        System.out.println("Destination:" +
destination);
                        a.set(ACTIVITY,
Activity.GOING_TO_DESTINATION);
```

```
                        System.out.println("Agent's "+agentName+"
activity changes from CHOOSE_NEXT_DESTINATION to
GOING_TO_DESTINATION");
                  }else{
                        System.out.println("No destinations found");
                        System.out.println("Setting occupation to
WANDERING");
                        a.set(ACTIVITY, Activity.SET_WANDERING_TIME);
                        a.set(OCCUPATION, new Text("WANDER"));
                        System.out.println("Agent's "+agentName+"
activity changes from CHOOSE_NEXT_DESTINATION to
SET_WANDERING_TIME");
                        System.out.println("The world has paused.");

                  }
                  break;

            case GOING_TO_WORK:
                  if(a.isAtDestination()){
                        System.out.println("Agent "+agentName+"
arrived at work");
                        int workDuration =
agentTypes.get(agentType).getWorkDuration();
                        int workDurationBlur =
agentTypes.get(agentType).getWorkDurationBlur();
                        EasyTime endWork = new
EasyTime(now).shift(workDuration,0).blur(workDurationBlur);
                        System.out.println("Calculated time for
ending work:" + endWork.toString());
                        a.set(WORK_END, endWork);
                        a.set(ACTIVITY, Activity.WORKING);
                        System.out.println("Agent's "+agentName+"
activity changes from GOING_TO_WORK to WORKING");
                        System.out.println("The world has paused.");
                        System.out.println();
                  }
                  break;

            case WORKING:
                  if(now.isAfter((EasyTime)a.get(WORK_END))){
                        System.out.println("The time is " +
now.toString() + " and my working hours has finished.");
                        EasyTime averageWorkStart = new
EasyTime(agentTypes.get(agentType).getWorkStart());
                        int workStartBlur =
agentTypes.get(agentType).getWorkStartBlur();
                        EasyTime workStart =
averageWorkStart.blur(workStartBlur);
                        a.set(WORK_START, workStart);
                        System.out.println("Tomorrow I'll start work
at " + workStart.toString());
                        a.set(ACTIVITY,
Activity.CHOOSE_NEXT_DESTINATION);
                        System.out.println("Agent's"+agentName+"
activity changes from WORKING to CHOOSE_NEXT_DESTINATION");
                        System.out.println("The world has paused");
                        System.out.println();
                        }
                  break;

            case GOING_TO_DESTINATION:
```

```
                  if(a.isAtDestination()){
                        System.out.println("Agent's "+agentName+"
activity:GOING_TO_DESTINATION");
                        System.out.println("Agent's "+agentName+"
activity changes from GOING_TO_DESTINATION to
ARRIVE_AT_DESTINATION");
                        a.set(ACTIVITY,
Activity.ARRIVE_AT_DESTINATION);
                        System.out.println("The world has paused.");
                        System.out.println();
                        }
                  break;

            case ARRIVE_AT_DESTINATION:
                  System.out.println("Agent's "+agentName+"
activity:ARRIVE_AT_DESTINATION");
                  if
((a.getNikosDest().getType().equals("Bar"))||(a.getNikosDest().getTyp
e().equals("Cafe"))){
                        String placeType =
a.getNikosDest().getType();
                        HashMap<String,PlaceTypeCharacteristics>
PlacesCharacteristics = WorldModel.placeTypes;
                        PlaceTypeCharacteristics typeChar =
PlacesCharacteristics.get(placeType);
                        TreeMap<EasyTime,String> trafficStates =
typeChar.getTrafficStates();
                        String trafficState="no";
                        boolean flag= false;
                        boolean ok=false;
                        for (int i=0 ;
i<trafficStates.keySet().size()&& !ok;i=i+1){
                              if
(now.isAfter((EasyTime)trafficStates.keySet().toArray()[i])){
                              if(i==trafficStates.keySet().size()-1){
                                    trafficState =
trafficStates.get((EasyTime)trafficStates.keySet().toArray()[i]);
                                    ok=true;
                                    }
                                    flag=true;
                              }else if
(flag&&now.isBefore((EasyTime)trafficStates.keySet().toArray()[i])){
                                    trafficState =
trafficStates.get((EasyTime)trafficStates.keySet().toArray()[i-1]);
                                    ok=true;
                                    }
                        }
                        if (trafficState.equals("no")){
                              trafficState =
trafficStates.get((EasyTime)trafficStates.keySet().toArray()[trafficS
tates.keySet().size()-1]);
                        }
                        System.out.println(trafficStates.toString());
                        System.out.println(now.toString());
                        System.out.println(trafficState);

                        Float unavailability =
typeChar.getUnavailability().get(trafficState);
                        Integer unavailabilityTime =
typeChar.getUnavailabilityTime().get(trafficState);
```

```
                        System.out.println("unavailability
"+unavailability+" unavailability time "+unavailabilityTime);

                        if
(RAND.nextDouble()<((FloatNumber)a.get(UNEXPECTED_FACTOR)).getNumber(
){
                              System.out.println("unexpected");
                              a.set(ACTIVITY,
Activity.CHOOSE_NEXT_DESTINATION);
                              a.setAvoidDest(a.getNikosDest());
                              System.out.println("Agent's
"+agentName+" activity changes from ARRIVE_AT_DESTINATION to
CHOOSE_NEXT_DESTINATION");
                           System.out.println("The world has
paused.");
                           System.out.println();
                           break;
                        }
                        else if (RAND.nextDouble()<unavailability){
                              System.out.println("unavailable");
                              a.set(ACTIVITY, Activity.WAITING);
                              int waitingTime =
Integer.parseInt(a.get(WAITING_TIME).toString());
                              int waitingBlur
=Integer.parseInt(a.get(WAITING_BLUR).toString());
                              int wait =
waitingTime+RAND.nextInt(waitingBlur)-(waitingBlur/2);
                              EasyTime nextEvent;
                              System.out.println(waitingTime+"time
"+waitingBlur+"blur "+wait);
                              if(wait>unavailabilityTime){
                                    a.setWillWait(true);
                                    nextEvent = new
EasyTime(now).shift(0,unavailabilityTime);
                              }
                              else{
                                    a.setWillWait(false);
                                    nextEvent = new
EasyTime(now).shift(0,wait);
                              }
                              a.set(NEXT_EVENT, nextEvent);
                              a.set(PREVIOUS_EVENT, new
EasyTime(now).shift(0,-1));
                              System.out.println(a.getWillWait());
                              System.out.println("Agent's
"+agentName+" activity changes from ARRIVE_AT_DESTINATION to
WAITING");
                           System.out.println("The world has
paused.");
                           System.out.println();

                           break;
                        }
                  }
                        System.out.println("Agent's "+agentName+"
activity changes from ARRIVE_AT_DESTINATION to SET_OCCUPATION_TIME");

                     a.set(ACTIVITY, Activity.SET_OCCUPATION_TIME);
                     System.out.println("The world has paused.");
                     System.out.println();
                     break;
```

```
            case WAITING:
                  if(!now.isIn(new
TimePeriod((EasyTime)a.get(PREVIOUS_EVENT),(EasyTime)a.get(NEXT_EVENT
))){
                        System.out.println("Agent's "+agentName+"
activity:WAITING"+a.getWillWait());
                        if (a.getWillWait()){
                              a.set(ACTIVITY,
Activity.SET_OCCUPATION_TIME);
                        }
                        else{
                              a.set(ACTIVITY,
Activity.CHOOSE_NEXT_DESTINATION);
                              a.setAvoidDest(a.getNikosDest());
                        }
                  }
                  System.out.println("Agent's "+agentName+"
activity:WAITING");
                  break;

            case SET_OCCUPATION_TIME:
                  System.out.println("Agent's "+agentName+"
activity:SET_OCCUPATION_TIME");
                  setNextEvent(a);
                  a.set(PREVIOUS_EVENT,new EasyTime(now).shift(0,-
1));
                  System.out.println("Occupation duration
calculated.NEXT_EVENT at " + (EasyTime)a.get(NEXT_EVENT));
                  System.out.println("Agent's "+agentName+" activity
changes from SET_OCCUPATION_TIME to BEING_OCCUPIED");
                  a.set(ACTIVITY, Activity.BEING_OCCUPIED);
                  System.out.println("The world has paused.");
                  System.out.println();
                  break;

            case BEING_OCCUPIED:
                  if(now.isIn(new
TimePeriod((EasyTime)a.get(WORK_START),(EasyTime)a.get(WORK_END)))){
                        System.out.println("The agent "+agentName+"
set yesterday the today's time for work at "+
(EasyTime)a.get(WORK_START));
                        System.out.println("Now the time is " + now +
" so he must leave whatever he's \n" + "doing and go to work.");
                        a.set(ACTIVITY, Activity.GOING_TO_WORK);
                        System.out.println("Agent's "+agentName+"
activity changes from BEING_OCCUPIED to GOING_TO_WORK");

                        a.setDestination((Place)a.get(OFFICE));
                        a.setNikosDest((Place)a.get(OFFICE));

                        System.out.println("The world has paused.");
                        System.out.println();
                        break;
                  }
                  if(!now.isIn(new
TimePeriod((EasyTime)a.get(PREVIOUS_EVENT),(EasyTime)a.get(NEXT_EVENT
))){
                        System.out.println("The NEXT_EVENT for the
agent was set to " + (EasyTime)a.get(NEXT_EVENT));
```

```
                        System.out.println("Now the time is " + now +
".");
                        a.set(ACTIVITY,
Activity.CHOOSE_NEXT_DESTINATION);
                        System.out.println("Agent's "+agentName+"
activity changes from BEING_OCCUPIED to CHOOSE_NEXT_DESTINATION");
                              System.out.println("The world has paused.");
                              System.out.println();
                              break;
                        }
                        break;

                  case SET_WANDERING_TIME:
                        System.out.println("Agent's "+agentName+"
activity:SET_WANDERING_TIME");
                        int duration =
((IntegerNumber)a.get(WANDER_DURATION)).getNumber();
                        EasyTime nextEvent = new
EasyTime(now).shift(0,duration);
                        a.set(NEXT_EVENT, nextEvent);
                        a.set(PREVIOUS_EVENT, new EasyTime(now).shift(0,-
1));
                        System.out.println("Previous Event at " +
now.shift(0,-1));
                        System.out.println("The NEXT_EVENT where the wander
will end is set to " + (EasyTime)a.get(NEXT_EVENT));
                        a.set(ACTIVITY, Activity.WANDERING);
                        System.out.println("Agent's "+agentName+" activity
changes from SET_WANDERING_TIME to WANDERING");
                        System.out.println("The world has paused.");
                        System.out.println();
                        break;

                  case WANDERING:
                        if(!now.isIn(new
TimePeriod((EasyTime)a.get(PREVIOUS_EVENT),(EasyTime)a.get(NEXT_EVENT
))))){
                              System.out.println("Agent's "+agentName+"
activity:WANDERING");
                              a.set(ACTIVITY,
Activity.CHOOSE_NEXT_DESTINATION);
                              System.out.println("Agent's "+agentName+"
activity changes from WANDERING to CHOOSE_NEXT_DESTINATION");
                              System.out.println("The world has paused.");
                              System.out.println();
                              }else{
                              int soberness = 30;
                              a.wander(soberness);
                        }
                        break;

                  default:
                        throw new RuntimeException("Unable to handle
activity " + (Activity) a.get(ACTIVITY));
                  }
            }
            private void setNextEvent(Agent a){
                  String agentType = ((Text)a.get(TYPE)).getText();
                  String occupationName =
((Text)a.get(OCCUPATION)).getText();
                  EasyTime nextEvent = new EasyTime(now);
```

```
      nextEvent.shift(agentTypes.get(agentType).getOccupations().
                        get(occupationName).getMeanDuration());
            int durationBlur =
agentTypes.get(agentType).getOccupations().
                        get(occupationName).getDurationBlur();
            nextEvent.blur(durationBlur);
            System.out.println("-setNextEvent method(called from
SET_OCCUPIED_TIME):" + " The occupation will end at " + nextEvent);
            a.set(NEXT_EVENT, nextEvent); }}
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Ubiquitous computing, Wikipedia, the free encyclopedia [Online]. Available: http://en.wikipedia.org/wiki/Ubiquitous_computing

[2] A. Dey, "Understanding and using context," *Personal and Ubiquitous Computing*, vol. 5(1), pp. 47-52, 2001.

[3] A. Dey and G. Abowd, "Towards a better understanding of context and contextawareness," in *Proceedings of the CHI 2000 Workshop on "The What, Who, Where, When, Why and How of Context-Awareness"*, 2000.

[4] G. Abowd, A. Dey, P. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a better understanding of context and context-awareness," in *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, 1999, pp. 304-307, Springer-Verlag.

[5] T. Reichenbacher, "Adaptive methods for mobile cartography," in *Proceedings of the 21st Inernational Cartographic Conference ICC: Cartographic Renaissance*, 10-16 August 2003, Durban, South Africa, pp. 1311-1321.

[6] S. Stieniger, M. Neun, A. Edwardes, "Foundations of location based services," Project CartouCHe - Lecture Notes on LBS, version 1.0 [Online]. Available: http://www.geo.unizh.ch/publications/cartouche/lbs_lecturenotes_steinigeretal2006.pdf

[7] A. Beresford, "Location privacy in ubiquitous computing," Ph.D. dissertation, University of Cambridge, Computer Laboratory, 2005.

[8] K. Lyytinen and Y. Yoo, "Issues and challenges in ubiquitous computing," *Communications of the ACM*, vol. 45(12), pp. 62-65, 2002.

[9] M. Weiser, "The computer for the 21st century," *Scientific American*, vol. 365(3), pp. 94-104, September 1991.

[10] B. Schilit, N. Adams, and R. Want, "Context-aware computing applications," in *Proceedings of the Workshop on Mobile Computing Systems and Applications*, 1993, pp. 85-90.

[11] D. Tennenhouse, "Proactive computing," *Communications of the ACM*, vol. 43(5),pp. 43-50, 2000.

[12] K. Virrantaus, J. Markkula, A. Garmash, Y. V. Terziyan, "Developing GISSupported
Location Based Services," in *Proceedings of WGIS'2001 – First International Workshop on Web Geographical Information Systems*, Kyoto, Japan, 2001, pp. 423-432.

[13] G. Orwell, *1984*. Everyman's Library, November 1992.

[14] M. Duckham and L. Kulik, “A formal model of obfuscation and negotiation for location privacy,” in *Pervasive Computing,* Springer Berlin / Heidelberg, 2005, pp. 152-170.

[15] H. Kido, Y. Yanagisawa, and T. Satoh, “Protection of location privacy using dummies for location-based services,” in *Proceedings of the 21st International Conference on Data Engineering Workshops,* 2005, p. 1248.

[16] A. Pfitzmann and M. Koehntopp, “Anonymity, unobservability, and pseudonymity a proposal for terminology,” in *Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability*, vol. 2009 of *LNCS*, Springer, 2000.

[17] M. F. Mokbel, C. Chow, and W. G. Aref, “The new Casper: query processing for location services without compromising privacy,” in *Proceedings of the 32nd International Conference on Very Large Data Bases*, Seoul, Korea, 2006, pp. 763-774.

[18]Du, Jing; Xu, Jianliang; Tang, Xueyan; Hu, Haibo, "iPDA: Supporting Privacy-Preserving Location-Based Mobile Services," *Mobile Data Management, 2007 International Conference on* , vol., no., pp.212-214, 1-1 May 2007

[19]Mokbel, M. F., Chow, C., and Aref, W. G. 2006. The new Casper: query processing for location services without compromising privacy. In *Proceedings of the 32nd international Conference on Very Large Data Bases* (Seoul, Korea, September 12 - 15, 2006). U. Dayal, K. Whang, D. Lomet, G. Alonso, G. Lohman, M. Kersten, S. K. Cha, and Y. Kim, Eds. Very Large Data Bases. VLDB Endowment, 763-774.

[20] X. Jiang and J. A. Landay, “Modeling privacy control in context-aware systems,” *Pervasive Computing*, IEEE, vol. 1(3), July-September 2002, pp. 59 – 63.

[21]Ποιότητα υπηρεσίας και προστασία ιδιωτικότητας σε εξατομικευμένες προσαρμοστικές κινητές υπηρεσίες, Διπλωματική εργασία ,Αθανάσιος Σ. Βουλόδημος, Ε.Μ.Π 2007

[22] C. E. Shannon, “The Mathematical Theory of Communication,” The Bell System Technical Journal, vol. 27, 1948, pp. 379-423, 623-656.

[23]Live Search (Windows local live homepage) [Online]. Available: http://local.live.com/

[24] Google Maps [Online]. Available: http://maps.google.com/

[25] Yahoo! Maps [Online]. Available: http://maps.yahoo.com/

[26] Mercator projection - Wikipedia, the free encyclopedia [Online]. Available: http://en.wikipedia.org/wiki/Mercator_projection

[27] Virtual Earth Tile System (V4) [Online]. Available: http://msdn2.microsoft.com/enus/library/bb259689.aspx

[28] H. Samet, "The quadtree and related hierarchical data structures," *ACM Computing Surveys*, vol.16(2), pp. 187-260, 1984.

[29] A. Klinger, "Patterns and search statistics," in *Optimizing Methods in Statistics*, J.S.Rustagi, FEd., Academic Press, New York, 1971, pp. 303-337.

[30] R. de Neufville, "Defining Capacity of Airport Passenger Buildings," lecture notes for the course *Airport Systems Planning, Design, and Management*, Massachusetts Institute of Technology, http://ardent.mit.edu/airports/ASP_current_lectures/ASP%2004/Defining_Capacity 04.pdf.

[31]Martin, Miquel; Nurmi, Petteri, "A Generic Large Scale Simulator for Ubiquitous Computing," *Mobile and Ubiquitous Systems: Networking & Services, 2006 Third Annual International Conference on* , vol., no., pp.1-3, July 2006

[32] M. C. Huebscher and J. A. McCann, "Simulation model for selfadaptive applications in pervasive computing," in *Proceedings of the 15th International Workshop on Database and Expert Systems Applications (DEXA)*. IEEE Computer Society, 2004, pp. 694 – 698.

[33] M. Bylund and F. Espinoza, "Using Quake III Arena to simulate sensors and actuators when evaluating and testing mobile services," in *CHI '01:CHI '01 extended abstracts on Human factors in computing systems*, 2001, pp. 241 – 243.

[34] "Testing and demonstrating services with Quake III Arena," *Communications of the ACM (CACM)*, vol. 45, no. 1, pp. 46 – 48, Jan. 2002.

[35] J. J. Barton and V. Vijayaraghavan, "UBIWISE, A simulator for ubiquitous computing systems design," Hewlett-Packard Laboratories Palo *Alto," HPL-2003-93, 2003.*

[36] K. Sanmugalingam and G. Coulouris, "A generic location event simulator," in *Proceedings of the 4th International Conference on Ubiquitous Computing (UbiComp)*, ser. Lecture Notes in Computer Science (LNCS), G. Borriello and L. Holmquist, Eds., vol. 2498. Berlin Heidelberg: Springer - Verlag, 2002, pp. 308 – 315.

[37] Μελέτη διασφάλισης της ιδιωτικότητας σε περιβάλλοντα κινητών επικοινωνιών με χρήση προσομοιωτή, Διπλωματική εργασία ,Κωνσταντίνος Ε. Οικονόμου, Ε.Μ.Π 2008