Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Επικοινωνιών, Ηλεκτρονικής και
Συστημάτων Πληροφορικής

Κοινωνικός Αποκλεισμός και Επανένταξη Εικονικών Οντοτήτων στο Διαδίκτυο των Πραγμάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΕΛΕΥΘΕΡΙΟΣ ΚΟΚΟΡΗΣ - ΚΟΓΙΑΣ

Επιβλέπων : Θεοδώρα Βαρβαρίγου
Καθηγήτρια ΕΜΠ

Αθήνα, Ιούλιος 2015

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Επικοινωνιών, Ηλεκτρονικής και
Συστημάτων Πληροφορικής

Κοινωνικός Αποκλεισμός και Επανένταξη Εικονικών Οντοτήτων στο Διαδίκτυο των Πραγμάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΕΛΕΥΘΕΡΙΟΣ ΚΟΚΟΡΗΣ - ΚΟΓΙΑΣ

Επιβλέπων : Θεοδώρα Βαρβαρίγου
Καθηγήτρια ΕΜΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 03η Ιουλίου 2015.

Θεοδώρα Βαρβαρίγου Καθηγήτρια ΕΜΠ	Βασίλειος Λούμος Καθηγητής Ε.Μ.Π.	Δημήτριος Ασκούνης Αναπληρωτής Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2015

Ελευθέριος Κόκορης - Κόγιας

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

Copyright © Ελευθέριος Κόκορης - Κόγιας, 2015.
Με επιφύλαξη παντός δικαιώματος. All rights reserved.

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

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

Περίληψη

Στον 21ο αιώνα, ο ψηφιακός κόσμος έχει ενσωματωθεί στην καθημερινότητά μας. Έξυπνες συσκευές θα είναι σύντομα σε θέση να αποφασίζουν μόνες τους πώς θα ενεργήσουν ώστε να διευκολύνουν την ζωή μας. Οι δράσεις αυτές θα πρέπει να βασίζονται στην χρήση γνώσης που αποκτήθηκε από παρόμοιες συσκευές που αντιμετώπισαν παρόμοια προβλήματα. Ωστόσο, καθώς αυτές οι συσκευές γίνονται όλο και ποιο αυτόνομες, πιθανές επιθέσεις μπορεί να οδηγήσουν σε προβλήματα όχι μόνο στον ψηφιακό κόσμο αλλά και στον πραγματικό

Ο σκοπός της παρούσας διπλωματικής εργασίας είναι, ο σχεδιασμός και η υλοποίηση ενός decentralized συστήματος για την διαχείριση της εμπιστοσύνης των συσκευών μεταξύ τους στο πλαίσιο του Κοινωνικού Διαδικτύου των Πραγμάτων (ΚΔτΠ) όπως αυτό απεικονίζεται από το project Cosmos .

Οι μέθοδοι διαχείρισης εμπιστοσύνης βασιζόμενη στη φήμη(reputation-based) έχουν χρησιμοποιηθεί με επιτυχία την προηγούμενη δεκαετία κυρίως σε Peer-to-Peer συστήματα. Για το λόγο αυτό, η εργασία αυτή χτίστηκε επάνω σε state-of-the-art αλγορίθμους που χρησιμοποιούνται σε τέτοια συστήματα, με στόχο την περαιτέρω ανάπτυξη και τροποποίηση των ιδεών ώστε να μπορούν να ενσωματωθούν στο πλαίσιο του Διαδικτύου των Πραγμάτων.

Αυτή η μελέτη προσδιορίζει τις πιθανές απειλές που μπορούν να προκύψουν στο ΚΔτΠ. Στην συνέχεια αναλύεται η προτεινόμενη αρχιτεκτονική του συστήματος διαχείρισης εμπιστοσύνης, η οποία είναι χτισμένη σε δύο παρατηρήσεις πάνω στης κοινωνικές αλληλεπιδράσεις των ανθρώπων. Πρώτον, όταν κάποιος θέλει να έχει πρόσβαση σε μια νέα υπηρεσία ζητά προτάσεις από τους φίλους του. Αυτή η διαδικασία χτίζει την φήμη (reputation) κάποιου. Δεύτερον, όταν κάποιος έχει αρκετές αλληλεπιδράσεις με κάποιον άλλο, η φήμη δεν έχει σημασία πια. Εκεί έχει χτιστεί εμπιστοσύνη.

Μετά από κατάλληλη μοντελοποίηση αυτών των εννοιών χρησιμοποιώντας ιδέες από την Θεωρία των Πιθανοτήτων, προτείνετε το σύστημα RTIoT (Reputation & Trust for the Internet of Things).Στην συνέχεια προστίθενται χαρακτηριστικά, όπως η ικανότητα μίας κακόβουλης Εικονικής Οντότητας (ΕΟ) να εξιλεωθεί , η γρήγορη αναγνώριση αλλαγών στη συμπεριφορά των ΕΟ από αξιόπιστες σε κακόβουλες και η δυναμική ενσωμάτωση νέων ΕΟ στο σύστημα. Στο τέλος παρουσιάζουμε τα αποτελέσματα μέσα από μία προσομοίωση στο TRMSIM-WSN και τα συγκρίνουμε με άλλα συστήματα Εμπιστοσύνης/Φήμης.

Λέξεις κλειδιά

εμπιστοσύνη, φήμη, ασφάλεια , Διαδίκτυο των πραγμάτων, Εικονική οντότητα, Cosmos, ασφάλεια κατανεμημένων συστημάτων, Κοινωνικό Διαδίκτυο των Πραγμάτων, κλιμάκωση

Abstract

In the 21st century the digital world is incorporated on everyday life. Smart devices will soon be able to decide on their own of actions needed to be taken in order to facilitate our lives. These actions will be based on acquiring knowledge from similar devices that have encountered similar problems. However, as these devices are becoming autonomous, potential attacks can result in problems not only in the digital world but also in the physical one.

The purpose of this thesis is the design and implementation of a decentralized system for Trust Management in the context of the Social Internet of Things as seen by the Cosmos project.

Reputation-based trust management methods have been successfully used in the past decade mostly on Peer-to-Peer systems. For this reason, this study is built upon state-of-the-art algorithms used on such systems, with the aim of further developing the ideas proposed and modifying them to fit the context of the Internet of Things.

This study identifies the potential threats that can emerge in SIoT. After that the proposed architecture is analyzed. It is built upon two observations of the social interactions of humans. Firstly, when someone wants to access a new service he asks for referrals from his friends. This feedback is called Reputation. Secondly, when someone has enough interactions with someone else, the reputation does not matter any more. There has been built Trust.

After appropriately modeling these notions using Probability Theory we propose the RTIoT(Reputation & Trust for the Internet of Things) system. then we add features like the ability of a malicious Virtual Entity (VE) to get redemption, the quick identification of behavioural changes of VE's from trustworthy to malicious and the dynamic integration of new VE's in the system. In the end we demonstrate the results of a simulation using the TRMSIM-WSN simulator and compare them with other reputation systems.

Key words

Internet of Things, trust, reputation, scalability, security, Cosmos, Social Internet of Things, Distributed Systems security

Ευχαριστίες

==========

TODO

==========

Ελευθέριος Κόκορης - Κόγιας,
Αθήνα, 03η Ιουλίου 2015

Περιεχόμενα

Κατάλογος σχημάτων

Κατάλογος πινάκων

Κατάλογος Κώδικα

Κεφάλαιο 1

Εισαγωγή

Όταν το 1969 γεννήθηκε το ARPANET η ιδέα που το ώθησε ήταν η κοινή χρήση υπολογιστικών πόρων από απομακρυσμένες περιοχές. Δηλαδή εάν σε κάποιο ερευνητικό κέντρο χρειαζόταν να γίνουν πολλοί υπολογισμοί, να μπορούσαν να γίνουν σε κάποιο εξειδικευμένο υπολογιστικό σύστημα που ήταν γεωγραφικά απομακρυσμένο από το ερευνητικό κέντρο.

Η ιδέα αυτή ποτέ δεν δούλεψε πραγματικά - για αρχή, όλοι οι υπολογιστές είχαν διαφορετικά λειτουργικά συστήματα και προγράμματα, ενώ η χρήση του μηχανήματος κάποιου άλλου ήταν πολύ δύσκολή. Επίσης μέχρι να υλοποιηθεί το ARPANET η τεχνολογία είχε φτάσει στο σημείο να μην το έχει ανάγκη αφού είχαν εμφανιστεί οι πρώτοι προσωπικοί υπολογιστές και ο καταμερισμός χρόνου σε υπολογιστικά συστήματα δεν πρόσφερε ποία τόσα οφέλη.

Έτσι είναι λογικό να πούμε ότι το ARPANET απέτυχε στο σκοπό του, αλλά στη διαδικασία αυτή έκανε μερικές σημαντικές ανακαλύψεις που είχαν ως αποτέλεσμα τη δημιουργία των τεχνολογιών του πρώτου Διαδικτύου. Σε αυτές περιλαμβάνονταν το e-mail, η μεταγωγή πακέτων εφαρμογών, και φυσικά η ανάπτυξη του Transport Control Protocol - Internet Protocol - Internet Protocol ή TCP / IP.

Στην συνέχεια και λόγω της ραγδαίας εξάπλωσης των προσωπικών υπολογιστών το Internet έγινε ή ποίο μεγάλη απόδειξη της ισχύς του νόμου του Metclafe κατά τον οποίο:

> ”Η αξία ενός δικτύου είναι ανάλογη με το τετράγωνο του αριθμού των κόμβων, δηλαδή του αριθμού των χρηστών του δικτύου.”

Σήμερα, το Διαδίκτυο με πάνω από 3 δισεκατομμύρια χρήστες είναι το βασικότερο μέσο εύρεσης και ανταλλαγής δεδομένων της ανθρωπότητας Ένας πολίτης του 21ου αιώνα δεν χρειάζεται να γνωρίζει κάθε μικρή πληροφορία, το μόνο που χρειάζεται να ξέρει είναι σε ποίο σημείο του διαδικτύου μπορεί να την βρει.

Έτσι τώρα, βρισκόμαστε σε μία μεταβατική περίοδο. Το Διαδίκτυο είναι ευρέως αποδεκτό και αναγκαίο, αλλά η έμφυτη ανθρώπινη περιέργεια καθώς και η δημιουργικότητα μας οδηγούν στο επόμενο βήμα. Στο Διαδίκτυο των Πραγμάτων.

Ποία είναι όμως η διάφορα του Διαδικτύου από το Διαδίκτυο των Πραγμάτων;

Τα λεγόμενα του Kevin Ashton, συνιδρυτή και εκτελεστικού διευθυντή του Auto-ID Center στο MIT είναι ξεκάθαρα:

> ”Σήμερα οι υπολογιστές - και, ως εκ τούτου, το Διαδίκτυο - εξαρτώνται σχεδόν εξ ολοκλήρου από τα ανθρώπινα όντα για πληροφορίες. Σχεδόν το σύνολο των περίπου 50 petabytes δεδομένων διαθέσιμων στο Διαδίκτυο, συλλαμβάνεται και δημιουργείται από τον άνθρωπο με την πληκτρολόγηση, το πάτημα ενός κουμπιού εγγραφής, την λήψη μιας ψηφιακής φωτογραφίας ή την σάρωση ενός barcode.
>
> Το πρόβλημα είναι πώς, οι άνθρωποι έχουν περιορισμένο χρόνο, προσοχή και ακρίβεια. Γιαυτό και δεν είναι πολύ καλοί στο καταγραφή των δεδομένων που αφορούν τα πράγματα του γύρω κόσμου. Αν είχαμε υπολογιστές που ήξεραν οτιδήποτε μπορούσαν να ξέρουν για τα πράγματα γύρω τους - με τη χρήση δεδομένων που συνέλεξαν αυτόνομα χωρίς καμία βοήθεια από εμάς - τότε θα ήταν σε θέση να παρακολουθούν και να μετράνε τα πάντα. Έτσι θα μειωνόντουσαν σε μεγάλο βαθμό η σπατάλη αγαθών, οι απώλειες από λάθη και το γενικότερο κόστος. Θα γνωρίζαμε από το πότε τα πράγματα θα χρειάζονται αντικατάσταση ή επισκευή έως και το αν ήταν φρέσκα ή όχι.”

1.1 Κοινωνικό Διαδίκτυο των Πραγμάτων και Προβλήματα Εμπιστοσύνης

Η Διπλωματική αυτή εργασία γεννήθηκε για να αντιμετωπισθούν ανησυχίες σχετικά με τις δράσεις των Εικονικών Οντοτήτων(Virtual Entities), οι οποίες αποτελούν την αναπαράσταση των πραγμάτων στον ψηφιακό κόσμο. Όταν λοιπόν σε αυτές τις Εικονικές Οντότητες προσθέσουμε την δυνατότητα να έχουν κοινωνικούς δεσμούς μέσο φιλίας, μέσο οικογένειας(ανήκουν στον ίδιο χρήστη) ή και μέσο ομοιότητας (έχουν τον ίδιο ρόλο στο σύστημα), τότε βρισκόμαστε στο Κοινωνικό Διαδίκτυο των Πραγμάτων [1]. Ειδικότερα αυτή η εργασία έγινε στα πλαίσια του COSMOS(Cultivate resilient smart Objects for Sustainable city applicatiOn) [2], σκοπός του οποίου είναι η δημιουργία έξυπνων αντικειμένων, προκειμένου να καταστεί δυνατή μια έξυπνη πόλη. Στο COSMOS

- Τα πράγματα θα είναι σε θέση να μάθουν βασιζόμενα σε εμπειρίες άλλον,
- ενώ μέσα από την απόκτηση και την ανάλυση της γνώσης τα πράγματα θα γνωρίζουν τις συνθήκες και τα γεγονότα που συμβαίνουν και ανάλογα θα μεταβάλουν τη συμπεριφορά τους.
- Οι διαχειριστικές αποφάσεις θα λαμβάνονται σε πραγματικό χρόνο για κάθε Εικονική Οντότητα βασιζόμενες στην ασφάλεια των πραγμάτων, την γεωγραφική τους θέση, τις σχέσεις που έχουν με άλλες εικονικές οντότητες καθώς και άλλες ψηφιακές πληροφορίες που θα εξάγονται από
- Complex Event Processing και άλλες τεχνολογίες Κοινωνικών Μέσων ώστε να εντοπίζεται η χρήσιμη πληροφορία μεσα στον τεράστιο αριθμό δεδομένων (Big Data)

[1] www.social-iot.org/
[2] www.iot-cosmos.eu/

Στα πλαίσια λοιπόν του COSMOS υπήρξε η ανάγκη να υπάρχει γνώση για το πόσο έμπιστος μπορεί να θεωρηθεί ένα πάροχος πληροφοριών ή/και υπηρεσιών. Με αυτή τη γνώση θα ήταν δυνατό:

- Να εντοπίζεται γρήγορα η πιο αξιόπιστη πηγή πληροφοριών και να μειώνεται η επικοινωνία με ταυτόχρονη βελτιστοποίηση της απόδοσης
- Να μπορεί μία Εικονική Οντότητα να κρίνει αυτόνομα πόσο πιθανό είναι να ικανοποιηθεί από μία προσφερόμενη υπηρεσία πριν προβεί στην δοσοληψία επειδή
  - μπορεί κάποια υπηρεσία να είναι προϊόν
  - μπορεί να μας δοθεί γνώση που θα ρισκάρει την ασφάλεια του συστήματος ή/και των τελικών χρηστών

1.2 Δομή της Διπλωματική Εργασίας

Η διπλωματική εργασία δομείται ως εξής:

Chapter 2
Σε αυτό το κεφάλαιο γίνεται μία εισαγωγή στη βασικές ιδέες που χαρακτηρίζουν γενικά τα συστήματα εμπιστοσύνης - φήμης. Στην συνέχεια αναλύονται οι βασικές αρχιτεκτονικές τους καθώς και οι διάφοροι τρόποι εξαγωγής των δεικτών φήμης και εμπιστοσύνης. Τέλος παρουσιάζονται βασικά συστήματα φήμης και εμπιστοσύνης που χρησιμοποιούνται σε διάφορους τομείς της υπολογιστικής επιστήμης.

Chapter 3
Σε αυτό το κεφάλαιο παρουσιάζονται τα είδη των απειλών που μπορούν να προκύψουν και προτάσεις αντιμετώπισής τους. Παράλληλα περιγράφεται ο σχεδιασμός του συστήματος και δίνονται βασικά παραδείγματα εκτέλεσης.

Chapter 4
Σε αυτό το κεφάλαιο παρουσιάζεται το εργαλείο προσημειώσεων TRMSim-WSN. Πάνω σε αυτό προστέθηκε ένα μοντέλο του συστήματος φήμης εμπιστοσύνης και αναλύεται ο τρόπος που έγινε αυτό.

Chapter ??
EVALUATION We cite our experience of using BlkKin in Archipelago and RADOS instrumentation and its use as a debugging and an alerting mechanism.

Chapter ??
CONCULSION-FUTURE WORK We provide some concluding remarks and give some future work that could be done to improve and evolve BlkKin.

Κεφάλαιο 2

Θεωρητικό και Τεχνολογικό Υπόβαθρο

2.1 Ορισμοί Εννοιών

Τα συστήματα εμπιστοσύνης και φήμης αποτελούν μία εξέλιξη της τελευταίας δεκαπενταετίας και χρησιμοποιούνται ευρέως σε διαδικτυακές αλληλεπιδράσεις μεταξύ τόσο ανθρώπων (e-bay, amazon) όσων και μηχανών (ευφυών δραστών, peer-to-peer συστημάτων κτλ.). Η ιδέα πάνω στην οποία βασίζονται είναι: η συλλογή κριτικής για έναν δράστη από άλλα μέλη της κοινότητας τα οποία έχουν ήδη αποκτήσει κάποιες εμπειρίες με τον πρώτο. Με αυτόν τον τρόπο υπάρχει η δυνατότητα να γνωρίζει κάποιος για ”το ποιόν” κάποιου άλλου χωρίς να χρειάζεται να τον γνωρίσει, δηλαδή να αλληλεπιδράσει με αυτόν. Αυτή η δυνατότητα με την σειρά της ”εξαναγκάζει” του δράστες ενός συστήμηματος να συμμορφωθούν με τους κανόνες της δικτυακή κοινότητας της οποίας είναι μέλη.

Όταν η αίτηση για παροχή υπηρεσιών ή για απόκτηση δεδομένων γίνεται online μεταξύ οντοτήτων πρότερα αγνώστων μεταξύ τους τότε μπορούν να προκύψουν 2 βασικά προβλήματα.

1. Το πρώτο πρόβλημα που δεν εξετάζεται στην παρούσα εργασία είναι πώς τα δεδομένα που παρέχονται (και στην περίπτωση του COSMOS η γνώση) μπορεί να χρησιμοποιηθούν από τον παραλήπτη για διαφορετικούς σκοπούς από αυτούς που ισχυρίζεται. Έτσι να τον βοηθήσουμε άθελα μας στου κακόβουλους ή γενικότερα αντίθετους από το καλό της κοινότητας σκοπούς του.
2. Αντίθετα αυτή η εργασία εστιάζει στο πρόβλημα του ρίσκου πού παίρνει ένας χρήστης όταν ζητάει μία υπηρεσία, επειδή μπορεί τελικά να μην την λάβει σε ικανοποιητικό βαθμό οπότε να πρέπει να την ξαναζητήσει. Ακόμα είναι δυνατόν να κινδυνέψει η ασφάλεια του (αντί για σωστό αρχείο να σταλεί κάτι που περιέχει ένα Trojan), ενώ στην δική μας περίπτωση μπορεί η γνώση και εν συνεχεία οι δράσεις που προτείνονται να προκαλέσουν προβλήματα ασφαλείας όχι μόνο στον ψηφιακό κόσμο των Εικονικών Οντοτήτων, όσο και στο φυσικό κόσμο των πραγμάτων και των ανθρώπων.

2.1.1 Soft Security

Ως ασφάλεια στην πιο γενική της έννοια μπορούμε να ορίσουμε ”την κατάσταση στην οποία κάποιος ή κάτι βρίσκεται μακριά από οποιονδήποτε κίνδυνο και οποιαδήποτε προσπάθεια αλλοίωσης αυτής της κατάστασης θα αποτύχει” (ή και να πετύχει θα είναι δυνατή η επαναφορά στην αρχική κατάσταση χωρίς κόστος). Αυτός ο ορισμός είναι πολύ γενικός και περιλαμβάνει την ασφάλεια ανθρώπων την ασφάλεια περιουσιών ακόμα και την ασφάλεια υγείας.

Στην επιστήμη των υπολογιστών η βασικότερη κατηγορία ασφάλειας είναι η ασφάλεια που παρέχει ως υπηρεσία τον αποκλεισμό δραστών εξωτερικών του συστήματος από την απόκτηση πρόσβασης σε προστατευόμενες πληροφορίες ή πόρους. Αυτό το είδος ”hard security” παρέχεται μέσα από την πιστοποίηση και τον έλεγχο πρόσβασης(access control) στα υπολογιστικά συστήματα καθώς και μέσα από την κρυπτογραφία στα κανάλια επικοινωνίας κατά την ανταλλαγή εμπιστευτικών πληροφοριών. Όμως το πρόβλημα μπορεί να αντιστραφεί και τότε όλες αυτές οι μέθοδοι είναι αναξιόπιστες. Αυτό θα συμβεί στην περίπτωση που η επίθεση γίνει εντός του συστήματος. Μία τέτοια επίθεση συμβαίνει όταν κάποιος δράστης ή γενικότερα κομμάτι του συστήματος, που έχει δικαίωμα να ενεργεί στο σύστημα, γίνει κακόβουλο ή ελαττωματικό.

Αυτά τα προβλήματα λύνονται μέσα από εφαρμογές του ”Soft Security” το οποίο είναι

Ορισμός 1 (Soft Security) 1: *Η συνεργασία των δραστών ενός συστήματος ώστε να αποκλείσουν μη κοινωνικά αποδεχτές συμπεριφορές.*

Με αυτόν τον τρόπο μπορεί να προστατευθεί η κοινότητα από κακόβουλους δράστες οι οποίοι όμως έχουν κάθε δικαίωμα να ενεργούν. Ουσιαστικά να εφαρμοστεί ένα είδος κοινωνικού αποκλεισμού. Ένας από τους βασικότερους μηχανισμούς για την επίτευξή αυτού του είδους ασφαλείας είναι με την χρήση των εννοιών της εμπιστοσύνης και της φήμης. [1]

2.1.2 Η Έννοια της Εμπιστοσύνης

Στον κόσμο των ανθρώπινων οντοτήτων η ύπαρξη της έννοιας τις εμπιστοσύνης είναι εμφανής στις καθημερινές μας αλληλεπιδράσεις. Θα μιλήσουμε για τα προβλήματα μας και της εμπειρίες μας -δηλαδή θα δώσουμε ευαίσθητες προσωπικές πληροφορίες- στους φίλους μας οι οποίοι περιμένουμε να μας βοηθήσουνε με δικές τους εμπειρίες, καθώς και να βοηθηθούν από τις πληροφορίες μας. Επίσης θα πάμε να αγοράσουμε προϊόντα από μαγαζιά που γνωρίζουμε και έχουμε ξαναπάει επειδή πιστεύουμε πως η ποιότητα ου προϊόντος ή της υπηρεσίας που θα λάβουμε θα είναι ικανοποιητική για εμάς.

Στον κόσμο όμως της υπολογιστικής επιστήμης ο ορισμός και η μοντελοποίηση της εμπιστοσύνης είναι αρκετά δύσκολη. Αυτό συμβαίνει επειδή τις ανθρώπινες σχέσεις τις κυβερνούνε συναισθήματα όπως αγάπη και αφοσίωση που δεν έχουν καμία θέση στον ψηφιακό κόσμο. Στο έργο του Audun Jøsang υπάρχει μία εξαιρετική συγκέντρωση για τους διαφόρους ορισμούς της εμπιστοσύνης και τις φήμης. Εκεί λοιπόν αναφέρεται πώς υπάρχουν δύο βασικοί

[1] Φυσικά και η έννοια της εμπιστοσύνης είναι χρήσιμη και για το ”hard security” (εμπιστοσύνη σε ένα πιστοποιητικό ή μία Certificate Authority που το υπογράφει), αλλά ουσιαστικά αυτό είναι μία εφαρμογή ”Soft Security” της ανθρώπινης κοινωνίας που χρησιμεύει στο hard security της υπολογιστικής επιστήμης.

ορισμοί για την εμπιστοσύνη τους οποίους ονομάζουμε εμπιστοσύνη αξιοπιστίας (reliability trust) και εμπιστοσύνη απόφασης (decision trust)

Όπως προϊδεάζει και το όνομα η εμπιστοσύνη αξιοπιστίας μπορεί να ερμηνευθεί ως η αξιοπιστία κάποιου δράστη ή αντικειμένου, και ο ορισμός του Gambetta παρέχει ένα παράδειγμα του πώς μπορεί να διατυπωθεί:

Ορισμός 2 (Εμπιστοσύνη Αξιοπιστίας) 1: *Εμπιστοσύνη είναι η υποκειμενική πιθανότητα από την οποία ένα άτομο, Α, περιμένει από ένα άλλο άτομο, Β, να κάνει μία συγκεκριμένη ενέργεια από την οποία εξαρτάται η ευημερία του.*

Αυτός ορισμός περιεχέι την έννοια της εξάρτησης του Α από τον Β, και την αξιοπιστία (δηλαδή την πιθανότητα) του Β όπως την αντιλαμβάνεται ο Α.

Όμως η εμπιστοσύνη μπορεί να είναι πολύ πιο σύνθετη από τον παραπάνω ορισμό. Αυτό συμβαίνει επειδή το γεγονός ότι η αξιοπιστία ενός δράστη είναι υψηλή δεν οδηγεί στην τυφλή εμπιστοσύνη του για οποιαδήποτε ενέργεια. Το ρίσκο τις αποτυχίας μίας συγκεκριμένη συναλλαγής μπορεί να είναι πολύ μεγάλο. Δηλαδή με απλά λόγια, ένας δράστης μπορεί να θεωρηθεί αξιόπιστος για μία απλή συναλλαγή, όμως για κάποια πιο κρίσιμη η αξιοπιστία του να μην είναι αρκετά καθησυχαστική. Για να συμπεριλάβουμε λοιπόν την πιο ευρεία έννοια που μπορεί να χαρακτηριστεί ως εμπιστοσύνη, μπορεί να χρησιμοποιηθεί ο ακόλουθος ορισμός των McKnight & Chervany

Ορισμός 3 (Εμπιστοσύνη Απόφασης) 1: *Η εμπιστοσύνη είναι ο βαθμός στον οποίο ένα άτομο είναι πρόθυμο να εξαρτηθεί από κάτι ή κάποιον σε μια δεδομένη κατάσταση με ένα αίσθημα σχετικής ασφάλειας, ακόμη και αν οι αρνητικές συνέπειες είναι δυνατές.*

Ο γενικότερος αυτός ορισμός μας παρέχει μία χρήσιμη ασάφεια αφού περιλαμβάνει πτυχές μίας ευρύτερης έννοιας της εμπιστοσύνης, η οποία περιέχει της έννοιες της εξάρτησης από αυτόν που απολαμβάνει την εμπιστοσύνη, της αξιοπιστίας αυτού καθώς και της ύπαρξης ενός παράγοντα κινδύνου που πρέπει να αποδεχτεί το άτομο που καλείτε να εμπιστευτεί κάποιο άλλο άτομο, ανάλογο με την εκάστοτε συναλλαγή.

2.1.3 Η Έννοια της Φήμης

Γυρίζοντας πίσω στον κόσμο τον ανθρώπων μπορούμε εύκολα να εντοπίσουμε άλλη μία έννοια η οποία χαρακτηρίζει εάν μεγάλο αριθμό ενεργειών και αποφάσεων μας. Αυτή η έννοια είναι η φήμη. Όταν για παράδειγμα θέλει ένα άτομο να αποκτήσει πρόσβαση σε μία υπηρεσία καινούργια (δηλαδή που δεν την έχει ξαναχρησιμοποιήσει), όπως για παράδειγμα να πάει σε ένα φροντιστήριο το παιδί του να μάθει μία ξένη γλώσσα ή να βρει έναν καλό μηχανικό για να του χτίσει το σπίτι του. Τότε ένα από τους βασικούς παράγοντες της επιλογής θα είναι η φήμη. Δηλαδή θα συλλέξει πληροφορίες από τον κοινωνικό του περίγυρο και αφού τις σταθμίσει ανάλογα με την εμπιστοσύνη του στης πηγές προέλευσής τους θα εξάγει την φήμη του πιθανού παρόχου της υπηρεσίας.

Στην Υπολογιστική Επιστήμη μοντελοποιούμε την φήμη με ακριβώς τον ίδιο τρόπο. Έτσι μπορούμε να ορίσουμε την φήμη ως:

Ορισμός 4 (Φήμη) 1: *Φήμη είναι το τι λέγεται ή πιστεύεται δημόσια για το ποιόν ή τον χαρακτήρα κάποιου ατόμου ή αντικειμένου*

Παρ'όλη την στενή συγγένεια των εννοιών της φήμης και της εμπιστοσύνης, έχουν μία θεμελιώδη διαφορά. Η φήμη βασίζεται στην εμπειρία του κοινωνικού συνόλου με ένα άτομο ενώ η εμπιστοσύνη είναι προσωπικό ζήτημα και δεν επηρεάζεται από το κοινωνικό σύνολο. Μπορούμε να δούμε ξεκάθαρα την διαφορά τους στης παρακάτω προτάσεις:

1. "Θα εμπιστευτώ τον Χ επειδή έχει καλή φήμη"
2. "Θα εμπιστευτώ τον Υ παρόλο που έχει κακή φήμη, επειδή μαζί μου είναι καλός"

Οι παραπάνω φράσεις δείχνουν ξεκάθαρα πώς παρόλο που η υπηρεσία που τα δυο άτομα θέλουν αιτηθούν είναι η ίδια, το άτομο 1, που δεν έχει αλληλεπιδράσει ποτέ με τον Υ, θα προτιμήσει τον Χ εμπιστευόμενος το κοινωνικό σύνολο από όπου εξήγε την καλή φήμη του Χ. Αντίθετα το άτομο 2 ένα και ξέρει πως ο Χ έχει καλύτερη φήμη από τον Υ θα διαλέξει τον Υ επειδή έχει καλή προϊστορία μαζί του, ουσιαστικά επειδή τον εμπιστεύεται. Αυτή η παρατήρηση δείχνει πώς η υποκειμενική γνώμη ενός ατόμου για ένα άλλο έχει μεγαλύτερο βάρος από την γνώμη του κοινωνικού σύνολου και έτσι στο τέλος η εμπιστοσύνη σε ένα άτομο θα υπερισχύσει της φήμης του.

2.2 Αρχιτεκτονικές Δικτύωσης Συστημάτων Εμπιστοσύνης

Όπως αναφέραμε οι κλασσικοί τρόποι που έχει η κοινωνία για την εξαγωγή φήμης και εμπιστοσύνης απουσιάζουν από τα online συστήματα που κατασκευάζουμε. Για αυτό τον λόγο χρειαζόμαστε ηλεκτρονικά υποκατάστατα. Οι βασικότερες ιδιότητες που πρέπει να έχει ένα ένα σύστημα βασιζόμενο στην φήμη είνα:

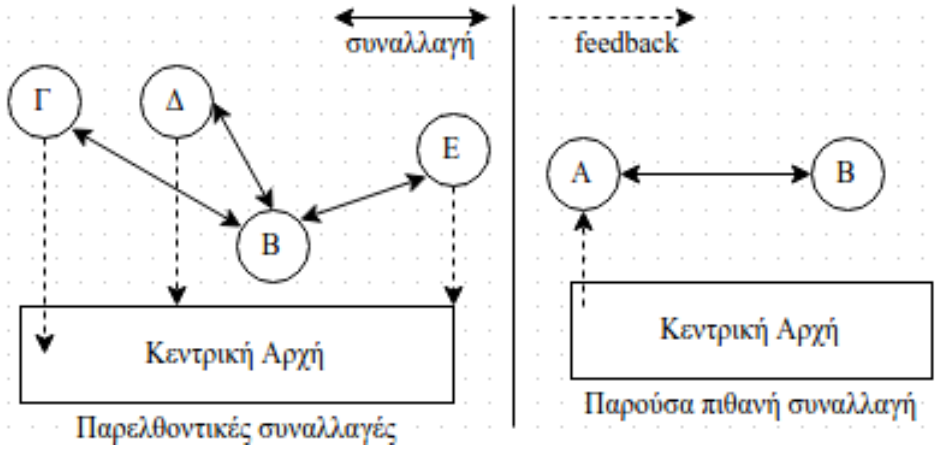
1. Οι οντότητες πρέπει να είναι μακρόβιες έτσι ώστε μετά από μία συναλλαγή να υπάρχει πιθανότητα μία μελλοντικής επαφής με την ίδια οντότητα
2. Οι συναλλαγές να αποτιμώνται και οι βαθμολογίες να διανέμονται σε άλλα μέλη τις κοινότητας
3. Οι αποφάσεις για της συναλλαγές μίας οντότητας να επηρεάζονται από τις βαθμολογίες των παλιών συναλλαγών

Οι τεχνικές αρχές για τα συστήματα φήμης περιγράφονται στο παρών και το ακόλουθο τμήμα. Η αρχιτεκτονική του δικτύου καθορίζει πώς οι αξιολογήσεις των συναλλαγών αξιοποιούνται και διανέμονται ανάμεσα στα μέλη των συστημάτων. Οι δύο κύριοι τύποι είναι οι συγκεντρωτικές και οι κατανεμημένες αρχιτεκτονικές.

2.2.1 Συγκεντρωτικές Αρχιτεκτονικές

Στις συγκεντρωτικές αρχιτεκτονικές, οι πληροφορίες για την επίδοση ενός μέλους συγκεντρώνονται από τις βαθμολογίες των άλλων μελών της κοινότητας οι οποίοι είχαν άμεση εμπειρία με τον πρώτο. Η κεντρική αρχή η οποία συγκεντρώνει τις βαθμολογίες εξάγει τελικά την δείκτη της φήμης του κάθε μέλους της κοινότητας τον οποίο μπορεί να δει όποιος ενδιαφέρεται. Τα άλλα μέλη με την σειρά του χρησιμοποιούν αυτούς τους δείκτες όταν θέλουν να αλληλεπιδράσουν με άγνωστα, για αυτούς, μέλη της κοινότητας και θέλουν να αποφασίσουν εάν θα προβούν σε κάποια συναλλαγή ή όχι. Η ουσία είναι πώς συναλλαγές με άτομα που έχουνε υψηλότερο δείκτη φήμης έχουν πιο πολλές πιθανότητες να είναι επιτυχημένες από συναλλαγές με άτομα χαμηλού δείκτη φήμης.

Στο Σχ.2.1 παρακάτω φαίνεται μία τυπική αρχιτεκτονική με κεντρική αρχή.



Σχήμα 2.1: Συγκεντρωτική Αρχιτεκτονική

Ο Α θέλει να προβεί σε μία συναλλαγή με τον Β, ενώ δεν έχει κάποια παρελθοντική εμπειρία με αυτόν. Για αυτό ρωτάει την κεντρική αρχή, η οποία με την σειρά της γυρίζει τον δείκτη φήμης που έχει υπολογίσει βασιζόμενη σε πληροφορίες από άλλα μέλη τις κοινότητας όταν αυτά αλληλεπίδρασαν με τον Β και κοινοποίησαν στην κεντρική αρχή την εμπειρία τους.

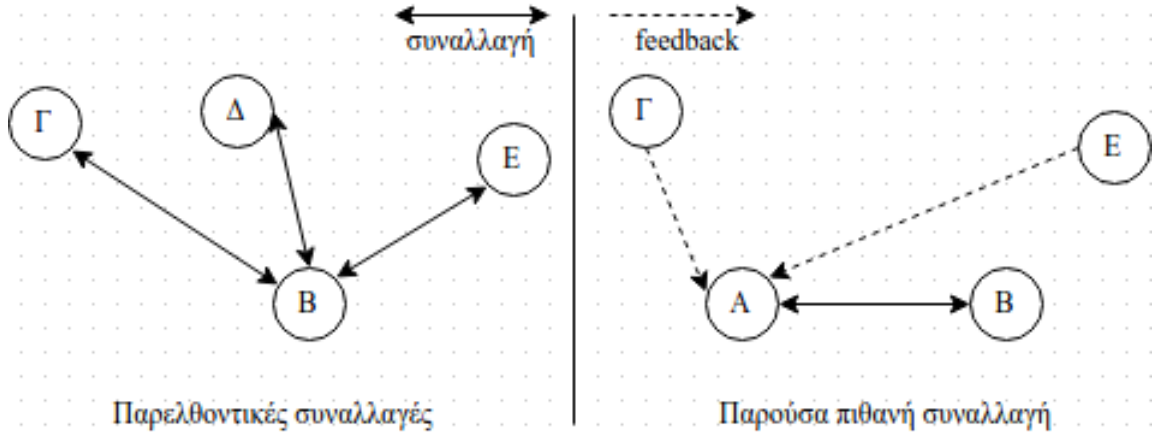
Τα δύο βασικά κομμάτια των συγκεντρωτικών συστημάτων φήμης είναι:

1. Τα συγκεντρωτικά πρωτόκολλα επικοινωνίας που επιτρέπουν στους συμμετέχοντες να παρέχουν αξιολογήσεις για τα μέλη των συναλλαγών στην κεντρική αρχή, καθώς και να αποκτήσουν έναν δείκτη φήμης των πιθανών εταίρων στην συναλλαγή από την κεντρική εξουσία.

2. Έναν τρόπο υπολογισμού φήμης που χρησιμοποιείται από την κεντρική αρχή να εξάγει δείκτες φήμη για κάθε μέλος, με βάση τις αξιολογήσεις που έλαβε, και ενδεχομένως άλλες πληροφορίες όπως θα δούμε στην παράγραφο 2.3 παρακάτω.

2.2.2 Κατανεμημένες Αρχιτεκτονικές

Υπάρχουν περιβάλλοντα όπου ένα κατανεμημένο σύστημα φήμης, δηλαδή χωρίς τις κεντρικές λειτουργίες, είναι καταλληλότερο από ένα συγκεντρωτικό σύστημα. Σε ένα κατανεμημένο σύστημα, δεν υπάρχει κεντρικό σημείο για την υποβολή των αξιολογήσεων ή την απόκτηση δεικτών φήμης των άλλων. Αντ'αυτού, μπορεί να υπάρχουν κατανεμημένα σημεία αποθήκευσης όπου μπορούν να υποβληθούν αξιολογήσεις ή ο κάθε συμμετέχοντας απλά να καταγράφει τη γνώμη του σχετικά με τις εμπειρίες του με άλλα μέλη της κοινότητας, και να παρέχει ο ίδιο τις πληροφορίες αυτές, κατόπιν αιτήματος. Ένας τρίτος συμβαλλόμενος, ο οποίος ενδιαφέρεται να συναλλαχθεί με δεδομένο άλλο μέλος της κοινότητας, πρέπει να βρει τις κατανεμημένες αποθήκες, ή να προσπαθήσει να αποκτήσει βαθμολογίες από όσα μέλη της κοινότητας μπορεί τα οποία είχαν άμεση εμπειρία με το εν λόγω μέλος-στόχο. Αυτό απεικονίζεται στο Σχ.2.2 παρακάτω.



Σχήμα 2.2: Κατανεμημένη Αρχιτεκτονική

Ο τρίτος συμβαλλόμενος (στο Σχ.2.2 ο Α) υπολογίζει μόνος του των δείκτη της φήμης του μέλους στόχου (Β) βασιζόμενος στην μερική ή ολική γνώση που συνέλεξες από ένα τους "γνωστούς" του. (εδώ έχει μερική γνώση επειδή δεν πήρε feedback από τον Δ).

Τα δύο βασικά κομμάτια των κατανεμημένων συστημάτων φήμης είναι:

1. Τα κατανεμημένα πρωτόκολλα επικοινωνίας που επιτρέπουν στους συμμετέχοντες να λάβουν και να στείλουν βαθμολογίες για άλλα μέλη της κοινότητας
2. Έναν τρόπο υπολογισμού φήμης που χρησιμοποιείται από κάθε μέλος της κοινότητας για να εξάγει δείκτες φήμης με βάση τις αξιολογήσεις που έλαβε, και ενδεχομένως άλλες πληροφορίες όπως θα δούμε στην παράγραφο 2.3 παρακάτω.

2.3 Τρόποι υπολογισμοί φήμης και εμπιστοσύνης

2.3.1 Bayesian

Τα Bayesian συστήματα λαμβάνουν δυαδικές αξιολογήσεις ως είσοδο (δηλαδή θετική ή αρνητική τιμή), και βασίζονται στον υπολογισμό των δεικτών με χρήση των *βήτα συναρτήσεων πυκνότητας πιθανότητας* (ΣΠΠ). Ο ενημερωμένος δείκτης υπολογίζεται συνδυάζοντας τις προηγούμενες τιμές των δεικτών φήμης με τη νέα βαθμολόγια.Ο δείκτης της φήμης ή/και της εμπιστοσύνης μπορεί να αναπαρασταθεί είτε με τη μορφή μιας πλειάδας παραμέτρων βήτα μορφής *(α, β)* (όπου *α* και *β* αντιπροσωπεύουν τον αριθμό των θετικών και αρνητικών αξιολογήσεων, αντίστοιχα), είτε με τη μορφή της τιμής της προσδοκία της βήτα ΣΠΠ. Προαιρετικά μπορεί να υπάρχει και ή διακύμανση. Το πλεονέκτημα των Bayesian συστημάτων είναι ότι παρέχουν μία θεωρητικά ορθή βάση για τον υπολογισμό των δεικτών, ενώ το μόνο μειονέκτημα που θα μπορούσε να καταλογιστεί είναι ότι είναι υπερβολικά πολύπλοκα για να τα κατανοήσει ο μέσος άνθρωπος.

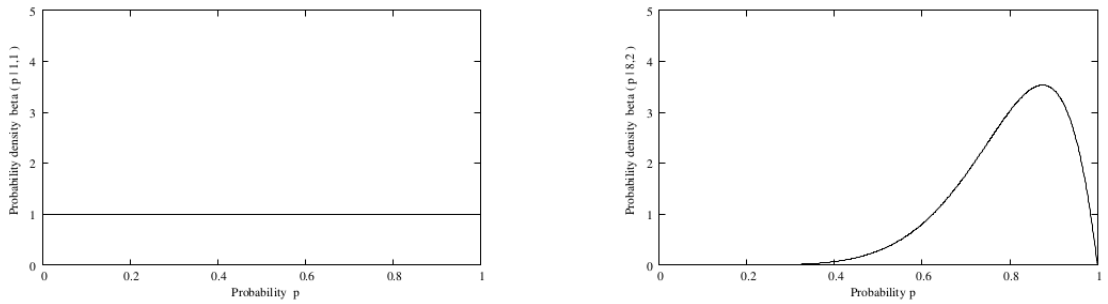
Η βήτα οικογένεια κατανομών είναι μια συνεχής οικογένεια συναρτήσεων κατανομής μεταβαλλόμενη από τις δύο παραμέτρους α και β. Η βήτα ΣΣΠ συμβολίζεται με beta(p | α, β) μπορεί να εκφραστεί χρησιμοποιώντας τη συνάρτηση γάμμα Γ ως εξής:

$$beta(p|\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\,\Gamma(\beta)} p^{\alpha-1}(1-p)^{\beta-1} \; o\pi ou \;\; 0 \le p \le 1,\; \alpha,\; \beta \;>\; 0 \qquad (2.1)$$

με τον περιορισμό ότι η πιθανότητα $p \neq 0 \;\; \alpha\nu \; \alpha \;<\; 1$, και $p \neq 1 \;\; \alpha\nu \; \beta \;<\; 1$. Η τιμή της προσδοκίας της κατανομής beta δίνεται από:

$$E(p) = \frac{\alpha}{(\alpha+\beta)} \qquad (2.2)$$

Στην αρχική κατάσταση όπου δεν είναι τίποτα γνωστό η εκ των προτέρων ΣΠΠ είναι η ομοιόμορφη κατανομή βήτα κατανομή όπου $\alpha = 1,\; \beta = 1$ η οποία φαίνεται στο σχήμα 2.3 παρακάτω. Στην συνέχεια αφού παρατηρηθούν ν θετικές και μ αρνητικές βαθμολογίες η εξαγόμενη κατανομή είναι η βήτα κατανομή(α,β) όπου α=ν+1 και β=μ+1. Για παράδειγμα εάν παρατηρηθούν 7 θετικές και 1 αρνητική βαθμολογίες τότε η ΣΠΠ φαίνεται στο σχήμα 2.3 .



(a) Uniform PDF beta$(p \mid 1, 1)$ (b) PDF beta$(p \mid 8, 2)$

Σχήμα 2.3: Βήτα Συναρτήσεις Πυκνότητας Πιθανότητας

Μία ΣΠΠ αυτού του τύπου εκφράζει την αβέβαιη πιθανότητα ότι οι μελλοντικές αλληλεπιδράσεις θα είναι θετικές. Το πιο φυσικό είναι να καθοριστεί ο δείκτης φήμης ως συνάρτηση της τιμής της προσδοκίας. Η τιμή αυτή (της πιθανότητας προσδοκίας) σύμφωνα με την Εξ. 2.2 είναι E(p) = 0,8. Αυτό μπορεί να ερμηνευθεί λέγοντας ότι η σχετική συχνότητα μιας θετικής έκβασης στο μέλλον είναι αβέβαιη, και ότι ή πιθανότερη τιμή της (της συχνότητας) είναι 0,8.

2.3.2 Διακριτών Καταστάσεων

Οι άνθρωποι μπορούν συχνά να αξιολογήσουν καλύτερα την απόδοση συστημάτων όταν οι τιμές έχουν διακριτή λεκτική μορφή, σε σχέση με όταν είναι συνεχείς μετρήσεις. Αυτό ισχύει επίσης και για τον καθορισμό των δεικτών εμπιστοσύνης. Ορισμένοι συγγραφείς, συμπεριλαμβανομένων έχουν προτείνει διακριτά μοντέλα εμπιστοσύνης. Για παράδειγμα, στο μοντέλο του Αμπντούλ Ραχμάν & Hailes (2000) , η αξιοπιστία ενός παράγοντα Χ μπορεί να έχει τιμές: Πολύ Αξιόπιστη, Αξιόπιστη, Αναξιόπιστη, Πολύ Αναξιόπιστη Το μέλος της κοινότητας (Υ) που θέλει να βρει την αξιοπιστία κάποιου μπορεί να συνδυάσει τη δική του άποψη για την αξιοπιστία του Χ με την φήμη που εξάγει από την κοινότητα.

Look-up πίνακες, με εγγραφές για την υποβάθμιση / αναβάθμιση της εμπιστοσύνης που έχει ο Υ στον Χ, χρησιμοποιούνται για να καθορίσουν την εξαγόμενη εμπιστοσύνη στο Χ. Κάθε φορά που ο Υ είχε προσωπική εμπειρία με το Χ, αυτό μπορεί να χρησιμοποιηθεί για τον προσδιορισμό τής αξιοπιστίας του. Η υπόθεση είναι ότι η προσωπική εμπειρία αντικατοπτρίζει την πραγματική αξιοπιστία του Χ και ότι η φήμη που λαμβάνει για τον Χ, η οποία διαφέρει από την προσωπική εμπειρία, δείχνει αν ο Υ υποτιμά ή υπερτιμά τον Χ. Παρατηρήσεις από ”φίλους” που υπερεκτιμούν των Υ θα υποβαθμιστεί πριν ληφθούν υπόψιν, και το αντίστροφο.

Το μειονέκτημα των διακριτών τιμών είναι ότι δεν είναι εύκολο να εφαρμοστούν σε αυτά κλασσικοί υπολογισμοί. Αντ ’αυτού, πρέπει να χρησιμοποιηθούν ευριστικοί μηχανισμοί όπως look-up πίνακες.

2.3.3 Ασαφούς Λογικής

Η εμπιστοσύνη και η φήμη μπορούν να παρασταθούν ως γλωσσικά ασαφείς έννοιες, όπου συναρτήσεις συμμετοχής περιγράφουν σε ποιο βαθμό ένας παράγοντας μπορεί να περιγραφεί ως π.χ. αξιόπιστος ή αναξιόπιστες. Η ασαφής λογική προβλέπει κανόνες για το συλλογισμό με ασαφείς μετρικές αυτού του είδους. Το σύστημα που προτείνει Manchala (1988,καθώς και η εξαγωγή τής φήμη στο συστήματος REGRET που προτείνουν οί Sabater & Sierra (2001,2002) εμπίπτουν σε αυτή την κατηγορία. Στο σύστημα των Sabater & Sierra, αυτό που αποκαλούν ατομική φήμη προέρχεται από ιδιωτικές πληροφορίες σχετικά με ένα συγκεκριμένο δράστη ενώ αυτό που ονομάζουν κοινωνική φήμη προέρχεται από την ενημέρωση του κοινού σχετικά με αυτόν τον δράστη, τέλος αυτό που αποκαλούν context dependent φήμη εξάγεται από τις συμφραζόμενες πληροφορίες.

2.3.4 Ροής Εμπιστοσύνης

Τα συστήματα που υπολογίζουν την εμπιστοσύνη ή τη φήμη με επανάληψη μέσω βρόχου ή με αυθαίρετα μεγάλες αλυσίδες μπορούν να ονομαστούν μοντέλα ροής.

Ορισμένα μοντέλα ροής έχουν σταθερή τιμή ολικής εμπιστοσύνης / φήμης για το σύνολο του συστήματος και η τιμή αυτή μπορεί να κατανεμηθεί μεταξύ των μελών της κοινότητας. Οι συμμετέχοντες μπορούν να αυξήσουν την εμπιστοσύνη / φήμη τους μόνο σε βάρος των άλλων. Ό PageRank της Google [52] ανήκει σε αυτή την κατηγορία. Σε γενικές γραμμές, η φήμη ενός συμμετέχοντος αυξάνει ως συνάρτηση της εισερχόμενης ροής, και μειώνεται ως συνάρτηση της εξερχόμενης ροής. Στην περίπτωση της Google, πολλοί υπερσύνδεσμοι εισερχόμενοι (που δείχνουν σε αυτή) σε μια ιστοσελίδα συμβάλλουν στην αύξηση του δείκτη PageRank ενώ πολλοί υπερσύνδεσμοι εξερχόμενοι (που υπάρχουν μέσα) από μια ιστοσελίδα οδηγούν στη μείωση του δείκτη PageRank για την εν λόγω ιστοσελίδα.

Τα μοντέλα ροής δεν απαιτούν πάντα το άθροισμα των βαθμολογιών φήμης / εμπιστοσύνης να είναι σταθερό. Ένα τέτοιο παράδειγμα είναι το σύστημα EigenTrust [35], το οποία υπολογίζει δείκτες εμπιστοσύνη σε δίκτυα P2P μέσω επαναλαμβανόμενου πολλαπλασιασμού και ομαδοποίησης των βαθμολογιών εμπιστοσύνης έως ότου οι δείκτες εμπιστοσύνης για όλα τα μέλη της P2P κοινότητας συγκλίνουν σε σταθερές τιμές.

2.3.5 Άθροιση ή εξαγωγή Μέσου Όρου

Η συνηθέστερη μορφή υπολογισμού φήμης / εμπιστοσύνης είναι να υπολογίζεται το συνολικό αριθμό των θετικών αξιολογήσεων και το συνολικό αριθμό των αρνητικών αξιολογήσεις ξεχωριστά, και να εξάγετε μια συνολική βαθμολογία ως το θετικό άθροισμα μείον το αρνητικό. Αυτή είναι η αρχή που χρησιμοποιείται στο φόρουμ του eBay η οποία περιγράφεται λεπτομερώς στο [55]. Το πλεονέκτημα είναι ότι ο καθένας μπορεί να καταλάβει την αρχή πίσω τον υπολογισμό της φήμη, το μειονέκτημα είναι ότι ο υπολογισμός είναι πρωτόγονος και, ως εκ τούτου ο δείκτης της φήμης δίνει μιας φτωχής εικόνα αν και αυτό οφείλεται επίσης στον τρόπο που παρέχεται αξιολόγηση.

Ένα ελαφρώς πιο προχωρημένο σύστημα που προτείνεται στο π.χ. [63] είναι να υπολογιστεί ο δείκτης της φήμη ως ο μέσος όρος όλων των αξιολογήσεων. Η αρχή αυτή χρησιμοποιείται στα συστήματα φήμης πολλών εμπορικές ιστοσελίδων, όπως της Amazon περιγράφεται στο 2.4

Προηγμένα μοντέλα σε αυτή την κατηγορία υπολογίζουν ένα σταθμισμένο μέσο όρο όλων των αξιολογήσεων, όπου το βάρος της αξιολόγησης μπορεί να καθορίζεται από παράγοντες όπως ο εκτιμητής της αξιοπιστίας / φήμη, τη ηλικία της αξιολόγησης, η σχέση μεταξύ αξιολογητή και αξιολογούμενου κ.λ.π. Το σύστημα που παρουσιάζεται στην υπόλοιπη εργασία εμπίπτει σε αυτήν την κατηγορία.

2.4 Γνωστά Συστήματα φήμης κ εμπιστοσύνης σε διαφόρους τομείς

Τα συστήματα φήμης και εμπιστοσύνης είναι ευρέως διαδεδομένα σε πολλούς τομείς όπου χρησιμοποιούνται πληροφοριακά συστήματα. Ο βασικός λόγος που εφαρμόζεται ένα σύστημα φήμης-εμπιστοσύνης είναι για να αντιμετωπίσει κακόβουλους συμμετέχοντες σε μία κοινότητα οντοτήτων και έτσι να βελτιωθεί τόσο η ασφάλεια όσο και η ποιότητα των υπηρεσιών.

2.4.1 Business

E-bay

Amazon

PageRank

2.4.2 Mobile-ad-hoc Networks

Le Boudec

hubeaux

stanford

2.4.3 Peer-to-Peer Systems

EingenTrust

PeerTrust

PowerTrust

2.4.4 Internet of Things

Atzori

Κεφάλαιο 3

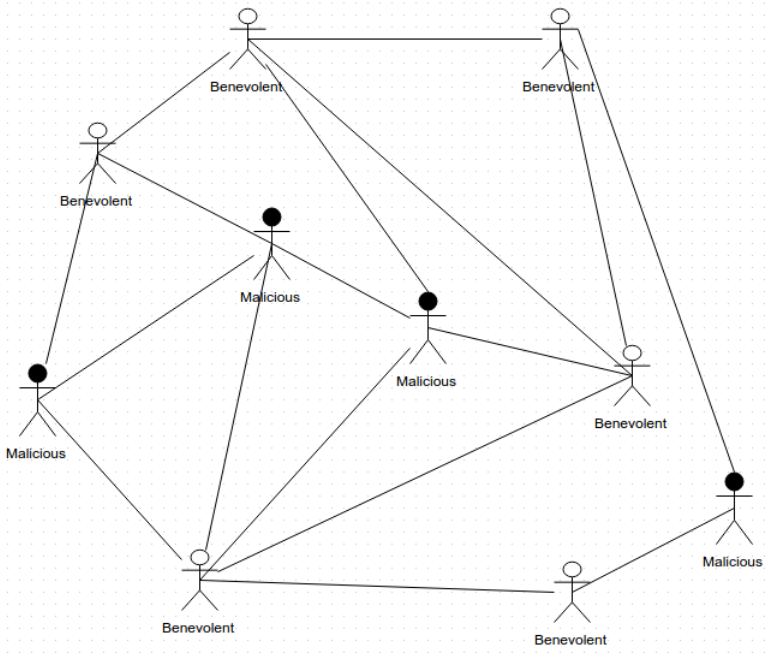
Τύποι επιθέσεων - Αντιμετώπιση με RT-IOT

3.1 Επιθέσεις στο Διαδίκτυο των πραγμάτων

Όπως αναφέρθηκε στο Κεφάλαιο 2 ένας από τους κινδύνους παραβίασης της ασφάλειας του Κοινωνικού Διαδικτύου των Πραγμάτων πηγάζει από επιθέσεις κακόβουλων οντοτήτων οι οποίες όμως νομίμως δρουν εντός της κοινωνίας των πραγμάτων. Σε αυτό το κομμάτι του κεφαλαίου θα αναλύσουμε τις βασικότερες επιθέσεις που μπορούν να συμβούν.

3.1.1 Μεμονωμένες κακόβουλες Εικονικές Οντότητες

Οι μεμονωμένες κακόβουλες Εικονικές Οντότητες Σχ.3.1 είναι ο απλούστερος και ο βασικότερος τύπος επίθεσης. Σε αυτήν την περίπτωση ένα ποσοστό των μελών της κοινότητας έχοντας κακόβουλους σκοπούς προσπαθούν να εκμεταλλευτούν την εμπιστοσύνη των υπόλοιπων και όταν τους ζητηθεί κάποια υπηρεσία ή γνώση αντί να τους εξυπηρετήσουν σωστά στέλνουν δεδομένα που θα οδηγήσουν στο να εκτεθεί ο παραλήπτης σε κίνδυνο ή να οδηγηθεί σε δυσλειτουργία.



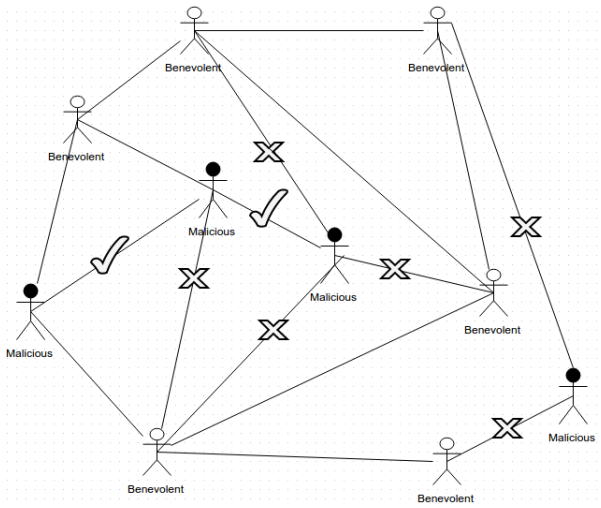
Σχήμα 3.1: Μεμονωμένες κακόβουλες Εικονικές Οντότητες

Η ύπαρξη κακόβουλων Εικονικών Οντοτήτων αποτελεί την βάση όλων των επιθέσεων και ο εντοπισμός τους είναι ο κύριος στόχος του RT-IOT. Για να βρεθούν χρειάζεται η συλλογική συμφωνία του συστήματος στο ποιες ακριβώς είναι.

Αλλά ακόμα και αν δεν τους εντοπίσει το σύστημα ώστε να σταματήσει να τους προτείνει ως πιθανούς φίλους με καλή φήμη, τα μέλη μπορούν αυτόνομα να προστατευθούν. Έτσι μετά από έναν αριθμό αλληλεπιδράσεων μία Εικονική Οντότητα θα χάσει την όποια εμπιστοσύνη τους στον κακόβουλο δράστη και θα τον απορρίψει από τον κοινωνικό του περίγυρο. Το θέμα είναι πόσο γρήγορα θα το κάνει ώστε να περιορίσει του κινδύνους.

3.1.2 Συνεργαζόμενες κακόβουλες Εικονικές Οντότητες

Σε αυτού του είδους της επίθεσης Σχ. 3.2 υπάρχει ένας αριθμός κακόβουλων Εικονικών Οντοτήτων όπως περιγράφηκε στην παράγραφο 3.1.1 οι οποίες όμως τώρα έχουν γνώση (μερική ή ολική) για την ύπαρξή και άλλον κακόβουλων Εικονικών Οντοτήτων. Με αυτή λοιπόν την γνώση συνεργάζονται μεταξύ τους και όταν στέλνουν, είτε στην κεντρική αρχή είτε σε άλλα μέλη της κοινότητας, πληροφορίες για να εξαχθεί ο δείκτης φήμης παραποιούν τα δεδομένα τους και να παρουσιάζουν τους κακόβουλους συνεργάτες τους ως φερέγγυους.



Σχήμα 3.2: Συνεργαζόμενες κακόβουλες Εικονικές Οντότητες

Αυτού του είδους η επίθεση είναι πολύ πιο επικίνδυνη από την προηγούμενη. Αυτό συμβαίνει επειδή δεν είναι δυνατόν να βασιστούν οι Εικονικές Οντότητες στην καλή θέληση των μελών της κοινωνίας για να εξαχθεί η φήμη αφού υπάρχει μεγάλη πιθανότητα να ψεύδονται για τις εμπειρίες τους. Για την αντιμετώπιση του πρέπει:

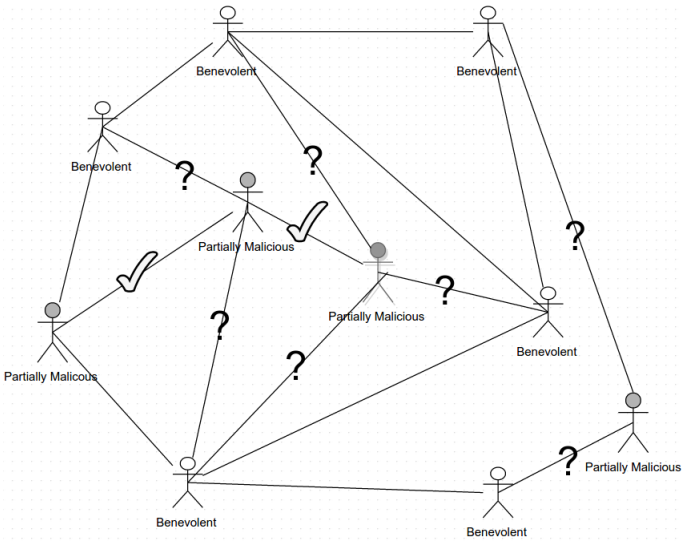
- Στην περίπτωση της κεντρικής αρχής να εντοπιστεί όλη η κοινότητα των κακόβουλων δραστών και να σταματήσει να λαμβάνεται υπόψιν η γνώμη τους κατά την εξαγωγή φήμης.
- Στην περίπτωση συστημάτων χωρίς κεντρική αρχή πρέπει η κάθε Εικονική Οντότητα να εντοπίσει κάποιου έμπιστους φίλους των οποίων η πιθανότητα να ψεύδονται για τις εμπειρίες τους θα είναι μειωμένη σε σχέση με το σύνολο και να εμπιστεύεται αυτούς για να εξάγει φήμη ενώ να αδιαφορεί για την γνώμη τον άλλον ή έστω να μην την λαμβάνει υπόψιν πάντα (Παρ. 3.2.3)

Μια εξελιγμένη μορφή αυτής της επίθεσης είναι η κακόβουλες Εικονικές Οντότητες να προσπαθούν να εντοπίσουν μέσω τις φήμης ανταγωνιστικές, προς τις υπηρεσίες που εκμεταλλεύονται, Εικονικές Οντότητες και να προσπαθήσουν να χαμηλώσουν την φήμη αυτών των Ε.Ο. με περαιτέρω ψευδής δηλώσεις για την εμπειρία τους με αυτές.

3.1.3 Μερικώς κακόβουλες Εικονικές Οντότητες

Σε αυτού του είδους την επίθεση Σχ. 3.3 υπάρχουν κακόβουλες συνεργασίες όπως αναφέρθηκαν στην παράγραφο 3.1.2. Η διαφορά τώρα όμως είναι πώς αυτές οι οντότητες προσπαθούν να νικήσουν το σύστημα φήμης αλλάζοντας την συμπεριφορά τους. Αυτό μπορεί να συμβεί με 2 τρόπους:

1. Αρχικά μπορούν να παρέχουν κακές υπηρεσίες μόνο p % των περιπτώσεων ώστε να μην καταποντίζεται η φήμη τους. Δηλαδή μόλις δουν ότι υπάρχει μία κίνηση αποκλεισμού τους, να γίνονται καλές ώστε να παραμείνουν ενσωματωμένες. Σε αυτή την περίπτωση πρέπει το σύστημα που υπολογίζει την φήμη να έχει μνήμη ώστε να βλέπει καχύποπτα τέτοιου είδους συμπεριφορές και να εφαρμόζει αποκλεισμό με το που ξαναγίνουν κακόβουλες. Παρόλα αυτά αυτή η επίθεση δεν είναι τόσο επικίνδυνη επειδή η κακή συμπεριφορά της Εικονικής Οντότητας είναι φραγμένη από το p
2. Ένας άλλος τρόπος να ξεγελάσουν το σύστημα είναι να προσθέσουν στο ενεργητικό τους και άλλες υπηρεσίες. Έτσι θα είναι πιο εύκολο να εκμεταλλεύονται μία υπηρεσία για τους κακόβουλους σκοπούς τους αλλά να παραμένουν στο απυρόβλητο από το σύστημα φήμης επειδή παρέχουν άλλες καλά. Για να αντιμετωπιστεί αυτού του είδους η επίθεση πρέπει να υπάρχει διαφοροποίηση των δεικτών φήμης ανά υπηρεσία. Αλλά για να μπορέσει παραμείνει κλιμακώσιμο πρέπει να γίνει ομαδοποίηση σε παρόμοιες υπηρεσίες ώστε να περιοριστεί το εύρος που μπορεί να εκμεταλλευτεί μία κακόβουλη Εικονική Οντότητα χωρίς να κοστίζει πολύ στο σύστημα.

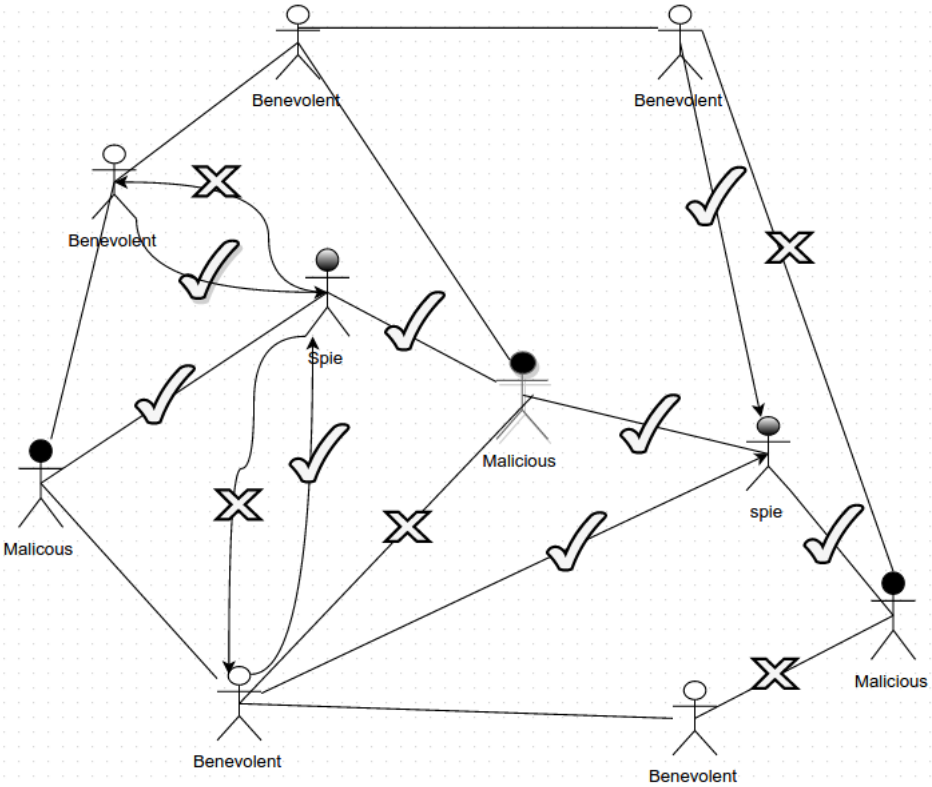


Σχήμα 3.3: Μερικώς κακόβουλες Εικονικές Οντότητες

3.1.4 Κακόβουλοι κατάσκοποι

Αυτού του είδους η επίθεση Σχ. 3.4 είναι βοηθητική στις 3.1.2 και 3.1.3. Οι Εικονικές Οντότητες - κατάσκοποι. Ονομάστηκαν έτσι επειδή στα μάτια του μεγαλύτερου μέρους των μελών του συστήματος παρουσιάζονται ως αξιόπιστες. Αυτό το καταφέρνουν παρέχοντας πάντα μία υπηρεσία καλά. Έτσι αυξάνουν την φήμη τους και την εμπιστοσύνη μεμονωμένων Εικονικών Οντοτήτων προς αυτές. Αυτό όμως που τις κάνει κακόβουλες είναι η συμπεριφορά τους όταν τους ζητηθεί αξιολόγηση ή να προτείνουν κάποιον άλλον πάροχο διαφορετικής υπηρεσίας. Τότε συνεργαζόμενοι με τις κακόβουλες κοινότητες του συστήματος ψεύδονται για τις εμπειρίες τους και προτείνουν κακόβουλες οντότητες για φίλους.

Ένα σύστημα που αντιμετωπίζει όλες τις παραπάνω επιθέσεις δεν μπορεί να αντιμετωπίσει και αυτήν. Για να γίνει αυτό πρέπει να υπάρξει αποσύνδεση της εμπιστοσύνης σε κάποιον ως πάροχο υπηρεσίας, από την εμπιστοσύνη ως πάροχο πληροφοριών. Με αυτόν τον τρόπο θα κρατάμε τους κατασκόπους για να μας παρέχουν σωστές υπηρεσίες αλλά η φήμη της ως παρόχους πληροφοριών θα είναι χαμηλή και άρα δεν θα λαμβάνετε υπόψιν.

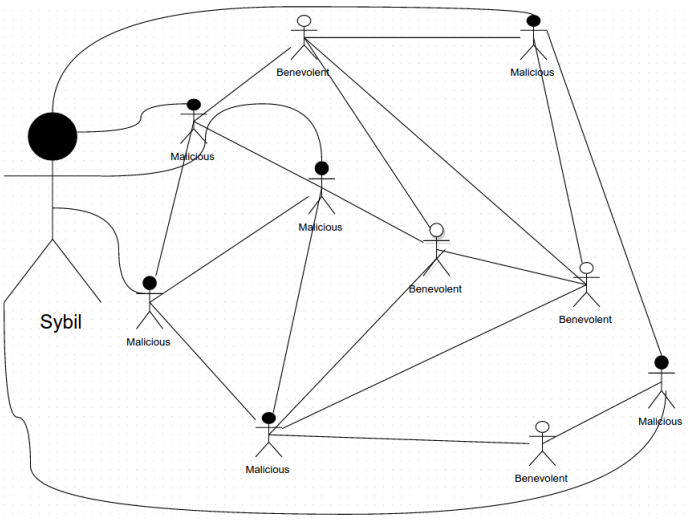


Σχήμα 3.4: Κακόβουλοι κατάσκοποι

3.1.5 Sybil attack

Σε αυτήν την επίθεση Σχ. 3.5 κάποιος καταφέρνει να δημιουργήσει ένα τεράστιο αριθμό εικονικών οντοτήτων και να τις χρησιμοποιήσει για τους κακόβουλους σκοπούς τους. Μόλις μειωθεί η φήμη τους τις διαγράφει και κάνει νέες. Για την αντιμετώπισή του πρέπει:

1. Αρχικά να μην υπάρχει πλεονέκτημα σε μία νέα οντότητα σε σχέση με κάποια με χαμηλή φήμη. Αυτό επιτυγχάνετε με την απουσία αρνητικής βαθμολογίας. Έτσι μία νέα οντότητα αντιμετωπίζεται με καχυποψία και θα πρέπει να αποδείξει την αξία της για να ενσωματωθεί
2. Ακόμα πρέπει να υπάρχει ένα ελάχιστο κόστος για την δημιουργία νέων εικονικών οντοτήτων. Στο σύστημα μας προτείνετε να πρέπει ο χρήστης να συνδέσει την οντότητα του με κάποιο user-id. Το οποίο να μην γίνετε αυτόματα (π.χ χρήση Captcha)



Σχήμα 3.5: Sybil attack

3.1.6 Ψευδής δήλωση στοιχείων

Στον Ατζορι έχει προταθεί ο πολλαπλασιασμός τις εμπιστοσύνης σε κάποιον με ένα βάρος για το πόσο εύκολο είναι για αυτόν να συμπεριφερθεί κακόβουλα. Έτσι αναφέρεται ότι θα εμπιστευτεί μία οντότητα πιο εύκολα εάν έχει χαμηλή υπολογιστική ισχύ π.χ. RFID από μία ισχυρή όπως ένα smartphone. Αλλά είναι δυνατόν κάποιος είτε να ισχυριστεί πώς είναι οντότητα διαφορετικού τύπου από την πραγματικότητα [1], είτε να έχει τον έλεγχο πραγματικών οντοτήτων τύπου RFID που θα είναι εύκολο να γίνουν έμπιστες και θα τις εκμεταλλεύεται.

Για αυτό εμείς αντιπροτείνουμαι το βάρος να είναι ανάλογα με το πόσο επικίνδυνο θα ήταν για μία Εικονική Οντότητα να παραπλανηθεί για μία συγκεκριμένη υπηρεσία. Έτσι υπηρεσίες που θα μπορούσε να δώσει ενα RFID που πιθανώς να ήταν ήσσονος σημασίας θα ζητούνται ακόμα και από Οντότητες μέτριας εμπιστοσύνης, ενώ κρίσιμες υπηρεσίας θα ζητούνται από έμπιστους παρόχους. Το πώς μοντελοποιήθηκε αυτό το βάρος φαίνεται στο ΡΕΦ παρακάτω.

[1] το οποίο ίσως να μπορούσε να ελεγχθεί μέσα από προσπάθεια εντοπισμού των ενεργειών της οντότητας, το οποίο είναι δύσκολο χωρίς κεντρική αρχή που βλέπει όλες τις συναλλαγές

3.2 RT-IOT

3.2.1 Εισαγωγή

Το RT-IOT (Reputation & Trust for the Internet Of Things) είναι ένα σύστημα που προτείνουμε για την αντιμετωπίσει των παραπάνω επιθέσεων στο περιβάλλον του Κοινωνικού Διαδικτύου των Πραγμάτων. Ο βασικός στόχος του ήταν να μπορεί να παρέχει μία αποτελεσματική άμυνα προς τις εσωτερικές επιθέσεις, παρέχοντας ταυτόχρονα την δυνατότητα εισόδου και εξόδου Εικονικών Οντοτήτων στο σύστημα και λαμβάνοντας υπόψιν των μεγάλο αριθμό Εικονικών Οντοτήτων που θα πρέπει να διαχειριστεί.

Για να δημιουργηθεί ένα τέτοιο σύστημα η βασική σχεδιαστική απόφαση έγκειται στο γεγονός ότι δεν είναι δυνατόν αυτό να γίνει ούτε με μία κεντρική αρχή ούτε κατανεμημένα, εάν η φήμη μίας Εικονικής Οντότητας ήταν ίδια για όλο το σύστημα, όπως συμβαίνει στα περισσότερα συστήματα εμπιστοσύνης/φήμης. Έπρεπε να εγκαταλειφθεί η ιδέα της ”αντικειμενικής αλήθειας” για την φήμη του άλλου. Για αυτό κοιτάξαμε την κοινωνία όπου όχι μόνο δεν γνωρίζουμε την φήμη όλων των μελών της αλλά αγνοούμε ακόμα και την ύπαρξη τους.

Με αυτόν τον περιορισμό ως θεμέλιο λίθο χτίσαμε ένα σύστημα με ανοχή σε σφάλματα όπου συνδυάζει ιδέες τόσο από συστήματα με κεντρική αρχή όσο και από συστήματα με κατανεμημένη λογική.

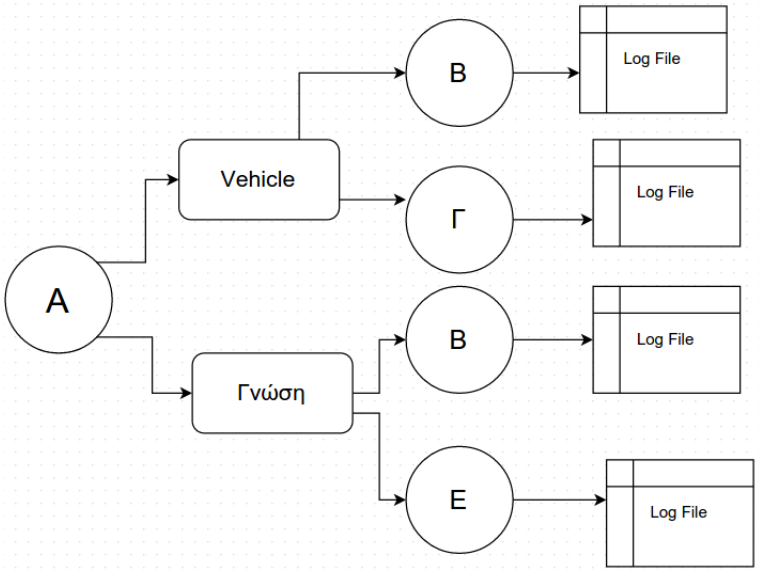
3.2.2 Υπολογισμός Εμπιστοσύνης

Όπως φαίνεται και από στον ορισμό της στην Παρ. 2.1.2, η εμπιστοσύνη είναι υποκειμενική. Για αυτό και είναι λογικό να ορίζεται και να διαφέρει για κάθε Εικονική Οντότητα.

Για να μοντελοποιήσουμε την εμπιστοσύνη χρειάζεται να μοντελοποιηθεί η εμπειρία. Άρα χρειαζόμαστε μνήμη. Για αυτό κάθε Εικονική Οντότητα έχει log files όπου αποθηκεύει της εμπειρίες των αλληλεπιδράσεών της με άλλες Ε.Ο. Εδικότερα όπως αναφέραμε στην παράγραφο 3.1.3 πρέπει να έχουμε διαφορετικά log file για διαφορετικά είδη υπηρεσιών (π.χ. άλλο Log File για υπηρεσίες στο domain ”home automation” ,αλλο στο domain ”vehicles” και άλλα για ανταλλαγή γνώσης). Επίσης για γρηγορότερους και ευκολότερους υπολογισμούς δέν έχουμε ένα log file ανα υπηρεσία, αλλά ένα ανά οντότητα εντός της υπηρεσία.

Επιπρόσθετα, ειδική μέριμνα πρέπει να υπάρξει για την αποσύνδεση της εμπιστοσύνης σε μία Εικονική Οντότητα για την παροχή υπηρεσιών από την παροχή προτάσεων για άλλους πιθανούς φίλους (recommendation). Αυτό έχει ως βασικό αίτιο το γεγονός ότι η παροχή συστάσεων είναι μία υποκειμενική πράξη που βασίζετε σε προσωπικά κριτήρια και δεν πρέπει να μπλέκεται με αντικειμενικές αλήθειες όπως το πόσο καλά παρέχει κάποιος μία υπηρεσία. Ως επακόλουθο αυτής τις πρακτικής θα υπάρχει και απόλυτη προστασία από την επίθεση της παραγράφου 3.1.4.

Άρα εάν μία Εικονική Οντότητα ζητάει την υπηρεσία ”vehicle”, όπου έχει γνωστούς για αυτή τον Β και τον Γ, και επίσης ζητάει και γνώση , όπου έχει γνωστούς για αυτή τον Β και τον Ε. Τα log file είναι όπως στο σχήμα 3.6



Σχήμα 3.6: Ιεραρχία Log File

Σε κάθε εγγραφή του log file βάζουμε τις εξής στήλες:

- **Ικανοποίηση:** Είναι η βαθμολογία που εξάγει μία εικονική οντότητα για την ποιότητα της υπηρεσίας/γνώσης που έλαβε
- **Βάρος:** Είναι μία τιμή που καταγράφει η εικονική οντότητα το πόσο κρίσιμη για αυτήν είναι η υπηρεσία που ζήτησε. Αυτή η τιμή χρησιμοποιείτε και για να ορίσει ένα κάτω όριο εμπιστοσύνης κατά την φάση αναζήτησης παρόχου. Επίσης το βάρος μπορεί να επηρεαστεί και από άλλους παράγοντες όπως εάν ανήκουν οι δύο οντότητες στον ίδιο χρήστη(co-ownership),εάν οι χρήστες γνωρίζονται από πρίν(friends), εάν βρίσκονται γεωγραφικά κοντά (co-location), εάν είναι ίδιου τύπου οντότητες (homophily) κ.α
- **Εξασθένηση:** Αναγνωρίζοντας την ανάγκη οι νεότερες συναλλαγές να έχουν μεγαλύτερο βάρος από τις παλαιότερες εισάγουμε την έννοια του παράγοντα εξασθένησης (fading effect). Με αυτόν τον τρόπο αναγκάζουμε της Εικονικές οντότητες να έχουν συνεπή συμπεριφορά και να μην μπορούν να παρεκτραπούν εκμεταλλευόμενη την καλή τους ιστορία. Επίσης με τον δείκτη εξασθένισης καταφέρνουμε να κρατήσουμε το σύστημα κλιμακόσημο αφού κάθε Εικονική Οντότητα κρατάει μικρό log file.Τέλος παρέχουμε την δυνατότητα εξιλεώσεις σε κακόβουλες οντότητες αφού μετά από κάποιο χρονικό διάστημα η συμπεριφορά τους τότε θα ξεχαστεί και θα κυριαρχεί η νέα καλή τους φύση.

Ικανοποίηση	Βάρος	Εξασθένηση
0.1	0.8	1.0
1.0	0.1	0.95
0.3	0.8	0.9
0.75	0.25	0.85

Πίνακας 3.1: Log File Εικονικής Οντότητας A για της αλληλεπιδράσεις με την B

Όταν η Εικονική Οντότητα Α θέλει να υπολογίσει την εμπιστοσύνη της στην εικονική οντότητα Β για μία συγκεκριμένη υπηρεσία βρίσκει το κατάλληλο log file και μετά υπολογίζει την μέση τιμή της ικανοποίησης λαμβάνοντας υπόψιν το βάρος και τον παράγοντα εξασθένησης. Άρα

$$\mu_t^k \ = \ \frac{\sum_{i=1}^{N} (s_i \ * \ w_i \ * \ f_i)}{W} \tag{3.1}$$

όπου

$$W = \sum_{i=1}^{N} (w_i \ * \ f_i) \tag{3.2}$$

Το W είναι ο συντελεστής κανονικοποίησής και εξασφαλίζει ότι η μέση τιμή θα παραμείνει εντός του εύρους [0,1]. Η μέση τιμή (μ) είναι μία μέτρηση της ολικής συμπεριφοράς της Εικονικής Οντότητας και μας δείχνει πια είναι η πιθανότερη τιμή ικανοποίησης που θα πάρουμε ένα ζητήσουμε την υπηρεσία από τον συγκεκριμένο πάροχο.

Παρόλα αυτά θέλουμε να ξέρουμε και πόσο σίγουροι μπορούμε να είμαστε για την τιμή του μ. Δηλαδή πόσο μπορεί να αποκλίνει η ικανοποίηση της υπηρεσίας από το μ. Για αυτό υπολογίζουμε και την τυπική απόκλιση της συμπεριφοράς. Για να κάνουμε αρκετά λιγότερες πράξεις το κάνουμε με τον ακόλουθο τύπο ταυτόχρονα με τον υπολογισμός της μέσης τιμής.

$$\sigma_t^k \ = \ \frac{1}{W} \sqrt{\sum_{i=1}^{N} (s_i^2 \ * \ w_i \ * \ f_i) * \sum_{i=1}^{N} (w_i \ * \ f_i) - \left(\sum_{i=1}^{N} s_i \ * \ w_i \ * \ f_i \right)^2} \tag{3.3}$$

Τέλος ορίζουμε την εμπιστοσύνη ως:

$$T^k \ = \ \mu_t^k \ - \sigma_t^k \tag{3.4}$$

Δηλαδή ,υποθέτοντας πώς η συμπεριφορά της ΕΟ ακολουθεί την κανονική κατανομή [2], μπορούμε να πούμε ότι εάν ζητήσουμε την υπηρεσία από τον συγκεκριμένο πάροχο έχουμε κάτω από 15% πιθανότητα να πάρουμε ικανοποίηση χαμηλότερη από Τ και άρα το ρίσκο είναι με το μέρος της Εικονικής Οντότητας.

Σύμβολο	Περιγραφή
s_i	Ικανοποίηση στην συγκεκριμένη συναλλαγή
w_i	Βάρος στην συγκεκριμένη συναλλαγή
f_i	Εξασθένηση της συγκεκριμένη συναλλαγή
μ_t^k	Μέση τιμή της εμπιστοσύνης στον k
σ_t^k	Τυπική απόκλιση της εμπιστοσύνης στον k
T^k	Εμπιστοσύνη στον k

Πίνακας 3.2: Πίνακας συμβόλων για υπολογισμό εμπιστοσύνης

[2] μπορεί μέσα από στατιστική ανάλυση να βρεθεί η πραγματική κατανομή και να αλλαχθούν τα ποσοστά

3.2.3 Υπολογισμός Φήμης

Στο RT-IOT η φήμη όπως αναφέραμε δεν υπολογίζεται ολικά για κάθε Εικονική Οντότητα. Δηλαδή εάν δύο εικονικές οντότητες έχουν η κάθε μια από ένα δείκτη φήμης για μία τρίτη Εικονική Οντότητα, τότε αυτές οι δύο τιμές δεν θα συμφωνούνε. Άρα και ο δείκτης φήμης είναι ατομική πληροφορία πού η κάθε Ε.Ο. διαθέτει. Παρόλα αυτά για να εξασφαλίσουμε ότι κάθε Εικονική Οντότητα θα μπορεί να εξάγει τον δείκτη φήμης μίας άλλης παρουσιάζουμε έναν μηχανισμό που έχει ανοχή σε σφάλμα/αποτυχία. Αυτό συμβαίνει επειδή η φήμη μπορεί να εξαχθεί με δύο ανεξάρτητους τρόπους. Δηλαδή:

1. **Κατανεμημένος τρόπος:** Όταν μία Ε.Ο. θέλει να βρει την φήμη μίας άλλης ακολουθεί την λογική των κατανεμημένων συστημάτων 2.2.2. Άρα οπως θα δούμε παρακάτω βασίζεται στον κοινωνικό της περίγυρο.
2. **Συγκεντρωτικός τρόπος:** Εάν αποτύχει ο πρώτος τρόπος η Ε.Ο. καταφεύγει σε μία κεντρική αρχή 2.2.1, η οποίο γνωρίζει όλες της οντότητες του συστήματος αλλά έχει ένα πολύ μικρό δείγμα (partial view) των αλληλεπιδράσεων ώστε να μπορεί να κλιμακώσει το όλο σύστημα. Παρόλα αυτά έχει μία καλή ιδέα για το πόσο καλή είναι γενικά μία Ε.Ο.

Κατανεμημένος τρόπος Υπολογισμού Φήμης

Όταν κάποιος έχει προταθεί έναν νέο φίλο πριν τον εμπιστευθεί προσπαθεί να υπολογίσει τον δείκτη φήμης του. Αυτό συμβαίνει σε δύο βήματα.

Βήμα 1ο: Εικόνα 3.7
Η Εικονική Οντότητα(π.χ. η Α) βρίσκει τους 5 καλύτερους recommenders , δηλαδή αυτούς που εμπιστεύεται περισσότερο για να συστήσουν κάποιον και τους ρωτάει την εμπειρία τους με την Εικονική Οντότητα(Β) που θέλει να ακολουθήσει[3], οι recommenders με την σειρά τους γυρίζουν την εμπιστοσύνη τους σε αυτόν[4]. Ο Α συλλέγει αυτές τις πληροφορίες και υπολογίζει μία *αρχική κατανομή φήμης* χρησιμοποιώντας πάλι τις εξισώσεις όπως οι 3.1,3.2,3.3 όπου το βάρος είναι η εμπιστοσύνη του Α στον εκάστοτε recommender για την προσφορά συστάσεων. Έτσι έχουμε πάλι:

$$\mu_r^k = \frac{\sum_{i=1}^{N} \left(T_i^k * T_{rec}^i\right)}{W'} \tag{3.5}$$

και

$$\sigma_r^k = \frac{1}{W'} \sqrt{\sum_{i=1}^{N} \left(T_i^2 * T_{rec}^i\right) * \sum_{i=1}^{N} \left(T_{rec}^i\right) - \sum_{i=1}^{N} \left(T_i^k * T_{rec}^i\right)^2} \tag{3.6}$$

όπου

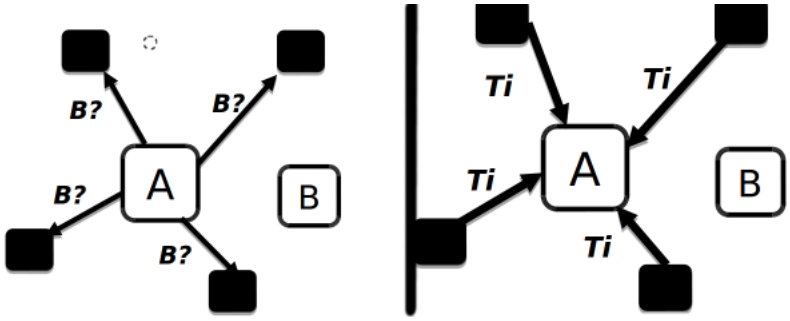
$$W' = \sum_{i=1}^{N} \left(T_{rec}^i\right) \tag{3.7}$$

[3] Στο Cosmos ο φίλος ονομάζεται followee και άρα ακολουθείτε

[4] Ο αναγνώστης μπορεί να προσέξει πώς δέν γυρνάμε το μ ή το σ αλλά το Τ=μ-σ, έτσι δυσκολεύουμε πιθανούς αντιπάλους από το να μας χειραγωγήσουν με τις υπηρεσίας που παρέχουν ώστε να έχουμε συγκεκριμένη εμπιστοσύνη

Και ορίζουμε τον δείκτη φήμης ως:

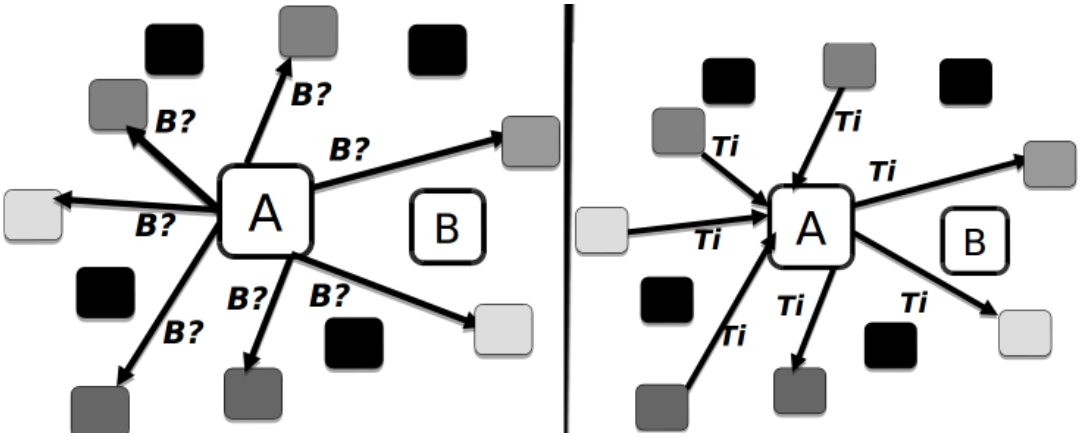
$$R^k = \mu_r^k - \sigma_r^k \tag{3.8}$$



Σχήμα 3.7: Βήμα 1 για κατανεμημένο υπολογισμό φήμης

Βήμα 2ο: Εικόνα 3.8
Έχοντας λοιπόν την αρχική κατανομή της φήμης (μ_r^k, σ_r^k) η Ε.Ο. συνεχίζει να ρωτά φίλους για την εμπειρία τους με τον Β μέχρι να μαζέψει έναν ικανό αριθμό από γνώμες. Αυτές όμως τις γνώμες δεν τις δέχεται άκριτα. Πρώτα προσπαθεί να αποκλείσει κακόβουλες προτάσεις. Στο αρχικό μοντέλο υποθέτουμε πώς και εδώ η συμπεριφορά ακολουθεί κανονική κατανομή οπότε το εύρος που θέλουμε να είναι μέσα η σύσταση ώστε να μην είναι κακόβουλη, είναι: $[\mu_r^k - 0.7\sigma_r^k, \mu_r^k + 0.7\sigma_r^k]$[5]. Αλλά εκ των υστέρων μπορεί να γίνει στατιστική ανάλυση και αυτό να αλλάξει ώστε να γίνει πιο ακριβές.

Μετά συνεχίζουμε με των υπολογισμό της τελικής κατανομής βάζοντας τις νέες καλές συστάσεις στις σχέσεις 3.5 - 3.8 και όταν βγει η τελική κατανομή αξιολογούμε και τους αρχικούς recommenders του βήματος 1.



Σχήμα 3.8: Βήμα 2 για κατανεμημένο υπολογισμό φήμης

[5] το οποίο αντιστοιχεί στο 51% τής κατανομής

Σύμβολο	Περιγραφή
T_i^k	Εμπιστοσύνη του i στον k
T_{rec}^i	Εμπιστοσύνη στον i για συστάσεις (recommendations)
μ_r^k	Μέση τιμή της φήμης του k
σ_r^k	Τυπική απόκλιση της φήμης του k
R^k	Φήμη του k

Πίνακας 3.3: Πίνακας συμβόλων για υπολογισμό φήμης

Συγκεντρωτικός τρόπος Υπολογισμού Φήμης

Για αυτό τον τρόπο υπολογισμού της φήμης δανειζόμαστε στοιχεία από τα συστήματα με κεντρική αρχή υπεύθυνη για τον υπολογισμό της φήμης (βλ. Παρ 2.2.1). Στο σύστημά μας υπάρχει μια οντότητα η οποία στο CosmoS ονομάζεται πλατφόρμα και ένας από τους λόγους ύπαρξης της είναι να παρέχει δείκτες φήμης σε Ε.Ο. όταν ο κατανεμημένος τρόπος αποτυγχάνει(φίλους η Ε.Ο που να έχουν εμπειρία με την νέα Ε.Ο. υποψήφια για φίλο).

Πώς το κάνει όμως αυτό;

Η πλατφόρμα γνωρίζει όλες τις Εικονικές Οντότητες, για να παραμείνει όμως ικανή να διαχειριστεί των τεράστιο αριθμό των πραγμάτων δεν έχει ολική εικόνα των συναλλαγών μεταξύ Εικονικών Οντοτήτων. Αντίθετα για να εξάγει έναν δείκτη φήμης δέχεται σε τυχαίες χρονικές στιγμές feedback από τις Εικονικές Οντότητες σχετικά με τις εμπειρίες τους. Αυτό το feedback είναι απλά ο δείκτης εμπιστοσύνης μίας εικονικής οντότητας σε μία άλλη μαζί με τον αριθμό των αλληλεπιδράσεών τους. Έτσι χρησιμοποιώντας ως βάρος τον αριθμό των αλληλεπιδράσεων χρησιμοποιεί ίδιες σχέσης με αυτές για την εξαγωγή εμπιστοσύνης παραπάνω και βγάζει έναν δείκτη φήμης. Αυτό ο δείκτης δεν είναι απόλυτα σωστός αλλά μπορεί να δείξει την τάση της συμπεριφοράς μία Εικονικής Οντότητας.

Καλό είναι να σημειώσουμε πώς και αυτός ο μηχανισμός μπορεί να αποτύχει στον βαθμό ακρίβειας του. Έτσι η κοινότητα σε περίπτωση sybil attack 3.1.5 μεγάλου μεγέθους μαζί με συνεργασία των κακόβουλων οντοτήτων 3.1.2 θα δίνει σκόπιμα κατασκευασμένο feedback στην πλατφόρμα, η οποία θα θεωρεί πώς οι κακοί είναι καλοί και αντίστροφα. Παρακάτω θα δούμε πώς αυτό αντιμετωπίζεται στην παράγραφο 3.3.2.

3.3 Βασικά Σενάρια Χρήσης

3.3.1 Αίτηση παροχής Υπηρεσίας (Service Request)

Ο λόγος που δημιουργήθηκε το RT-IOT είναι για να βρίσκει μία Εικονική Οντότητα τον καλύτερο δυνατό πάροχο με βάση την δικιά της εικόνα για την κοινωνία και τις δικές της ανάγκες. Αυτό το καταφέρνει αποδοτικά με την χρήση των δεικτών εμπιστοσύνης και φήμης που αναλύθηκαν στην προηγούμενη ενότητα.

Όταν λοιπόν μία Εικονική Οντότητα αποφασίζει πώς χρειάζεται μία συγκεκριμένη υπηρεσία ξέρει και πόσο πρόθυμη είναι να ρισκάρει.[6] Αυτή την τιμή την χρησιμοποιεί ως κατώφλι(threshold) για την τιμή των δεικτών που υπολογίζει. Δηλαδή εάν δεν είναι πρόθυμη να ρισκάρει πολύ μπορεί να βάλει το κατώφλι 0.8 και άρα να ζητήσει την υπηρεσία μόνο από Εικονικές Οντότητες-φίλους που εμπιστεύεται τουλάχιστον κατά 0.8/1.0. Φυσικά η εύρεση αυτών μπορεί να είναι δυσκολότερη οπότε να χρειαστεί παραπάνω πόρους για να το επιτύχει και αυτό είναι ένα trade-off που πρέπει να ληφθεί υπόψιν κατά τον ορισμό του κατωφλίου.

Όταν λοιπόν οριστεί το κατώφλι αρχίζει η αναζήτηση κατάλληλου παρόχου. Αυτό μπορεί να γίνει με 2 διαφορετικούς τρόπους ανεξάρτητους μεταξύ τους

1) Αίτηση σε Φίλο

Ο πρώτος τρόπος που δοκιμάζει μία Εικονική Οντότητα να βρει έναν κατάλληλο πάροχο είναι να βρει κάποιον φίλο της που εμπιστεύεται αρκετά. Για να γίνει αυτό πάει στην κατάλληλη υπηρεσία (βλ. σχήμα 3.6) και αποτιμά την εμπιστοσύνη της σε κάθε ένα από τους φίλους της.

Εφόσον με κάποια εικονική οντότητα υπάρχει αρκετή προϊστορία συναλλαγών (log file > 5) τότε ο υπολογισμός της εμπιστοσύνης γίνεται δύο φορές ταυτόχρονα. Χρησιμοποιώντας της σχέσης 3.1 - 3.4 υπολογίζεται το T_{long} γιά N ίσο με το μέγεθος του log file και το T_{short} για N ίσο με $\frac{size(log\_file)}{10}$. Εδώ καλό είναι να αναφέρουμε πώς το log file είναι μορφής LIFO οπότε η πρώτη εγγραφή είναι και η κοντινότερη χρονικά. Στην συνέχεια θεωρούμε πώς:

$$T = min(T_{long}\ ,\ T_{short})$$

Με αυτόν τον τρόπο η τιμή του T_{long} δείχνει την ολική εμπειρία μας με κάποια Ε.Ο. για αυτήν την υπηρεσία ενώ το T_{short} δείχνει την συμπεριφορά της στις τελευταίες χρονικές στιγμές. Οπότε μπορούμε να τόσο να ανιχνεύσουμε την μεταβολή μίας καλόβουλης Ε.Ο. σε κακόβουλη γρήγορα ,αφού το T_{short} θα μειωθεί ραγδαία, όσο και να μην ξεγελαστούμε από την ξαφνική καλοσύνη μίας κακόβουλης Ε.Ο. ,αφού το T_{long} έχει αρκετή "αδράνεια" ώστε να το εμποδίσει. Έτσι αντιμετωπίζονται μερικώς κακόβουλες οντότητες 1ου τύπου (3.1.3).

[6] δηλαδή ο developer της έχει αποφασίσει ή σε πιο εξελιγμένο επίπεδο το κάνει μόνη της υπολογίζοντας πόσο θα της κοστίσει αν κάτι πάει στραβά

Αφού γίνει ο υπολογισμός των δεικτών εμπιστοσύνης της Ε.Ο. για όλους τους φίλους της στην συγκεκριμένη υπηρεσία, επιλέγει τον πιο έμπιστο φίλο και συγκρίνει τον δείκτη εμπιστοσύνης του με το κατώφλι. Εάν το ξεπερνάει τότε έχει βρεθεί μία αρκετά καλή λύση για το πρόβλημα της με ένα ικανοποιητικά χαμηλό ρίσκο, οπότε ζητάει την υπηρεσία. Εδώ καλό είναι να σημειώσουμε πώς εάν πληρούνται κάποιες προϋποθέσεις ασφαλείας [7] τότε υπάρχει μία πιθανότητας 10% να ζητηθεί η υπηρεσία από ένα φίλο που δεν έχω πολλές αλληλεπιδράσεις μαζί του (π.χ. μου τον έδωσε η πλατφόρμα χωρίς να υπάρχει δείκτης φήμης, άρα είναι κάποια Εικονική Οντότητα που μόλις εισηχθεί στο σύστημα). Αυτό συμβαίνει για να μπορούν να ενσωματωθούν δυναμικά στο σύστημα νέοι κόμβοι και να υπάρχει καταμερισμός εργασιών.

Σε περίπτωση που αποτύχει ο πρώτος τρόπος και δέν βρεθεί κάποιος η Ε.Ο. προχωράει στον 2ο τρόπο.

Algorithm 1 Service Request Method: Ask_Friend

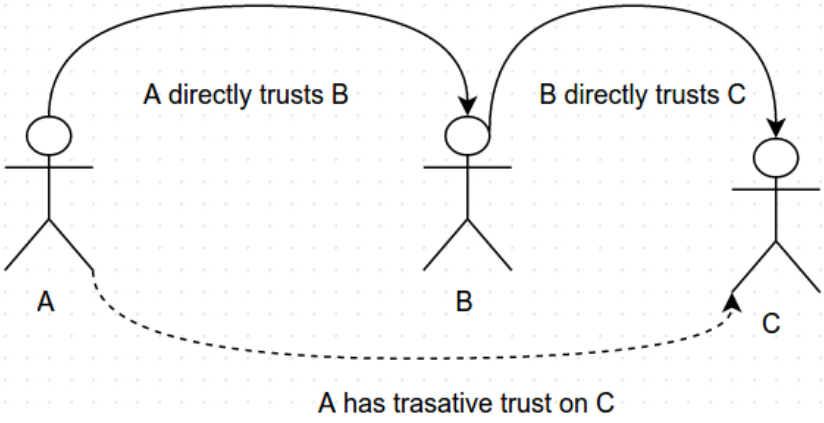
```
1:  procedure Ask–Friend
2:      Threshold = t
3:      Best = 0
4:      Trust = 0
5:      for each friend i ∈ Service do
6:          log_file=get_log_file(Service,i)
7:          if size(log_file) > 5 then
8:              T_short =caculate_Trust(i,size(log_file)/10)
9:          else
10:             T_short = 1.0
11:         end if
12:         T_long =caculate_Trust(i,size(log_file))
13:         T = min(T_long , T_short)
14:         if T > Trust then
15:             Best=i
16:             Trust=T
17:         end if
18:     end for
19:     if Trust ≥ threshold then
20:         RequestService(Service,Best)
21:     else
22:         if (threshold indicates low severity) AND (Math.random() ≤ 0.1) then
23:             RequestService(Service,Random friendwith zero trust)
24:         end if
25:     end if
26: end procedure
```

[7] η υπηρεσία δεν είναι κρίσιμης σημασίας και δέν έχω κάποιον πολύ καλό πάροχο

2) Αίτηση σε φίλο φίλου

Αν δεν καταφέρει η εικονική οντότητα να βρει κάποιον έμπιστο γνωστό της πάροχο τότε θα στραφεί στους φίλους της. Ένας τρόπος να το κάνει αυτό είναι να υπολογίσει με τον ίδιο τρόπο τους δείκτες εμπιστοσύνης σε φίλους, όχι όμως για την συγκεκριμένη υπηρεσία αλλά για παροχή βοήθειας (assistance).

Εάν λοιπόν υπάρχουν μερικοί αρκετά έμπιστοι φίλοι θα τους στείλει μία αίτηση για βοήθεια (request for assistance). Όταν αυτοί λάβουν την αίτηση θα ψάξουν τους δικούς τους φίλους και αν βρουν κάποιον ικανό και έμπιστο να βοηθήσει τότε θα αιτηθούν την υπηρεσία και θα προωθήσουν το αποτέλεσμα στον αρχικό αιτούντα. Αυτή η δυνατότητα είναι αποτέλεσμα του *transative trust*. Δηλαδή του γεγονότος ότι εάν εμπιστεύομαι κάποιον και αυτός με την σειρά του εμπιστεύεται έναν τρίτο τότε εμπιστεύομαι και εγώ τον τρίτο. Μετά την λήψη της υπηρεσίας η Ε.Ο. αποθηκεύει την ικανοποίηση της ως ικανοποίηση στον άμεσο φίλο της αφού δεν έχει ως φίλο τον τελικό πάροχο.



Σχήμα 3.9: Transative Trust

Στο RT-IOT αυτό γενικεύεται και η αίτηση για βοήθεια έχει και ένα Time To Live (TTL) που δείχνει πόσα hops μακριά από την εικονική οντότητα μπορεί να είναι ο πάροχος. Όσο πιο μακρυά πάει τόσο πιο πιθανό είναι να βρεθεί κάποιος πάροχος, αλλά ανάλογα αυξάνει και το ρίσκο να υπάρχει κακόβουλη οντότητα στο path. Θεωρούμε πώς στο δίκτυο ισχύει η παραδοχή του six degrees of separation[8] οπότε το μέγιστο TTL είναι 6.

[8] en.wikipedia.org/wiki/Six_degrees_of_separation

Algorithm 2 Service Request Method: Ask_for_assistance

1: **procedure** Ask–Assistance
2: $Threshold = t$
3: $Best = 0$
4: $Trust = 0$
5: **for** each friend $i \in Recommendation$ **do**
6: log_file=get_log_file (Recommendation,i)
7: **if** size(log_file) > 5 **then**
8: T_{short} =caculate_trust_for_assists(i,$\frac{size(log\_file)}{10}$)
9: **else**
10: $T_{short} = 1.0$
11: **end if**
12: T_{long} =caculate_trust_for_assists(i,$size\ (logfile)$)
13: $T = min(T_{long}\ ,\ T_{short})$
14: **if** $T\ >\ Trust$ **then**
15: Best=i
16: Trust=T
17: **end if**
18: **end for**
19: **if** $Trust \geq threshold$ **then**
20: RequestAssist(Service,Best)
21: **end if**
22: **end procedure**

3.3.2 Μέθοδος Απόκτησης Νέων Φίλων(Friend Acquisition Method)

Σε αυτή την μέθοδο , σε αντίθεση με την προηγούμενη, βασικό ρόλο παίζει ο δείκτης φήμης. Για αυτό και ή απόκτηση νέων φίλων γίνεται με 2 τρόπους. Την απόκτηση φίλων κατανεμημένα από προτάσεις φίλων και την απόκτηση φίλων από την πλατφόρμα (κεντρική αρχή). Αυτή η μέθοδος καλείται κάθε φορά που αποτυγχάνει η προηγούμενη 3.3.1 να βρει ικανό πάροχο υπηρεσίας και μπορεί να κληθεί και κατευθείαν μετά τον βήμα 1 εάν κριθεί καλύτερο.

Απόκτηση φίλων κατανεμημένα

Εδώ μία εικονική οντότητα ενεργεί παρόμοια με τον 2ο τρόπο της προηγούμενη ενότητας. Αρχικά βρίσκει φίλους που εμπιστεύεται αρκετά για παροχή προτάσεων φίλων(recommendations). Τους κατατάσσει σε φθίνουσα σειρά εμπιστοσύνης και τους ρωτάει αν έχουν κάποια γνωστή Ε.Ο. η οποία να είναι έμπιστη για παροχή κάποιας συγκεκριμένη υπηρεσίας. Αυτό το κάνει για έναν φίλο κάθε φορά μέχρι να μαζέψει αρκετές προτάσεις.

Όταν τις μαζέψει (τις προτεινόμενες για φίλους Εικονικές Οντότητες) υπολογίζει τον δείκτη φήμης αυτών, όπως περιγράφηκε στην ενότητα 3.2.3, και κρατάει την πιο φημισμένη μαζί με την Εικονική Οντότητα που την πρότεινε. Στη συνέχεια ζητάει την υπηρεσία που υποτίθεται ότι παρέχει καλά και αν όντως είναι αρκετά καλή την εισάγει ως φίλο στην κατάλληλη δομή από log files. Ό,τι και να γίνει αξιολογεί την αρχική Εικονική Οντότητα που έκανε την πρόταση ανάλογα με το αποτέλεσμα.

Εάν αποτύχει αυτή η μέθοδος επειδή οι φίλοι δεν μπορούσαν να βρουν προτάσεις συνεχίζει στην συγκεντρωτική μέθοδο.

Απόκτηση φίλων από την Πλατφόρμα

Εδώ τα πράγματα είναι πολύ απλά. Όπως αναφέραμε στην ενότητα 3.2.3 η πλατφόρμα έχει γνώση όλων τον εικονικών Οντοτήτων, αλλά έχει μόνο μερική εικόνα των δράσεών τους. Σε περίπτωση που της ζητηθεί πρόταση χρησιμοποιεί το feedback που τις έχει σταλεί για να υπολογίσει τους δείκτες φήμης των εικονικών οντοτήτων και να τις κατατάξει. Στην συνέχεια θα γυρίσει κάποια ως πρόταση στον αιτούντα μαζί με τον δείκτη φήμης του και αυτός θα ζητήσει την υπηρεσία. Όμοια με πριν αν συμπεριφερθεί καλά θα τον κρατήσει ως φίλο.

Η ερώτηση είναι πώς επιλέγεται ο ένας που θα προταθεί;

Όπως αναφέραμε υπάρχει κάποια, αν και αρκετά απίθανη, περίπτωση να ξεγελαστεί η πλατφόρμα. Για αυτό ο τρόπος που διαλέγει να προτείνει κάποιον προσομοιώνει ένα random walk ώστε να μπορούν να αντιμετωπιστούν κακόβουλες συνεργασίες. Έτσι υπάρχει 80% να προταθεί κάποιος από τις 3 πιο φημισμένες Ε.Ο. (ισοπίθανη επιλογή για λογους load balancing) ενώ υπάρχει και μία πιθανότητα 20% να προταθεί κάποιος με μηδενική φήμη.

Αυτό σημαίνει πως στην περίπτωση χειραγώγησης της πλατφόρμας, όπου οι καλοί θα φαίνονται κακοί, υπάρχει 20% πιθανότητα να προταθεί κάποιος καλός και να κρατηθεί αυτός ως φίλος από την Εικονική Οντότητα που ζήτησε φίλους. Επίσης αυτό το 20% βοηθάει στην ενσωμάτωση νέων Εικονικών Οντοτήτων στο σύστημα, μίας και θα έχουν μηδενική φήμη. Σίγουρα έτσι δίνουμε μία μικρή ευκαιρία σε κακόβουλους να εκμεταλλευτούν το σύστημα αλλά θεωρούμε πώς το trade-off είναι θεμιτό.

3.3.3 Μέθοδος Απόρριψης Κακόβουλων φίλων(BlackListing)

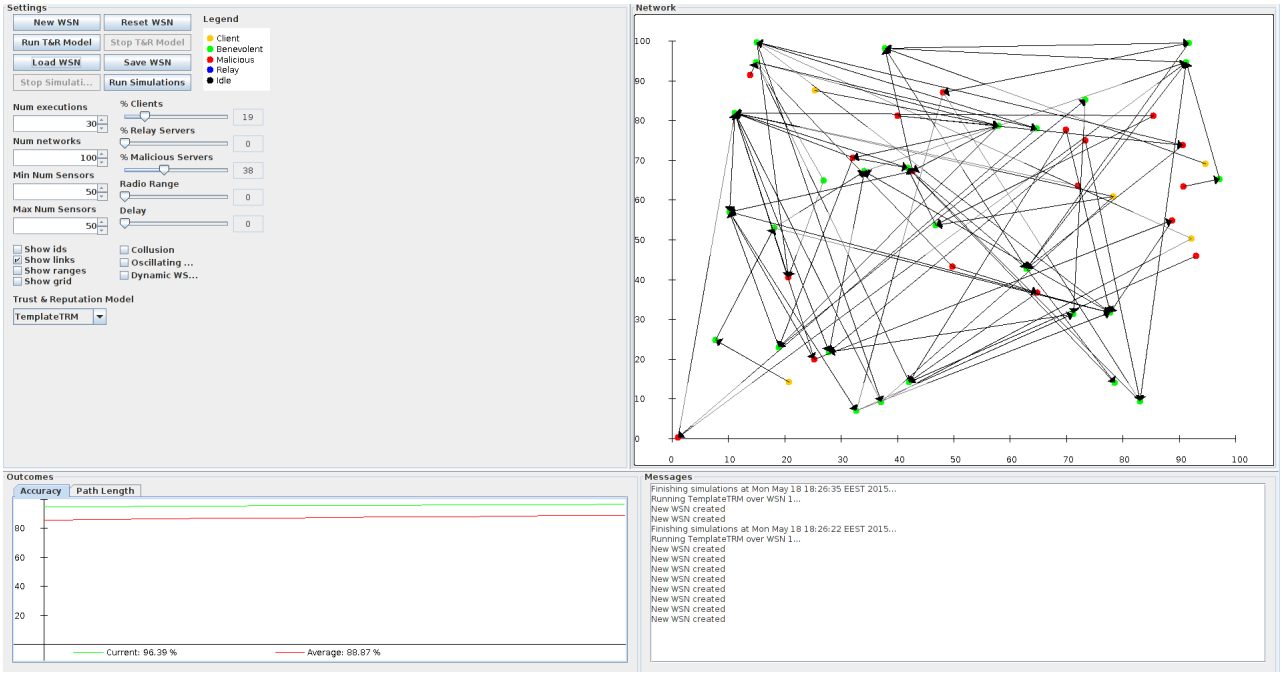
Για να μπορέσει μία εικονική Οντότητα να διατηρήσει ένα καλό επίπεδο ικανοποίησης χρειάζεται έναν τρόπο να διαγράφει φίλους οι οποίοι είναι ή έγιναν κακόβουλοι. Αυτό πρέπει να συμβεί επειδή αλλιώς θα γέμιζε η οντότητα με κακόβουλους φίλους. Αλλά λόγω της εξασθένισης σε συνδυασμό με τον ποσοστό που υπάρχει για επιλογή τυχαίου φίλου η πιθανότητα αυτός να είναι κακόβουλος θα ήταν υψηλή.

Έτσι αποφασίστηκε οι Εικονικές Οντότητες να έχουν μία μέθοδο απόρριψης που τρέχει περιοδικά ελέγχοντας εάν η εμπιστοσύνη σε κάποιον φίλο είναι χαμηλή και τότε τον διαγράφει. Με αυτό τον τρόπο το μεγαλύτερο ποσοστό των φίλων θα είναι καλόβουλοι.

Κεφάλαιο 4

Υλοποίηση προσομοίωσης RT-IOT με χρήση του TRMSim-WSN

4.1 TRMSim-WSN



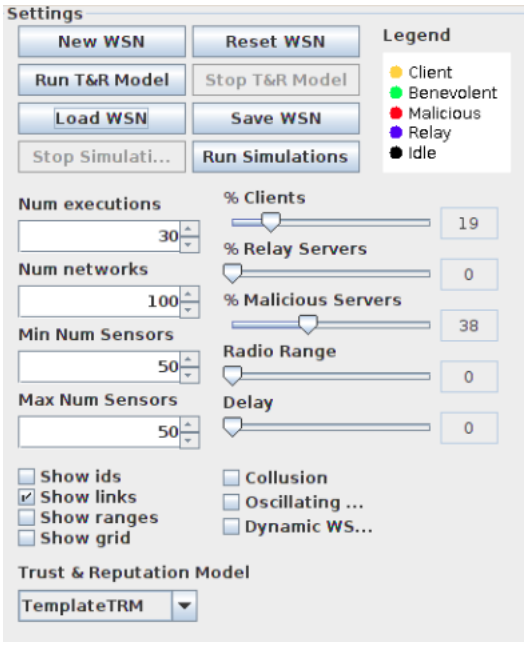
Σχήμα 4.1: TRMSim-WSN περιβάλλον εργασίας

Το TRMSim-WSN είναι ,στο βαθμό που γνωρίζουμε, η state-of-the-art πλατφόρμα προσομοίωσης συστημάτων εμπιστοσύνης-φήμης. Δημιουργήθηκε για να προσομοιώνει αλγορίθμους διαχείρισης φήμης και εμπιστοσύνης σε wireless sensor networks συστήματα, αλλά οι ίδιες αρχές εφαρμόζονται και σε συστήματα Διαδικτύου των Πραγμάτων. Αυτό συμβαίνει επειδή και στο Διαδίκτυο των Πραγμάτων υπάρχουν διεσπαρμένοι κόμβοι στο χώρο με την διαφορά πως οι σχέσεις φιλίας δεν είναι βάση φυσικής απόστασης αλλά τυχαίες.

Η προσομοίωση γίνεται για μία υπηρεσία, άλλα λόγω της αποσύνδεσης των υπηρεσιών μεταξύ τους σε ότι αφορά το σύστημα διαχείρισης φήμης-εμπιστοσύνης, τα αποτελέσματα θα είναι ίδια και για πολλές υπηρεσίες. Ουσιαστικά ένα σύστημα με πολλές υπηρεσίες μπορεί να αναπαρασταθεί ως το άθροισμα πολλών ανεξάρτητων προσομοιώσεων με ίδιους κόμβους και διαφορετική κατανομή συμπεριφορών (malicious,benevolent,client). Παρακάτω παρουσιάζονται τα βασικά χαρακτηριστικά του προσομοιωτή.

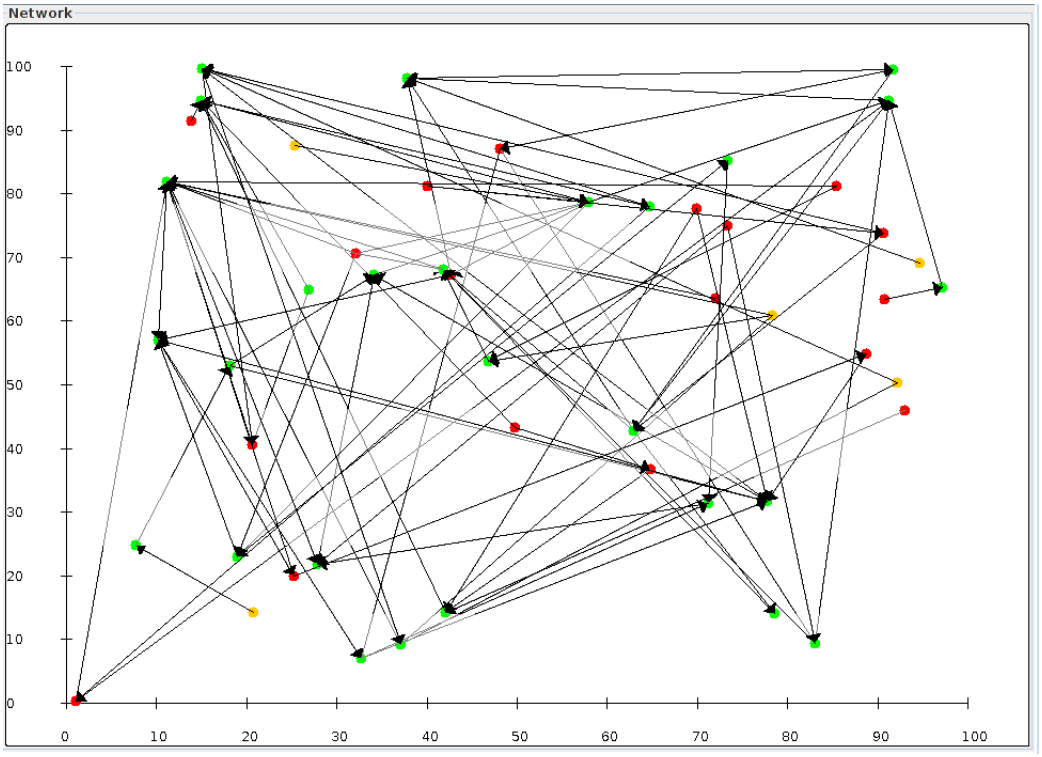
Όπως φαίνεται στην εικόνα 4.2 το σύστημα έχει ένα πάνελ για ρύθμιση των παραμέτρων. Οι παράμετροι είναι:

- **Num executions:** Πόσες φορές κάνει request ένας πελάτης σε ένα συγκεκριμένο δίκτυο.
- **Num networks:** Σε πόσα διαφορετικά τυχαία δίκτυα θα τρέξει η προσομοιώσει ώστε να ελεγχτεί η συμπεριφορά του συστήματος. Σε κάθε δίκτυο γίνονται Num executions αιτήσης για την υπηρεσία από την κάθε Εικονική Οντότητα.
- **Min Num Sensors:** Ο ελάχιστος αριθμός Εικονικών Οντοτήτων ανά δίκτυο προσομοίωσης.
- **Max Num Sensors:** Ο μέγιστος αριθμός Εικονικών Οντοτήτων ανά δίκτυο προσομοίωσης.
- **% Clients:** Το ποσοστό των Εικονικών Οντοτήτων που δεν παρέχουν υπηρεσίες.
- **% Relay Servers:** Το ποσοστό των Εικονικών Οντοτήτων που δέν παρέχει την συγκεκριμένη υπηρεσία, για τον σκοπό των δικών μας προσομοιώσεων συμπεριφέρεται σαν Client και για αυτό το κρατάμε στο 0%.
- **% Malicous Servers:** Το ποσοστό των Εικονικών Οντοτήτων που παρέχουν κακόβουλες υπηρεσίες.
- **Radio Range:** Η απόσταση που φτάνει το σήμα. Δεν έχει νόημα στο IoT και το κρατάμε στο 0.
- **Delay:** Χρόνος καθυστέρησης μεταξύ δυο διαδοχικών αιτήσεων. Το χρησιμοποιούμε για καλύτερη εποπτεία κατά την διάρκεια της εκτέλεσης.
- **Collusion:** Εάν είναι ενεργοποιημένο υπάρχουν κακόβουλες συνεργασίες στο σύστημα
- **Oscilating:** Εάν είναι ενεργοποιημένο ανά τακτά χρονικά διαστήματα κάποιοι servers αλλάζουν συμπεριφορά απο καλόβουλη σε κακόβουλη. Τα ποσοστά(malicous,relay,clients) παραμένουν όμως σταθερά.
- **Dynamic:** Εάν είναι ενεργοποιημένο ανά τακτά χρονικά διαστήματα κάποιες Εικονικές Οντότητες απενεργοποιούνται για 0.5sec προσομοιώνοντας την δυναμική ενεργοποίηση και απενεργοποίηση των πραγμάτων/αισθητήρων



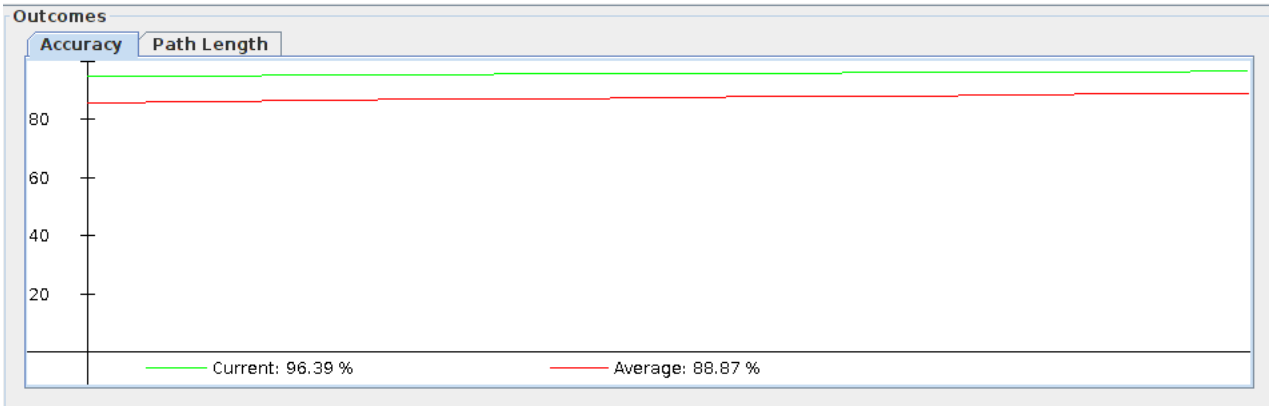
Σχήμα 4.2: Πάνελ Ρυθμίσεων

Όπως φαίνεται στην εικόνα 4.3 το σύστημα έχει ένα πάνελ για την αναπαράσταση των Εικονικών Οντοτήτων και των σχέσεων μεταξύ τους. Επειδή οι σχέσεις είναι μονόδρομες υπάρχει βέλος κατεύθυνσης. Παρακάτω θα δούμε πώς αλλάζουν αυτές οι σχέσεις ώστε οι κακόβουλες οντότητες να μην έχουν σχεδόν κανένα βέλος πάνω τους ενώ οι καλόβουλες να έχουν πολλά.

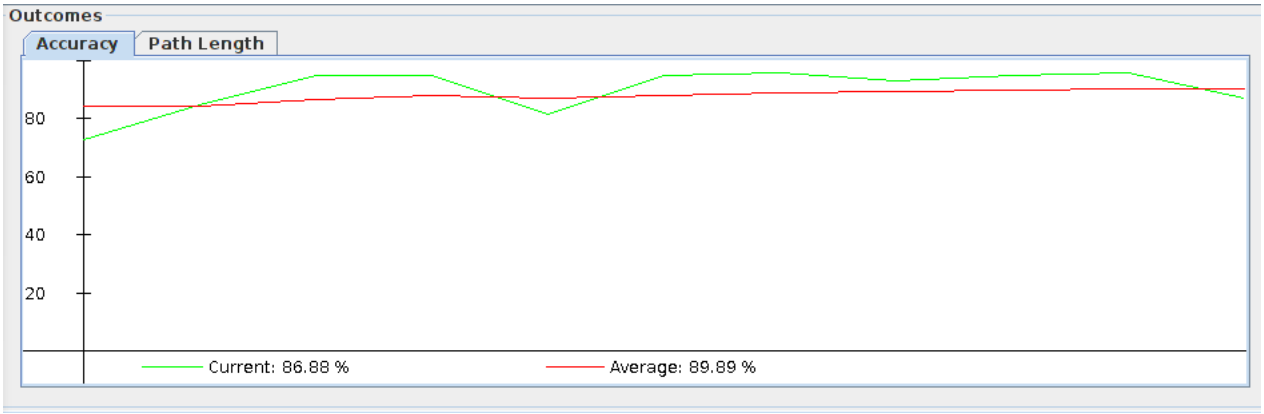


Σχήμα 4.3: Πάνελ Δικτύου

Όπως φαίνεται στις εικόνες 4.4, 4.5 το σύστημα έχει ένα πάνελ για την αναπαράσταση του μέσου όρου της ικανοποίησης των Εικονικών Οντοτήτων. Όταν η προσομοίωση γίνεται για ένα δίκτυο (RUN T& R Model κουμπί) τότε η πράσινη γραμμή δείχνει την ικανοποίηση για κάθε κύκλο αιτήσεων (π.χ εάν Num execution = 30 τότε θα έχει 30 σημεία) και η κόκκινη τον μέσο όρο. Αντίθετα εάν γίνεται προσομοίωση για πολλά δίκτυα (Run Simulations κουμπί) τότε η πράσινη γραμμή δείχνει την ικανοποίηση μετά από την προσομοίωση σε ένα συγκεκριμένο δίκτυο και η κόκκινη πάλι την μέση τιμή. Δηλαδή εάν Num networks =100 , Num execution = 30 η πράσινη γραμμή θα έχει 100 σημεία όπου κάθε σημείο θα είναι το current satisfaction μετά από 30 εκτελέσεις σε ένα τυχαίο δίκτυο. Όπως είνα λογικό σε ένα δίκτυο χωρίς oscillating και dynamic το πρώτο πάνελ θα έχει μία αύξουσα συνάρτηση αφού τα δίκτυα θα προσκολλώνται σε έμπιστους φίλους ενώ το δεύτερο μία τυχαία με απότομες μεταβάσεις αφού τα δίκτυα δημιουργούνται με τυχαία διάταξη και αρχικές συνθήκες.



Σχήμα 4.4: Πάνελ ικανοποίησης σε ένα δίκτυο



Σχήμα 4.5: Πάνελ ικανοποίησης σε πολλά δίκτυα

4.2 Υλοποίηση RT-IOT

Η Υλοποίηση του RT-IOT έγινε ως επέκταση του συστήματος προσομοίωσης και χρησιμοποιεί της δομές που παρέχει έτοιμες το πακέτο TemplateTRM και οι οποίες κάνουν τα σωστά extend ώστε να μπορεί να λειτουργήσει ένα νέο σύστημα Trust & Reputation πάνω στο TRMSim-WSN. Οι βασικές λειτουργικότητες της πλατφόρμας καθώς και η αρχικοποίηση του δικτύου προστέθηκαν στην κλάση TemplateTRM_Network ενώ οι λειτουργικότητες των Εικονικών Οντοτήτων προστέθηκαν στην κλάση TemplateTRM_Sensor. Επίσης για λόγους απομόνωσης,ευκολότερης υλοποίηση και επαναχρησιμοποίησης κώδικα δημιουργήθηκαν βοηθητικές κλάσεις της οποίες και παρουσιάζουμε παρακάτω:

4.2.1 ComparableFriend & FriendComparator

Η κλάση ComparableFriend 4.1 μοντελοποιεί μόνο τα απαραίτητα στοιχεία ενός φίλου για να μπορέσει να συγκριθεί με τους άλλους. Δηλαδή έχει το id που αναφέρεται στην εικονική οντότητα και το value με το οποίο θα καταταγεί, το οποίο ανάλογα την εφαρμογή μπορεί να πάρει τιμή δείκτη εμπιστοσύνης ή δείκτη φήμης.

```
public class ComparableFriend {
  String id;
  double value;

  public ComparableFriend(String id, double value) {
    this.id = id;
    this.value = value;
  }
}
```

Listing 4.1: ComparableFriend

Η κλάση FriendComparator 4.2 είναι απλά ένας custom-made Comparator που χειρίζεται των τρόπο κατάταξης των φιλών σε μία PriorityQueue

```
public class FriendComparator implements Comparator<ComparableFriend>{

  @Override
  public int compare(ComparableFriend o1, ComparableFriend o2) {

    if (o1.value >=  o2.value)
      return 1;
    else
      return -1;
  }
}
```

Listing 4.2: FriendComparator

4.2.2 MyOutcome,SatisfactionInterval,MyTransaction

Η κλάση MyOutcome 4.3 είναι ένα instance της abstract κλάσης Outcome που χρησιμοποιείτε από τον TRMSim-WSN για να αναπαραστήσει το αποτέλεσμα μίας συναλλαγής μεταξύ δύο Οντοτήτων.

```
public class MyOutcome extends BasicOutcome {

    protected SatisfactionDiscreteScale satisfaction;

  public MyOutcome(SatisfactionInterval satisfaction) {
    super(satisfaction,satisfaction.getSatisfactionValue(),1);

  }

}
```

Listing 4.3: MyOutcome

Τα βασικότερα στοιχεία αυτής της κλάσης είναι το Satisfaction που στην περίπτωσή μας έχει οριστεί ως SatisfactionInterval 4.4 , δηλαδή παίρνει τιμές από 0 έως 1, και η μέθοδος aggregate που παίρνει ένα collection από Outcomes και βγάζει την ολική ικανοποίηση το οποίο και προβάλετε στο Πάνελ ικανοποίησης 4.4

```
public class SatisfactionInterval implements Satisfaction {
  /** Lower bound of the interval */
  private double min;
  /** Upper bound of the interval */
  private double max;
  /** Actual satisfaction value */
  private double value;

  public SatisfactionInterval(double min, double max, double value) {
    if (min > max)
      min = max;
    if ((min > value) || (max < value))
      value = min;
    this.min = min;
    this.max = max;
    this.value = value;
  }
```

Listing 4.4: SatisfactionInterval

Τέλος οι κλάση MyTransaction 4.5 περιεχέι τα βασικά στοιχεία μίας συναλλαγής όπως το βάρος,η εξασθένηση και φυσικά το outcome που περιεχέι την ικανοποίηση. Δηλαδή ένα MyTransaction Object αναπαριστά μία εγγραφή στο log_file μίας εικονικής Οντότητας(μία γραμμή στον πίνακα 3.1).

```
public class MyTransaction {
  private MyOutcome outcome;

  protected final double severity;
  protected double fading;
  protected String id=null;

  public MyTransaction(MyOutcome outcome, double severity,String id) {
```

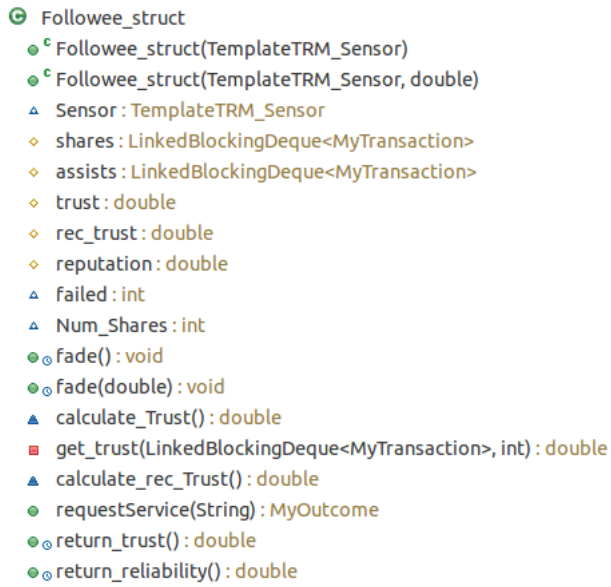
```
9       this.outcome = outcome;
10      fading = 1.0;
11      this.severity = severity;
12      this.id=id;
13    }
14  }
```

Listing 4.5: MyTransaction

4.2.3 Followee_struct

Αυτή η κλάση αντιπροσωπεύει έναν φίλο (ή αλλιώς followee στο CosmoS). Περιεχέι όλες τις πληροφορίες που χρειάζεται να ξέρει μία Εικονική Οντότητα για κάποια άλλη, την οποία έχει φίλη για κάποια υπηρεσία, όπως το log_file, τον δείκτη εμπιστοσύνης, τον δείκτη φήμης κ.α. (βλ. σχήμα 4.6)

Διαισθητικά, κοιτώντας την ιεραρχία στο σχήμα 3.6, είναι λογικό πώς θα υπάρχουν 4 Followee_struct. Αυτό σημαίνει ότι θα υπάρχει διαφορετικό για την Εικονική Οντότητα B στην υπηρεσία Vehicle και διαφορετικό για την ίδια E.O. όταν παρέχει γνώση. Αυτό είναι αποτέλεσμα της αποσύνδεσης των υπηρεσιών μεταξύ τους. Έτσι είναι δυνατός ο αποκλεισμός της Εικονικής Οντότητας για παροχή γνώσης χωρίς τον αποκλεισμό της για παροχή υπηρεσιών σε οχήματα.



Σχήμα 4.6: Followee

Εκτός από δεδομένα εσωκλείεται και λειτουργικότητα στο Followee_struct . Περιέχει μεθόδους για την εφαρμογή εξασθένησης στο log_file, για τον υπολογισμό του δείκτη εμπιστοσύνης και βασικότερο όλων, περιεχέι την μέθοδο που υλοποιεί την αίτηση για την υπηρεσία στην Εικονική-Οντότητα φίλο. Μία περίληψη των μεθόδων φαίνεται στον κώδικα 4.6

```
1   public synchronized void fade() {
2     for (MyTransaction t : shares)
3       if ((t.fading -= 0.02) <= 0)
4         shares.remove(t);}
5
```

```
  double calculate_Trust() {
    if ((Math.random() < 0.2) && (shares.size() < 2))
      return 1.0; // if new friend (20% chance to trust him)
    if (shares.size() >10){
    double temp1 = get_trust(shares, shares.size()/10);
    double temp2 = get_trust(shares, shares.size());
    trust = Math.min(temp1, temp2);
    }
    else if(shares.size() >0) trust = get_trust(shares, shares.size());
    return trust;
  }

  private double get_trust( LinkedBlockingDeque<MyTransaction> assists2, int size) {
    double sum = 0.0;// wi*xi
    double weights = 0.0;// wi
    double wx2 = 0.0;// wi*xi^2
    for (int i = 0; i < size; i++) {
      MyTransaction t = assists2.poll();
      if (t!=null){
      double weight = t.fading * t.severity;
      double xi = t.getSatisfaction().getSatisfactionValue();
      double temp = weight * xi;
      sum += temp; // wi*xi
      wx2 += temp * xi;// wi*xi*xi
      weights += weight;// wi
      }}
    fade();
    double m = sum / weights;
    double s = (wx2 * weights - Math.pow(sum, 2)) / Math.pow(weights, 2);
    return m - s;
  }

  public MyOutcome requestService(String Service) {
    SatisfactionInterval satisfaction = null;
      double temp=Sensor.serve(Service);
      satisfaction = new SatisfactionInterval(0.0, 1.0,
          temp);
    return new MyOutcome(satisfaction);
  }
```

Listing 4.6: Followee_struct

4.2.4 Application_struct

Η τελευταία βοηθητική κλάση που θα δούμε είναι το Application Struct. Σε αυτήν την κλάση συγκεντρώνουμε όλες τις λειτουργικότητες που χρειάζεται το RT-IOT για να μπορέσει να διαχειριστεί του δείκτες φήμης και εμπιστοσύνης μίας συγκεκριμένης υπηρεσίας. Άρα εδώ περιέχονται όλοι οι φίλοι που παρέχουν αυτήν την υπηρεσία καθώς και το κατώφλι που χρειάζεται να περνάει ο δείκτης εμπιστοσύνης για να εμπιστευθεί η Εικονική Οντότητα τον φίλο της.

Διαισθητικά, κοιτώντας την ιεραρχία στο σχήμα 3.6, είναι λογικό πώς θα υπάρχουν 2 Application_struct ένα για κάθε υπηρεσία τα οποία περιέχουν από 2 Followee_struct το καθένα.

Επίσης εδώ περιέχεται η λογική για να συγκεντρωθούν οι διάφοροι δείκτες φήμης και εμπιστοσύνης των φίλων και να καταταγούν ώστε να βρεθεί ο πιο έμπιστος τόσο για ιδία χρήση μετά από σύγκριση με το threshold όσο και για παροχή recommendation. Ακόμα εδώ γίνεται ή εισαγωγή νέων εγγραφών στην κορυφή των log_files, όπως φαίνεται στο 4.7

```
/**This method request service from the most trusted followee*/
public synchronized Outcome request_Service() {
  MyOutcome outcome = null;
  ComparableFriend friend = rank_friends2();//get most trusted VE
  if (friend.value > threshold) {// then trust him
    outcome = get_followee(friend.id).requestService(service.id);
    if (outcome!=null)
      addNewShare(friend.id, outcome);
    return outcome;
  }
  return null;
};
/**This method returns a followee as recommendation to someone else*/
public synchronized TemplateTRM_Sensor request_Sensor(boolean collusion, boolean
  malicious) {
  TemplateTRM_Sensor sensor = null;
  ComparableFriend friend = rank_friends2();
  if (friend.value > threshold) {
    sensor = get_followee(friend.id).Sensor;
  }
  if (malicious && collusion) {
    friend = rank_friends_reverse();
    if (friend.value<0.5)
      sensor = get_followee(friend.id).Sensor;
  }
  return sensor;
};

/**This method returns all frinds sorted by trust descending */
public synchronized PriorityQueue<ComparableFriend> rank_friends(String list) {
  PriorityQueue<ComparableFriend> evaluationQeue;

  Comparator<ComparableFriend> comparator = new FriendComparator();
  evaluationQeue = new PriorityQueue<ComparableFriend>(1,
      comparator);
  ArrayList<String> keysAsArray = new ArrayList<String>(
      followees.keySet());
  if (list.equals("trust")){
    for (String id : keysAsArray){
      if (get_followee(id).Sensor.isActive())
  evaluationQeue.add(new ComparableFriend(id, get_followe(id).calculate_Trust()));}}
  else{

  for (String id : keysAsArray){
    if (get_followee(id).Sensor.isActive()){
  evaluationQeue.add(new ComparableFriend(id, get_followee(id).calculate_rec_Trust()
  ));}}
  }
  return evaluationQeue;
}

/**This method returns the most trusted friend only*/
public synchronized ComparableFriend rank_friends2() {// compute trust for all
```

```
    followees
52  // and return higher trustee
53  ComparableFriend selected = new ComparableFriend("no", -0.1);
54
55  ArrayList<String> keysAsArray = new ArrayList<String>(
56      followees.keySet());
57  for (String id : keysAsArray) {
58    if (get_followee(id).Sensor.isActive()){
59
60      double ttrust = get_followee(id).calculate_Trust();
61      if (ttrust > selected.value) {
62        selected.id = id;
63        selected.value = ttrust;
64      }}}
65  return selected;
66  }
67
68  /** This method adds a new Share to the collection of shares of this sensor */
69  public synchronized void addNewShare(String id, MyOutcome outcome) {
70    Followee_struct f = get_followee(id);
71    if (f!=null)//mporei na edwsa service se emena apo recommendation
72      f.shares.addFirst(new MyTransaction(outcome));
73  }
```

Listing 4.7: Application_struct

4.2.5 Η πλατφόρμα

Η πλατφόρμα έχει δύο βασικές λειτουργικότητες. Πρώτον αρχικοποιεί το σύστημα και δεύτερον παρέχει προτάσεις για νέου φίλους με βάση κάποιον δείκτη φήμης που υπολογίζει.

Αρχικοποίηση συστήματος

Κατά την αρχικοποίηση του συστήματος η Πλατφόρμα δημιουργεί ένα Application_struct για κάθε υπηρεσία όπως και για recommendation στα οποία κρατάει Followee_struct για κάθε Εικονική Οντότητα που παρέχει την συγκεκριμένη υπηρεσία. Στην συνέχεια δέχεται ως είσοδο από το σύστημα τις παραμέτρους του πάνελ ρυθμίσεων 4.2 και δημιουργεί ένα δίκτυο με τα συγκεκριμένα χαρακτηριστικά. Με βάση τις προδιαγραφές του TRMSim-WSN θεωρείτε πώς οι malicious servers παρέχουν υπηρεσία ικανοποίησης 0 και οι benevolent server παρέχουν υπηρεσία ικανοποιησης 1.0. Στο τέλος προστίθενται σε κάθε Εικονική Οντότητα δύο τυχαίοι φίλοι για μπορέσει αν εκκινήσει σωστά.

```
1
2  public class TemplateTRM_Network extends Network {
3
4    protected HashMap<String, Application_struct> Service_VEs; //
5
6    /**
7     * This constructor creates a new random TemplateTRM Network using the given
8     * parameters
9     *
10    * @param numSensors
11    *            Network number sensors
12    * @param probClients
13    *            The network will have a number of clients depending of this
14    *            parameter.
15    * @param rangeFactor
16    *            Range Factor.
17    * @param probServices
18    *            Probability of servers offers services and clients requesting
19    *            services.
20    * @param probGoodness
21    *            Probability of servers being good.
22    * @param services
23    *            Services that servers offers to clients.
24    */
25   public TemplateTRM_Network(int numSensors, double probClients,
26       double rangeFactor, Collection<Double> probServices,
27       Collection<Double> probGoodness, Collection<Service> services) {
28     super(null, null, null);
29     Service_VEs = new HashMap<String, Application_struct>();//struct me tis listes ton
       VE ana service gia na ginoun recommendations
30     clients = new ArrayList<Sensor>();
31     servers = new ArrayList<Sensor>();
32     sensors = new ArrayList<Sensor>();
33     this.services = new ArrayList<Service>();// ola ta services tou
34
35     for (Service service : services)
36       Service_VEs.put(service.id,new Application_struct(service.id,
37               Math.random() * 0.5 + 0.5)); // create log+files
38     Sensor.resetId();
39
40     for (int i = 0; i < numSensors; i++) {
41       if (Math.random() <= probClients) { // client intialize
42         TemplateTRM_Sensor client = newSensor();
43         clients.add(client);
44         sensors.add(client);
45         Iterator<Double> itProbServices = probServices.iterator();
46         for (Service service : services)
47           // add requested services
48           if (Math.random() <= itProbServices.next().doubleValue()) {
49             client.addNewService(service.id);
50           }
51
52       } else { // server intialize
53         TemplateTRM_Sensor server = newSensor();
54         clients.add(server);
55         servers.add(server);
56         sensors.add(server);
```

```
57
58        Iterator<Double> itProbServices1 = probServices.iterator();
59        Iterator<Double> itProbServices2 = probServices.iterator();
60        Iterator<Double> itProbGoodness = probGoodness.iterator();
61
62        for (Service service : services)
63          // add requested services
64          if (Math.random() <= itProbServices1.next().doubleValue()) {
65            server.addNewService(service.id);
66          }
67
68        for (Service service : services){
69          // add provided services
70          if (Math.random() <= itProbServices2.next().doubleValue()) {
71            Service_VEs.get(service.id).followee_put(server,0.0);
72            if (Math.random() <= itProbGoodness.next()
73                .doubleValue()){// check quality of service
74             server.addService(service, 1.0);
75            }else{
76              server.addService(service, 0.0);
77              server.malicious=true;
78            }
79
80        }}}}
81
82    // We are going to add some friends to the sensors two for each service
83    // it requests
84    for (Sensor client : clients) {
85      TemplateTRM_Sensor myclient = (TemplateTRM_Sensor) client;
86      Set<String> ServiceSet = myclient.getRequestedServices();
87      for (String service : ServiceSet) {
88        if (service.equals("My service")){
89        myclient.addfollowee(Service_VEs.get(service).get_random_entry(), service,0.9)
    ;
90             myclient.addfollowee(Service_VEs.get(service).get_random_entry(), service
    ,0.9);}
91    }}
```

Listing 4.8: Αρχικοποίηση Πλατφόρμας

Παροχή Προτάσεων

Αυτή η διαδικασία αποτελείτε από δύο διαφορετικά βήματα αναγκαία για την σωστή λειτουργία της πλατφόρμας.

1. **Αποδοχή feedback:** Αυτή η μέθοδος καλείτε περιοδικά από της Εικονικές Οντότητες και προσομοιώνει την αποστολή feedback στην πλατφόρμα. Αυτό γίνεται στέλνοντας την εμπιστοσύνη στις διάφορες Εικονικές Οντότητες φίλους για διαφορές υπηρεσίες. Η Πλατφόρμα με την σειρά της τα αποθηκεύει στα δικά της Application_stuct για περαιτέρω επεξεργασία.

2. **Παροχή προτάσεων:** Όταν ζητηθεί από την πλατφόρμα να προτείνει κάποιον νέο φίλο αυτή υπολογίζει τους δείκτες φήμης όλων τον Εικονικών Οντοτήτων που παρέχουν την υπηρεσία και τους κατατάσσει όμοια με την διαδικασία υπολογισμού εμπιστοσύνης.

Στην συνέχει όπως αναφέραμε υπάρχει 80% πιθανότητα να προτείνει μία από της πιο φημισμένες Εικονικές Οντότητες και 20% να προτείνει μία τυχαία.

```
/**
 * This method is called by VE want it reports its experience
 *
 * @param reportedService
 *            the application struct of the VE containing its experience for the service
 * @param value
 *            the value of the service it provides it can be changed to boolean malicous on another version
 *
 *   @param collusion
 *      if there is collusion on the network
 */

public synchronized void report(Application_struct reportedService, Double value , Boolean collusion) {
  Application_struct myStruct = Service_VEs.get(reportedService.service.id); // retrieve the platforms struct for the service
  ArrayList<String> keysAsArray = new ArrayList<String>(
      reportedService.followees.keySet());//get the friends of the VE reporting
  for (String id : keysAsArray) { //for each friend add a share containing satisfaction as the trust index
    Followee_struct global = myStruct.followees.get(id);
    Followee_struct local = reportedService.followees.get(id);

    if (value>=0)   myStruct.addNewShare(id, new MyOutcome(new SatisfactionInterval(0.0, 0.9,local.calculate_Trust())));
    global.fade(0.005);//fading effect for the struct of platform
  }

}
/**
 * This method is called by VE that wants a new friend
 *
 * @param s
 *            service that the friend should have
 */
public synchronized TemplateTRM_Sensor get_recommendation(String s) {
  Application_struct myStruct = Service_VEs.get(s);
  PriorityQueue<ComparableFriend>  MyQueue = new PriorityQueue<ComparableFriend>();
  MyQueue = myStruct.rank_friends("trust");
  double propability = Math.random();//0.4 o prwtos 0.4 o deyteros 0.2 random me miden trust
  if (propability<0.4 && !MyQueue.isEmpty())
  {
    String f = MyQueue.poll().id;
    if (myStruct.followees.get(f)!=null)
      return myStruct.followees.get(f).Sensor;
  }
  MyQueue.poll();
  if (propability<0.8 && !MyQueue.isEmpty())
  {
    String f = MyQueue.poll().id;
    if (myStruct.followees.get(f)!=null)
      return myStruct.followees.get(f).Sensor;
```

```
49      }
50      if(!MyQueue.isEmpty()){
51        ComparableFriend f1 = MyQueue.poll();
52        while (!MyQueue.isEmpty() & f1.value>0.0)
53          MyQueue.poll();
54        if(!MyQueue.isEmpty()){
55          String f = MyQueue.poll().id;
56          if (myStruct.followees.get(f)!=null)
57            return myStruct.followees.get(f).Sensor;
58        }
59      }return null;
60    }
61
62
63  }
```

Listing 4.9: Παροχή Προτάσεων

4.2.6 Η Εικονική Οντότητα

Τα μόνα πεδία που χρειάζεται μία Εικονική Οντότητα είναι ένα HashMap με όλα τα Application_struct για τις υπηρεσίες που ζητάει και ένα HashMap που περιεχέι της υπηρεσίες τις οποίες παρέχει. Στην προσομοίωση οι υπηρεσίες είναι συνδεδεμένες με έναν αριθμό που είναι η ικανοποίηση που θα παρέχουν, αλλά στον πραγματικό κόσμο εκεί θα υπάρχει μία δομή με την λογική για την παροχή της εκάστοτε υπηρεσίας.

Σε ότι αφορά την λογική που περιέχεται στις μεθόδους της κλάσης, αυτή αφορά αρκετά κομμάτια της λειτουργίας της Εικονική Οντότητας. Βασικότερη όλων είναι η μέθοδος run η οποία περιεχέι την λογική με την οποία θα γίνει η αίτηση στον κατάλληλο πάροχο τον οποίο θα προσπαθήσει αρχικά η Εικονική Οντότητα να τον βρει μέσα από τις δικές της εμπειρίες χρησιμοποιώντας το Application_struct. Στην συνέχεια εάν αποτύχει, είτε θα ψάξει για assistance από φίλους, είτε θα ζητήσει να τις προταθούν φίλοι. Αυτό θα γίνει κατανεμημένα μιλώντας με τις φιλικές Εικονικές Οντότητες που περιλαμβάνονται στο Application_struct για την υπηρεσία recommendations. Εάν και αυτό αποτύχει τότε θα ερωτηθεί η πλατφόρμα για προτάσεις. Πριν το τέλος κάθε κύκλου υπάρχει η πιθανότητα αποστολής report στην πλατφόρμα, η οποία ακολουθείτε από καθαρισμό της λίστας φίλων από κακόβουλες Εικονικές Οντότητες

```
1   public void run() {
2     outcome = null;
3     Set<String> Services = getRequestedServices();
4     for (String s : Services)// gia kathe service pou thelw
5     {
6       if (s.equals("My␣service")) {
7         Application_struct new_requiredService = get_app(s);
8         outcome = new_requiredService.request_Service(); // request service base on
    Trust on followees
9         if (outcome == null)
10        { if (Math.random() < 0.3 ) outcome=get_assistance(s,5,this.id);
11        else outcome=get_recommendation(s);
12
13        }
14        if (outcome == null)
15        { TemplateTRM_Sensor recommended=platform.get_recommendation(s);
```

```
        if (recommended!=null){
          addfollowee(recommended,s,0.9);
          outcome = new_requiredService.request_Service();

      if (Math.random() <0.1) { this.report();
      this.blacklist();
}}}
```

Listing 4.10: Μέθοδος run()

Επιπρόσθετα μέσα στην Εικονική Οντότητα είναι και η λογική για την απόκτηση προτάσεων για φίλους και για την εξαγωγή δεικτών φήμης μέσα από τον κοινωνικό της κύκλο ο οποίος τελικά οδηγεί στην αίτηση για υπηρεσία από τον πιο φημισμένο όπως περιγράφεται στην παράγραφο 3.3.2.

```

  /**
   * This method returns the outcome of finding a new friend from recommendation and
    requesting the service s
   * @param String s the Service
   * */
  private  Outcome get_recommendation(String s) {
    Application_struct struct1=null;
    TemplateTRM_Sensor my_new_sensor =null;
    Application_struct recommenders = friends.get("recommendation");
    PriorityQueue<ComparableFriend> MyQueue=recommenders.rank_friends("trust");
    ArrayList<Followee_struct> recommendations =  new ArrayList<Followee_struct>();
    LinkedList<TemplateTRM_Sensor> recommender =  new LinkedList<TemplateTRM_Sensor>()
    ;
    while (!MyQueue.isEmpty() && recommendations.size()<4){//mazeyw protaseiw apo most
     trusted filous
      ComparableFriend temp = MyQueue.poll();
      if ( temp.value > 0.5)
      {
        TemplateTRM_Sensor current = recommenders.get_followee(temp.id).Sensor;
        if (current!=null)  struct1 = current.get_app(s);
        if (struct1!=null)  my_new_sensor = struct1.request_Sensor(collusion,malicious
    );
        if (my_new_sensor!=null && my_new_sensor.isActive()){Followee_struct  next =
    new Followee_struct(my_new_sensor);
        recommendations.add(next);  recommender.add(current);}
      }else break;
    }
    //Exw tis protaseis zhtaw feedback gia ton kathena kai vriskw reputation apo ton
    most reputable  ton kanw filo kai zhtaw service
    double reputation=0.0;
    for (Followee_struct current : recommendations){
      current.reputation = calculate_reputation(current.Sensor.id,s);
      // System.out.println(current.reputation);
      if (current.reputation > reputation) current.reputation = reputation;}
    for (Followee_struct current : recommendations){
      TemplateTRM_Sensor his_recommender = recommender.pollFirst();
      if (current.reputation ==  reputation)
      {   Application_struct Service = friends.get(s);
      addfollowee(current.Sensor,s,0.0);      //vazw filo kai zhtaw service
      Service.change_reputation(current.Sensor.id,s,current.reputation);
      MyOutcome outcome1=current.requestService(s);
      Service.addNewShare(Integer.toString(current.Sensor.id), outcome1);
```

```
38        recommenders.addNewShare(Integer.toString(his_recommender.id), outcome1);//kanw
      evaluate ton arxiko recommender
39        return outcome1;
40        }
41
42
43      }
44
45      return null;
46    }
47    /**
48     * This method caluclates the reputation of a VE for a Service
49     * @param id the VE
50     * @param String service the Service
51     * */
52    private double calculate_reputation(int id, String service) {
53      //zhtaw feedback apo high reputable kai to vazw se lista mazi me to pios to edwse
54      LinkedList<MyTransaction> feedback = new LinkedList<MyTransaction>();
55      double m = 0,s=0;
56      Application_struct recommenders = friends.get("recommendation");
57      PriorityQueue<ComparableFriend> MyQueue=recommenders.rank_friends("trust");
58      for(int i=0; i<5 && !MyQueue.isEmpty() ;i++ ){
59        ComparableFriend temp = MyQueue.poll();
60        double experience=recommenders.get_followee(temp.id).Sensor.ask_experience(
      Integer.toString(id),service);
61        if (experience >=0){
62          double weight=temp.value;
63          feedback.add(new MyTransaction(new MyOutcome(new SatisfactionInterval(0.0,
      1.0, experience)),weight,temp.id));}
64      }
65      // calculate intial reputation distribution
66      double sum = 0.0;// wi*xi
67      double weights = 0.0;// wi
68      double wx2 = 0.0;// wi*xi^2
69      int j=0;
70
71      if (feedback.size()>0){
72        for ( j = 0; j < feedback.size(); j++) {
73          MyTransaction t = feedback.get(j);
74          double weight = t.fading * t.severity;
75          double xi = t.getSatisfaction().getSatisfactionValue();
76          double temp = weight * xi;
77          sum += temp; // wi*xi
78          wx2 += temp * xi;// wi*xi*xi
79          weights += weight;// wi
80        }
81        m = sum / weights;
82        s = (wx2 * weights - Math.pow(sum, 2)) / Math.pow(weights, 2);
83      }
84      //next get more feedback
85      for(int i=0; i<15 && !MyQueue.isEmpty() ;i++ ){
86        ComparableFriend temp = MyQueue.poll();
87        double experience=recommenders.get_followee(temp.id).Sensor.ask_experience(
      Integer.toString(id),service);
88        if (experience >=0){
89          double weight=temp.value;
90          if ((experience < m-0.5*s) || (experience > m+0.5*s)) {
91            recommenders.addNewShare(temp.id, (new MyOutcome(new SatisfactionInterval
```

```
    (0.0, 1.0, 0.3))));
92          }
93          else feedback.add(new MyTransaction(new MyOutcome(new SatisfactionInterval
    (0.0, 1.0, experience)),weight,temp.id));}
94      }
95      if (feedback.size()>=j){ //continue calculating the reputation with the new
    feedbacks
96        for ( ; j < feedback.size(); j++) {
97          MyTransaction t = feedback.get(j);
98          double weight = t.fading * t.severity;
99          double xi = t.getSatisfaction().getSatisfactionValue();
100         double temp = weight * xi;
101         sum += temp; // wi*xi
102         wx2 += temp * xi;// wi*xi*xi
103         weights += weight;// wi
104       }
105       m = sum / (weights+0.000001);
106       s = (wx2 * weights - Math.pow(sum, 2)) / Math.pow(weights+0.00001, 2);
107       for (int i=0; i<feedback.size();i++){       //kane evaluate tous arxikous
108         MyTransaction t = feedback.get(i);
109         if ((t.getSatisfaction().getSatisfactionValue() < m-0.5*s) || (t.
    getSatisfaction().getSatisfactionValue() > m+0.5*s)) {
110           recommenders.addNewShare(t.id, (new MyOutcome(new SatisfactionInterval(0.0,
    1.0, 0.0))));
111         }else   recommenders.addNewShare(t.id, (new MyOutcome(new SatisfactionInterval
    (0.0, 1.0, 0.9))));
112       }
113       return m-s;
114     }
115     else return -1;
116   }
```

Listing 4.11: Μέθοδοι εύρεσης φίλων

Τέλος μπορεί να βρεθεί και η λογική για την παροχή υπηρεσιών. Δηλαδή υπάρχουν και οι μέθοδοι που δίνουν απάντηση σε κάποια εικονική οντότητα όταν προσπαθεί να μαζέψει προτάσεις, όταν ψάχνει μία υπηρεσία μέσα από την μέθοδο αίτησης σε φίλο φίλου και φυσικά όταν ζητάει μία υπηρεσία οπότε κοιτάμε στο HashMap και εφόσον υπάρχει εγγραφή γυρίζεται η ικανοποίηση

```
1   /**
2    * This Method returns the experience the VE had with another VE for a service
3    * @param id the sensor
4    * @param service the service
5    * @return the trust level
6    */
7   public synchronized double ask_experience(String id, String service) {
8
9     Application_struct f = friends.get(service);
10    if (f!=null)  {
11      Followee_struct f1 = f.get_followee(id);
12      if (f1!=null){
13       return f1.return_trust();
14
15      }
16    }
17
18    return -0.1; //negative means no experience at all
```

```
19  }
20
21  /**
22   *
23   * @param s the service
24   * @param i the TTL
25   * @param id id of the requester so that i do not lie to myself when colluding
26   * @return
27   */
28  public synchronized Outcome get_assistance(String s, int i,int id) {
29    Application_struct recommenders = friends.get("recommendation");
30    PriorityQueue<ComparableFriend> MyQueue=recommenders.rank_friends("assists");
31    while (!MyQueue.isEmpty() && outcome ==  null){
32      ComparableFriend temp = MyQueue.poll();
33      if ( temp.value > 0.5)
34      {
35
36          TemplateTRM_Sensor current = recommenders.get_followee(temp.id).Sensor;
37          Application_struct new_requiredService = current.get_app(s);
38          if (new_requiredService!=null){
39
40            outcome = new_requiredService.request_Service();
41          }if (i>0 && outcome==null) outcome = current.get_assistance(s, i-1,this.id);
42        }
43
44        if (outcome!=null){
45          recommenders.addNewAssist(temp.id,(MyOutcome)outcome);}
46      }else break;
47
48    }
49    return outcome;
50  }
51
52  /**
53   * This methods returns the actual service value
54   * @param service
55   * @return the satisfaction
56   */
57  public synchronized double serve(String service) {
58    if(servicesGoodness2.get(service)!=null && this.isActive()){
59      double quality = servicesGoodness2.get(service);
60      return Math.round(10 * (quality)) / 10.0;}
61    else return -0.1;
62  }
```

Listing 4.12: Μέθοδοι παροχής υπηρεσιών

4.3 Υλοποίηση χαρακτηριστικών προσομοιωτή

Εκτός από την υλοποίηση του συστήματος στον προσομοιωτή πρέπει να υλοποιηθούν και άλλα χαρακτηριστικά που θα βοηθήσουν στην προσομοίωση των επιθέσεων . Αυτά είναι η εναλλαγή συμπεριφοράς (oscilattin VE's), η δυναμική είσοδος και έξοδος από το σύστημα (dynamic) και η συνεργασία μεταξύ κακοόβουλων οντοτήτων.

Εναλλαγή συμπεριφοράς

Η εναλλαγή συμπεριφοράς υλοποιείτε κατευθείαν στην λογική του προσομοιωτή για ν είναι εύκολη η εναλλαγή στην αναπαράσταση τον κόμβων και να μπορεί να παραμείνει ίδιο το ποσοστό των κακόβουλων οντοτήτων σε όλη την προσομοίωση.

Έτσι ανά 20 κύκλους κάποιες εικονικές Οντότητες αλλάζουν την ικανοποίηση στην παρεχόμενη υπηρεσία από καλή σε κακή και αντίστροφα. Με αυτόν τον τρόπο θα μπορέσουμε να παρατηρήσουμε την δυναμική αναδιοργάνωση τον σχέσεων φιλίας μεταξύ των κόμβων όπως θα δούμε στο επόμενο κεφάλαιο.

```
1  /**
2     * This method turns every benevolent server in the network into malicious and
3     * counts the number of swapped servers. Then (when every server is malicious)
4     * it randomly selects malicious servers and converts them into benevolent until
5     * the number of benevolent servers is equal to the one before calling this method
6     * @param service Service over what the oscillation is to be carried out
7     */
8    public void oscillate(Service service) {
9
10           int numBenevolentServers = 0;
11           for (Sensor server : servers)
12               if ((server.offersService(service)) && (server.get_goodness(service)
     >= 0.5)) {
13                   numBenevolentServers++;
14                   server.set_goodness(service,0.0);
15               }
16           double prob = ((double)numBenevolentServers/servers.size());
17           while (numBenevolentServers > 0)
18               for (Sensor server : servers)
19                   if ((Math.random() < prob) && (server.offersService(service)) && (
     server.get_goodness(service)< 0.5)) {
20                       server.set_goodness(service,1.0);
21                       numBenevolentServers--;
22                       if (numBenevolentServers == 0)
23                           break;
24                   }
25   }
```

Listing 4.13: Εναλλαγή συμπεριφοράς

Δυναμική Εκτέλεση

Για να υλοποιηθεί η δυναμική εκτέλεση έχουμε βάλει στα threads την ώρα που εξυπηρετούν μία αίτηση να ελέγχουν μία συνθήκη η οποία ισχύει κάθε 20 αιτήσεις. Όταν συμβεί αυτό τα

threads και άρα οι Εικονικές Οντότητες που αναπαριστούν γίνονται ανενεργές. Έτσι για όσο χρόνο είναι ανενεργές οι Εικονικές Οντότητες δεν απαντάνε σε αιτήματα ακριβώς όπως θα συνέβαινε και στην πραγματικότητα.

```
numRequests++;
    if (numRequests == numRequestsThreshold) { // Edited by Hamed Khiabani for dynamic
     nodes
      numRequests = 0;
      if (dynamic && runningSimulation) {
        activeState = false;
        numRequestsTimer = new Timer();
        numRequestsTimer.schedule(new TimerTask(){
          @Override
          public void run() {
            activeState = true;
            numRequestsTimer.cancel();
          }
        },sleepingTimeoutMilis);
      }
    }
```

Listing 4.14: Δυναμική Εκτέλεση

Κακόβουλες συνεργασίες

Οι κακόβουλες συνεργασίες επηρεάζουν διαφορές λειτουργίες του συστήματος και για αυτό έπρεπε να αλλαχτούν αρκετές μέθοδοι. Έτσι :

1. Οι κακόβουλες εικονικές οντότητες δεν διαγράφουν τους κακόβουλους φίλους τους επειδή τους χρησιμοποιούν για να χειραγωγήσουν το σύστημα
2. Όταν ζητάει κάποιος recommendation από κακόβουλη Ε.Ο. αυτή γυρίζει την αντίθετη τιμή εμπιστοσύνης από την πραγματική. Έτσι δίνει κακές συστάσεις για καλές οντότητες με βάση την γνώση της και καλές στους κακόβουλους φίλους της.
3. Όταν ζητάει κάποιος την υπηρεσία από φίλο φίλου η κακόβουλη Ε.Ο. γυρίζει την υπηρεσία την οποία παρέχει μία άλλη κακόβουλη Εικονική Οντότητα. Αυτό οδηγεί σε χαμηλή ικανοποίηση του αρχικού αιτούντα.
4. Όταν στέλνει report στην πλατφόρμα η κακόβουλη Ε.Ο. όλοι οι δείκτες εμπιστοσύνης είναι αντίθετοι έτσι ώστε να δυσκολέψει την διαδικασία παροχής προτάσεων από την πλατφόρμα.

```

 /**
  * This Method returns the experience the VE had with another VE for a service
  * @param id the sensor
  * @param service the service
  * @return the trust level
  */
 public synchronized double ask_experience(String id, String service) {

   Application_struct f = friends.get(service);
```

```
    if (f!=null)  {
      Followee_struct f1 = f.get_followee(id);
      if (f1!=null){
        if(malicious && collusion){ //if malicous and collusion return reverse
    experience
          return 1-f1.return_trust();
        }else return f1.return_trust();

      }
    }

    return -0.1; //negative means no experience at all
  }

  public synchronized Application_struct get_app(String S){
    return friends.get(S);


  }

  /**
   *
   * @param s the service
   * @param i the TTL
   * @param id id of the requester so that i do not lie to myself when colluding
   * @return
   */
  public synchronized Outcome get_assistance(String s, int i,int id) {
    Application_struct recommenders = friends.get("recommendation");
    PriorityQueue<ComparableFriend> MyQueue=recommenders.rank_friends("assists");
    while (!MyQueue.isEmpty() && outcome ==  null){
      ComparableFriend temp = MyQueue.poll();
      if ( temp.value > 0.5)
      {
        if(collusion && (id!=this.id) && malicious){ //collude in order to return
    outcome of a malicous friend

          Application_struct bad_struct = get_app(s);
          if (bad_struct!=null){
            PriorityQueue<ComparableFriend> BadQueue= bad_struct.rank_friends("trust")
    ;
            while (!BadQueue.isEmpty()){
              ComparableFriend bad_temp = BadQueue.poll();
              if (bad_temp.value < 0.3){ //found bad friend
                Followee_struct ff = bad_struct.get_followee(temp.id);
                if (ff!=null){
                  TemplateTRM_Sensor current = ff.Sensor;
                  Application_struct new_requiredService = current.get_app(s);
                  if (new_requiredService!=null){
                    outcome = new_requiredService.request_Service();}
                }}
            }
          }


        }
        if (outcome!=null){
          recommenders.addNewAssist(temp.id,(MyOutcome)outcome);}
```

```
66        }else break;
67
68      }
69      return outcome;
70    }
```

Listing 4.15: Κακόβουλες συνεργασίες