ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΜΣ: ΨΗΦΙΑΚΑ ΣΥΣΤΗΜΑΤΑ ΚΑΙ ΥΠΗΡΕΣΙΕΣ

ΚΑΤΕΥΘΥΝΣΗ: ΔΙΚΤΥΟΚΕΝΤΡΙΚΑ ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη και αξιολόγηση εμπιστοσύνης μεταξύ συμμετεχόντων σε υπολογιστικά περιβάλλοντα και πιστοποίηση αποστολής ποιοτικών δεδομένων

Επιμέλεια

ΤΡΙΑΝΤΑΦΥΛΛΟΥ ΝΙΚΟΛΑΟΣ

Επιβλέπων: ΕΠΙΚΟΥΡΟΣ ΚΑΘΗΓΗΤΗΣ ΔΗΜΟΣΘΕΝΗΣ ΚΥΡΙΑΖΗΣ

Ιούνιος 2016

ΕΥΧΑΡΙΣΤΙΕΣ

Για την περάτωση της διπλωματικής μου εργασίας, πέραν της προσωπικής προσπάθειας που κατεβλήθη καθ' όλη τη διάρκεια της ακόλουθης μελέτης, βασικό ρόλο στη διαμόρφωση της τελικής εργασίας διαδραμάτισαν διάφοροι συνεργάτες, οι οποίοι με την προσφορά και τη βοήθειά τους διευκόλυναν και υποστήριξαν σημαντικά το έργο μου.

Θα ήθελα να ευχαριστήσω τον επίκουρο Καθηγητή του Πανεπιστημίου Πειραιώς και επιβλέποντα αυτής της εργασίας κ. Δημοσθένη Κυριαζή για την πολύτιμη συμβολή του στη διεκπεραίωση αυτής της εργασίας με την υποστήριξη, τη συμπαράσταση, το χρόνο και την προσωπική γνώση που μου παρείχε για την κατανόηση του αντικειμένου της διπλωματικής εργασίας και το ευχάριστο κλίμα συνεργασίας κάτω από το οποίο αυτή πραγματοποιήθηκε.

Τέλος, θα ήθελα να ευχαριστήσω τους δικούς μου ανθρώπους, οικογένεια και φίλους που στάθηκαν αρωγοί και συμπαραστάτες στην προσπάθειά μου να ολοκληρώσω με επιτυχία το συγκεκριμένο κύκλο σπουδών μου.

ΠΕΡΙΛΗΨΗ

Η ταχύτατη ανάπτυξη του Διαδικτύου των Πραγμάτων (Internet of Things), σε συνδυασμό με την εξάπλωση των λύσεων και εφαρμογών cloud (δηλαδή ένα εικονικό νέφος από συστοιχίες υπολογιστών που προσφέρουν υπολογιστικούς πόρους μέσω του διαδικτύου) και τη συνεχή εξέλιξη στις τεχνολογίες και υποδομές των δικτύων, πέρα από τα πλεονεκτήματα που προσφέρει δημιουργεί ταυτόχρονα ανησυχίες ως προς την αξιοπιστία των συσκευών που μετέχουν στο δίκτυο και την ασφαλή μεταφορά των δεδομένων μεταξύ αυτών.

Η παρούσα διπλωματική έχει σα στόχο την ανάπτυξη και παρουσίαση ενός νέου μοντέλου για την πιστοποίηση της διακίνησης ποιοτικών δεδομένων μέσα σε ένα δίκτυο διασυνδεδεμένων συσκευών και την ανάπτυξη εμπιστοσύνης μεταξύ των μελών ενός cloud δικτύου. Αυτό επιτυγχάνεται χρησιμοποιώντας ως βασική ιδέα την τεχνολογία του πλέον επιτυχημένου ψηφιακού νομίσματος, του Bitcoin.

Μέσα στο δίκτυο συναλλαγών με το Bitcoin, η διαφάνεια και η ασφάλεια των συναλλαγών εξασφαλίζονται με τη χρήση μιας κατανεμημένης βάσης δεδομένων όπου καταγράφονται όλες οι επικυρωμένες συναλλαγές. Για να γίνει αυτό απαιτείται η ύπαρξη χρηστών που ελέγχουν και πιστοποιούν τις συναλλαγές αυτές. Οι χρήστες αυτοί ονομάζονται miners.

Στο πλαίσιο υλοποίησης της εργασίας αναπτύχθηκε πρόγραμμα που προσομοιώνει τις συναλλαγές δεδομένων μεταξύ των συσκευών σε ένα δίκτυο και χρησιμοποιεί τα βασικά χαρακτηριστικά του Bitcoin (με τις απαραίτητες τροποποιήσεις) για την πιστοποίηση της αποστολής και λήψης ποιοτικών δεδομένων.

ABSTRACT

The rapid development of the Internet of Things, combined with the spread of solutions and cloud applications (i.e. a virtual cloud of computer arrays that provide computing resources over the Internet) and the continuous evolution of technologies and network infrastructures, besides the advantages offered, simultaneously creates concerns about the reliability of the devices involved in the network and secure data transfer between them.

This thesis aims at the development and presentation of a new model for certification of qualitative data handling within a network of interconnected devices and the development of trust among members of a network cloud. This could be achieved by using as a basic idea the technology of the most successful digital currency, the Bitcoin.

Inside the trading network that uses the Bitcoin, transparency and security of transactions are ensured by the use of a distributed database where all valid transactions are recorded. This requires users who check and certify these transactions, and these users are called miners.

In order to implement this thesis, a program was developed. This particular program simulates data transactions between devices on a network and uses the basic features of the Bitcoin (with the necessary changes) for the certification of sending and receiving qualitative data.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΔΙΑΓΡΑΜΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

Τα τελευταία χρόνια παρατηρείται μια ταχεία εξέλιξη στην τεχνολογία και στις δυνατότητες που παρέχει το hardware. Αυτό σε συνδυασμό με την αναβάθμιση και την ταχύτητα των δικτύων ευρείας περιοχής (WAN) καθώς και τη διαθεσιμότητα των virtualization λύσεων επιτρέπουν την ανάπτυξη νέων μοντέλων για το IT. Ένα από αυτά είναι το Cloud Computing. Όταν πρωτοεμφανίστηκε επρόκειτο για καινοτόμο σύστημα διότι απελευθέρωνε το χρήστη από το hardware προσφέροντας υπηρεσίες και υπολογιστικούς πόρους μέσα από το διαδίκτυο.

Αν και ήταν επαναστατική η τεχνολογία αυτή, εντούτοις υπήρχε και φυσικά υπάρχει ακόμη αρκετό περιθώριο για βελτιστοποίηση και περαιτέρω ανάπτυξη. Ένα από τα πρώτα θέματα που εμφανίστηκαν και έπρεπε να αντιμετωπιστούν άμεσα ήταν η κατακόρυφη αύξηση του αριθμού των συσκευών που συνδέονται στο διαδίκτυο. Αυτό οδηγούσε σε καθυστερήσεις κατά τη μεταφορά δεδομένων μέσα από αυτό. Μια πρώτη λύση στο πρόβλημα ήταν η μεταφορά υπολογιστικών πόρων πιο κοντά στο άκρο του δικτύου. Κάπως έτσι ήρθε στη ζωή μας το fog computing ενώ με τη σύνδεση πολλών συσκευών στο διαδίκτυο εκμεταλλευόμαστε αυτό που σήμερα ονομάζεται Internet of Things (IoT).

Σύμφωνα με την τελευταία έκθεση της Ericsson Mobility Report, το διαδίκτυο των αντικειμένων (Internet of Things) θα έχει εκθρονίσει τα κινητά τηλέφωνα από την κατηγορία των περισσότερων συνδεδεμένων συσκευών, μέχρι το 2018. Μεταξύ 2015 και 2021, ο αριθμός των συσκευών που είναι θα συνδεδεμένες στο IoT, αναμένεται να παρουσιάσει αύξηση 23%, ετησίως, με το μεγαλύτερο ρυθμό αύξησης να εντοπίζεται στο κινητό IoT. Από τα 28 δισεκατομμύρια συσκευών που θα έχουν συνδεθεί έως το 2021, σχεδόν 16 δισεκατομμύρια θα αντιπροσωπεύουν συσκευές IoT. [1]

Ανάμεσα στις συσκευές που έχουν ήδη συνδεθεί ή θα συνδεθούν στο IoT βρίσκονται και διάφορες συσκευές που δε διαχειρίζονται άμεσα από ανθρώπους αλλά λειτουργούν με βάση τον τρόπο που έχουν προγραμματιστεί (π.χ. αισθητήρες, «έξυπνες» συσκευές κλπ.). Η αύξηση των διασυνδεδεμένων συσκευών παρασέρνει σε αύξηση και τον όγκο

των δεδομένων που μεταφέρονται στο δίκτυο διαμέσου αυτών. Μάλιστα πολλές φορές τα δεδομένα χρησιμοποιούν τέτοιες συσκευές σα δίαυλο για την κίνησή τους. Λόγω του όγκου των δεδομένων, δημιουργούνται διάφοροι προβληματισμοί για την ποιότητα τους, την ασφάλεια τους κατά τη μεταφορά καθώς και για τους τρόπους που αυτά ελέγχονται. Όλοι μας έχουμε βρεθεί στη θέση να θέλουμε να κάνουμε ένα download χωρίς να είμαστε σίγουροι για την ποιότητα των δεδομένων που θα κατεβάσουμε.

Η ασφαλής ανταλλαγή ποιοτικών δεδομένων μεταξύ των συσκευών στο άκρο του δικτύου είναι μια περίπτωση που απασχολεί ιδιαίτερα όσους ασχολούνται με την ασφάλεια στο cloud. Στην παρούσα εργασία παρουσιάζεται ένα μοντέλο ασφάλειας που κινείται σε αυτό το πλαίσιο. Δίδεται ιδιαίτερη βαρύτητα στον προκαταβολικό έλεγχο της ποιότητας των δεδομένων, προτού αυτά παραληφθούν από κάποια συσκευή. Δε γίνεται έλεγχος όμως στα δεδομένα αυτά καθ' αυτά αλλά στην «πηγή» τους, στη συσκευή από την οποία αποστέλλονται. Συγκεκριμένα, κάθε συσκευή που θέλει να στείλει δεδομένα ελέγχεται ως προς την αξιοπιστία της και αναλόγως της επιτρέπεται ή όχι να προχωρήσει στην αποστολή. Με τον τρόπο αυτό δημιουργείται μια σχέση εμπιστοσύνης μεταξύ των μελών ενός δικτύου και προστατεύονται οι χρήστες που μέσα σε ένα τεράστιο δίκτυο αγνώστων συσκευών αναζητούν πληροφορίες ή δεδομένα.

Έμπνευση για την ανάπτυξη του μοντέλου αυτού και βασικό στοιχείο αποτελεί η τεχνολογία που βρίσκεται πίσω από το πλέον επιτυχημένο και ευρέως διαδεδομένο ψηφιακό νόμισμα, το Bitcoin. Χρησιμοποιώντας μια distributed βάση δεδομένων, διάφοροι χρήστες-εθελοντές επιβεβαιώνουν τις συναλλαγές ή τις απορρίπτουν ανάλογα με τη εγκυρότητά τους. Με παρόμοιο τρόπο λειτουργεί και το μοντέλο που παρουσιάζουμε στη συνέχεια.

ΚΕΦΑΛΑΙΟ 2

CLOUD COMPUTING – EDGE COMPUTING

2.1 ΓΕΝΙΚΕΣ ΠΛΗΡΟΦΟΡΙΕΣ ΓΙΑ CLOUD ΚΑΙ EDGE COMPUTING

Η βασική αρχή του cloud computing είναι η χρήση ενός δικτύου από απομακρυσμένους υπολογιστικούς πόρους (δίκτυα, εξυπηρετητές, αποθηκευτικούς χώρους, εφαρμογές, υπηρεσίες), οι οποίοι φιλοξενούνται στο Διαδίκτυο, για αποθήκευση, διαχείριση και επεξεργασία δεδομένων αντί να χρησιμοποιείται ένας τοπικός διακομιστής ή ένας προσωπικός υπολογιστής. Πρόκειται για μια τάση για τα υπολογιστικά συστήματα όπου δυναμικά κλιμακωτοί (dynamically scalable) και εικονοποιημένοι (virtualized) πόροι προσφέρονται σαν υπηρεσίες μέσα από το διαδίκτυο. Το μοντέλο αυτό προωθεί τη διαθεσιμότητα και αποτελείται από 5 σημαντικά χαρακτηριστικά, 3 μοντέλα υπηρεσίας και 4 μοντέλα ανάπτυξης.

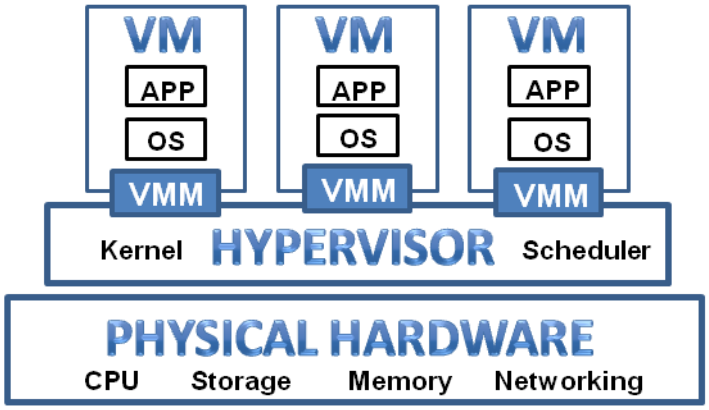
Υπολογίζεται ότι μέχρι το 2020 πάνω από 20 δισεκατομμύρια συσκευές θα είναι συνδεδεμένες στο Internet. Αυτός ο αριθμός πέρα από κινητά, τάμπλετ και τηλεοράσεις περιλαμβάνει και δισεκατομμύρια αισθητήρες. Όλες αυτές οι συνδεδεμένες στο διαδίκτυο συσκευές συνθέτουν το Internet of Things (IoT) και ανοίγουν νέες δυνατότητες σε διάφορους χώρους όπως για παράδειγμα στην υγεία, στα smart homes και στον κατασκευαστικό κλάδο. Το κλειδί για να επιτύχει ένα τόσο φορτωμένο δίκτυο και να λειτουργεί απρόσκοπτα είναι η μεταφορά ενεργειών στο άκρο του δικτύου (edge). Αυτές οι ενέργειες αφορούν στην αποθήκευση, επεξεργασία και ανάλυση δεδομένων. Το edge computing δηλαδή φέρνει το cloud computing πιο κοντά στο άκρο του δικτύου που εξυπηρετεί, πιο κοντά στις συσκευές.

2.2 VIRTUALIZATION

Μια από τις βασικές τεχνολογίες στην οποία είναι βασισμένη η ανάπτυξη του cloud computing είναι το virtualization. Με τον όρο virtualization εννοούμε την τεχνολογία με

την οποία τα φυσικά συστήματα (hardware) και οι πόροι αυτοί μετατρέπονται σε ιδεατά (virtual). Οι πόροι τμηματοποιούνται ώστε να καλυφθούν αιτήματα υπηρεσίας από διάφορα εικονικά συστήματα προς το συγκεκριμένο hardware. Δίνεται έτσι η δυνατότητα να λειτουργούμε παραπάνω από μια virtual machines, οι οποίες είναι ανεξάρτητες μεταξύ τους, σε μια πλατφόρμα hardware. Οι virtual machines χρησιμοποιούν τους πόρους του hardware (μνήμη, επεξεργαστή, αποθηκευτικό χώρο) σύμφωνα με κατανομή που εμείς ορίζουμε και μπορεί είτε να είναι σταθερή είτε να μεταβάλλεται ανάλογα με τις ανάγκες των virtual machines και τους διαθέσιμους πόρους του συστήματος.

Με την τεχνολογία αυτή το hardware διαχωρίζεται από το software. Υπάρχει ένα ενδιάμεσο layer (virtualization layer), ο hypervisor, που τμηματοποιεί τους φυσικούς πόρους και τους διαμοιράζει στα virtual machines. Τα virtual machines νομίζουν ότι επικοινωνούν με το hardware αλλά στην πραγματικότητα επικοινωνούν με τον ενδιάμεσο layer. Είναι εφικτή έτσι η μετακίνηση πόρων του hardware από ένα virtual machine σε ένα άλλο ανάλογα με τις εκάστοτε ανάγκες. Με τον τρόπο αυτό βελτιστοποιείται η χρήση των διαθέσιμων πόρων και εκμεταλλευόμαστε πλήρως τις δυνατότητές του.



Εικόνα : 1 Βασική αρχιτεκτονική cloud

Ένα πολύ σημαντικό θετικό του virtualization είναι ότι προσφέρει υψηλή διαθεσιμότητα. Με τη χρήση διάφορων τεχνικών clustering [2] και multi-pathing είναι εφικτή η αντιμετώπιση προβλημάτων σε φυσικούς servers χωρίς να διακοπεί η λειτουργία.

Επίσης μπορεί να αντίγραφα ασφαλείας των virtual machines ενώ αυτές λειτουργούν και να επαναφέρονται σε οποιαδήποτε παρελθούσα κατάσταση απαιτηθεί.

Από τη στιγμή που δε δεσμεύεται κάποιο hardware μπορεί αυτό να χρησιμοποιείται μόνο όταν υπάρχει μεγάλη ζήτηση. Με τον τρόπο αυτό μειώνεται το κόστος συντήρησης και διαχείρισης καθώς και η κατανάλωση ηλεκτρικής ενέργειας από το χρήστη λιγότερων servers.

Εδώ πρέπει να τονίσουμε ότι εν μέρει επικρατεί η άποψη ότι τα virtual machines δεν έχουν την ίδια απόδοση με τα φυσικά μηχανήματα λόγω του επιπλέον επιπέδου μεταξύ των εφαρμογών και του hardware. Παρόλα αυτά η συγκεκριμένη τεχνολογία έχει εξελιχθεί πολύ και διάφορες μετρήσεις που γίνονται κατά καιρούς δείχνουν ότι η επίπτωση είναι πολύ μικρή, πράγμα που οφείλεται κυρίως στον hypervisor και στην καλύτερη χρήση hardware που προσφέρει. [3],[4]

2.3 ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ CLOUD COMPUTING

Τα βασικά χαρακτηριστικά του Cloud Computing είναι τα παρακάτω [5] [6]:

2.3.1 ΠΡΟΣΦΟΡΑ ΒΑΣΕΙ ΖΗΤΗΣΗΣ (ON-DEMAND-SELF-SERVICE)

Ο χρήστης μπορεί να αιτηθεί μονομερώς τη χρήση υπολογιστικών πόρων όπως ο χρόνος που θα απασχολήσει το server, το μέγεθος του αποθηκευτικού χώρου που χρειάζεται ή την επεξεργαστική ισχύ και όλο αυτό να γίνει αυτόματα, χωρίς να χρειάζεται επικοινωνία ή αλληλεπίδραση με τον πάροχο της εκάστοτε υπηρεσίας.

2.3.2 ΕΥΡΕΙΑ ΠΡΟΣΒΑΣΗ (BROAD NETWORK ACCESS)

Οι δυνατότητες που παρέχει το cloud computing είναι προσπελάσιμες από παντού μέσα από το δίκτυο και μέσω διάφορων μηχανισμών γίνεται εφικτή η χρήση και πρόσβαση σε αυτές από διάφορες πλατφόρμες στη μεριά του χρήστη (π.χ. κινητά τηλέφωνα, laptop, PDA κλπ.)

2.3.3 ΠΡΟΣΒΑΣΗ ΑΝΕΞΑΡΤΗΤΗ ΤΗΣ ΤΟΠΟΘΕΣΙΑΣ (LOCATION INDEPENDENT RESOURCE POOLING)

Οι υπολογιστικοί πόροι του παρόχου χρησιμοποιούνται με τρόπο που εξυπηρετούν πολλούς ενοικιαστές χρησιμοποιώντας multi-tenant μοντέλο (μοντέλο πολλαπλών μισθωτών). Οι διάφοροι φυσικοί και εικονικοί πόροι ανατίθενται δυναμικά μετά από απαίτηση του χρήστη. Ο χρήστης δε γνωρίζει, ούτε μπορεί να ελέγξει την ακριβή τοποθεσία του παρεχόμενου πόρου αλλά μπορεί να προσδιορίσει ένα υψηλότερο επίπεδο αφαίρεσης όπως η χώρα, η πόλη ή συγκεκριμένο data-center. Παραδείγματα τέτοιων πόρων είναι ο αποθηκευτικός χώρος, η επεξεργασία, η μνήμη, το εύρος ζώνης δικτύου καθώς και οι Virtual Machines.

2.3.4 ΕΛΑΣΤΙΚΟΤΗΤΑ (RAPID ELASTICITY)

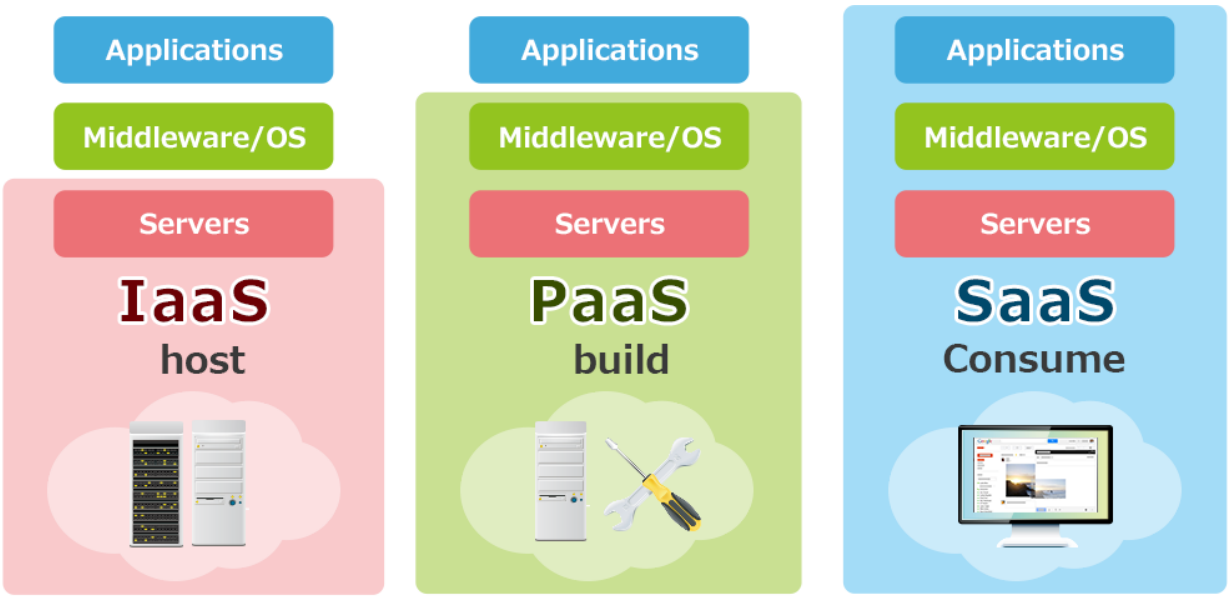
Οι πόροι του συστήματος μπορούν να δεσμευτούν για χρήση πολύ γρήγορα και ελαστικά ενώ το μέγεθος τους δύναται να αυξηθεί ή να μειωθεί σχεδόν άμεσα. Για τον καταναλωτή οι διαθέσιμοι πόροι για δέσμευση και χρήση μοιάζουν να είναι άπειροι και μπορούν να αγορασθούν-αποκτηθούν ανά πάσα στιγμή και σε οποιαδήποτε ποσότητα.

2.3.5 ΜΕΤΡΗΣΙΜΗ ΥΠΗΡΕΣΙΑ (MEASURED SERVICE)

Τα συστήματα cloud ελέγχουν και βελτιστοποιούν αυτόματα τη χρήση πόρων χρησιμοποιώντας ένα μηχανισμό μέτρησης ανάλογα με τον τύπο της προσφερόμενης υπηρεσίας (αποθηκευτικός χώρος, επεξεργαστική ισχύς, εύρος ζώνης κλπ). Οι πόροι που χρησιμοποιούνται μπορούν να παρακολουθηθούν και να ελεγχθούν και από τις δύο πλευρές, τελικού χρήστη και παρόχου της χρησιμοποιούμενης υπηρεσίας προσφέροντας έτσι διαφάνεια.

2.4 ΜΟΝΤΕΛΑ ΥΠΗΡΕΣΙΑΣ CLOUD COMPUTING

Τα διαθέσιμα μοντέλα του cloud computing είναι τα Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS) και το Infrastructure-as-a-Service (IaaS). Κάθε ένα από τα μοντέλα αυτά εξυπηρετεί διαφορετικές ανάγκες και προσφέρει διαφορετικές υπηρεσίες.[7]



Εικόνα : 2 Μοντέλα υπηρεσίας cloud computing

2.4.1 SOFTWARE-AS-A-SERVICE (SaaS)

Στο μοντέλο αυτό ο καταναλωτής έχει τη δυνατότητα να χρησιμοποιεί εφαρμογές του παρόχου οι οποίες τρέχουν σε υποδομή cloud. Οι εφαρμογές αυτές είναι προσβάσιμες από διάφορες client συσκευές μέσω μιας διεπαφής ή εργαλείου όπως για παράδειγμα ένας Internet Browser. Ο χρήστης δεν έχει τη δυνατότητα να ελέγχει ή να διαχειρίζεται την υποδομή του δικτύου ή τις ειδικές δυνατότητες της εφαρμογής. Χαρακτηριστικό παράδειγμα εφαρμογής σε ένα μοντέλο Saas είναι το web mail. Ο χρήστης ενδιαφέρεται απλά να χρησιμοποιήσει την υπηρεσία και δεν τον ενδιαφέρει να την κατανοήσει.

Σε ένα μοντέλο SaaS, ο πελάτης δεν αγοράζει λογισμικό, αλλά το νοικιάζει για χρήση σε ένα συνδρομητικό ή pay-per-use μοντέλο. Σε ορισμένες περιπτώσεις, η υπηρεσία είναι δωρεάν για περιορισμένη χρήση. Ο χρήστης έχει πρόσβαση στην υπηρεσία μέσω οποιασδήποτε εξουσιοδοτημένης συσκευής. Ένας άλλο όρος για το μοντέλο αυτό είναι το «Software on Demand».

Τα βασικά πλεονεκτήματα του μοντέλου Saas είναι:

- Δίνεται η δυνατότητα σε οργανισμούς να αναθέτουν τη διαχείριση και αποθήκευση διάφορων εφαρμογών σε έναν τρίτο φορέα (πωλητής λογισμικού και παροχής υπηρεσιών) και με τον τρόπο αυτό μειώνεται το κόστος των αδειών

του λογισμικού, των servers που απαιτούνται καθώς και ό, τι εφαρμογή ή υποδομή είναι απαραίτητη για τη φιλοξενία της εφαρμογής εσωτερικά.

- Οι προμηθευτές λογισμικού ελέγχουν καλύτερα και πιο αποτελεσματικά όλες τις εκδόσεις του λογισμικού τους και τους δίνεται η δυνατότητα να περιορίζουν τη χρήση κατά το δοκούν απαγορεύοντας την αντιγραφή και τη διανομή αυτού.
- Οι αιτήσεις παροχής που χρησιμοποιούν το μοντέλο Saas χρησιμοποιούν συνήθως προσέγγιση παράδοσης one-to-many με το Web ως υποδομή. Ο τελικός χρήστης μπορεί να έχει πρόσβαση σε μια εφαρμογή SaaS μέσω ενός Web Browser.

.

2.4.2 PLATFORM-AS-A-SERVICE (PaaS)

Στο μοντέλο αυτό ο προμηθευτής προσφέρει ένα περιβάλλον ανάπτυξης για προγραμματιστές εφαρμογών, οι οποίοι αναπτύσσουν εφαρμογές και προσφέρουν τις υπηρεσίες αυτές μέσω της πλατφόρμας του παρόχου. Δουλειά του παρόχου είναι να αναπτύσσει και να προσφέρει εργαλεία και πρότυπα για την ανάπτυξη καθώς και κανάλια για τη διανομή και την πληρωμή. Ο πάροχος λαμβάνει μια πληρωμή για να παρέχει την πλατφόρμα, τις πωλήσεις και τις υπηρεσίες διανομής.

Το Paas είναι μια παραλλαγή του SaaS στην οποία το περιβάλλον ανάπτυξης προσφέρεται ως υπηρεσία. Ο χρήστης δε διαχειρίζεται ούτε ελέγχει τη cloud υποδομή η οποία συμπεριλαμβάνει τα δίκτυα, τους servers, τα λειτουργικά συστήματα ή τα αποθηκευτικά μέσα. Χρησιμοποιούν όμως δομικά στοιχεία (π.χ. προκαθορισμένα τμήματα κώδικα) που προσφέρει ο προμηθευτής, δηλαδή το περιβάλλον ανάπτυξης , για να δημιουργήσουν τις δικές τους εφαρμογές.

Το εργαλείο ανάπτυξης φιλοξενείται το ίδιο στο cloud και η πρόσβαση σε αυτό γίνεται μέσω browser. Με το μοντέλο αυτό οι developers μπορούν να χτίζουν web εφαρμογές χωρίς να χρειάζεται να έχουν εγκατεστημένα εργαλεία στον υπολογιστή τους. Και αυτό είναι που κάνει τα συστήματα PaaS χρήσιμα διότι επιτρέπει σε ανεξάρτητους χρήστες ή start up επιχειρήσεις να αναπτύσσουν web-based εφαρμογές χωρίς το κόστος και την πολυπλοκότητα της αγοράς server και της ρύθμισής αυτών.

Το βασικό πλεονέκτημα των εφαρμογών PaaS αφορά στην αύξηση του αριθμού των χρηστών που μπορούν να προγραμματίσουν, να διατηρήσουν και να αναπτύξουν web εφαρμογές. Πολύ γνωστές PaaS υπηρεσίες είναι το Windows Azure της Microsoft και το App Engine της Google.

2.4.INFRASTRUCTURE –AS-A-SERVICE (IaaS)

Στο μοντέλο αυτό παρέχεται στο χρήστη η δυνατότητα να δεσμεύσει πόρους του cloud (αποθηκευτικά μέσα, επεξεργαστική ισχύ, δίκτυα και άλλους θεμελιώδεις πόρους) με σκοπό την ανάπτυξη και εκτέλεση λογισμικού το οποίο μπορεί να περιλαμβάνει συστήματα και εφαρμογές. Ο χρήστης δε μπορεί να ελέγξει την cloud υποδομή. Από την πλευρά του παρόχου IaaS, μπορεί να δημιουργήσει υποδομή που να διαχειρίζεται τη ροή και τη ζήτηση από πελάτες. Επίσης στο μοντέλο αυτό ο πάροχος μπορεί να καλύψει μόνο το hosting της εφαρμογής ή μπορεί να επεκταθεί και σε άλλε υπηρεσίες όπως υποστήριξη και ανάπτυξη εφαρμογής.

Ένα σύστημα IaaS περιλαμβάνει τα παρακάτω:

- Επεκτασιμότητα (Scalability): Δίνεται η δυνατότητα αναβάθμισης των απαιτήσεων σε υποδομή (υπολογιστικοί πότοι, αποθηκευτικός χώρος κλπ) ανάλογα με τις απαιτήσεις του χρήστη
- Χρέωση ανά χρήση (Pay-as-you-go): Ο χρήστης μπορεί να αγοράσει το ακριβές ποσό της υποδομής που χρειάζεται
- Βέλτιστες τεχνολογίες και πόροι (Best-of-breed technology and resources): Πρόσβαση σε βέλτιστες τεχνολογικές λύσεις και ανώτερο IT ταλέντο για ένα μέρος του κόστους.

Πολύ γνωστές υπηρεσίες IaaS παρέχουν η Amazon καθώς και η Rackspace.

2.5 ΜΟΝΤΕΛΑ ΑΝΑΠΤΥΞΗΣ CLOUD COMPUTING

Όπως προαναφέραμε υπάρχουν 4 βασικά μοντέλα ανάπτυξης σε ένα cloud σύστημα. Αυτά είναι το ιδιωτικό (private) cloud, το δημόσιο (public) cloud, το κοινοτικό (community) το υβριδικό (hybrid) μοντέλο. Τα μοντέλα αυτά είναι τεχνικά και λειτουργικά

άσχετα μεταξύ των μοντέλων υπηρεσιών που είδαμε στην προηγούμενη ενότητα. Οποιοδήποτε δηλαδή μοντέλο υπηρεσίας μπορεί να υπάρξει σε οποιοδήποτε μοντέλο ανάπτυξης αν και πολλές φορές ένας συγκεκριμένος συνδυασμός μπορεί να είναι πιο συχνός από άλλους (π.χ. το μοντέλο SaaS συνδυάζεται συνήθως με public cloud). [7],[8]

2.5.1 ΙΔΙΩΤΙΚΟ (PRIVATE) CLOUD

Στα ιδιωτικά ή εσωτερικά clouds η υποδομή του cloud λειτουργεί και είναι δεσμευμένη αποκλειστικά για έναν οργανισμό. Μπορεί να διαχειρίζεται και να ελέγχεται από τον ίδιο τον οργανισμό ή κάποιον που βρίσκεται εντός του οργανισμού. Οι χρήστες έχουν άμεση πρόσβαση σε υπολογιστικούς πόρους οι οποίοι φιλοξενούνται στην υποδομή του οργανισμού. Επίσης οι χρήστες μπορούν και ελέγχουν το μέγεθος των πόρων που χρησιμοποιούν.

Αν αναλογιστούμε ότι αναπτύσσεται εσωτερικά σε έναν οργανισμό (στο πλαίσιο ενός ήδη υπάρχοντος data center) το ιδιωτικό cloud υπόκειται σε περιορισμούς ασφαλείας από τον οργανισμό και αυτό το κάνει πιο ασφαλές σχετικά με ευαίσθητα δεδομένα και κώδικα. Από την άλλη όμως εκτιμήσεις σχετικά με την εξασφάλιση του εικονικού περιβάλλοντος γίνονται από τον πελάτη και όχι από τον πάροχο και αυτό επιβαρύνει κάτα κάποιο τρόπο τον οργανισμό. Το μεγαλύτερο πλεονέκτημα του ιδιωτικού cloud είναι ότι μπορεί να στηθεί με τρόπο που θα βρίσκεται πιο κοντά στις συγκεκριμένες απαιτήσεις και ανάγκες που θα έχει ο εκμισθωτής.

2.5.2 ΔΗΜΟΣΙΟ (PUBLIC) CLOUD

Στα δημόσια ή εξωτερικά clouds οι χρήστες έχουν πρόσβαση στους υπολογιστικούς πόρους που εκμισθώνουν μέσω του διαδικτύου. Η εκμίσθωση γίνεται με βάση την πληρωμή ανάλογα με τη χρήση (pay-as-you-go). Οι χρήστες ελέγχουν τους πόρους που έχουν ζητήσει μέσω μια δικτυακής επαφής. Τα δημόσια cloud έχουν μοναδικά στοιχεία ασφαλείας σε σχέση με τα ιδιωτικά. Επειδή είναι διαμορφωμένα για υποδομές μεγάλης συναμικότητας και μεγάλο εύρος πελατών, τα δεδομένα που βρίσκονται στα data centers πρέπει να είναι κωδικοποιημένα για μεγαλύτερη ασφάλεια.

2.5.3 ΚΟΙΝΟΤΙΚΟ (COMMUNITY) CLOUD

Η υποδομή του μοντέλου αυτού μοιράζεται από πολλούς οργανισμούς μέσα σε μία κοινότητα. Οι οργανισμοί αυτοί έχουν κοινούς στόχους ή ενδιαφέροντα όπως για παράδειγμα κοινές απαιτήσεις ασφαλείας ή αποστολή. Η διαχείριση του μπορεί να γίνεται από τους οργανισμούς που συμμετέχουν αλλά και από κάποιον τρίτο οργανισμό ή επιχείρηση.

2.5.4 ΥΒΡΙΔΙΚΟ (HYBRID) CLOUD

Το μοντέλο αυτό συνδυάζει πόρους τόσο από τα ιδιωτικά όσο και από τα δημόσια clouds. Αυτό το μοντέλο υιοθετείται από οργανισμούς που διαθέτουν ιδιωτικά clouds και θέλουν να έχουν και διασύνδεση με τα δημόσια clouds για δικούς τους σκοπούς. Ένας λόγος που επιχειρήσεις χρησιμοποιούν αυτό το μοντέλο είναι για να έχουν σε ιδιωτικό cloud την κρίσιμη υποδομή τους και να ικανοποιούν ανάγκες που δεν είναι οικονομικές με τη χρήση δημόσιων clouds. Για παράδειγμα μια επιχείρηση χρησιμοποιεί ιδιωτικό cloud για την υποδομή της και να εκμισθώνει υπολογιστικούς πόρους από ένα δημόσιο cloud για να δοκιμάζει αναβαθμίσεις.

2.6 EDGE COMPUTING ΚΑΙ INTERNET OF THINGS

Στα επόμενα χρόνια συνεχώς περισσότερες συσκευές θα παράγουν και θα καταναλώνουν δεδομένα. Αυτές οι συσκευές προφανώς αποτελούν μέρος μεγαλύτερων συστημάτων τα οποία χρειάζονται υπολογιστική ισχύ και αποθηκευτικό χώρο για να επεξεργάζονται και να αποθηκεύουν τα δεδομένα. Μέχρι ένα σημείο η επεξεργασία κάποιων δεδομένων γίνεται στη συσκευή. Αλλά είναι λογικό να απαιτούνται επιπλέον πόροι. Το που τοποθετούνται αυτοί οι πόροι δεν είναι καθόλου εύκολο να απαντηθεί. Βασικός παράγοντας είναι οι συσκευές που συνδέονται στο διαδίκτυο και φυσικά το γεγονός ότι απαιτούν πόρους για επεξεργασία και αποθήκευση δεδομένων. Πρέπει επίσης να λάβουμε υπόψη ότι ο αριθμός των συσκευών αυτών είναι τεράστιος και ολοένα αυξανόμενος και ότι οι συσκευές αυτές είναι διασκορπισμένες γεωγραφικά. Επιπλέον, κάποιες συσκευές θα έχουν διαφορετικές απαιτήσεις ανάλογα με το σκοπό που εξυπηρετούν. Για παράδειγμα συσκευές που κατά βάση τρέχουν real-time εφαρμογές έχουν πολύ μικρή ανοχή σε καθυστερήσεις.

Μια πρώτη απάντηση για το που θα βρίσκονται αυτοί οι πόροι είναι το cloud. Δυστυχώς όμως το επίπεδο των απαιτήσεων καθιστά το cloud πολλές φορές ανέφικτο, όταν

στόχος είναι η ανάπτυξη πλατφόρμας που να υποστηρίζει ένα μεγάλο εύρος IoT εφαρμογών. Η απάντηση σε αυτό το πρόβλημα είναι το fog computing το οποίο προτείνει ένα μοντέλο στο οποίο τα δεδομένα αναλύονται και επεξεργάζονται από συσκευές που βρίσκονται στο δίκτυο και όχι σε ένα κεντρικό cloud. Αυτό μας οδηγεί στο edge computing. Με την τοποθέτηση των πόρων στο άκρο το δικτύου και έξυπνη οργάνωση και διαχείριση αυτών το δίκτυο μπορεί να ανταπεξέλθει αποδοτικότερα στις απαιτήσεις των διασυνδεδεμένων συσκευών και των ολοένα αυξανόμενων απαιτήσεων.

Φυσικά, όσο οι περιορισμοί και οι απαιτήσεις των εφαρμογών ικανοποιούνται, είναι στη διακριτική ευχέρεια του κατασκευαστή της πλατφόρμας να επιλέξει που θα τοποθετήσει τα endpoints, αν δηλαδή θα τα τοποθετήσει στο cloud ή στο edge. Πρέπει να τονίσουμε ότι η μεταφορά πόρων στο edge δεν αναπτύχθηκε για να ανταγωνιστεί το cloud. Αντιθέτως αναπτύχθηκε για να το συμπληρώσει και να το επεκτείνει. Για το λόγο αυτό τοποθετήθηκαν μικρότεροι servers στο άκρο του δικτύου. Αυτές οι τεχνολογίες καλούνται να συνεργαστούν και να βοηθηθούν αφού για να ικανοποιηθούν οι απαιτήσεις του IoT πρέπει να υπάρξει έξυπνος συνδυασμός της επικοινωνίας, ενορχήστρωσης και ανάθεσης των πόρων. [9]

2.6.1 ΠΛΕΟΝΕΚΤΗΜΑΤΑ EDGE COMPUTING

Η μεταφορά πόρων στο άκρο ενός δικτύου παρουσιάζει τα παρακάτω πλεονεκτήματα [10],[11]:

- Οι υπηρεσίες εφαρμογής Edge μειώνουν σημαντικά τον όγκο των δεδομένων που πρέπει να μετακινηθούν και έτσι αποφεύγεται η κίνηση που θα δημιουργούταν. Ταυτόχρονα μειώνεται και η απόσταση που τα δεδομένα πρέπει να διανύσουν και έτσι μειώνονται τα έξοδα μετάδοσης, το latency και βελτιώνεται η ποιότητα της παρεχόμενης υπηρεσίας (Quality of Service-QoS)
- Εξαλείφονται, ή τουλάχιστον δεν εμφανίζεται τόσο στη διαδικασία ο πυρήνας του υπολογιστικού περιβάλλοντος περιορίζοντας έτσι τα σημεία συμφόρησης των δεδομένων και πιθανά σημεία αποτυχίας.
- Η ασφάλεια βελτιώνεται καθώς κρυπτογραφημένα δεδομένα μετακινούνται περαιτέρω προς το κεντρικό δίκτυο. Καθώς πλησιάζουν μια επιχείρηση, τα δεδομένα ελέγχονται για ιούς περνώντας μέσα από προστατευόμενα firewalls και άλλα σημεία ελέγχου και ανιχνεύονται νωρίς.

2.7 ΕΦΑΡΜΟΓΕΣ IoT

Η ανάπτυξη του IoT μπορεί φέρνει επαναστατικά αποτελέσματα και μεγάλη ανάπτυξη σε μια πληθώρα εφαρμογών. Η δυνατότητα παρακολούθησης (monitoring), συλλογής δεδομένων και οι υπόλοιπες διαδικασίες οι οποίες γίνονται μέσα από τις διασυνδεδεμένες συσκευές σε συνδυασμό με το edge computing και τη μεταφορά συστημάτων στο άκρο του δικτύου μόνο θετικά μπορεί να προσφέρει. Παρακάτω είναι συγκεντρωμένες οι βασικές εφαρμογές τις οποίες εκτοξεύει το IoT. [12]

2.7.1 ΠΕΡΙΒΑΛΛΟΝΤΙΚΗΣ ΠΑΡΑΚΟΛΟΥΘΗΣΗΣ (ENVIRONMENTAL MONITORING)

Οι εφαρμογές αυτές χρησιμοποιούν κάποιους αισθητήρες (sensors) για να βελτιώσουν και να βοηθήσουν την προστασία του περιβάλλοντος με την παρακολούθηση της ποιότητας του νερού, της κατάστασης της ατμόσφαιρας ή του εδάφους. Παρακολουθούν επίσης της κινήσεις στη φύση και στους βιότοπους. Συσκευές χωρίς περιορισμό σε χρήση πόρων μπορούν επίσης να τρέχουν εφαρμογές που συλλέγουν δεδομένα σχετικά με φυσικές καταστροφές (σεισμούς, τσουνάμι κλπ) και με την ανάλυση αυτών να προειδοποιούν σχετικά με την εκδήλωση φαινομένων. [13]

2.7.2 ΔΙΑΧΕΙΡΙΣΗ ΥΠΟΔΟΜΩΝ (INFRASTRUCTURE MANAGEMENT)

Η παρακολούθηση και ο έλεγχος αστικών και αγροτικών υποδομών όπως γέφυρες, σιδηροδρομικές γραμμές, αιολικά πάρκα είναι πολύ σημαντική παρεχόμενη υπηρεσία του IoT. Η υποδομή από αισθητήρες και συσκευές παρακολούθησης μπορεί να επιβλέπει τη δομική κατάσταση τους και σε περίπτωση που ανιχνεύσει κάποια αλλαγή που μπορεί να επηρεάσει την ασφαλή και απρόσκοπτη λειτουργία ενημερώνει κατάλληλα. Χρησιμοποιούνται επίσης για το σχεδιασμό της συντήρησης και επισκευών συντονίζοντας τα καθήκοντα μεταξύ διάφορων παρόχων υπηρεσιών συντήρησης και των χρηστών των υποδομών. Με τη χρήση διασυνδεδεμένων συσκευών μπορεί να γίνεται έλεγχος σημαντικών υποδομών όπως γέφυρες και πως αυτές θα κινούνται παρέχοντας χρόνο για διέλευση σε διάφορα πλοία. [14]

2.7.3 ΙΑΤΡΙΚΟΣ ΚΑΙ ΦΑΡΜΑΚΕΥΤΙΚΟΣ ΚΛΑΔΟΣ (MEDICAL AND HEALTHCARE SYSTEMS)

Η χρήση IoT συσκευών επιτρέπει την απομακρυσμένη παρακολούθηση της υγείας διάφορων ασθενών και την ανάπτυξη συστημάτων προειδοποίησης. Αυτές οι συσκευές παρακολούθησης μπορούν να ελέγχουν από πίεση και καρδιακούς παλμούς σε έναν άνθρωπο μέχρι και την κατάσταση εμφυτευμάτων ή πρόσθετων μελών. Ειδικοί αισθητήρες δύνανται να τοποθετούνται επίσης σε οίκους ευγηρίας ώστε να παρακολουθούν και να επιβεβαιώνουν την καλή κατάσταση της υγείας των ηλικιωμένων που ζουν σε αυτούς. Με την ανάπτυξη αισθητήρων παρακολούθησης υγείας συλλέγονται επίσης πολλά δεδομένα και γίνεται ίσως περισσότερο κατανοητή κάποια ασθένεια ενώ μπορεί να ελέγχεται και η φαρμακευτική αγωγή σε ασθενείς και κατά πόσο αυτή παρέχεται σωστά. [15]

2.7.4 ΔΙΑΧΕΙΡΙΣΗ ΕΝΕΡΓΕΙΑΚΩΝ ΠΟΡΩΝ (ENERGY MANAGEMENT)

Η ενσωμάτωση και χρήση αισθητήρων και συστημάτων κίνησης και η σύνδεση αυτών στο διαδίκτυο δύναται να βελτιώσει την ενεργειακή κατανάλωση. Με την τοποθέτηση τους σε οποιοδήποτε σύστημα καταναλώνει ενέργεια (διακόπτες, πρίζες, λαμπτήρες, τηλεοράσεις κλπ) θα υπάρχει δυνατότητα επικοινωνίας με εταιρείες παροχής ενέργειας με στόχο την προσαρμογή της παρεχόμενης ενέργειας στις ποσότητες που πραγματικά χρειάζονται για τη λειτουργία των συστημάτων αυτών. Ταυτόχρονα οι χρήστες θα μπορούν να ελέγχουν απομακρυσμένα τις συσκευές αυτές.

Με την ανάπτυξη έξυπνων δικτύων (smart grid) παρέχεται δυνατότητα για συλλογή και επεξεργασία σε πληροφορίες που σχετίζονται με την ενέργεια με στόχο την βελτίωση της αποτελεσματικότητας, της σταθερότητας, της οικονομίας και αξιοπιστίας της παραγωγής και κατανομής του ηλεκτρισμού. [16]

2.7.5 ΜΕΤΑΦΟΡΕΣ – ΣΥΓΚΟΙΝΩΝΙΕΣ (TRANSPORTATION)

Το IoT μπορεί να βελτιώσει την ποιότητα των μετακινήσεων των ανθρώπων αφού μπορεί να βοηθήσει στην ενσωμάτωση επικοινωνίας, ελέγχου και ροής-επεξεργασίας πληροφοριών σε διάφορα συστήματα μεταφοράς ενώ οι εφαρμογές που μπορεί να χρησιμοποιηθεί τέτοια τεχνολογία εκτείνονται σε όλες τα τμήματα των μεταφορών (π.χ. οχήματα, υποδομή, και οδηγούς ή χρήστες). Η δυναμική αλληλεπίδραση μεταξύ αυτών επιτρέπει τον έλεγχο της κίνησης των δρόμων, μειώνει το χρόνο αναζήτησης θέσεως στάθμευσης, επιτρέπει την τη χρήση ηλεκτρονικών διοδίων, τον έλεγχο οχημάτων και την καλύτερη αξιοποίηση της οδικής βοήθειας. [17]

2.7.6 ΑΥΤΟΜΑΤΙΣΜΟΙ ΣΤΗΝ ΚΑΤΟΙΚΙΑ (BUILDING AND HOME AUTOMATICATION)

Συσκευές IoT μπορούν να χρησιμοποιηθούν για την παρακολούθηση και τον έλεγχο των μηχανικών, ηλεκτρικών και ηλεκτρονικών συστημάτων που βρίσκονται σε διάφορους τύπους κτιρίων είτε αυτά είναι δημόσια ή ιδιωτικά, είτε είναι ιδρύματα, εργοστάσια ή απλά κατοικίες. Με τον τρόπο αυτό αυτοματοποιούνται τα εν λόγω συστήματα και γίνονται πιο απλές και παραγωγικές όλες οι διαδικασίες. Δύνανται να συμβάλλουν στην ασφάλεια των κτιρίων αυτών με διάφορες εφαρμογές (π.χ. αυτόματες ειδοποιήσεις για την κατάσταση του κτιρίου μέσα κλειστού κυκλώματος καμερών), στην αποδοτικότερη διαχείριση ενέργειας (αυτόματη λειτουργία ή παύση λειτουργίας συσκευών ανάλογα με την τιμή της ενέργειας, έλεγχος της θερμοκρασίας για μέγιστη αποδοτικότητα των σχετικών συσκευών κλπ), στη ψυχαγωγία και τη διασκέδαση καθώς και σε άλλους τομείς. [18]

ΚΕΦΑΛΑΙΟ 3

ΑΣΦΑΛΕΙΑ ΚΑΙ ΑΠΕΙΛΕΣ ΣΤΟ CLOUD

Στο κεφάλαιο αυτό θα ασχοληθούμε με θέματα ασφάλειας στο cloud computing και τις απειλές που υπάρχουν. Το cloud computing συνδυάζει διάφορες τεχνολογίες (δίκτυα, βάσεις δεδομένων, λειτουργικά συστήματα, εικονοποίηση, συστήματα διαχείρισης μνήμης κλπ). Για το λόγο αυτό όποια θέματα ασφαλείας υπάρχουν σε αυτά τα συστήματα ξεχωριστά, μεταφέρονται και στο cloud. Και αν αναλογιστούμε ότι τα δεδομένα του κάθε χρήστη (απόρρητα ή μη) είναι αποθηκευμένα στο διαδίκτυο τότε καταλαβαίνουμε ότι πρέπει να υπάρξουν μέτρα για την ασφάλεια ώστε να μην είναι εύκολο, αν όχι ακατόρθωτο, για έναν επιτιθέμενο να εκμαιεύσει κάποια από αυτά.

Το πιο σημαντικό στοιχείο είναι η αναγνώριση από τον οργανισμό των πραγματικών απαιτήσεων για τα θέματα ασφαλείας. Υπάρχουν τρεις κύριες πηγές για το σκοπό αυτό:

- Η αποτίμηση των κινδύνων (risk assessment) που αντιμετωπίζει ο οργανισμός. Μέσα από αυτή γίνονται γνωστές οι πιθανές απειλές προς τους πόρους του οργανισμού. Επίσης γίνεται εκτίμηση της ευπάθειας (vulnerability) του οργανισμού στις συγκεκριμένες απειλές, η πιθανότητα υλοποίησής τους καθώς και κόστος που θα έχουν στον οργανισμό.
- Το νομικό πλαίσιο και οι συμβατικές υποχρεώσεις του οργανισμού απέναντι στο κράτος, το προσωπικό και τους συνεργάτες του.
- Το σύνολο των αρχών, των απαιτήσεων και στόχων που ορίζει ο ίδιος ο οργανισμός σχετικά με την επεξεργασία των πληροφοριών που είναι απαραίτητες στη λειτουργία του. Το βασικότερο εργαλείο που χρησιμοποιείται για τον ορισμό των απαιτήσεων αυτών είναι η πολιτική ασφάλειας.

3.1 ΒΑΣΙΚΑ ΣΗΜΕΙΑ ΕΞΑΣΦΑΛΙΣΗΣ ΑΣΦΑΛΕΙΑΣ

Υπάρχουν 6 συγκεκριμένα σημεία στο περιβάλλον του cloud όπου ο εξοπλισμός και το λογισμικό απαιτούν ιδιαίτερη προσοχή για το θέμα της ασφάλειας. Αυτά τα 6 σημεία είναι τα παρακάτω [19]:

1) Η ασφάλεια των δεδομένων εκεί που είναι αποθηκευμένα:

Για την εξασφάλιση των αποθηκευμένων δεδομένων η χρήση μηχανισμών κρυπτογράφησης είναι η καλύτερη επιλογή. Οι κατασκευαστές σκληρών δίσκων προωθούν αυτό-κρυπτογραφούμενους (self-encrypting) δίσκους που εφαρμόζουν πρότυπα αξιόπιστης και ασφαλούς αποθήκευσης που εκδόθηκαν από το trusted computing group. [19]. Αυτοί οι σκληροί δίσκοι φέρουν κρυπτογραφημένο υλικό εσωτερικά παρέχοντας έτσι αυτόματα ένα επίπεδο κρυπτογράφησης δεδομένων με ελάχιστο κόστος και χωρίς αρνητική επίπτωση στην απόδοση. Αντίστοιχα μπορεί να χρησιμοποιηθεί και κρυπτογράφηση λογισμικού αλλά πέφτει η απόδοση του συστήματος και είναι πιο επισφαλές εξαιτίας του κλειδιού της κρυπτογράφησης που μπορεί να κλαπεί

2) Η ασφάλεια των δεδομένων κατά τη μεταφορά τους:

Η κρυπτογράφηση είναι επίσης η καλύτερη δυνατή λύση για κρυπτογράφηση δεδομένων κατά τη μεταφορά τους. Επιπλέον, μηχανισμοί ελέγχου ταυτότητας και προστασίας της ακεραιότητας των δεδομένων διασφαλίζουν πως τα δεδομένα όχι μόνο πηγαίνουν εκεί που ο χρήστης θέλει αλλά και πως δεν μεταβάλλονται από κανέναν κατά τη μεταφορά τους.

3) Η πιστοποίηση των χρηστών, εφαρμογών, διαδικασιών:

Ο έλεγχος ταυτότητας είναι υποχρεωτική απαίτηση για οποιαδήποτε υλοποίηση cloud. Πρόκειται για τη βασική αρχή για να του δοθεί πρόσβαση στον έλεγχο των δεδομένων. Άλλωστε σε ένα cloud περιβάλλον όλα τα δεδομένα είναι στο διαδίκτυο και μπορεί ο καθένας να έχει πρόσβαση. Επομένως είναι πιο σημαντικά από οτιδήποτε άλλο ο έλεγχος ταυτότητας κατά την πρόσβαση των χρηστών. Κάποια σχετικά με αυτό το θέμα πρότυπα που έχουν αναπτυχθεί επιτρέπουν τη real-time επικοινωνία μεταξύ του παρόχου μιας cloud υπηρεσίας και του πελάτη μόνο όταν προηγουμένως έχει γίνει

πιστοποίηση του χρήστη. Όταν αφαιρούνται ή τροποποιούνται δικαιώματα πρόσβασης από το χρήστη τότε αυτόματα ενημερώνεται ο πάροχος σε πραγματικό χρόνο και μπορεί να τροποποιήσει ή να αφαιρέσει τη δυνατότητα πρόσβασης του χρήση στο cloud.

4) Ο καθαρός διαχωρισμός μεταξύ των χρηστών:

Μια από τις πιο προφανείς ανησυχίες στο cloud είναι πως θα γίνει ο διαχωρισμός από τον πάροχο των χρηστών οι οποίοι μπορεί να είναι εταιρείες, μπορεί όμως να είναι και κακόβουλοι χρήστες. Πρέπει επομένως να αποφεύγεται η ακούσια ή εκούσια πρόσβαση σε ευαίσθητα δεδομένα. Ο πάροχος του cloud κανονικά χρησιμοποιεί όπως είδαμε εικονικές μηχανές και ένα hypervisor για να διαχωρίζει τους πελάτες και να τους δίνει την πρόσβαση στα δεδομένα τους.

5) Τα νομικά και ρυθμιστικά θέματα του cloud:

Είναι πολύ σημαντικό για όλα τα μέλη σε ένα cloud να μπορεί ο πάροχος να επιβεβαιώσει ότι εφαρμόζει αυστηρές στρατηγικές και μοντέλα σχετικά με νομικά θέματα και ο κάθε πελάτης πρέπει να ελέγχει τη νομιμότητα και το κατά πόσο ο πάροχος εφαρμόζει αυτά που δηλώνει ότι χρησιμοποιεί για ασφάλεια δεδομένων και όχι μόνο. Θέματα για τα οποία ο πάροχος θα πρέπει να είναι νομικά καλυμμένος και να εφαρμόζει ό,τι υπόσχεται αφορούν στην ασφάλεια των δεδομένων, στον τρόπο εξαγωγής αυτών, στον έλεγχο, διατήρηση και καταστροφή δεδομένων.

6) Η αντιμετώπιση σχετικών περιστατικών:

Σε κάθε περίπτωση οι πελάτες θα πρέπει να έχουν ένα σχέδιο για την πιθανότητα να υπάρξει κάποια παραβίαση στην ασφάλεια του cloud ή κάποια ανάρμοστη συμπεριφορά των άλλων χρηστών. Στην περίπτωση αυτή μία αυτόματη αντίδραση ή έστω μια άμεση και σε πραγματικό χρόνο ενημέρωση για την κατάσταση είναι η καλύτερη λύση για το σκοπό αυτό.

3.2 ΠΙΘΑΝΕΣ ΑΠΕΙΛΕΣ

Οι οργανισμοί ή οι απλοί χρήστες που επιλέγουν το cloud επιλέγουν συνειδητά τη λειτουργιά σε ένα περιβάλλον όπου υπάρχει ανταλλαγή πληροφορίας μεταξύ δύο

σημείων (πελάτης και server) και ανάμεσα στα δύο αυτά σημεία παρεμβάλλεται το διαδίκτυο μέσα από το οποίο γίνεται η μεταφορά των δεδομένων. Θα πρέπει επομένως να υπάρξει ένας έλεγχος για το κατά πόσο η αποστολή γίνεται με ασφαλή τρόπο. Πολλές φορές οι πληροφορίες που αποστέλλονται μπορεί να περιέχουν προσωπικά δεδομένα για θέματα όπως ο ιατρικός φάκελος ενός ασθενή, η ημερομηνία γέννησης, το Α.Φ.Μ. κλπ και επομένως είναι πολύς σημαντικό να διασφαλίσουμε πως η πληροφορία αυτή δε θα υποκλαπεί.

Επομένως η ασφαλής μεταφορά δεδομένων αποτελεί πολύ σημαντικό κομμάτι. Το ίδιο σημαντικό όμως είναι και ο τόπος που αποθηκεύονται τα δεδομένα που αποστέλλονται να είναι εξίσου ασφαλής. Το data center δηλαδή να παρέχει συνθήκες ασφάλειας και αξιοπιστίας. Πρέπει να γνωρίζουμε δηλαδή αν αποθηκεύονται κρυπτογραφημένα ή συμπιεσμένα και ακόμη αν ο πάροχος-διαχειριστής του data center έχει δυνατότητα πρόσβασης και χρήσης στα δεδομένα αυτά. Πρέπει δηλαδή να υπάρχει ασφάλεια στην υποδομή.

Με τη στροφή των εταιρειών και τον προσανατολισμό τους σε συστήματα βασισμένα στο cloud αναδύονται νέες πιθανές απειλές ενώ οι παλιές δύνανται να εξελιχθούν. Υπάρχουν διάφορες απαιτήσεις ασφαλείας τις οποίες πρέπει να πληροί ένα δίκτυο cloud για να θεωρείται ασφαλές. Υπάρχουν πολλοί που δεν αισθάνονται το cloud ιδιαίτερα ασφαλές και αυτό είναι λογικό αν λάβουμε υπόψη μας ότι όταν αποθηκεύουμε πληροφορίες στο cloud δε μπορούμε να γνωρίζουμε με ποιους μοιραζόμαστε στους πόρους και αυτό μας οδηγεί σε σκέψεις ότι χάνουμε τον έλεγχο και την ασφάλεια για τα δεδομένα μας.

Για έναν πάροχο, το να έχει πολλούς πελάτες μπορεί να τον οδηγήσει σε παραγωγή περισσότερης πληροφορίας και με σωστή ανάλυση αυτής οδηγείται σε πιο αποτελεσματική παρακολούθηση των απειλών. Επίσης, η ομοιογένεια που προσφέρει ένα cloud σύστημα κάνει την κεντρική διαχείριση ασφάλειας και παρακολούθηση πιο αποτελεσματική. Αυτό όμως σε καμία περίπτωση δε σημαίνει ότι η μεταφορά των δεδομένων στο διαδίκτυο δε τα κάνει επισφαλή και ευαίσθητα. Αντιθέτως φέρνει στην επιφάνεια ένα νέο μπλοκ απειλών όπως μια επίθεση στον κεντρικό μηχανισμό διαχείρισης το οποίο μπορεί να επιφέρει ένα μικρό χάος αφού κάθε χρήστης ίσως αποκτά πρόσβαση σε δεδομένα που δεν είναι δικά του.

Ένα είδος ασφαλείας της πληροφορίας στο cloud είναι η ταυτοποίηση των χρηστών για την είσοδο σε μια υπηρεσία. Με την εισαγωγή στοιχείων του χρήστη για την είσοδο σε ένα σύστημα ξεκινάει η πρόσβαση στους φακέλους και τα δεδομένα που βρίσκονται στο server. Βέβαια στη δομή του cloud ο επιτιθέμενος αν εντοπίσει με κάποιο τρόπο τα πιστοποιητικά ενός χρήστη αποκτά εύκολα πρόσβαση στο λογαριασμό του στο cloud άρα και στα δεδομένα του. Επομένως είναι απαραίτητη η σωστή χρήση των δεδομένων και η διαγραφή όσων δε χρειάζονται.

Το επόμενο πρόβλημα που προκύπτει είναι η ασφαλής μεταφορά των δεδομένων ενός χρήστη μέσα από το διαδίκτυο. Το πιο απλό μέτρο προστασίας είναι η κρυπτογράφηση των δεδομένων που μεταφέρονται. Πρέπει δηλαδή τα με κάποιο τρόπο τα μηνύματα που μεταφέρουν πληροφορίες να μετασχηματίζονται σε τέτοια μορφή ώστε αν κάποιος δεν έχει τον κώδικα για την αποκρυπτογράφηση να μη μπορεί να τα διαβάσει. Συγκεκριμένα να μπορούν να φτάσουν με ασφαλή τρόπο οι πληροφορίες από τον πελάτη στο server και το ανάποδο. [20]

3.3 ΑΣΦΑΛΕΙΑ ΥΠΟΔΟΜΩΝ ΤΟΥ ΠΕΛΑΤΗ – ΧΡΗΣΤΗ

3.3.1 ΕΠΙΠΕΔΟ ΔΙΚΤΥΟΥ

Όταν αναφερόμαστε σε επίπεδο δικτύου σχετικά με την ασφάλεια των υποδομών θα πρέπει αρχικά να γίνεται μια διάκριση μεταξύ των δημόσιων και των ιδιωτικών cloud. Στα ιδιωτικά, οι απειλές που υπάρχουν είναι σταθερές και για το λόγο αυτό θα πρέπει να εξετάζονται τα πιθανά κενά ασφαλείας και ιδιαίτερα τα κενά σχετικά με την τοπολογία του δικτύου στην οποία θα πρέπει να δίνεται ιδιαίτερη βαρύτητα. Αν ένας οργανισμός αρχίσει να χρησιμοποιεί ιδιωτικό cloud πιθανότατα θα αλλάξει και η αρχιτεκτονική του. Αντίθετα, η τοπολογία του πιθανότατα δε θα αλλάξει σε μεγάλο βαθμό. Για παράδειγμα εάν μια extranet δε διαφέρει και πολύ από τη λογική του ιδιωτικού cloud για μια επιχείρηση. Σε πολλά σημεία ο τρόπος φύλαξης και ασφάλειας που εφαρμόζεται πρακτικά μπορεί να χρησιμοποιηθεί και σε ένα ιδιωτικό cloud. Αυτό ισχύει και για τα εργαλεία ασφαλείας που πρακτικά χρησιμοποιούνται και τα οποία είναι απαραίτητα για ένα ιδιωτικό cloud.

Από την άλλη, στην περίπτωση που ένας οργανισμός επιλέξει δημόσιο cloud και στη συνέχεια επιλέξει να αλλάξει και τις απαιτήσεις ασφαλείας του δικτύου θα πρέπει να προβεί σε αλλαγές και στην τοπολογία του δικτύου. Πρέπει αν βρεθεί το πώς αλληλεπιδρούν οι δύο αυτέ τοπολογίες μεταξύ τους. Υπάρχουν τέσσερα σημαντικά σημεία που πρέπει να προσεχθούν ιδιαίτερα.

- Η διασφάλιση της εμπιστευτικότητας και της ακεραιότητας των δεδομένων που μεταφέρονται ανάμεσα στον οργανισμό και στο δημόσιο πάροχο cloud.
- Η σωστή πιστοποίηση κατά τον έλεγχο πρόσβασης στους πόρους που ο οργανισμός νοικιάζει από το δημόσιο πάροχο.
- Η εξασφάλιση της διαθεσιμότητας των πόρων του Διαδικτύου σε ένα δημόσιο cloud και οι οποίοι έχουν εκχωρηθεί και χρησιμοποιούνται από τον οργανισμό.

3.3.2 ΔΙΑΣΦΑΛΙΣΗ ΕΜΠΙΣΤΕΥΤΙΚΟΤΗΤΑΣ ΚΑΙ ΑΚΕΡΑΙΟΤΗΤΑΣ ΔΕΔΟΜΕΝΩΝ

Ένας οργανισμός που επιλέγει να χρησιμοποιήσει ένα cloud δημόσιο δίκτυο γνωρίζει ότι δεδομένα και πόροι του που στο παρελθόν περιορίζονταν σε ένα ιδιωτικό δίκτυο εκτίθενται σήμερα στο διαδίκτυο και μάλιστα σε κοινόχρηστο δίκτυο που ανήκει σε έναν τρίτο cloud πάροχο. Αυτό από μόνο του αποτελεί έναν παράγοντα κινδύνου. Ένα πρώτο βήμα για να μετριαστεί ο κίνδυνος είναι η χρήση του πρωτόκολλου HTTPS αντί του κλασσικού HTTP ώστε τα δεδομένα να είναι κρυπτογραφημένα. Οι οργανισμοί που συνεχίζουν να χρησιμοποιούν το HTTP φυσικά αντιμετωπίζουν μεγαλύτερο κίνδυνο για απώλεια δεδομένων ή αλλοίωση αυτών κατά τη μεταφορά τους.[21]

3.3.3 ΕΞΑΣΦΑΛΙΣΗ ΣΗΜΕΙΟΥ ΠΡΟΣΒΑΣΗΣ

Από τη στιγμή που ένα υποσύνολο των πόρων εκτίθεται στο διαδίκτυο, ένας οργανισμός που χρησιμοποιεί δημόσιο cloud αντιμετωπίζει αυξημένο κίνδυνο σχετικά με την ασφάλεια των δεδομένων του. Δεν είναι σε θέση να ελέγχει τις διεργασίες του δικτύου του παρόχου πόσο μάλλον να τις παρακολουθεί live όπως θα έκανε σε ένα δικό του δίκτυο. Η πρόσβαση στις καταγραφές σε επίπεδο δικτύου και δεδομένων είναι μειωμένη. Το ίδιο μειωμένη είναι και η δυνατότητα να διεξάγει ο οργανισμός έρευνα σε περίπτωση που υπάρξει κάποιο υποκλοπή των δεδομένων του.

Ένα χαρακτηριστικό παράδειγμα για την τελευταία περίπτωση είναι η επαναχρησιμοποίηση των IP διευθύνσεων. Όταν ένας πελάτης σταματήσει να χρησιμοποιεί μια διεύθυνση IP πάροχος δε τη θεωρεί παλιά, αντίθετα οι διευθύνσεις αυτές παρέχονται σε άλλους πελάτες αφού είναι διαθέσιμες. Αυτό είναι λογικό να συμβαίνει για τον πάροχο από τη στιγμή που του ανήκουν. Από τη μεριά του χρήστη όμως και όσον αφορά την ασφάλεια των δεδομένων του, η διατήρηση των διευθύνσεων IP από τη στιγμή που δε χρησιμοποιούνται πλέον μπορεί να επιφέρει προβλήματα. Αν δε απελευθερωθεί η διεύθυνση IP από έναν πελάτη, δε μπορεί να υποθέσει ότι η πρόσβαση στους πόρους του δικτύου έχει τερματιστεί. Σίγουρα υπάρχει μια μικρή καθυστέρηση ανάμεσα στην αλλαγή της διεύθυνσης IP στο DNS και την εκκαθάριση της εν λόγω διεύθυνσης από τις cashes του DNS. Αυτό σημαίνει ότι ακόμη και αν έχει γίνει αλλαγή στις διευθύνσεις ενός οργανισμού, οι «παλιές» διευθύνσεις παραμένουν διαθέσιμες στη μνήμη cache και έτσι μπορεί ο οργανισμός να χρησιμοποιεί ακόμη αυτούς τους μη διαθέσιμους πόρους.

3.4 ΑΣΦΑΛΕΙΑ ΥΠΟΔΟΜΩΝ ΤΟΥ ΠΑΡΟΧΟΥ

Για να εξετάσουμε την ασφάλεια των υποδομών θα πρέπει να εξετάσουμε και το μοντέλο παροχής υπηρεσιών του cloud (SaaS, PaaS, IaaS) καθώς και τα μοντέλα ανάπτυξης αυτού (ιδιωτικό, δημόσιο, υβριδικό). Επιπλέον, η ομοιογένεια του λειτουργικού συστήματος που χρησιμοποιείται από τους παρόχους καθώς και η δύναμη σε υπολογιστικούς κόμβους που υπάρχει οδηγεί μερικές φορές σε γρήγορη και εύκολη αύξηση της απειλής. Είναι σημαντικό επομένως να κατανοήσουν όλοι οι χρήστες τα όρια εμπιστοσύνης που πρέπει να δείχνουν καθώς και τις ευθύνες τους για την ασφάλεια της υποδομής που διαχειρίζονται. Αντίστοιχα φυσικά και οι πάροχοι οι οποίοι με τη σειρά τους πρέπει να γνωρίζουν τόσο για τις ευθύνες τους όσο και για το κατά πόσο έχουν λάβει όλα τα προβλεπόμενα μέτρα ασφαλείας. [20]

3.4.1 ΑΣΦΑΛΕΙΑ ΥΠΟΔΟΜΩΝ ΣΕ SaaS ΚΑΙ PaaS

Στην περίπτωση SaaS και PaaS μοντέλων υποδομής, οι πάροχοι συνήθως δε μοιράζονται με τους χρήστες δημόσια τις πληροφορίες που σχετίζονται με την πλατφόρμα, τα λειτουργικά συστήματα και τα θέματα ασφαλείας διότι είναι εύκολο για

κάποιον να χρησιμοποιήσει τις πληροφορίες αυτές και να εισβάλλει στην υπηρεσία cloud. Για το λόγο αυτό, στα ανωτέρω μοντέλα υποδομής η ασφάλεια και οι τρόποι επίτευξης παραμένουν κρυφά και η ευθύνη για την εφαρμογή μέτρων ασφαλείας επιβαρύνει τον πάροχο. Κάθε χρήστης έχει δικαίωμα να ζητήσει διαβεβαίωση για την ύπαρξη του προβλεπόμενου επιπέδου ασφαλείας και οι πληροφορίες αυτές θα ανταλλαχθούν με συγκεκριμένους πιστοποιημένους κανόνες ασφαλείας.

Η εικονοποίηση (virtualization) είναι η βασική τεχνολογία η οποία χρησιμοποιείται για να κάνει αποδοτικότερη τη χρήση της υποδομής. Ο πάροχος θα πρέπει να διαβεβαιώνει τον πελάτη ότι χρησιμοποιεί κατάλληλες διαδικασίες ώστε να ασφαλίζει το virualization επίπεδο. Τελικά στα δύο αυτά συστήματα υποδομής υπεύθυνος για την ασφάλεια της υποδομής είναι ο πάροχος. Παρόλο όμως που οι πελάτες δε χρειάζεται να ανησυχούν για το πώς οι ίδιοι θα εφαρμόσουν τα πρότυπα ασφαλείας των υποδομών του cloud, θα διατηρούν ένα ρίσκο για το κατά πόσο γίνεται σωστή διαχείριση των δεδομένων τους που βρίσκονται αποθηκευμένα στο cloud. [22], [23]

3.4.2 ΑΣΦΑΛΕΙΑ ΥΠΟΔΟΜΩΝ ΣΕ IaaS

Αντίθετα με τα δύο παραπάνω μοντέλα, στην περίπτωση της IaaS υποδομής οι πελάτες είναι αυτοί που ευθύνονται για το θέμα της ασφάλειας. Επειδή όμως όλες οι υπηρεσίες IaaS συμπεριλαμβάνουν και το virualization, το θέμα της ασφάλειας στην υποδομή περιλαμβάνει τόσο την ασφάλεια του λογισμικού του virtualization επιπέδου που βρίσκεται μεταξύ του hardware και των εικονικών servers που εκμισθώνονται όσο και η ασφάλεια του λογισμικού αυτών των εικονικών server. Για το πρώτο μέρος και για δημόσια υπηρεσία cloud οι πελάτες δεν έχουνε πρόσβαση στο εν λόγω λογισμικό στρώμα το οποίο διαχειρίζεται μόνο από τον πάροχο. Αντίθετα οι πελάτες έχουν πλήρη πρόσβαση στο λογισμικό του εικονικού server αφού επιλέγουν οι ίδιοι την έκδοση λογισμικού που θα χρησιμοποιήσουν. [24],[25]

3.5 ΑΣΦΑΛΕΙΑ ΔΕΔΟΜΕΝΩΝ

Οι απειλές για τις πληροφορίες και τα δεδομένα που είναι αποθηκευμένα στο cloud ποικίλουν ανάλογα με το μοντέλο που χρησιμοποιεί ο κάθε χρήστης και ο κάθε

οργανισμός. Υπάρχει ένας μεγάλος αριθμός απειλών για την ασφάλεια του cloud οι οποίες αυξάνουν την ευαισθησία του δικτύου. Μια γενική κατηγοριοποίηση των απειλών για τους χρήστες μπορεί να γίνει με βάση τα παρακάτω:

- Εμπιστευτικότητα
- Ακεραιότητα
- Διαθεσιμότητα

Στη συνέχεια αναλύονται οι παραπάνω παράγοντες σε συνδυασμό με τα μοντέλα παροχής υπηρεσιών. [20]

3.5.1 ΕΜΠΙΣΤΕΥΤΙΚΟΤΗΤΑ

Είναι έννοια στενά συνδεδεμένη με την ιδιωτικότητα και τη μυστικότητα. Αφορά τη μη αποκάλυψη ευαίσθητων πληροφοριών σε άτομα που δεν έχουν την κατάλληλη εξουσιοδότηση. Όταν πρόκειται για την προστασία του απορρήτου των δεδομένων που αποθηκεύονται σε ένα cloud, υπάρχουν τρεις πιθανές ανησυχίες:

- ✓ Εσωτερικές απειλές από χρήστες, από τον πάροχο ή από κάποιο τρίτο μέλος-χρήστη που υποστηρίζει τον πελάτη-οργανισμό ή τον πάροχο. Η εσωτερική απειλή είναι πολύ μεγάλη διότι και στα τρία μοντέλα υπάρχουν πολλοί εσωτερικοί χρήστες. Αν έχουμε SaaS πρόσβαση στα δεδομένα έχει ο πάροχος του cloud και οι πελάτες. Αν έχουμε PaaS μοντέλο οι προγραμματιστές των υπηρεσιών και αν έχουμε IaaS οι διάφορες συμβουλευτικές που ασχολούνται.
- ✓ Απειλές από το εξωτερικό περιβάλλον. Αυτό περιλαμβάνει απομακρυσμένη επίθεση στην υποδομή του cloud, απομακρυσμένη επίθεση στις εφαρμογές του cloud, απομακρυσμένη επίθεση στο λογισμικό και στο υλισμικό ενός χρήστη – οργανισμού στα endpoint μεταξύ λογισμικού και υλισμικού. Οι απειλές από εξωτερικούς κακόβουλους χρήστες θεωρητικά αφορά περισσότερο το δημόσιο Διαδίκτυο οπότε και το δημόσιο cloud. Παρόλ’ αυτά στην περίπτωση του ιδιωτικού cloud τα endpoints των χρηστών γίνονται πιο εύκολα στόχος. Όμως από τη στιγμή που οι πάροχοι διατηρούν ένα μεγάλο όγκο πληροφοριών με κάποιες να είναι ιδιαίτερα ευαίσθητες (προσωπικά δεδομένα, στοιχεία πιστωτικών καρτών κλπ) είναι σίγουρο ότι θα οι πληροφορίες αυτές θα γίνονται στόχος επιθέσεων τόσο στο λογισμικό όσο και στο υλισμικό τους.

- ✓ Διαρροή δεδομένων που οφείλεται σε αστοχία και αδυναμία διατήρησης ασφαλούς εισόδου και αστοχία είτε στα φυσικά είτε στα ηλεκτρονικά κανάλια μεταφοράς δεδομένων του cloud. Μια τέτοια απειλή, διαρροή δεδομένων δηλαδή, ίσως και μεγάλου εύρους, μπορεί να συμβεί είτε από ανθρώπινο σφάλμα είτε από αστοχία υλικού. Σε κάθε περίπτωση όμως θα βάλει σε κίνδυνο πολύ σημαντικές πληροφορίες.

3.5.2 ΑΚΕΡΑΙΟΤΗΤΑ

Η ακεραιότητα είναι η εγγύηση ότι τα δεδομένα προστατεύονται από τυχαία ή σκόπιμη τροποποίηση. Διασφαλίζει την ορθότητα, την εγκυρότητα και την πληρότητα των δεδομένων κατά τη φάση της εισαγωγής, της αποθήκευσης και της μεταφοράς τους. Όσον αφορά την ακεραιότητα των δεδομένων στο cloud υπάρχουν τρεις πιθανές ανησυχίες:

- ✓ Διαχωρισμός δεδομένων τόσο λόγω εσφαλμένου καθορισμού των παραμέτρων ασφαλείας όσο και λόγω εσφαλμένης διαμόρφωσης των εικονικών μηχανών και των hypervisors. Η ακεραιότητα των δεδομένων μέσα σε πολύπλοκα περιβάλλοντα cloud, κυρίως σε SaaS υποδομή, τα οποία διαμοιράζουν τους πόρους τους σε διάφορους πελάτες μπορεί να αποδειχθεί απειλή ενάντια στη ακεραιότητα των δεδομένων αν οι πόροι δεν διαχωρίζονται αποτελεσματικά.
- ✓ Η μη εφαρμογή ισχυρού ελέγχου εισόδου μπορεί να δημιουργήσει πολλές απειλές. Για παράδειγμα ένας απολυμένος υπάλληλος ενός οργανισμού ίσως να έχει τη δυνατότητα να διατηρήσει την απομακρυσμένη είσοδο στη διαχείριση των cloud υπηρεσιών του οργανισμού και να βλάψει έτσι τα δεδομένα του οργανισμού.
- ✓ Η αποθήκευση προβληματικών εφαρμογών ή προβληματικών εξαρτημάτων υποδομής ειδικά αν αυτά ζητηθούν από διάφορους χρήστες μπορούν να οδηγήσουν σε προβλήματα σχετικά με την ακεραιότητα των δεδομένων που παρέχονται και σε άλλους χρήστες.

3.5.3 ΔΙΑΘΕΣΙΜΟΤΗΤΑ

Υποθέτοντας ότι τα στοιχεία και τα δεδομένα που αφορούν έναν πελάτη διασφαλίζονται ως προς την εμπιστευτικότητα και ως προς την ακεραιότητά τους, ανησυχία θα πρέπει να υπάρχει και σχετικά με τη διαθεσιμότητα αυτών. Η διαθεσιμότητα αφορά στην άμεση πρόσβαση στις υπηρεσίες του συστήματος για τους νόμιμους χρήστες του. Υπάρχουν τρεις μεγάλες απειλές ως προς τη διαθεσιμότητα των δεδομένων οι οποίες είναι οι εξής:

- ✓ Η αλλαγή της διαχείρισης του λογισμικού και του υλισμικού σε υπάρχουσες υπηρεσίες cloud ή η καταστροφή μέρους της υπάρχουσας υποδομής.
- ✓ Η άρνηση της εφαρμογής (denial of service) είναι μια απειλή κατά την οποία οι επιτιθέμενοι, οι οποίοι συνήθως είναι εξωτερικές για το cloud απειλές χρησιμοποιούν τέτοιες επιθέσεις (denial of service) με στόχο να συντρίψουν μια εφαρμογή ώστε οι πελάτες να μην έχουν πρόσβαση στη συγκεκριμένη εφαρμογή.
- ✓ Οι μη ικανοποιητικές διαδικασίες ανάκτησης δεδομένων σε περίπτωση καταστροφής κάποιου μέρους της υποδομής καθώς και οι διαφορετικές διαδικασίες ανάμεσα σε πάροχο και ενοικιαστή της υποδομής μπορεί να οδηγήσει σε πολύ μεγάλη αύξηση του χρόνου ανάκτησης δεδομένων. Επομένως χάνεται η απαίτηση ενός χρήστη για άμεση πρόσβαση στα δεδομένα του.

ΚΕΦΑΛΑΙΟ 4

BITCOIN ΚΑΙ BLOCKCHAIN

Η οικονομική κρίση καθώς και ο φόβος-έλλειψη εμπιστοσύνης προς τα τραπεζικά συστήματα γέννησε την ανάγκη για τη δημιουργία ενός μέσου συναλλαγών το οποίο θα είναι ανεξάρτητο και δε θα βρίσκεται υπό την αιγίδα ή τον έλεγχο μιας κεντρικής αρχής. Διανύοντας μια περίοδο που συνεχώς νέα τεχνολογικά επιτεύγματα παρουσιάζονται, ήταν φυσικό να βρεθεί κάποιος που θα προσφέρει μια πιθανή εναλλακτική που θα σταθεί ισάξια απέναντι σε τραπεζικά ιδρύματα και θα προσελκύσει τον απλούς χρήστες. Η πρώτη ολοκληρωμένη πρόταση έγινε από τον software developer με το ψευδώνυμο Satoshi Nakamoto και την ομάδα του οι οποίοι παρουσίασαν το Bitcoin. [26]

4.1 ΤΙ ΕΙΝΑΙ ΤΟ BITCOIN

Το Bitcoin είναι ένα ψηφιακό νόμισμα και δεν υπάρχει επισήμως σε καμία φυσική μορφή κερμάτων ή χαρτονομισμάτων. Δεν παράγεται από κάποια συγκεκριμένη χώρα και φυσικά δεν ελέγχεται από καμία τράπεζα. Πρόκειται για ένα συναινετικό δίκτυο που προσφέρει τη δυνατότητα ενός νέου συστήματος πληρωμών και μιας εντελώς ψηφιακής μορφής χρημάτων. Χρησιμοποιώντας τεχνολογία peer-to-peer λειτουργεί ανεξάρτητα από κεντρικές αρχές και η διαχείριση των συναλλαγών καθώς και η έκδοση των Bitcoin πραγματοποιείται συλλογικά από το δίκτυο. Κανένας δεν κατέχει ή ελέγχει Bitcoin και ο καθένας μπορεί να πάρει μέρος στο δίκτυο αυτό. Παρέχει μοναδικές ιδιότητες και επιτρέπει λειτουργίες που δεν καλύπτονται προς το παρόν από άλλα συστήματα πληρωμών.

Ένα πολύ σημαντικό χαρακτηριστικό που αποτελεί και εχέγγυο για την επιτυχία του Bitcoin είναι ότι πρόκειται για λογισμικό ανοιχτού κώδικα (Open Source). Ενώ οι προγραμματιστές βελτιώνουν το λογισμικό, δεν μπορούν να εξαναγκάσουν καμία αλλαγή στο πρωτόκολλο Bitcoin, διότι όλοι οι χρήστες είναι ελεύθεροι να επιλέξουν την έκδοση του λογισμικού που χρησιμοποιούν. Για να διατηρηθεί η συμβατότητα, όλοι οι χρήστες πρέπει να χρησιμοποιούν το λογισμικό που υπακούει στους ίδιους κανόνες. Το

Bitcoin μπορεί να λειτουργήσει σωστά μόνο με την πλήρη συναίνεση μεταξύ όλων των χρηστών. Ως εκ τούτου, όλοι οι χρήστες και οι προγραμματιστές έχουν ισχυρό κίνητρο να προστατεύουν αυτήν την γενική συναίνεση.[26], [27]

4.2 ΠΟΙΟΣ ΔΗΜΙΟΥΡΓΗΣΕ ΤΟ BITCOIN

Το Bitcoin αποτελεί την πρώτη εφαρμογή μιας έννοιας που ονομάζεται «κρυπτονόμισμα» (crypto-currency). Η έννοια αυτή προτάθηκε εν μέρει το 1998 από τον Wei Dai στην λίστα αλληλογραφίας cypherpunks, υποστηρίζοντας την ιδέα μιας νέας μορφής χρήματος η οποία κάνει χρήση κρυπτογραφίας για να ελέγξει τη δημιουργία και τις συναλλαγές του, παρά μια κεντρική αρχή. Εντούτοις, χρειάστηκαν δέκα χρόνια μέχρι να υπάρξουν οι πρώτες ενδείξεις πρακτικής εφαρμογής αυτής της ιδέας. Το 2008, πιθανότατα με έμπνευση την οικονομική κρίση, έγινε η πρώτη επιστημονική δημοσίευση που περιέγραφε το bitcoin. Η πρώτη λειτουργική εμφάνιση και η κυκλοφορία του πρώτου open source client καθώς και η δημιουργία των αντίστοιχων Bitcoin έγινε το 2009. Ο συγγραφέας της δημοσίευσης και δημιουργός του αντίστοιχου λογισμικού παρέμεινε ανώνυμος, χρησιμοποιώντας το ψευδώνυμο Satoshi Nakamoto. Ο Satoshi αποσύρθηκε από το έργο αυτό στα τέλη του 2010 χωρίς να αποκαλύψει πολλά για τον εαυτό του. Από τότε, η κοινότητα του Bitcoin μεγάλωσε εκθετικά με πολλούς προγραμματιστές που ασχολούνται με το Bitcoin.[27]

4.3 ΛΕΙΤΟΥΡΓΙΑ BITCOIN

Από τη μεριά του χρήστη, το Bitcoin δεν είναι τίποτα περισσότερο από μια εφαρμογή κινητού τηλεφώνου ή υπολογιστή η οποία παρέχει ένα προσωπικό πορτοφόλι (Bitcoin wallet) και επιτρέπει στο χρήστη να στέλνει και να λαμβάνει bitcoins μέσω αυτού. Το πορτοφόλι αυτό μπορεί να βρίσκεται είτε σαν πρόγραμμα στον υπολογιστή μας, είτε να φιλοξενείται σε κάποια ιστοσελίδα, πχ ένα Bitcoin Exchange. Έτσι λειτουργεί το Bitcoin για τους περισσότερους χρήστες.

Ουσιαστικά, το σύνολο των bitcoin είναι μοιρασμένο στα Bitcoin wallets των ατόμων και των εταιρειών που συμμετέχουν στο δίκτυο του νομίσματος. Γι' αυτό λέμε πως το σύστημα είναι της μορφής peer to peer, [28], καθώς εξαρτάται από τους χρήστες (peers) και όχι από κάποιο κεντρικό server πχ μιας τράπεζας. Όταν γίνεται μια συναλλαγή (πχ μια αγοραπωλησία) με Bitcoin, το σύστημα επιβεβαιώνει τις διευθύνσεις των δύο διαφορετικών wallets, του αγοραστή και του πωλητή καθώς και την πραγματοποίηση της συναλλαγής.

Πρακτικά, το δίκτυο Bitcoin μοιράζεται ένα δημόσιο λογιστικό βιβλίο (ledger) που ονομάζεται "block chain" (αλυσίδα των μπλοκ). Αυτό το βιβλίο περιέχει κάθε συναλλαγή που έχει ποτέ λάβει χώρα, επιτρέποντας στον υπολογιστή του χρήστη να εξακριβώνει την εγκυρότητα της κάθε συναλλαγής. Η αυθεντικότητα της κάθε συναλλαγής προστατεύεται από ψηφιακές υπογραφές που αντιστοιχούν στις διευθύνσεις αποστολής, επιτρέποντας σε όλους τους χρήστες να έχουν πλήρη έλεγχο κατά την αποστολή bitcoins από τις δικές τους διευθύνσεις Bitcoin. Επιπλέον, ο καθένας μπορεί να επεξεργαστεί συναλλαγές και να τις επιβεβαιώσει κερδίζοντας έτσι μια ανταμοιβή σε bitcoins για την υπηρεσία αυτή. Οι χρήστες που συμμετέχουν στην επιβεβαίωση των συναλλαγών ονομάζονται miners και διαδραματίζουν πολύ σημαντικό ρόλο στο όλο σύστημα καθώς είναι οι εγγυητές της ορθής λειτουργίας του δικτύου και της ασφάλειας των συναλλαγών. [29]

4.4 ΣΥΝΑΛΛΑΓΕΣ ΜΕ BITCOIN

Οι συναλλαγές στέλνονται από και προς τα ηλεκτρονικά πορτοφόλια ενώ παράλληλα υπογράφονται ψηφιακά για λόγους ασφαλείας. Ο καθένας που συμμετέχει στο δίκτυο γνωρίζει για μία συναλλαγή που πραγματοποιήθηκε και μπορεί να εντοπίσει το σημείο όπου τα συγκεκριμένα Bitcoins δημιουργήθηκαν.

Το πιο περίεργο στοιχείο σχετικά με τα Bitcoin είναι ότι δεν υπάρχουν πρακτικά πουθενά, ούτε καν σε κάποιο σκληρό δίσκο ή μέσο αποθήκευσης. Αναφέρουμε ότι κάποιος κατέχει bitcoins αλλά αν ελέγξουμε το κάποια διεύθυνση bitcoin δε θα βρούμε ψηφιακά νομίσματα όπως ο καθένας μας τα έχει στο μυαλό του. Αντίθετα υπάρχουν μόνο καταγραφές από συναλλαγές μεταξύ διαφορετικών διευθύνσεων. Κάθε συναλλαγή

που πραγματοποιήθηκε οποτεδήποτε είναι αποθηκευμένη στο block chain. Αν θελήσει κάποιος να βρει το υπόλοιπο κάποιας διεύθυνσης bitcoin δε θα πρέπει να ψάξει στη διεύθυνση αυτή αλλά θα πρέπει να το ανακατασκευάσει ανατρέχοντας στο block chain.

Για να πραγματοποιηθεί μια συναλλαγή θα πρέπει να είναι γνωστά 3 πράγματα:

- Μια είσοδος (input). Πρόκειται για τη διεύθυνση από την οποία ο αποστολέας της συναλλαγής που μελετάμε δέχτηκε τα bitcoin.
- Το ποσό της συναλλαγής δηλαδή τον αριθμό των bitcoin που θα αποσταλλούν.
- Μια έξοδο (output). Πρόκειται για τη διεύθυνση bitcoin του δέκτη.

Για να στείλει κάποιος χρήστης bitcoin πρέπει να έχει 2 πράγματα, μία διεύθυνση Bitcoin και ένα private key. Η διεύθυνση Bitcoin δημιουργείται τυχαία και πρόκειται για μια απλή ακολουθία γραμμάτων και αριθμών. Το private key είναι επίσης μια ακολουθία από γράμματα και αριθμούς αλλά σε αντίθεση με τη διεύθυνση διατηρείται κρυφό.

Όταν ο αποστολέας θέλει να στείλει μερικά bitcoin σε κάποιον χρησιμοποιεί το private key του για να υπογράψει το μήνυμα της συναλλαγής που περιλαμβάνει τα 3 προαναφερθέντα στοιχεία. Αφού στείλει στο δίκτυο το μήνυμα αυτό πρέπει να περιμένει έως ότου οι miners πιστοποιήσουν τη συναλλαγή και την τοποθετήσουν σε ένα block μέσα στο block chain.

Επειδή κάθε συναλλαγή πρέπει να πιστοποιηθεί από τους miners του δικτύου μερικές φορές οι συμμετέχοντες σε μια συναλλαγή πρέπει να περιμένουν έως ότου τελειώνει η διαδικασία mining. Το πρωτόκολλο του bitcoin είναι φτιαγμένο με τέτοιο τρόπο ώστε να απαιτείται χρονικό διάστημα 10 λεπτών για τη διαδικασία.

Επειδή τα Bitcoins φαίνονται μόνο ως καταγεγραμμένες συναλλαγές μεταξύ διευθύνσεων είναι φυσικό ένας μεγάλος αριθμός από τέτοιες να είναι συνδεδεμένες με μια συγκεκριμένη διεύθυνση bitcoin. Δηλαδή κάθε χρήστης έχει δεχθεί ένα ποσό από περισσότερες από μία διευθύνσεις. Στην περίπτωση αυτή το σύνολο δεν αθροίζεται αλλά διατηρείται κάθε συναλλαγή ξεχωριστά. Όταν κάποιος χρήστης θελήσει να στείλει ένα ποσό το wallet της θα συνδυάσει προηγούμενες συναλλαγές των οποίων το ποσό αθροίζεται στο ποσό που θέλει να μεταφέρει στη νέα συναλλαγή. Στην περίπτωση που αυτό δεν είναι εφικτό χρησιμοποιείται μια προηγούμενη συναλλαγή όπου το ποσό είναι μεγαλύτερο από αυτό που θέλει ο χρήστης να στείλει και το output θα περιλαμβάνει

τόσο τη διεύθυνση του παραλήπτη με το ακριβές ποσό της συναλλαγής ενώ το υπόλοιπο ποσό αποστέλλεται σε μια άλλη διεύθυνση του αποστολέα που έχει δημιουργηθεί για το λόγο αυτό και ουσιαστικά επιστρέφουν πάλι στον ίδιο. [29]

4.5 BLOCK CHAIN

Οι χρήστες ανταλλάσσουν συνεχώς bitcoin μεταξύ τους αλλά μόνο αν κάποιος κρατά ένα αντίγραφο όλων των συναλλαγών είναι σε θέση να εντοπίσει ποιος έχει πληρώσει τι. Για το λόγο αυτό το δίκτυο του bitcoin συλλέγει όλες τις συναλλαγές που πραγματοποιήθηκαν σε ένα χρονικό διάστημα μέσα σε μια λίστα που ονομάζεται block. Είναι δουλειά των miners να επιβεβαιώσουν τις συναλλαγές και να τις καταγράψουν στο block chain.

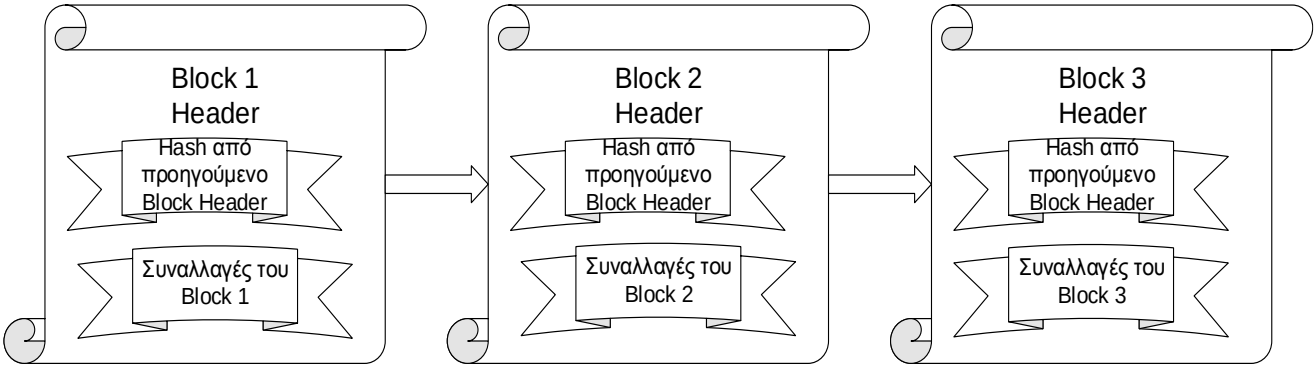
Κάθε φορά που ένα block συναλλαγών δημιουργείται, αυτόματα προστίθεται στο block chain δημιουργώντας έτσι μια πολύ μεγάλη λίστα όλων των συναλλαγών που έλαβαν χώρα. Ένα συνεχώς ανανεωμένο αντίγραφο των blocks δίνεται σε όσους συμμετέχουν στο δίκτυο έτσι ώστε να γνωρίζουν τι συμβαίνει με τις συναλλαγές.

Ένα δημόσιο βιβλίο όπως το block chain πρέπει να είναι αξιόπιστο και να το εμπιστεύονται όλοι. Πρέπει να μένει ακέραιο και να μη μπορεί κανείς να το τροποποιήσει. Αυτό το εξασφαλίζουν οι miners.

Κάθε φορά που δημιουργείται ένα μπλοκ συναλλαγών οι miners το υποβάλλουν σε μια διαδικασία. Λαμβάνουν το περιεχόμενο του, εφαρμόζουν ένα μαθηματικό μοντέλο σε αυτό και το μετατρέπουν σε κάτι άλλο. Πρόκειται για μια τυχαία ακολουθία γραμμάτων και αριθμών η οποία είναι πολύ μικρότερη σε μέγεθος από το αντίστοιχο μπλοκ. Η ακολουθία αυτή είναι γνωστή ως hash. Κάθε hash αποθηκεύεται μαζί με το αντίστοιχο μπλοκ του στο τέλος του block chain. [30]

Τα hashes έχουν μερικές πολύ ενδιαφέρουσες ιδιότητες. Είναι εύκολο να παραχθεί ένα hash από μια συλλογή δεδομένων όπως το μπλοκ αλλά είναι πρακτικά αδύνατο να καταλάβει κάποιος τι δεδομένα περιέχονται μέσα σε ένα hash. Επίσης κάθε hash είναι μοναδικό ενώ αν αλλαχθεί έστω και ένας χαρακτήρας από ένα μπλοκ το αντίστοιχο hash του αλλάζει εντελώς.

Οι miners δε χρησιμοποιούν μόνο τις συναλλαγές ενός μπλοκ για να παράγουν ένα hash αλλά χρειάζονται και κάποια επιπλέον δεδομένα. Ένα από αυτά είναι το hash από το τελευταίο μπλοκ που είναι αποθηκευμένο στο block chain. Επειδή το hash από κάθε μπλοκ δημιουργείται χρησιμοποιώντας δεδομένα από το hash προηγούμενου μπλοκ μπορούμε να χαρακτηρίσουμε το block chain ως ψηφιακά συμπαγές. Αν κάποιος προσπαθήσει να τροποποιήσει ένα μπλοκ όλοι θα το καταλάβουν αφού πλέον δε θα ταιριάζει με το αντίστοιχο μπλοκ του.



Εικόνα : 3 Αρχιτεκτονική block chain

Είναι εύκολο να ελεγχθεί η αυθεντικότητα ενός μπλοκ. Απλά κάποιος χρησιμοποιώντας τη συνάρτηση για την παραγωγή των hashes σε ένα παλιό μπλοκ το οποίο έχει τροποποιηθεί θα δει ότι το καινούργιο hash είναι διαφορετικό από το ήδη αποθηκευμένο. Στην περίπτωση αυτή το εν λόγω μπλοκ θα στιγματιστεί ως πλαστό. Στην περίπτωση αυτή βέβαια υπάρχει ο κίνδυνος να επηρεαστεί ολόκληρη η συνέχεια μέσα στο block chain καθώς κάθε hash από κάθε μπλοκ χρησιμοποιεί πληροφορία από το αμέσως προηγούμενο hash.

4.6 ΔΗΜΙΟΥΡΓΙΑ BITCOIN

Τα νέα bitcoins δημιουργούνται κατά τη διαδικασία του mining. Η διαδικασία αυτή περιλαμβάνει ότι τα άτομα επιβραβεύονται από το δίκτυο για τις υπηρεσίες τους. Η miners Bitcoin επεξεργάζονται συναλλαγές και ασφαλίζουν το δίκτυο και σε αντάλλαγμα συλλέγουν νέα bitcoins.

Το πρωτόκολλο bitcoin είναι σχεδιασμένο με τέτοιο τρόπο ώστε νέα bitcoins να δημιουργούνται με σταθερό ποσοστό. Αυτό καθιστά την εξόρυξη (mining) Bitcoin μια πολύ ανταγωνιστική επιχείρηση. Όταν περισσότεροι miners συμμετέχουν στο δίκτυο, γίνεται όλο και περισσότερο δύσκολο να βγει κέρδος και miners πρέπει να αναζητήσουν αποδοτικότητα για να περιορίσουν τα λειτουργικά κόστη. Καμία κεντρική εξουσία ή προγραμματιστής δεν έχει τη δύναμη να ελέγξει ή να κατευθύνει το σύστημα για να αυξήσει τα κέρδη τους. Κάθε κόμβος Bitcoin ανά τον κόσμο θα απορρίψει καθετί που δεν συμμορφώνεται με τους κανόνες που το σύστημα περιμένει να ακολουθήσει.

Τα bitcoins δημιουργούνται με μειούμενο και προβλέψιμο ρυθμό. Ο αριθμός νέων bitcoins που δημιουργείται κάθε χρόνο αυτόματα μειώνεται στο μισό προοδευτικά μέχρι η έκδοση bitcoin να σταματήσει εντελώς όταν υπάρχουν σε σύνολο 21 εκατομμύρια bitcoins.

Αυτό βέβαια δεν αποτελεί περιορισμό διότι οι συναλλαγές μπορούν να έχουν ονομαστική αξία με μικρότερες υπομονάδες ενός bitcoin, όπως τα bits - υπάρχουν 1,000,000 bits σε ένα 1 bitcoin. Τα bitcoins μπορούν να διαιρεθούν μέχρι σε 8 δεκαδικά ψηφία (0.000 000 01) και ενδεχομένως σε ακόμα μικρότερες μονάδες αν αυτό ποτέ χρειαζόταν στο μέλλον καθώς ο μέσος όρος του μεγέθους μιας συναλλαγής μειώνεται. [27]

ΚΕΦΑΛΑΙΟ 5

ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΥΣΤΗΜΑΤΟΣ

Σε προηγούμενα κεφάλαια μελετήσαμε και παρουσιάσαμε το cloud computing, το fog computing που αποτελεί κατά μία έννοια επέκταση του πρώτου καθώς και τη μεταφορά πόρων στο άκρο του δικτύου (edge). Επίσης αναφέραμε οι παραπάνω τεχνολογίες σε συνδυασμό με την κατακόρυφη αύξηση του αριθμού των συσκευών που μπορούν να συνδεθούν πλέον στο διαδίκτυο προσφέρουν τη δυνατότητα να ζούμε σε έξυπνα σπίτια (smart homes) και έξυπνες πόλεις (smart cities) απολαμβάνοντας όσα πλεονεκτήματα μας προσφέρει το Internet of Things.

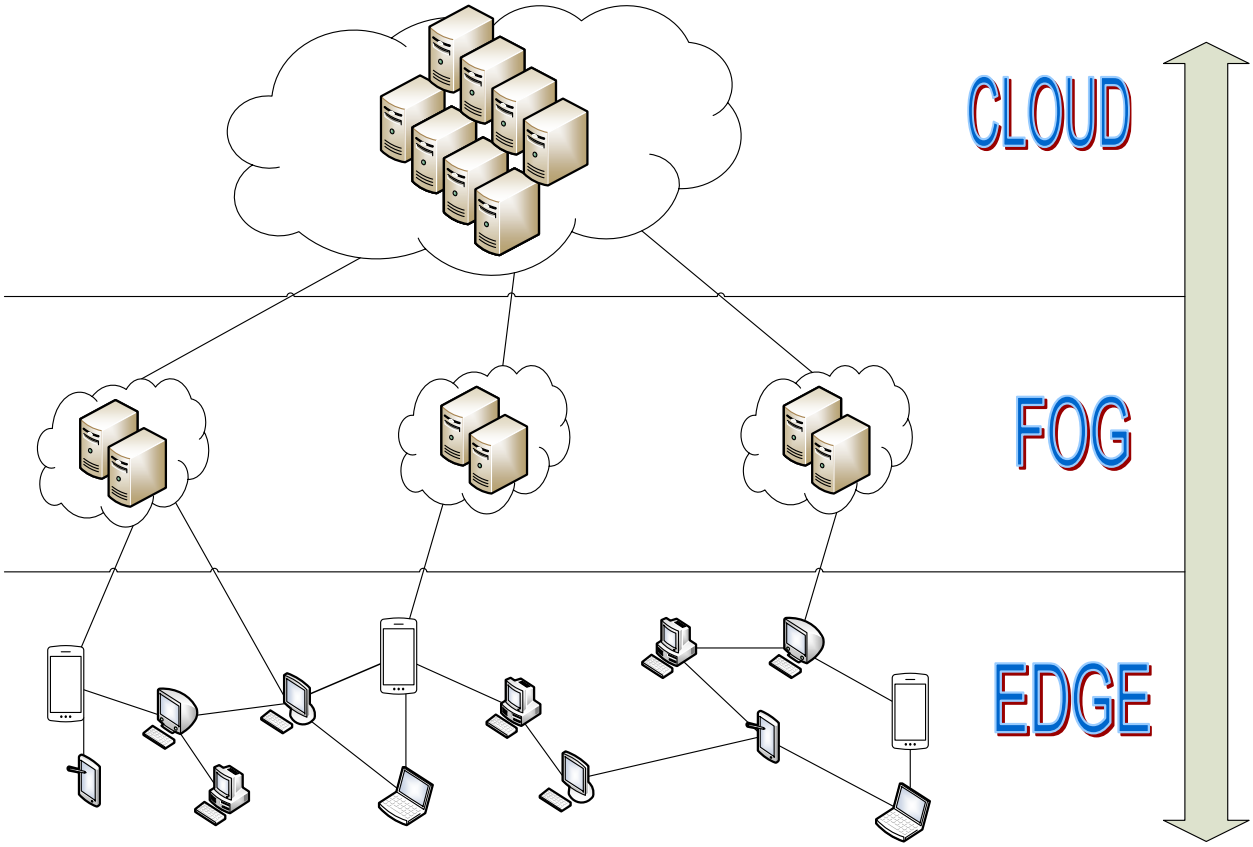
Επίσης παρουσιάσαμε το πρώτο ευρέως διαδεδομένο και χρησιμοποιούμενο ψηφιακό νόμισμα, το bitcoin. Είδαμε πως λειτουργεί, πως γίνονται οι συναλλαγές και ποιά είναι η βασική τεχνολογία που χρησιμοποιείται για την επικύρωση των συναλλαγών με το νόμισμα αυτό (ύπαρξη miner) και πως εξασφαλίζεται η διαφάνεια (block chain).

Στο κεφάλαιο αυτό συνδυάζουμε τις δύο παραπάνω τεχνολογίες. Χρησιμοποιώντας τις βασικές ιδέες του ψηφιακού αυτού νομίσματος προτείνουμε ένα αντίστοιχο μοντέλο στο cloud και πιο συγκριμένα στο άκρο του δικτύου, το οποίο δύναται να διασφαλίζει ανταλλαγές δεδομένων, ως προς την ποιότητα τους, μεταξύ συσκευών που πολλές φορές είναι μεταξύ τους άγνωστες. Για την πιστοποίηση των συναλλαγών γίνεται έλεγχος των συσκευών και όχι των δεδομένων. Επιχειρούμε δηλαδή να αναπτύξουμε ένα μοντέλο με στόχο οι ανταλλαγές δεδομένων να γίνονται με περισσότερη διαφάνεια η οποία διασφαλίζεται με τον προκαταβολικό έλεγχο των συσκευών ως προς την αξιοπιστία τους με στόχο να παρέχεται μεγαλύτερη ασφάλεια και σιγουριά στον παραλήπτη των δεδομένων.

5.1 ΠΑΡΑΔΟΧΕΣ

Για την ανάπτυξη του συστήματος έγιναν κάποιες βασικές παραδοχές. Αυτές θα παρουσιασθούν σε πρώτη φάση συνοπτικά και στη συνέχεια κατά την παρουσίαση της αρχιτεκτονικής θα αναλυθούν περαιτέρω για καλύτερη κατανόηση του συνόλου.

Το σύστημα που παρουσιάζουμε αναπτύχθηκε για μια δομή όπως αυτή που φαίνεται στην εικόνα που ακολουθεί.



Εικόνα : 4 Δομή του υπό μελέτη δικτύου

Στο παραπάνω σχήμα παρατηρούμε πως εμφανίζεται η δομή ενός δικτύου το οποίο περιλαμβάνει 3 επίπεδα.

- Το ανώτερο επίπεδο στο οποίο συναντάμε το Cloud με τους servers που ανήκουν σε αυτό και αποτελεί τον πυρήνα του συστήματος.
- Το μεσαίο επίπεδο περιλαμβάνει κάποιους micro-servers που φέρνουν πιο κοντά τους servers στο Cloud με τις συσκευές που βρίσκονται στο edge. Εκεί αποθηκεύονται δεδομένα και εφαρμογές που χρησιμοποιούνται πιο συχνά ώστε να είναι πιο εύκολα προσβάσιμα σε λιγότερο χρόνο (low latency).

- Στο πιο χαμηλό επίπεδο συναντάμε τις συσκευές που συνδέονται στο διαδίκτυο και για τις οποίες ουσιαστικά αναπτύσσουμε το σύστημα. Οπότε κατά βάση θα μας απασχολήσει το τελευταίο επίπεδο.

Η πρώτη και βασικότερη παραδοχή είναι πως για να γίνει μια ανταλλαγή δεδομένων και να απαιτείται η επιβεβαίωση αυτής πρέπει να υπάρχουν:

a. Ένας ή περισσότεροι χρήστες/συσκευές που θέλουν να στείλουν δεδομένα (sender)
b. Ένας ή περισσότεροι χρήστες/συσκευές που θέλουν να λάβουν δεδομένα (receiver)
c. Ένας χρήστης/συσκευή που θα λειτουργήσει ως miner για ένα συγκεκριμένο χρονικό διάστημα ή έως ότου θελήσει να συμμετάσχει σε μια συναλλαγή.

Οι δύο πρώτες περιπτώσεις υπάρχουν ούτως ή άλλως σε κάθε δίκτυο και είναι ο βασικός λόγος που αυτό υπάρχει. Βέβαια με την εξέλιξη του Internet of Things πολλές φορές κάποιες συσκευές χρησιμοποιούνται απλά και ως ενδιάμεσοι για τη μεταφορά δεδομένων, σα δίαυλοι για την επικοινωνία συσκευών που λειτουργούν με διαφορετικά πρωτόκολλα.

Κάθε ένας από τα παραπάνω μέλη προχωρά σε μια σειρά ενεργειών κατά τη μεταφορά δεδομένων. Τον πιο σημαντικό ρόλο φυσικά διαδραματίζει η τρίτη συσκευή που ουσιαστικά αποτελεί τη νέα προσθήκη σε ένα δίκτυο. Αναλυτικά οι ενέργειες που πραγματοποιούνται παρουσιάζονται στη συνέχεια ενώ αναλύεται επίσης ο τρόπος με τον οποίο πραγματοποιείται η αποστολή δεδομένων, η λήψη και η επιβεβαίωση της συναλλαγής.

Για να αξιοποιηθεί πλήρως το προτεινόμενο μοντέλο πρέπει κάποιες από τις συσκευές-μέλη του συστήματος είτε να είναι άγνωστες μεταξύ τους, δηλαδή να είναι άγνωστη η αξιοπιστία της κάθε συσκευής, είτε να μη μπορούν να επικοινωνήσουν απευθείας (π.χ. διαφορετικό πρωτόκολλο) και να απαιτείται η ύπαρξη τρίτης συσκευής για να γεφυρώσει αυτό το κενό. Βέβαια το μοντέλο αυτό εφαρμόζεται και σε μια απλή ανταλλαγή δεδομένων μεταξύ δύο ή περισσότερων «φιλικών» συσκευών, δηλαδή μεταξύ χρηστών που γνωρίζει ο παραλήπτης τι ακριβώς περιμένει από τον αποστολέα και φυσικά γνωρίζει το κατά πόσο αυτός είναι έμπιστος. Στην περίπτωση αυτή μάλιστα γίνεται και επιβεβαίωση της σωστής λειτουργίας του μοντέλου.

Κάθε συσκευή που είναι καταγεγραμμένη ότι συμμετέχει στο δίκτυο χαρακτηρίζεται από ένα επίπεδο αξιοπιστίας (trust level). Το μέγεθος χαρακτηρίζει ως αξιόπιστη ή μη κάθε συσκευή. Το trust level θα καταγράφεται στη συναλλαγή μαζί με το ID της κάθε συσκευής και είναι το βασικό στοιχείο που θα ελέγχει ο miner για να προχωρήσει η συναλλαγή. Ταυτόχρονα, με κριτήριο το trust level θα γίνεται και η επιλογή της συσκευής που θα εκτελεί χρέη miner. Προφανώς κάθε φορά θα επιλέγεται η πιο αξιόπιστη συσκευή.

Βασικό κομμάτι στο σύστημά μας αποτελεί μια distributed βάση δεδομένων (κατ' αντιστοιχία με το block chain στο bitcoin) όπου καταγράφονται ο/οι αποστολέας/εις με το επίπεδο αξιοπιστίας που τους χαρακτηρίζει, ο/οι δέκτης/ες καθώς και πληροφορίες για τη χρονική στιγμή που πραγματοποιήθηκε η αποστολή και λήψη των δεδομένων. Επιλέγεται ο συγκεκριμένος τύπος βάσης για τα πλεονεκτήματα που προσφέρει. [31].Τη βάση δεδομένων μπορεί να τη δει οποιοσδήποτε χρήστης και ας μη λειτουργεί ως miner τη δεδομένη χρονική στιγμή.

Επιπλέον, κάθε συσκευή έχει ένα μοναδικό προφίλ – ID (αντίστοιχα με το bitcoin address) το οποίο τη χαρακτηρίζει και καταγράφεται στο block chain σε κάθε συναλλαγή της συσκευής. Θεωρούμε ID διαφορετικό της IP κάθε συσκευής. Το ID για κάθε συσκευή/χρήστη θα δίνεται από το κεντρικό σύστημα (Cloud Server) και σε κάθε νέα συσκευή/χρήστη που μπαίνει στο σύστημά εκχωρείται ένα νέο ID. Με τον τρόπο αυτό όλες οι συσκευές θα είναι εύκολα αναγνωρίσιμες και μοναδικές. Επίσης αυτό το ID θα εμφανίζεται στις συναλλαγές τόσο για τον αποστολέα όσο και για το δέκτη.

Σε αντίθεση με ό, τι ισχύει στο Bitcoin στην περίπτωση μας θα υπάρχει απλή καταγραφή **της ανταλλαγής δεδομένων και ο miner δε θα είναι επιφορτισμένος με την δημιουργία των hashes αλλά ο ρόλος του θα είναι πιο απλός**.[30]. Θα ανατρέχει στην τελευταία αποστολή δεδομένων του αποστολέα και θα ελέγχει το επίπεδο αξιοπιστίας του. Στη συνέχεια θα επιτρέπει ή θα απορρίπτει την ανταλλαγή ανάλογα αυτό. Θέτουμε δηλαδή ένα συγκεκριμένο threshold πάνω από το οποίο ο αποστολέας θεωρείται αξιόπιστος και η ανταλλαγή μπορεί να προχωρήσει. Σε αντίθετη περίπτωση ο miner δεν επιτρέπει να επιτευχθεί η μεταφορά των δεδομένων. Ταυτόχρονα με τον έλεγχο του επίπεδου αξιοπιστίας του αποστολέα και προτού επιτρέψει ή όχι την μεταφορά δεδομένων ενημερώνει το δέκτη για το επίπεδο αξιοπιστίας του αποστολέα.

Μετά από κάθε αποστολή και λήψη δεδομένων ο δέκτης θα στέλνει μια θετική ή αρνητική αξιολόγηση ανάλογα με την ποιότητα δεδομένων που έλαβε. Με βάση αυτή την αξιολόγηση ο miner θα μεταβάλλει ανάλογα το επίπεδο αξιοπιστίας του αποστολέα στη βάση δεδομένων και συγκεκριμένα στην τελευταία καταγραφή στην οποία συμμετείχε.

Τη ιδιότητα του miner μπορεί λάβει οποιαδήποτε συσκευή στο σύστημα με την προϋπόθεση ότι έχει επίπεδο αξιοπιστίας μεγαλύτερο από τις υπόλοιπες συσκευές του cluster και μεγαλύτερο από ένα κατώτατο επίπεδο αξιοπιστίας (threshold).

Κάθε συσκευή που μπαίνει στο σύστημα για πρώτη φορά, μαζί με το ID παίρνει και ένα αρχικό επίπεδο αξιοπιστίας. Αυτό συμβαίνει διότι όταν η συσκευή αυτή θα προσπαθήσει να στείλει δεδομένα, ο miner θα κληθεί να εντοπίσει για τη συσκευή αυτή μια συναλλαγή αποθηκευμένη στη βάση για να δει το επίπεδο αξιοπιστίας αυτής.

Τα δύο βασικά συστήματα που θα μας απασχολήσουν στη συνέχεια είναι τα παρακάτω:

1) Το πρώτο περιλαμβάνει την εν γένει δομή του συστήματος για την ανάπτυξη εμπιστοσύνης και επιβεβαίωσης αποστολής δεδομένων μεταξύ πιστοποιημένων μελών ενός fog community.

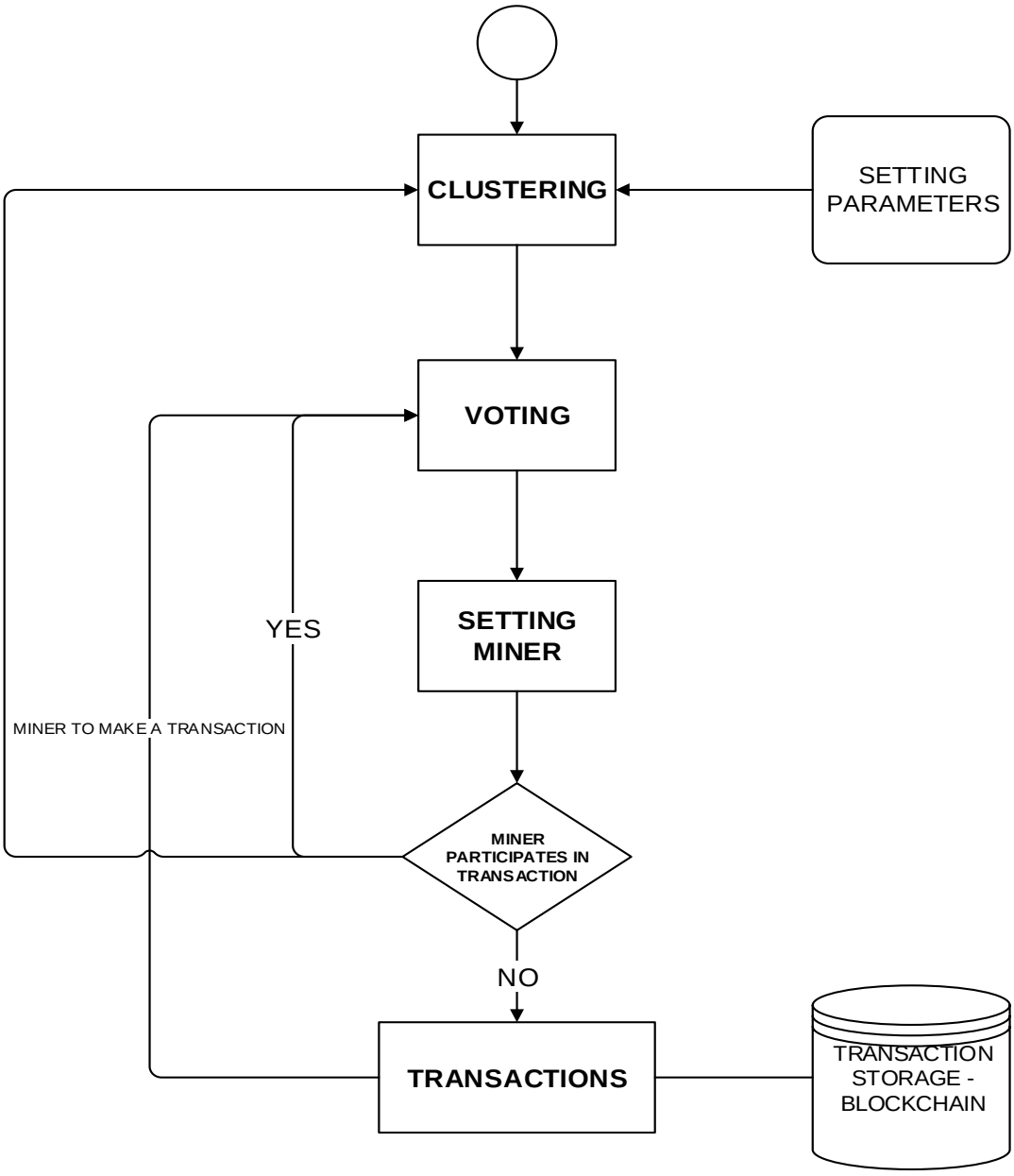
2) Το δεύτερο περιλαμβάνει τις ενέργειες που γίνονται κατά τη διάρκεια της αποστολής δεδομένων από όλα τα εμπλεκόμενα μέλη δηλαδή με την αρχιτεκτονική της συναλλαγής.

5.2 ΔΟΜΗ ΤΗΣ ΠΡΟΤΕΙΝΟΜΕΝΗΣ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ

Για την επίτευξη του στόχου της εργασίας αναπτύχθηκε μια αρχιτεκτονική που αποτελείται από μια απλή σειρά διαδικασιών-εργασιών οι οποίες κατά βάση εκτελούνται σειριακά ενώ όπου απαιτείται υπάρχει και η επιστροφή σε προηγούμενη εργασία. Μια γενική εικόνα της αρχιτεκτονικής φαίνεται στο παρακάτω σχήμα (εικόνα 5).

Με την πρώτη ματιά επιβεβαιώνεται η σειριακή δομή με τις απαιτούμενες ανατροφοδοτήσεις. Παρατηρούμε πως ο χρήστης που ξεκινά τη λειτουργία του εν λόγω συστήματος θέτει κάποιες παραμέτρους. Οι παράμετροι αυτοί σχετίζονται με τον τρόπο που θα ομαδοποιηθούν οι συσκευές, το επίπεδο αξιοπιστίας που θεωρείται επαρκές να

έχει μια συσκευή για να είναι σε θέση να στείλει δεδομένα και το χρονικό διάστημα που κάθε συσκευή θα λειτουργεί ως miner από τη στιγμή που θα επιλεγεί.

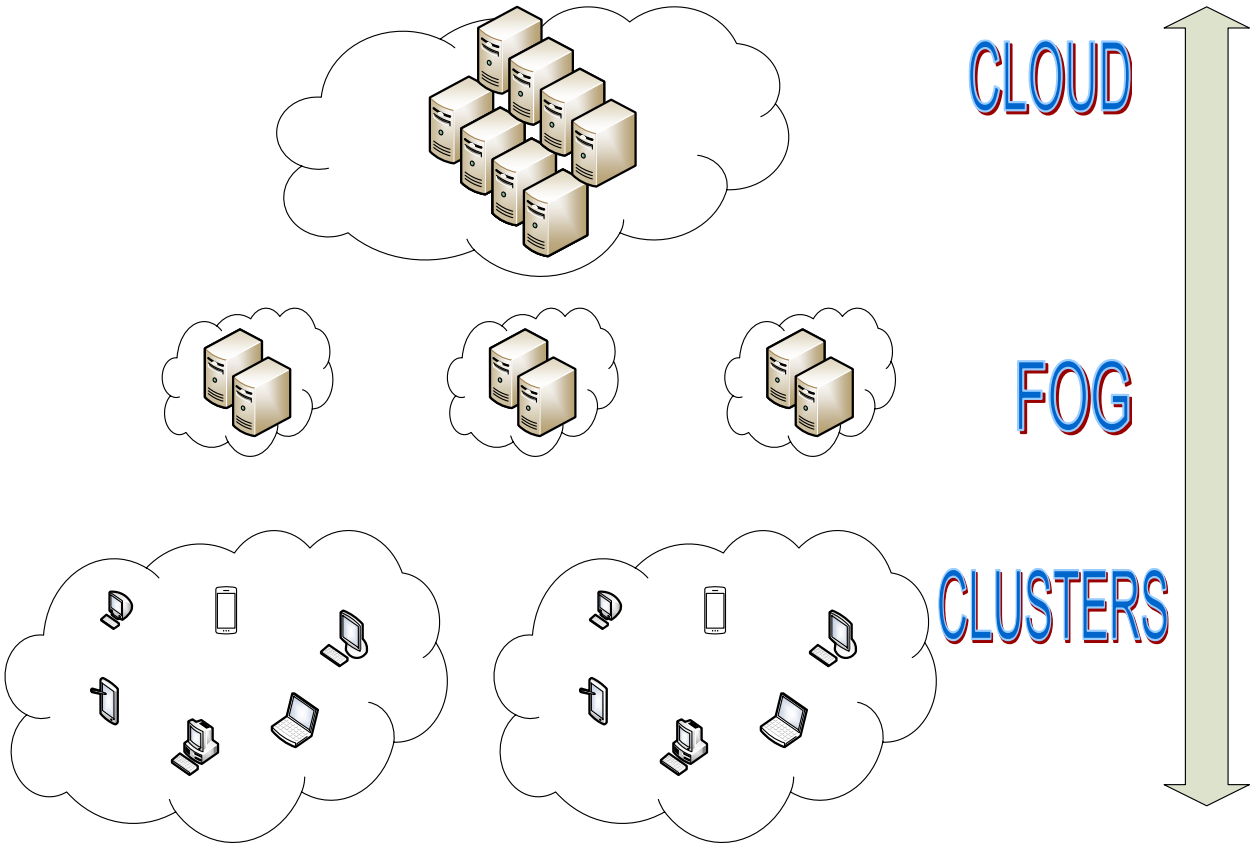


Εικόνα 5: Διάγραμμα ροής της προτεινόμενης αρχιτεκτονικής

Στην αρχή της διαδικασίας απαιτείται να γίνει πρώτα μια ομαδοποίηση (clustering) των συσκευών που βρίσκονται στο edge με στόχο την αποδοτικότερη μεταξύ τους επικοινωνία και τη διευκόλυνση των miner στην αναζήτηση και επιβεβαίωση του

επιπέδου αξιοπιστίας όσων συμμετέχουν σε ανταλλαγές δεδομένων. Αυτό επιτυγχάνεται με χρήση clustering αλγορίθμων και σα βασικό κριτήριο για την ομαδοποίηση χρησιμοποιούνται οι συντεταγμένες της κάθε συσκευής. Ένα cluster επομένως θα αποτελέσουν συσκευές που είναι σε κοντινή απόσταση μεταξύ τους. Φυσικά η ομαδοποίηση που θα γίνει δε είναι παντοτινή. Αντίθετα απαιτείται πολλές φορές να ξαναγίνει ομαδοποίηση συσκευών. Επίσης κάθε φορά που τρέχει το σύστημα από την αρχή, μπορούμε να μεταβάλλουμε τις παραμέτρους ως προς τις οποίες θα γίνεται το clustering των συσκευών.

Έστω λοιπόν ότι έγινε με επιτυχίας η ομαδοποίηση των συσκευών. Η νέα δομή του συστήματος θα είναι η παρακάτω:



Εικόνα : 6 Δομή του υπό μελέτη δικτύου μετά την ομαδοποίηση

Στην εικόνα αυτή φαίνεται η δομή που παρουσιάσαμε στην αρχή του κεφαλαίου με τη διαφορά ότι στο edge έχουν δημιουργηθεί clusters με έναν αριθμό συσκευών στο

καθένα. Αφού λοιπόν έγινε η ομαδοποίηση σειρά έχει η ανάδειξη της συσκευής που θα εκτελέσει τα καθήκοντα του miner.

Ο τρόπος με τον οποίο επιλέγεται ο miner σε κάθε cluster έχει να κάνει με το επίπεδο αξιοπιστίας των συσκευών που ανήκουν στο cluster. Ουσιαστικά επιλέγεται η πιο έμπιστη συσκευή, δηλαδή αυτή με το υψηλότερο επίπεδο αξιοπιστίας στο cluster. Η διαδικασία για την ανάδειξη του miner τρέχει ξεχωριστά σε κάθε cluster. Από τις συσκευές δηλαδή που ανήκουν σε ένα συγκεκριμένο cluster θα επιλεγεί αυτή με το μεγαλύτερο επίπεδο αξιοπιστίας να λειτουργήσει ως miner. Θα βρεθεί αρχικά μέσα στη βάση δεδομένων η τελευταία αποστολή δεδομένων που πραγματοποίησε. Μέσα σε αυτή την καταγραφή φαίνεται και το επίπεδο αξιοπιστίας αυτής. Θα γίνει δηλαδή σύγκριση των τιμών αυτών και η συσκευή με την πιο υψηλή τιμή θα πάρει την ιδιότητα του miner. Κάθε φορά που τελειώνει το χρονικό διάστημα που μια συσκευή λειτουργεί ως miner, η συσκευή αυτή επιλέγει με την παραπάνω διαδικασία αυτήν που θα τη διαδεχτεί και θα συνεχίσει να πιστοποιεί τις ανταλλαγές δεδομένων μεταξύ των υπόλοιπων συσκευών. Μετά και το βήμα αυτό ξεκινάει η ανταλλαγή δεδομένων.

Ο miner σε κάθε cluster αλλάζει σε 2 συγκεκριμένες περιπτώσεις:

1) Εάν παρέλθει ένα χρονικό διάστημα t ορισμένο από το χρήστη του συστήματος.

2) Εάν η συσκευή που λειτουργεί ως miner θελήσει να στείλει δεδομένα σε μία άλλη συσκευή.

Όπως βλέπουμε και στην εικόνα 5 υπάρχει και ένα βέλος που ξεκινά από το σημείο όπου γίνεται έλεγχος κατά πόσο ο miner που επιλέχθηκε θέλει να συμμετάσχει σε ανταλλαγή δεδομένων. Σε περίπτωση που υπάρχει ένας αριθμός αποτυχημένων επιλογών miner (όπου αποτυχημένη επιλογή θεωρούμε το να θέλει να συμμετάσχει σε αποστολή/λήψη δεδομένων) τότε απαιτείται να γίνει εκ νέου ομαδοποίηση.

Αυτή ήταν η γενική δομή του συστήματος. Καταγράφηκε η συνολική εικόνα χωρίς να έχουμε αναφερθεί στις ενέργειες που γίνονται μέσα σε κάθε βήμα. Θα παρουσιάσουμε επομένως στη συνέχεια το πώς υλοποιείται η μεταφορά δεδομένων από μια συσκευή σε μια άλλη, πως δηλαδή υλοποιείται μια αποστολή και λήψη δεδομένων, πως επεμβαίνει ο miner στη διαδικασία και τι ενέργειες κάνουν τα εμπλεκόμενα μέλη.

5.3 Η ΔΙΑΔΙΚΑΣΙΑ ΤΗΣ ΣΥΝΑΛΛΑΓΗΣ

Για να γίνει πιο ολοκληρωμένο και κατανοητό το μοντέλο θα πρέπει να γίνει επεξήγηση και της διαδικασίας της αποστολής και λήψης δεδομένων. Τι ενέργειες κάνει ο αποστολέας, τι ενέργειες κάνει ο παραλήπτης και τί ο miner. Και φυσικά τι περιλαμβάνει μια «συναλλαγή», δηλαδή μια ανταλλαγή δεδομένων, πως αυτή αποθηκεύεται στη βάση δεδομένων (block chain) και γιατί αυτό είναι χρήσιμο.

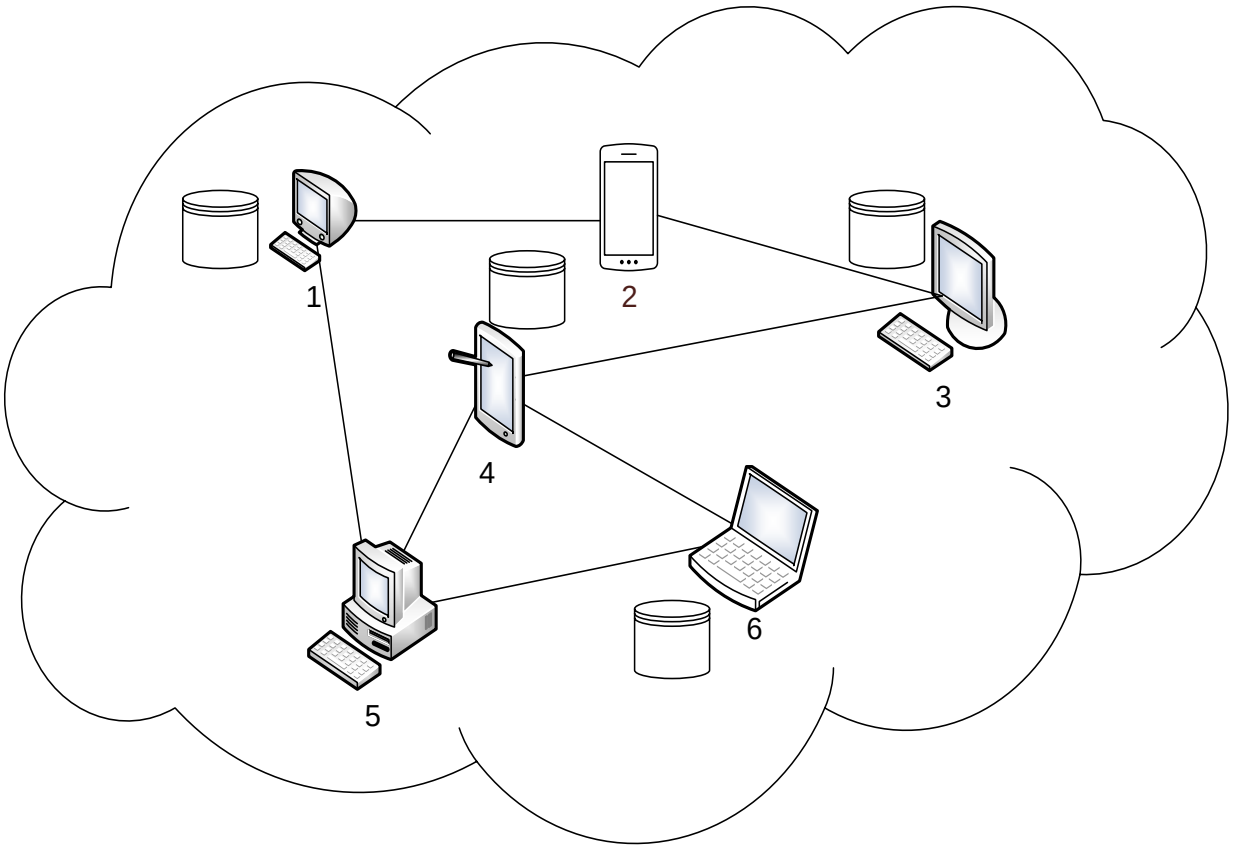
Κάθε φορά που πραγματοποιείται μια ανταλλαγή δεδομένων προστίθεται μια καταγραφή, μια σειρά με τα στοιχεία αυτής στη βάση δεδομένων. Η καταγραφή αυτή περιλαμβάνει τα παρακάτω στοιχεία:

1) Το χρονικό σημείο της αποστολής των δεδομένων
2) Το ID του αποστολέα
3) Το επίπεδο αξιοπιστίας του αποστολέα
4) Το ID του παραλήπτη

Συνοπτικά φαίνεται ως εξής:

Timestamp	$ID_{transmiter}$	$Trust\_level_{transm}$	$ID_{receiver}$

Κάθε φορά που μια τέτοια ανταλλαγή δεδομένων λαμβάνει χώρα, αυτόματα καταγράφεται και στη βάση δεδομένων σε χρονολογική σειρά. Έστω λοιπόν ότι είμαστε στο σημείο όπου έχει γίνει ομαδοποίηση συσκευών και έχει επιλεγεί ο miner. Η δομή του συστήματος το χρονικό αυτό σημείο αναπαρίσταται με την παρακάτω εικόνα (εικόνα 7).



Εικόνα : 7 Παράδειγμα ενός cluster του δικτύου

Στο σχήμα βλέπουμε κατ' αρχάς ένα cluster, τις συσκευές που βρίσκονται μέσα σε αυτό και μια αναπαράσταση της βάσης δεδομένων. Κάθε συσκευή μπορεί να δει, να κατεβάσει και να ελέγξει τη βάση αυτή. Αυτός είναι άλλωστε ο τρόπος με τον οποίο εξασφαλίζεται η διαφάνεια μεταξύ των μελών. Τονίζεται ότι στο σχήμα οι συνδέσεις έχουν τοποθετηθεί τυχαία. Στην πραγματικότητα όλοι μπορούν να επικοινωνήσουν με όλους.

Έστω λοιπόν ότι ο 4 έχει επιλεγεί ως miner. Επίσης θεωρούμε ότι ο 5 θέλει να στείλει δεδομένα στον 6. Αρχικά θα έρθει στο miner (4) ειδοποίηση ότι 2 συσκευές θέλουν να ανταλλάξουν δεδομένα. Αυτός με τη σειρά του θα εντοπίσει το ID του αποστολέα (5) και αφού το βρει ανατρέχει στη βάση δεδομένων για να βρει την τελευταία αποστολή δεδομένων στην οποία αυτός συμμετείχε. Αφού λοιπόν την εντοπίσει ελέγχει το επίπεδο αξιοπιστίας που είναι καταγεγραμμένο σε αυτή για τον αποστολέα.

Στο σημείο αυτό πρέπει να τονιστεί ότι θα μπορούσε να μπει μια πληθώρα επιλογών ώστε να κάνει πιο αντικειμενικό τον τρόπο αξιολόγησης της αξιοπιστίας. Θα μπορούσε

για παράδειγμα να λαμβάνει υπόψη παραπάνω από μια καταγεγραμμένες συναλλαγές καθώς επίσης και το χρονικό διάστημα που αυτές πραγματοποιήθηκαν. Αλλά σε αυτή την πρώτη φάση λειτουργούμε με την πιο απλή προσέγγιση λαμβάνοντας υπόψη την τελευταία συναλλαγή.

Αφού λοιπόν ελέγξει το επίπεδο αξιοπιστίας του αποστολέα, ενημερώνει σχετικά το δέκτη που θα λάβει τα δεδομένα και εγκρίνει ή απορρίπτει τη μεταφορά. Αν το επίπεδο αξιοπιστίας είναι πάνω από το ορισμένο κατώφλι τότε την εγκρίνει. Σε αντίθετη περίπτωση την απορρίπτει. Σε περίπτωση που τελικά εγκριθεί, καταγράφεται στη βάση δεδομένων.

Παράλληλα αφού ο δέκτης πάρει όλα τα δεδομένα που περιμένει, στέλνει μια θετική ή αρνητική αξιολόγηση για τον αποστολέα. Ο miner διαβάζει την αξιολόγηση και ανάλογα με αυτή μεταβάλλει (αυξάνει ή μειώνει) το trust level του αποστολέα στη βάση δεδομένων και συγκεκριμένα στη συναλλαγή που μόλις πραγματοποιήθηκε. Με τον τρόπο αυτόν επιταχύνεται η διαδικασία της εύρεσης του επιπέδου αξιοπιστίας του αποστολέα την επόμενη φορά που θα χρειαστεί να γίνει αναζήτηση αυτού.

Έτσι γίνεται ένα φιλτράρισμα των συσκευών και αυτές που δεν πληρούν τα κριτήρια εμπιστοσύνης δε μπορούν να συμμετέχουν ισότιμα στο δίκτυο. Ασφαλώς θα υπάρξουν και περιπτώσεις που κάποιες συσκευές θα έχουν επίπεδο αξιοπιστίας χαμηλότερο από το κατώφλι που ορίζουμε. Και εφόσον είναι σε αυτή την κατάσταση δε θα μπορέσουν ποτέ να στείλουν δεδομένα καθώς δε θα τους επιτρέπεται από το miner και δε θα έχουν δυνατότητα να αυξήσουν το επίπεδο αξιοπιστίας τους. Σε αυτή την περίπτωση ο miner είναι αυτός που θα κληθεί να δώσει λύση. Ζητάει μικρά κομμάτια δεδομένων από αυτές τις συσκευές, έπειτα αξιολογεί ο ίδιος αυτά τα δεδομένα και σταδιακά αυξάνει το επίπεδο αξιοπιστίας του. Κάποια στιγμή επομένως θα ξεπεράσει το κατώφλι και θα ξανασυμμετάσχει ομότιμα στο δίκτυο των ανταλλαγών.

ΚΕΦΑΛΑΙΟ 6

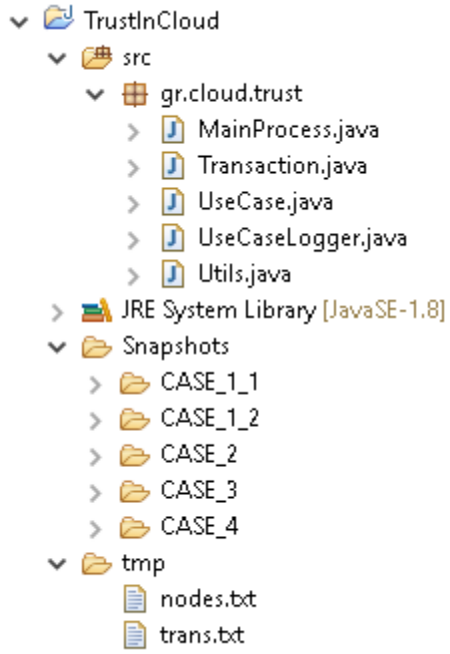
ΥΛΟΠΟΙΗΣΗ - ΑΠΟΤΕΛΕΣΜΑΤΑ

6.1 ΠΕΡΙΓΡΑΦΗ ΤΗΣ ΔΙΑΔΙΚΑΣΙΑΣ ΥΛΟΠΟΙΗΣΗΣ

Στο κεφάλαιο αυτό παρουσιάζεται μια υλοποίηση της αρχιτεκτονικής που συζητήθηκε προηγουμένως. Αναπτύχθηκε ένα πρόγραμμα στο eclipse με χρήση java που προσομοιώνει τη διαδικασία που μελετάμε. Φυσικά λόγω έλλειψης πραγματικών δεδομένων η υλοποίηση έγινε με δεδομένα που δημιουργούνται κατά την εκτέλεση του προγράμματος με ψευδοτυχαίο τρόπο. Τα δεδομένα που δημιουργούνται και χρησιμοποιούνται για την παραγωγή των αποτελεσμάτων είναι τα παρακάτω:

- ✓ Συσκευές (nodes) που θεωρούμε ότι ανήκουν στο άκρο του δικτύου (edge),
- ✓ Συσκευές (nodes) που ανήκουν στην ίδια ομάδα (cluster) μέσα στο δίκτυο,
- ✓ Καταγραφές με ανταλλαγές δεδομένων που έχουν πραγματοποιηθεί πριν την έναρξη της μελέτης,
- ✓ Καταγραφές με ανταλλαγές δεδομένων που πραγματοποιούνται κατά την εκτέλεση του κώδικα και τις οποίες μελετάμε και χαρακτηρίζουμε.

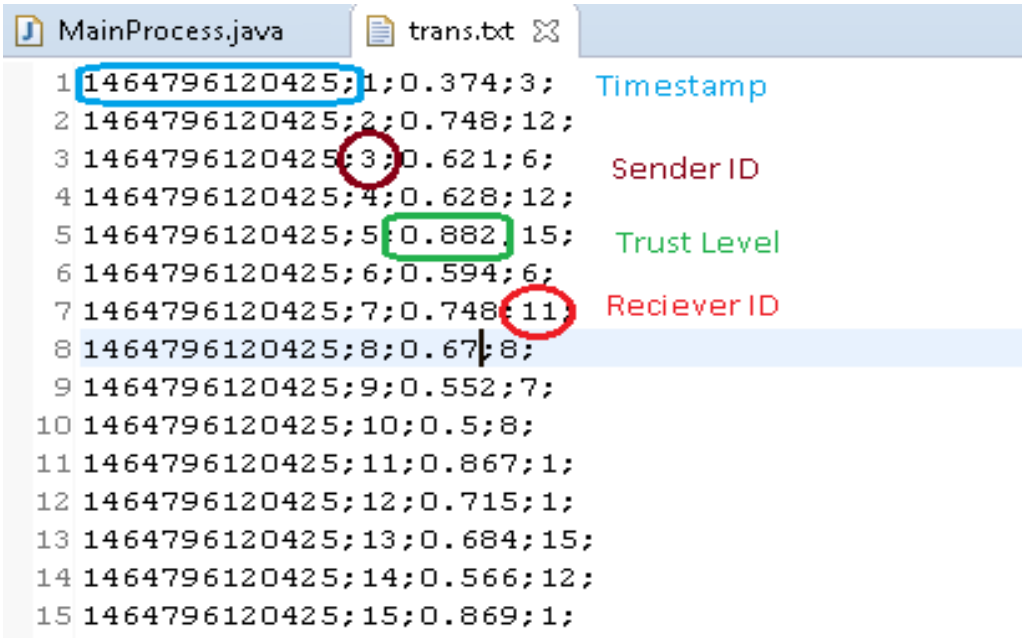
Το πρόγραμμα περιλαμβάνει πέντε (5) κλάσεις java για την υλοποίηση της διαδικασίας καθώς και έναν αριθμό αρχείων .txt για την παρουσίαση των αποτελεσμάτων, Τα βασικά αρχεία πάνω στα οποία γίνονται όλες οι ενέργειες και ουσιαστικά χρησιμοποιούνται σαν είσοδο είναι δύο και δημιουργούνται στην αρχή του προγράμματος. Από αυτά, το **πρώτο αρχείο (nodes.txt) περιλαμβάνει τα ID όλων των συσκευών** που θεωρούμε ότι υπάρχουν στο δίκτυο και **το δεύτερο (trans.txt) περιλαμβάνει καταγραφές με ανταλλαγές δεδομένων που έχουν πραγματοποιηθεί ή πραγματοποιούνται** και αντιστοιχεί στο block chain. Στο πρώτο αρχείο σημειώνονται επίσης ποιες συσκευές ανήκουν στο cluster καθώς και το ποια συσκευή είναι ο miner ενώ στο δεύτερο θα καταγράφεται κάθε ανταλλαγή δεδομένων που πραγματοποιείται με επιτυχία. Ο φάκελος Snapshots περιέχει ξεχωριστά αρχεία για κάθε περίπτωση που μελετάμε οι οποίες αναλύονται στη συνέχεια. (Εικόνα 8)



Εικόνα : 8 Απεικόνιση των περιεχομένων του project

Στο σημείο αυτό πρέπει να τονίσουμε ότι το πρόγραμμα αναπτύχθηκε με την παραδοχή ότι οι ανταλλαγή των δεδομένων θα γίνεται από συσκευές που ανήκουν στο ίδιο cluster και για το λόγο αυτό και ο miner επιλέγεται να είναι συσκευή από το ίδιο cluster. Προφανώς στην πραγματικότητα είναι σχεδόν αδύνατο να συμβαίνει αυτό και φυσικά να γίνει μια τέτοια ομαδοποίηση. Ας προχωρήσουμε λοιπόν στην επεξήγηση του προγράμματος.

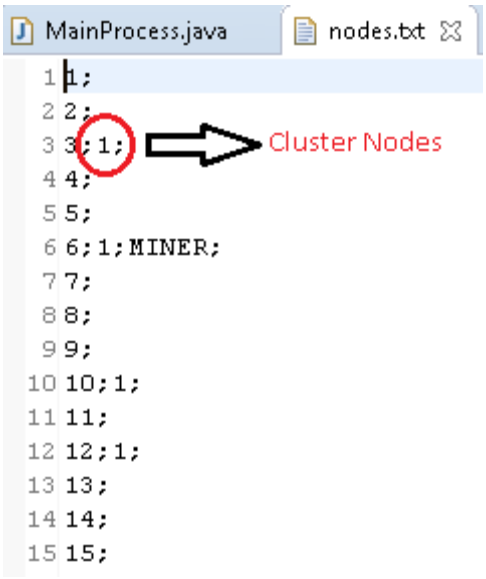
Αρχικά δημιουργείται ο φάκελος tmp με τα αρχεία που φαίνονται στην ανωτέρω εικόνα. Ο χρήστης επιλέγει τον αριθμό των συσκευών που υπάρχουν στο δίκτυο για τη δημιουργία του πρώτου αρχείου και με βάσει αυτές δημιουργείται το δεύτερο αρχείο (εικόνα 9) που περιλαμβάνει τις προηγούμενες ανταλλαγές δεδομένων μεταξύ αυτών. Δημιουργείται τουλάχιστον μια αποστολή δεδομένων για κάθε μια συσκευή με σκοπό να δημιουργηθεί το αρχικό επίπεδο αξιοπιστίας ενώ για κάθε μία αποστολή επιλέγεται ένας τυχαίος παραλήπτης. Το αρχικό επίπεδο αξιοπιστίας παράγεται τυχαία με βάση τη Gaussian κατανομή ενώ τα επιπλέον στοιχεία (κέντρο και απόκλιση) που απαιτούνται τα δίνει ο χρήστης.



Εικόνα : 9 Αρχείο με τις νεοδημιουργηθείσες συναλλαγές

Στη συνέχεια γίνεται μια επιλογή ενός αριθμού τυχαίων συσκευών από το σύνολο για τη δημιουργία της ομάδας (cluster). Ο αριθμός των συσκευών του cluster καθορίζεται στην αρχή του προγράμματος από το χρήστη. Οι συσκευές που ανήκουν στην ίδια ομάδα δεν καταγράφονται σε διαφορετικό αρχείο. Αντίθετα στο ήδη υπάρχον αρχείο τοποθετείται ένας χαρακτηριστικός αριθμός («1») δίπλα από κάθε συσκευή που επιλέγεται στην ομάδα.

Το επόμενο βήμα είναι να γίνει επιλογή του miner από το αρχείο που περιλαμβάνει όλες τις συσκευές. Αρχικά γίνεται εύρεση όλων των συσκευών που ανήκουν στο cluster και έπειτα στο αρχείο με τις καταγραφές βρίσκεται για κάθε μία συσκευή η πιο πρόσφατη ανταλλαγή στην οποία συμμετείχε ως αποστολέας. Ελέγχεται το επίπεδο αξιοπιστίας και η συσκευή με το πιο υψηλό επιλέγεται ως miner. Στην εικόνα 10 φαίνεται το αρχείο με το σύνολο των κόμβων έπειτα από την ομαδοποίηση και την πρώτη επιλογή miner.



Εικόνα : 10 Αρχείο με τα ID των συσκευών

Στο σημείο αυτό είναι που ξεκινάει και η λειτουργία που πρακτικά μας ενδιαφέρει, αυτή του ελέγχου και διεκπεραίωσης ανταλλαγής δεδομένων. Για να γίνει αυτό αρχικά επιλέγονται δύο συσκευές από το cluster για αποστολέας και δέκτης. Έχοντας και σαν καταγραφή το χρονικό σημείο (timestamp) και ένα ενδεικτικό επίπεδο αξιοπιστίας δημιουργείται μια πιθανή ανταλλαγή η οποία πρέπει να ελεγχθεί και να επικυρωθεί από το miner.

Οι συσκευές επιλέγονται τυχαία και για το λόγο αυτό υπάρχει πιθανότητα να επιλεχθεί σαν αποστολέας η συσκευή miner. Επειδή όμως στην περίπτωση αυτή ίσως δε θα έχουμε αντικειμενικότητα στον έλεγχο της αξιοπιστίας, δηλαδή δε μπορεί ο αποστολέας να είναι ίδιος με το miner, γίνεται ένας έλεγχος και αν διαπιστωθεί ότι είναι η ίδια συσκευή επιλέγεται ξανά άλλος miner αποκλείοντας τον προηγούμενο, ο οποίος θέλει να στείλει δεδομένα.

Ο τελικός miner θα πιστοποιήσει την αποστολή δεδομένων. Από την πιθανή ανταλλαγή που δημιουργήθηκε βρίσκει το ID του αποστολέα, στη συνέχεια ανατρέχει στο αρχείο με τις προηγούμενες καταγραφές, εντοπίζει την τελευταία αποστολή δεδομένων του ίδιου αποστολέα και ελέγχει το επίπεδο αξιοπιστίας. Αν το αυτό είναι μεγαλύτερο από το όριο που έχουμε θέσει στην αρχή επιτρέπει στην αποστολή να πραγματοποιηθεί. Αν είναι μικρότερο η αποστολή δεν πραγματοποιείται.

Από τη στιγμή που πραγματοποιηθεί μια αποστολή δεδομένων μένει το τελευταίο μέρος της υλοποίησης, αυτό της αξιολόγησης των δεδομένων από το δέκτη. Για το λόγο αυτό δημιουργούμε τυχαία βαθμολογίες μέσα σε ένα ρυθμιζόμενο εύρος. Η βαθμολογία αυτή παράγεται μετά από κάθε επιτυχημένη αποστολή και προστίθεται ή αφαιρείται (ανάλογα με το αν είναι θετική ή αρνητική) από το προηγούμενο επίπεδο αξιοπιστίας του αποστολέα. Το ανανεωμένο πλέον επίπεδο αξιοπιστίας του αποστολέα τοποθετείται δίπλα από το ID του στην πιο πρόσφατη καταγραφή.

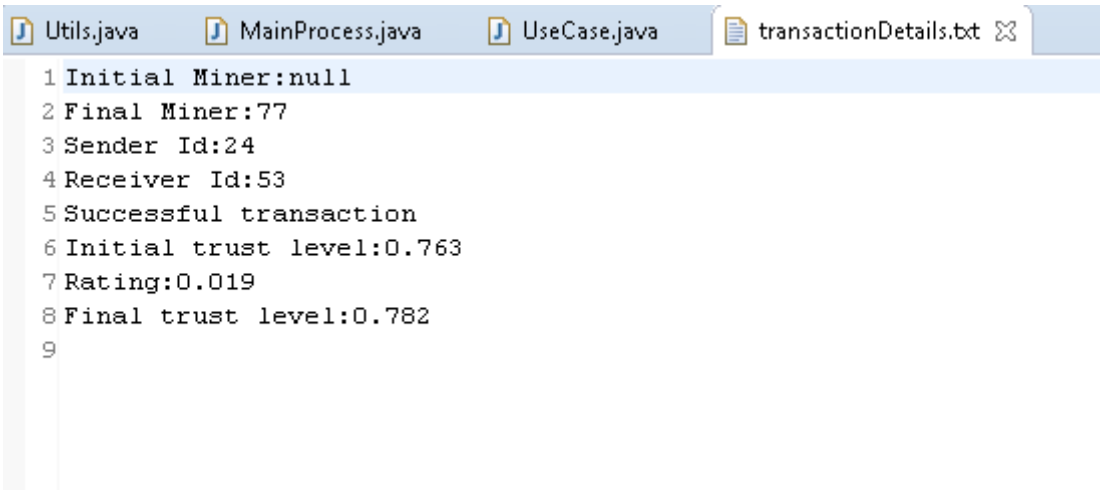
6.2 ΠΑΡΟΥΣΙΑΣΗ ΤΩΝ ΠΕΡΙΠΤΩΣΕΩΝ ΠΟΥ ΜΕΛΕΤΗΘΗΚΑΝ

Στην εξέλιξη της όλης διαδικασίας ξεχωρίζουμε 5 περιπτώσεις τις οποίες μελετάμε ως προς τη συμπεριφορά τους και τη συχνότητα εμφάνισής τους. Για να γίνει πιο κατανοητή κάθε περίπτωση δημιουργούμε ένα αρχείο το οποίο ενημερώνεται κατά την εξέλιξη της διαδικασίας και αφορά την κάθε ανταλλαγή ξεχωριστά. Συγκεκριμένα για κάθε ανταλλαγή κρατάμε τα παρακάτω στοιχεία:

- ✓ τα ID του αρχικού και τελικού miner,
- ✓ τα ID του αποστολέα και του δέκτη,
- ✓ εμφάνιση της επιτυχία ή όχι της ανταλλαγής,
- ✓ το αρχικό επίπεδο αξιοπιστίας του αποστολέα,
- ✓ τη βαθμολογία του παραλήπτη προς τον αποστολέα και
- ✓ το τελικό επίπεδο αξιοπιστίας του αποστολέα.

6.2.1 ΠΕΡΙΠΤΩΣΗ 1_1 (CASE 1_1)

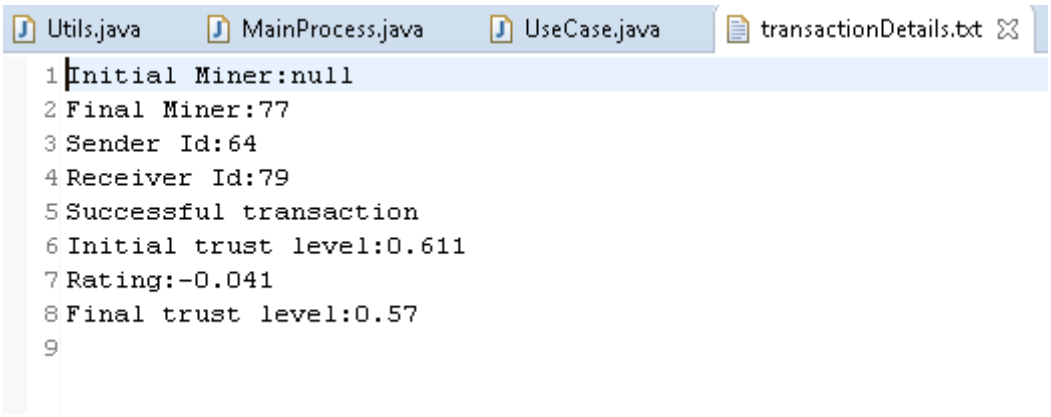
Στην περίπτωση αυτή η αποστολή και λήψη δεδομένων πραγματοποιείται με επιτυχία καθώς ο miner πληροί τα κριτήρια αξιοπιστίας. Ταυτόχρονα ο δέκτης αξιολογεί τα δεδομένα με θετική βαθμολογία και το επίπεδο αξιοπιστίας του αποστολέα αυξάνεται**.** Στην εικόνα 11 φαίνονται συνοπτικά τα στοιχεία μιας τυχαίας συναλλαγής που εμπίπτει στην περίπτωση αυτή.



Εικόνα : 11 Αποτέλεσμα επιτυχημένης αποστολής-Θετική αξιολόγηση (Case 1_1)

6.2.2 ΠΕΡΙΠΤΩΣΗ 1_2 (CAΣΕ 1_2)

Και στην περίπτωση αυτή έχουμε επιτυχία στην πραγματοποίηση της ανταλλαγής. Η διαφορά σε σχέση με την προηγούμενη περίπτωση είναι ότι η βαθμολογία του παραλήπτη είναι αρνητική. Στην εικόνα 12 φαίνονται τα στοιχεία μιας τέτοιας περίπτωσης.

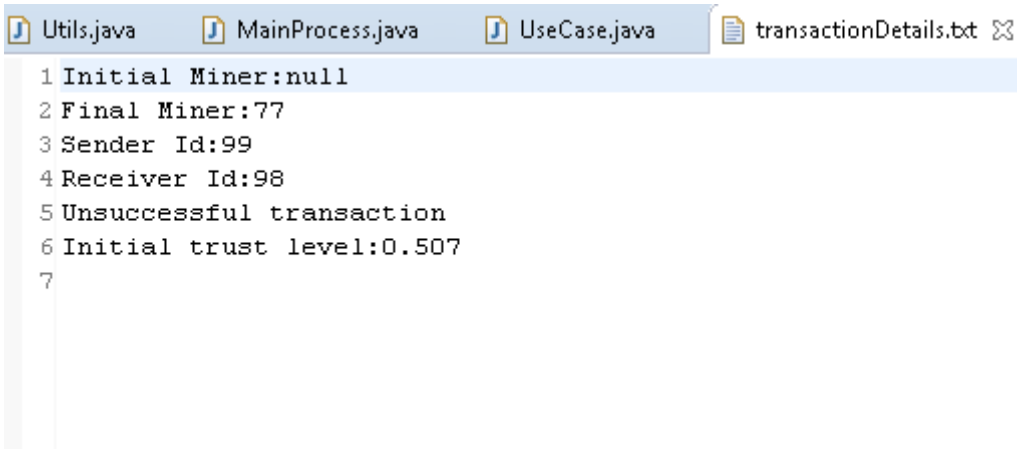


Εικόνα : 12 Αποτέλεσμα επιτυχημένης αποστολής-Αρνητική αξιολόγηση (Case 1_2)

6.2.3 ΠΕΡΙΠΤΩΣΗ 2 (CASE 2)

Στην περίπτωση αυτή δεν έχουμε επιτυχημένη ανταλλαγή δεδομένων. Ο αποστολέας δεν πληροί τα κριτήρια αξιοπιστίας και για το λόγο αυτό δεν υπάρχει καμία καταγραφή

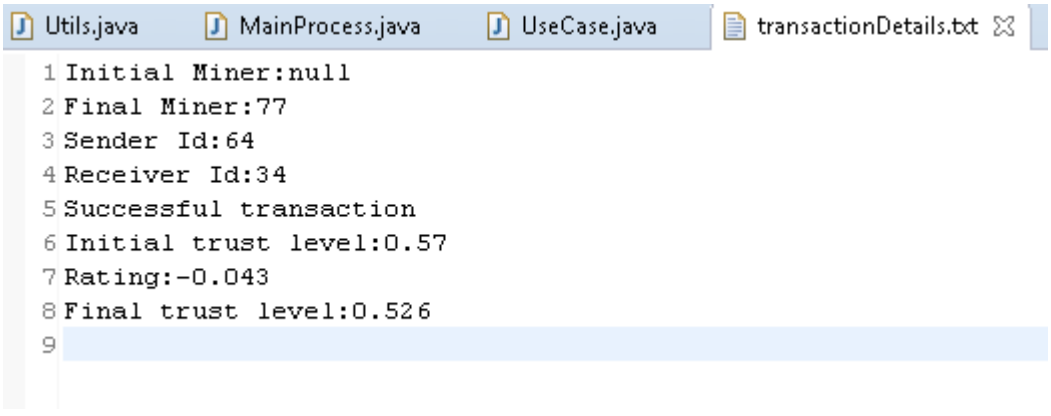
σχετικά με την ανταλλαγή αυτή. Στην εικόνα 13 φαίνονται συνοπτικά τα στοιχεία μιας τυχαίας συναλλαγής που εμπίπτει στην περίπτωση αυτή.



Εικόνα : 13 Αποτέλεσμα αποτυχημένης αποστολής (Case 2)

6.2.4 ΠΕΡΙΠΤΩΣΗ 3 (CASE 3)

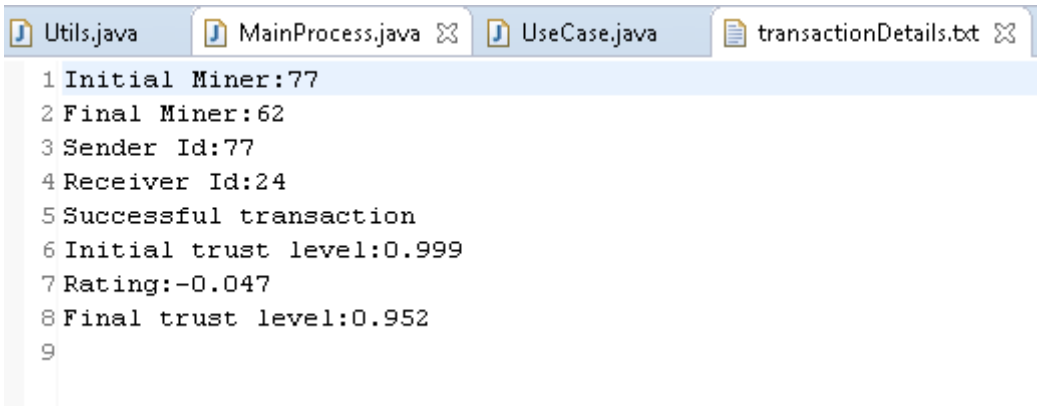
Στην περίπτωση αυτή η ανταλλαγή δεδομένων πραγματοποιείται με επιτυχία καθώς ο αποστολέας θεωρείται έμπιστος. Βέβαια μετά την αξιολόγηση του από τον παραλήπτη, το επίπεδο αξιοπιστίας του πέφτει κάτω από το όριο που έχουμε θέσει οπότε σε εκ νέου προσπάθεια αποστολής δεδομένων δε θα πάρει έγκριση. Την περίπτωση αυτή τη θεωρούμε ως σφάλμα καθώς θα πραγματοποιηθεί λήψη δεδομένων αμφιβόλου ποιότητας. Στην εικόνα 14 φαίνονται τα στοιχεία μιας τέτοιας περίπτωσης.



Εικόνα : 14 Αποτέλεσμα επιτυχημένης αποστολής - Σφάλμα (Case 3)

6.2.5 ΠΕΡΙΠΤΩΣΗ 4 (CASE 4)

Η περίπτωση αυτή περιλαμβάνει την επανεκλογή του miner επειδή ο ήδη επιλεγμένος θέλει να στείλει δεδομένα. Κατά τον έλεγχο του αποστολέα διαπιστώνεται ότι αυτός είναι ο miner και εκ νέου επιλέγεται άλλος. Η κατάληξη της διαδικασίας εμπίπτει έπειτα σε μία από τις προαναφερθείσες περιπτώσεις. Στην εικόνα 15 φαίνονται συνοπτικά τα στοιχεία μιας τυχαίας συναλλαγής που ανήκει στην περίπτωση αυτή.



Εικόνα : 15 Αποτέλεσμα συναλλαγής που απαιτείται αλλαγή miner (Case 4)

6.3 ΠΑΡΟΥΣΙΑΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

Η παραπάνω διαδικασία έτρεξε πολλές φορές δοκιμάζοντας διάφορους συνδυασμούς μεταξύ του συνολικού αριθμού των συσκευών, του αριθμού των συσκευών που ανήκουν στο cluster και τον αριθμό των transactions που πιστοποιήθηκαν και εντάχθηκαν σε κάθε μία από τις περιπτώσεις που μας ενδιαφέρουν. Συγκεκριμένα για κάθε διαφορετικό συνδυασμό που δοκιμάσαμε έτρεξε το πρόγραμμα για συνολικά 1500 πιθανές συναλλαγές. Από τη διαδικασία αυτή εξήχθησαν ενδιαφέροντα συμπεράσματα τα οποία παρουσιάζουμε στη συνέχεια.

6.3.1 ΑΠΟΤΕΛΕΣΜΑΤΑ ΓΙΑ ΔΙΚΤΥΟ ΑΠΟΤΕΛΟΥΜΕΝΟ ΑΠΟ 100 ΣΥΣΚΕΥΕΣ

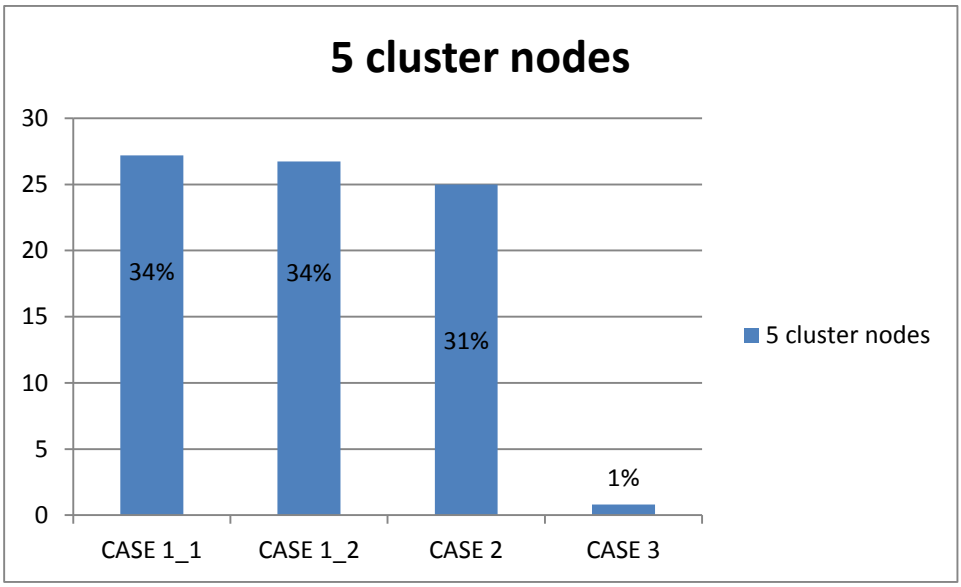
Αρχικά μελετήσαμε τα αποτελέσματα της διαδικασίας με επιλογή των συσκευών του cluster μέσα από έναν σύνολο 100 συσκευών. Έχοντας αυτό σα δεδομένο δοκιμάσαμε τρεις διαφορετικές περιπτώσεις clusters αλλάζοντας τον αριθμό των συσκευών που το απαρτίζουν.

- Cluster 5 συσκευών

Για αρχή επιλέξαμε έναν μικρό αριθμό συσκευών για τη συμπλήρωση του cluster. Τα αποτελέσματα της δοκιμής φαίνονται στο Διάγραμμα 1**.**

Παρατηρούμε ότι συνολικά οι ανταλλαγές δεδομένων πραγματοποιήθηκαν επιτυχώς με ποσοστό 68% (Case 1_1, Case 1_2) ενώ το ποσοστό αυτών που απέτυχαν ήταν 31% καθώς ο αποστολέας δεν είχε το απαιτούμενο επίπεδο αξιοπιστίας. Το σφάλμα στην περίπτωση αυτή εμφανίζεται με ποσοστό 1% το οποίο είναι αρκετά χαμηλό.

Ο miner χρειάστηκε να αλλάξει συνολικά 20% των περιπτώσεων το οποίο είναι αρκετά υψηλό ποσοστό αλλά και φυσιολογικό αν σκεφτούμε ότι οι συσκευές που θα αντάλλασαν δεδομένα μεταξύ τους ήταν μόλις 5.



Διάγραμμα 1: Αποτέλεσμα για σύνολο 100 συσκευές και 5 συσκευές στο cluster

Επίσης στον πίνακα που ακολουθεί φαίνονται οι περισσότερες φορές που εμφανίστηκε κάθε περίπτωση στο σύνολο των δοκιμών. Παρατηρούμε ότι η διαφορά ανάμεσα στο

μέγιστο και στο ελάχιστο είναι περίπου 10 με 15 συναλλαγές με εξαίρεση βέβαια την περίπτωση 3 όπου είναι μόλις 2 συναλλαγές.

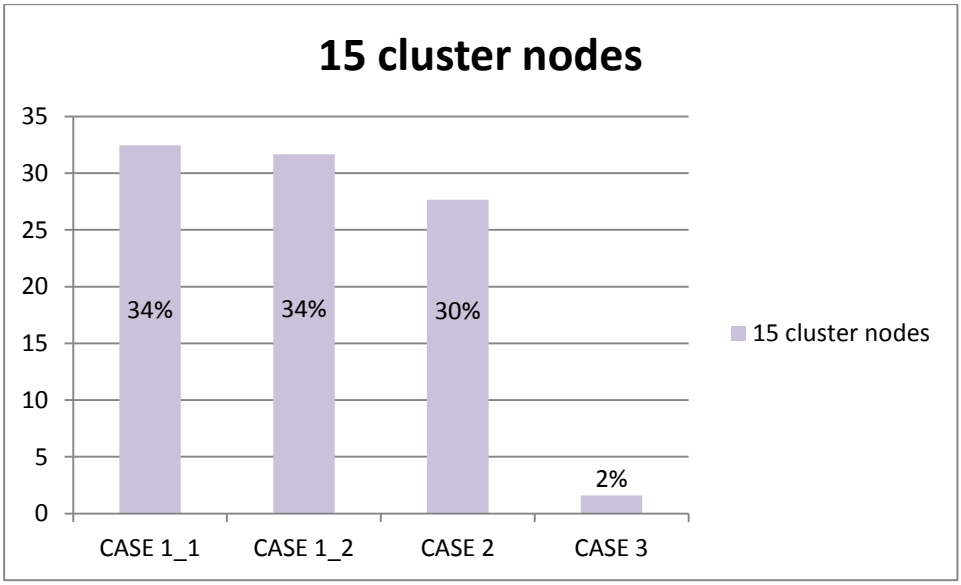
	case 1_1	case 1_2	case 2	case 3	case 4
max	31	36	32	2	25
min	23	21	19	0	15

Πίνακας 1: Μέγιστα και ελάχιστα κάθε περίπτωσης για δοκιμές με 100 συσκευές στο δίκτυο και 5 συσκευές στο cluster

- Cluster 15 συσκευών

Στη συνέχεια αυξήσαμε τον αριθμό των συσκευών του cluster κατά 10 και ακολουθήσαμε την ίδια διαδικασία. Τα αποτελέσματα της δοκιμής στην περίπτωση αυτή φαίνονται στο Διάγραμμα 2.

Ομοίως με την προηγούμενη περίπτωση οι επιτυχημένες ανταλλαγές δεδομένων πραγματοποιήθηκαν με ποσοστό 64% σε αντίθεση με τις μη πραγματοποιημένες οι οποίες άγγιξαν το 30%. Το σφάλμα στην περίπτωση αυτή ήταν οριακά μεγαλύτερο με ποσοστό 2%.



Διάγραμμα 2: Αποτέλεσμα για σύνολο 100 συσκευές και 15 συσκευές στο cluster

Ο miner χρειάστηκε να αλλάξει μέσο όρο 7 φορές σε 100 ανταλλαγές δεδομένων. Το ποσοστό αυτό είναι αρκετά μικρότερο από την προηγούμενη περίπτωση κάτι το οποίο προκύπτει από την αύξηση των συσκευών του cluster.

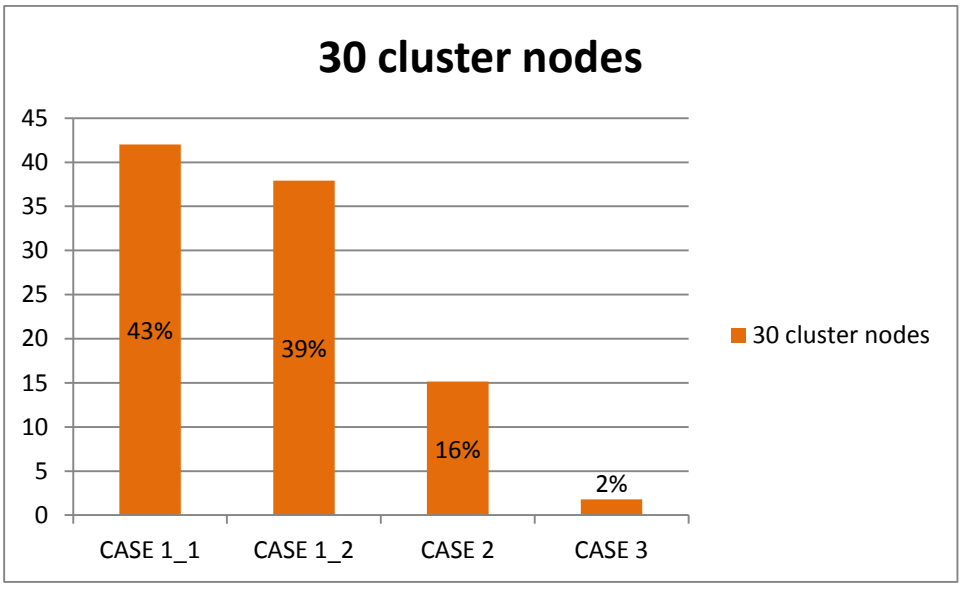
Στον πίνακα που ακολουθεί φαίνονται οι μέγιστος και ο ελάχιστος αριθμός που εμφανίστηκε η κάθε περίπτωση στη διάρκεια των δοκιμών. Η διαφορές είναι αρκετά μεγάλες σε κάποιες από τις περιπτώσεις ενώ ιδιαίτερη εντύπωση προκαλεί το μέγιστο των αποτυχημένων συναλλαγών που ξεπέρασε το 50% με 51 εμφανίσεις.

	case 1_1	**case 1_2**	**case 2**	**case 3**	**case 4**
max	43	43	51	3	9
min	18	20	8	1	2

Πίνακας 2: Μέγιστα και ελάχιστα κάθε περίπτωσης για δοκιμές με 100 συσκευές στο δίκτυο και 15 συσκευές στο cluster

- Cluster 30 συσκευών

Η επόμενη δοκιμή έγινε με το διπλασιασμό των συσκευών που ανήκουν στο cluster χρησιμοποιώντας το 30% του συνολικού αριθμού των συσκευών. Τα αποτελέσματα στην περίπτωση αυτή παρουσιάζονται με το Διάγραμμα 3.



Διάγραμμα 3: Αποτέλεσμα για σύνολο 100 συσκευές και 30 συσκευές στο cluster

Το ποσοστό των φορών που τα δεδομένα στάλθηκαν με επιτυχίας είναι αυξημένο στην περίπτωση αυτή και αγγίζει το 82%. Στον αντίποδα, οι αποτυχημένες αποστολές δεδομένων φτάνουν μόλις στο 16% ενώ το σφάλμα συνεχίζει να κυμαίνεται σε χαμηλά ποσοστά και είναι της τάξεως του 2%.

Η περίπτωση 4 (αλλαγή miner) εμφανίστηκε συνολικά σε ποσοστό 3% το οποίο είναι απολύτως λογικό αφού ο πιθανός αριθμός των συσκευών που μέσα από τις οποίες θα γινόταν η επιλογή ήταν αρκετά αυξημένος.

Στον πίνακα που ακολουθεί φαίνονται για την περίπτωση αυτή τα μέγιστα και τα ελάχιστα για κάθε περίπτωση.

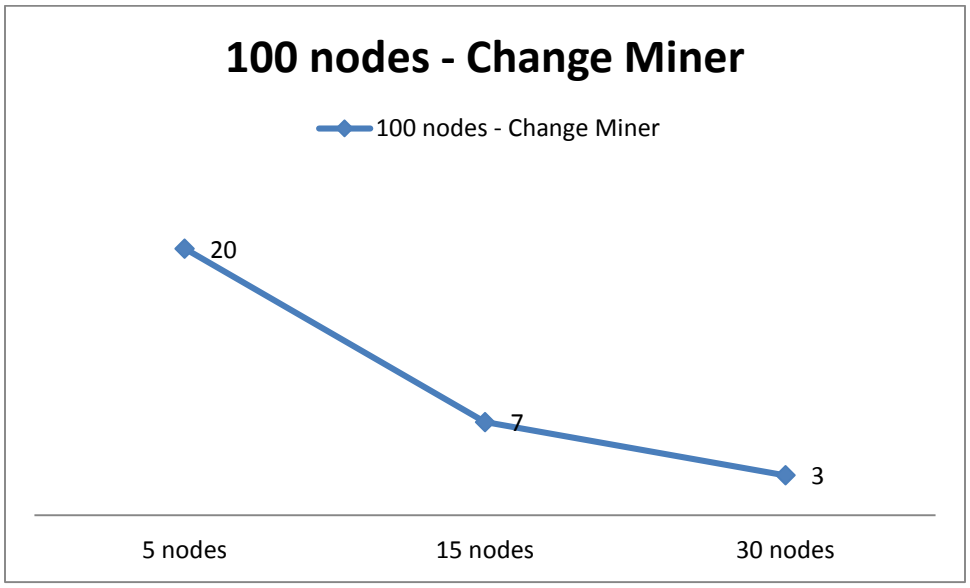
	case 1_1	**case 1_2**	**case 2**	**case 3**	**case 4**
max	54	48	30	4	5
min	32	25	5	1	1

Πίνακας 3: Μέγιστα και ελάχιστα κάθε περίπτωσης για δοκιμές με 100 συσκευές στο δίκτυο και 30 συσκευές στο cluster

Μεγαλύτερο ποσοστό σφάλματος 4% ενώ οι επιτυχημένες αποστολές δεδομένων με θετική αξιολόγηση για τον αποστολέα έφτασε στο 54%.

- Συγκεντρωτικά αποτελέσματα Case 4 (Αλλαγή miner)

Στο διάγραμμα που ακολουθεί φαίνονται συγκεντρωμένες οι φορές όπου ο miner χρειάστηκε να αλλάξει για τις περιπτώσεις που παρουσιάσαμε έως τώρα.



Διάγραμμα 4: Εξέλιξη της περίπτωσης 4 (Αλλαγή miner) για το σύνολο των δοκιμών με 100 συσκευές στο δίκτυο

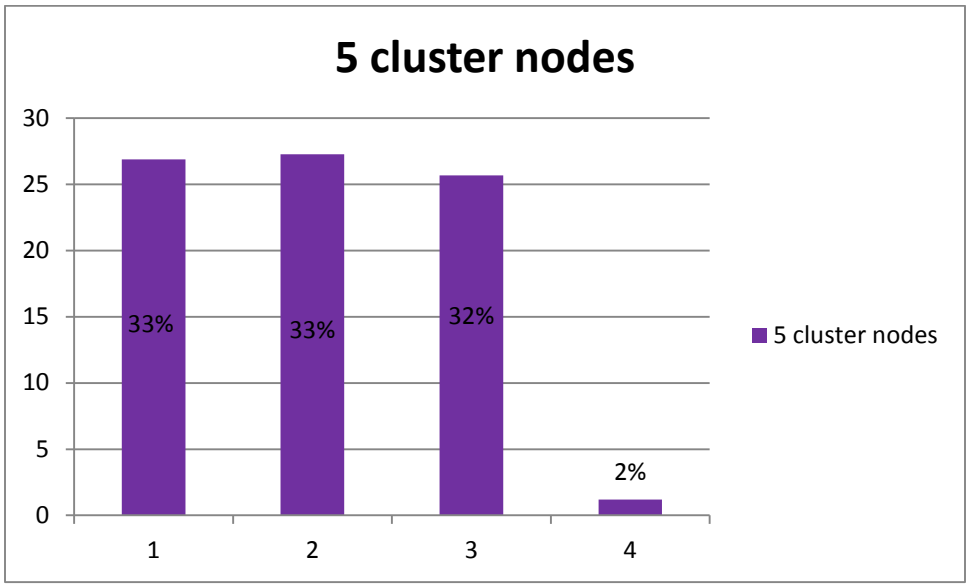
Παρατηρούμε πτωτική πορεία εμφανίσεων όσο αυξάνεται ο αριθμός των συσκευών στο cluster το οποίο όπως αναφέραμε είναι φυσιολογικό αφού όσο αυξάνεται ο αριθμός των συσκευών τόσο μειώνεται η πιθανότητα να χρειαστεί η συσκευή miner να στείλει δεδομένα.

6.3.2 ΑΠΟΤΕΛΕΣΜΑΤΑ ΓΙΑ ΔΙΚΤΥΟ ΑΠΟΤΕΛΟΥΜΕΝΟ ΑΠΟ 200 ΣΥΣΚΕΥΕΣ

Είδαμε τη συμπεριφορά του συστήματος για σύνολο 100 συσκευών. Στη συνέχεια διπλασιάσαμε τις συνολικές συσκευές και δοκιμάσαμε πάλι τις τρεις διαφορετικές περιπτώσεις των cluster.

- Cluster 5 συσκευών

Στην πρώτη περίπτωση χρησιμοποιήσαμε μόλις πέντε συσκευές για τη συμπλήρωση του cluster Τα αποτελέσματα της δοκιμής αυτής φαίνονται στο Διάγραμμα 5.



Διάγραμμα 5: Αποτέλεσμα για σύνολο 200 συσκευές και 5 συσκευές στο cluster

Παρατηρούμε πως οι φορές που τα δεδομένα στάλθηκαν με επιτυχίας αγγίζει το 66% ενώ η περίπτωση που η αποστολή εμφανίζεται με σχεδόν μισό ποσοστό και συγκεκριμένα με 32%. Το σφάλμα είναι σταθερά σε χαμηλά ποσοστά και συγκεντρώνει το 2%.

Στο ίδιο μοτίβο, αυξημένο είναι και το ποσοστό των περιπτώσεων που χρειάστηκε να αλλάξει ο miner το οποίο φτάνει στο 19% και επηρεάστηκε ελάχιστα έως καθόλου από την αύξηση του συνολικού αριθμού των συσκευών στο σύνολο του συστήματος.

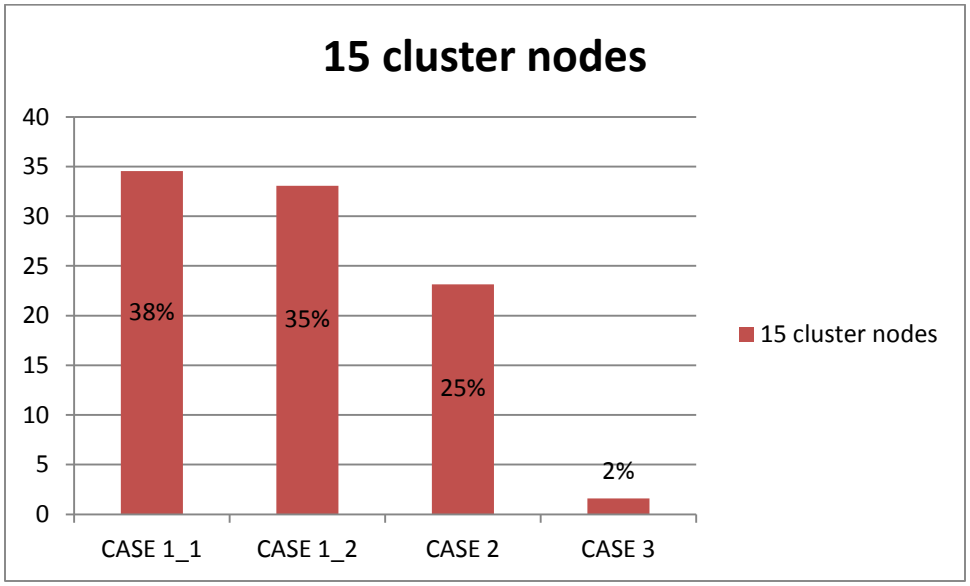
Στον πίνακα 4 εμφανίζονται οι μέγιστες και οι ελάχιστες φορές που εμφανίστηκε η κάθε περίπτωση στο σύνολο των δοκιμών. Παρατηρούμε σχετικά μικρές διαφορές (10-15) εξαιρώντας βέβαια την περίπτωση του σφάλματος που είναι μόλις 2.

	case 1_1	**case 1_2**	**case 2**	**case 3**	**case 4**
max	31	36	32	2	25
min	23	21	19	0	15

Πίνακας 4: Μέγιστα και ελάχιστα κάθε περίπτωσης για δοκιμές με 200 συσκευές στο δίκτυο και 5 συσκευές στο cluster

- Cluster 15 συσκευών

Συνεχίσαμε αυξάνοντας τον αριθμό των συσκευών του cluster σε 15. Στην περίπτωση αυτή τα αποτελέσματα φαίνονται στο Διάγραμμα 6.



Διάγραμμα 6: Αποτέλεσμα για σύνολο 200 συσκευές και 15 συσκευές στο cluster

Υψηλά ποσοστά για την αποστολή δεδομένων με επιτυχία η οποία φτάνει στο 73% ενώ οι αποτυχημένες φορές φτάνουν στο 25%. Η διαφορά στις περιπτώσεις με θετική και αρνητική αξιολόγηση είναι πολύ μικρή (3%). Το σφάλμα παραμένει σταθερά στο 2% όπως σχεδόν σε όλες τις προηγούμενες δοκιμές.

Στην περίπτωση αυτή ο miner άλλαξε στο 8% των περιπτώσεων, ποσοστό μειωμένο σε σχέση με την προηγούμενη περίπτωση με 5 συσκευές στο cluster.

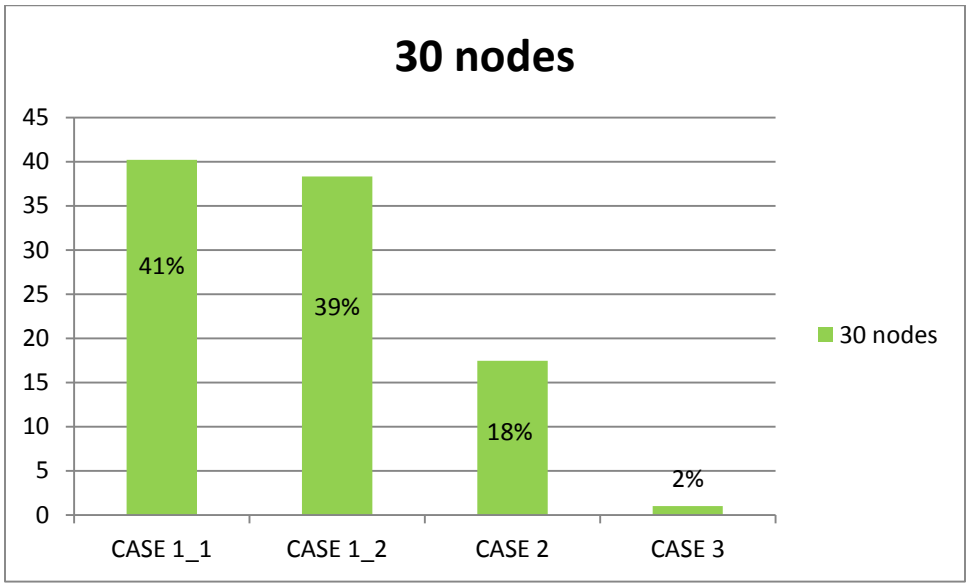
Στον πίνακα 5 εμφανίζονται το μέγιστο και το ελάχιστο των εμφανίσεων κάθε περίπτωσης στη διάρκεια των δοκιμών. Παρατηρούμε σχετικά μεγάλες αποκλίσεις για όλες τις περιπτώσεις ενώ αίσθηση προκαλεί το μέγιστο του σφάλματος που έφτασε το 6%, τριπλάσιο δηλαδή από το μέσο όρο του.

	case 1_1	**case 1_2**	**case 2**	**case 3**	**case 4**
max	42	47	49	6	12
min	21	19	3	0	4

Πίνακας 5: Μέγιστα και ελάχιστα κάθε περίπτωσης για δοκιμές με 200 συσκευές στο δίκτυο και 15 συσκευές στο cluster

- Cluster 30 συσκευών

Στη συνέχεια διπλασιάσαμε τις συσκευές του cluster και προχωρήσαμε σε δοκιμές. Τα αποτελέσματα από αυτές παρουσιάζονται συνοπτικά στο Διάγραμμα 7.



Διάγραμμα 7: Αποτέλεσμα για σύνολο 200 συσκευές και 30 συσκευές στο cluster

Το ποσοστό επιτυχίας στην αποστολή δεδομένων συνολικά έφτασε το 80% ενώ αυτό της αποτυχίας άγγιξε το 18%. Και στην τελευταία αυτή περίπτωση το σφάλμα παρέμεινε σταθερό στο 2%.

Το ποσοστό των περιπτώσεων που ο miner άλλαξε στην διάρκεια του προγράμματος ήταν μόλις 3% το οποίο είναι ίδιο με το ποσοστό της περίπτωσης αυτής για σύνολο 100 συσκευών με 30 συσκευές στο cluster.

Ακολουθεί πίνακας με τις μέγιστες και τις ελάχιστες εμφανίσεις κάθε περίπτωσης στο σύνολο των δοκιμών. Το πιο περίεργο είναι το ελάχιστο της αποτυχίας αποστολής όπου έφτασε το 2%.

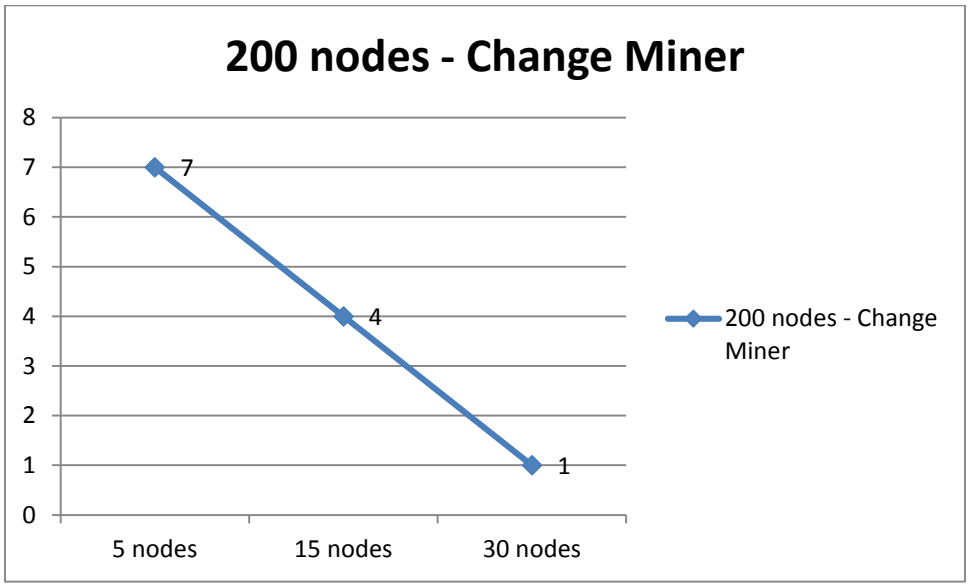
	case 1_1	**case 1_2**	**case 2**	**case 3**	**case 4**

max	52	48	32	2	8
min	33	28	2	0	1

Πίνακας 6: Μέγιστα και ελάχιστα κάθε περίπτωσης για δοκιμές με 100 συσκευές στο δίκτυο και 5 συσκευές στο cluster

- Συγκεντρωτικά αποτελέσματα Case 4 (Αλλαγή miner)

Στο διάγραμμα που ακολουθεί φαίνονται συγκεντρωμένες οι φορές όπου ο miner χρειάστηκε να αλλάξει για τις περιπτώσεις που παρουσιάσαμε έως τώρα.



Διάγραμμα 8: Εξέλιξη της περίπτωσης 4 (Αλλαγή miner) για το σύνολο των δοκιμών με 100 συσκευές στο δίκτυο

Παρατηρούμε σταθερή πτωτική πορεία εμφανίσεων όσο αυξάνεται ο αριθμός των συσκευών στο cluster. Λογικό και σε αυτή την περίπτωση γιατί δεν εξαρτάται από το συνολικό αριθμό των συσκευών αλλά μόνο από τις συσκευές που ανήκουν στο ίδιο cluster.

ΚΕΦΑΛΑΙΟ 7

ΣΥΜΠΕΡΑΣΜΑΤΑ – ΠΡΟΤΑΣΕΙΣ ΓΙΑ ΣΥΝΕΧΙΣΗ ΕΡΕΥΝΑΣ

7.1 ΣΥΜΠΕΡΑΣΜΑΤΑ

Στην παρούσα εργασία αναπτύξαμε ένα μοντέλο για την πιστοποίηση της ανταλλαγής δεδομένων μέσα σε ένα δίκτυο cloud και κυρίως στο άκρο του δικτύου. Η καινοτόμος και βασική ιδέα αυτού του μοντέλου είναι η αρχιτεκτονική στην οποία βασίζεται το ψηφιακό νόμισμα bitcoin, η οποία είναι το block chain και η ύπαρξη χρηστών (miners) που πιστοποιούν και επιβεβαιώνουν τις συναλλαγές με το νόμισμα αυτό. Στην περίπτωση μας δοκιμάσαμε το μοντέλο αυτό για ανταλλαγή δεδομένων λαμβάνοντας υπόψη ένα μέγεθος που χαρακτηρίζει το πόσο έμπιστη είναι κάθε συσκευή, το επίπεδο αξιοπιστίας (trust level). Το μέγεθος αυτό είναι που κρίνει το αν μία συσκευή θα μπορέσει να στείλει δεδομένα καθώς κάτω από ένα όριο αυτού η συσκευή δε θεωρείται αξιόπιστη να στείλει δεδομένα ή καλύτερα θεωρούμε ότι τα δεδομένα αυτά θα είναι αμφιβόλου ποιότητας.

Αναπτύχτηκε κώδικας που προσομοιώνει μια απλή διαδικασία ανταλλαγής δεδομένων όπου μία τρίτη συσκευή, ο miner, επιτρέπει ή απαγορεύει την αποστολή ανάλογα με το επίπεδο αξιοπιστίας του αποστολέα. Βέβαια λόγω της έλλειψης πραγματικών δεδομένων η διαδικασία αυτή δοκιμάστηκε με δεδομένα που παρήχθησαν με ψευδοτυχαίο τρόπο μέσα από το πρόγραμμα.

Η ιδέα για το μοντέλο αυτό γεννήθηκε από την ανάγκη για έλεγχο της ποιότητας των μεταφερόμενων δεδομένων ενώ η χρήση συσκευών - miner που πιστοποιούν τα επίπεδα αξιοπιστίας των αποστολέων προσφέρει στην κατά το δυνατό ελεγχόμενη χρήση υπολογιστικών πόρων. Ταυτόχρονα η συσκευή αυτή είναι πιστοποιημένα έμπιστο μέλος του δικτύου αφού επιλέγεται με βάση το κριτήριο αυτό.

Από τη διαδικασία και την ανάλυση των αποτελεσμάτων προέκυψαν κάποια ενδιαφέροντα συμπεράσματα. Κατ’ αρχάς σε πρώτη φάση και στην απλή μορφή που δοκιμάστηκε φάνηκε να λειτουργεί με επιτυχία καθώς σε όλες τις περιπτώσεις με όλους τους συνδυασμούς είχαμε ικανοποιητικά ποσοστά τόσο για τις επιτυχημένες όσο για τις

μη πραγματοποιηθείσες αποστολές δεδομένων. Το σφάλμα, δηλαδή οι περιπτώσεις που αν και το επίπεδο αξιοπιστίας του αποστολέα ήταν πάνω από το κατώφλι εν τέλει τα δεδομένα που έστειλε δεν ήταν ποιοτικά, κυμάνθηκε σε πολύ χαμηλά επίπεδα της τάξεως του 2% το οποίο πρόκειται για ανεκτό ποσοστό αν λάβουμε υπόψη το πλήθος των δεδομένων που μεταφέρονται στο δίκτυο από τις συσκευές.

Παρατηρήθηκε επίσης ότι υπάρχει αναλογία στη σχέση του σφάλματος με τις περιπτώσεις που μια αποστολή τελικά δεν πραγματοποιείται λόγω αναξιοπιστίας του αποστολέα. Αυτό συμβαίνει διότι όσο αυξάνονται οι συσκευές που το επίπεδο αξιοπιστίας τους έπειτα από μια αποστολή πέφτει κάτω από το προβλεπόμενο όριο, τόσο μειώνονται και οι συσκευές που μπορούν τελικά να προχωρήσουν σε αποστολή δεδομένων εφόσον αναφερόμαστε σε cluster που δε μεταβάλλεται όσον αφορά τις συσκευές που το απαρτίζουν.

Επίσης μια πολύ σημαντική αλλά αναμενόμενη παρατήρηση αφορά στη συχνότητα αλλαγής του miner και στην άμεση σχέση που έχει με τον αριθμό των συσκευών που συμμετέχουν στο cluster. Μεγάλος αριθμός συσκευών οδηγεί σε μικρότερη πιθανότητα αλλαγής του miner ενώ όσο το cluster γίνεται πιο κλειστό οδηγούμαστε σε περισσότερες αλλαγές του miner στην εξέλιξη της διαδικασίας.

Δοκιμάστηκαν διαφορετικοί συνδυασμοί ανάμεσα στο συνολικό αριθμό των συσκευών του δικτύου και στον αριθμό των συσκευών που συνθέτουν ένα cluster. Από τις δοκιμές και συγκεκριμένα για το μοντέλο αυτό δε φαίνεται να υπάρχει κάποια άμεση σύνδεση ανάμεσα στο συνολικό αριθμό των συσκευών που υπάρχουν στο δίκτυο και στα ποσοστά των πραγματοποιηθέντων ή μη συναλλαγών. Η μόνη πολύ μεγάλη διαφορά ήταν πως η επιλογή ενός cluster 5 συσκευών ανάμεσα από ένα σύνολο 100 συσκευών οδήγησε σε αλλαγή του miner 13% περισσότερες φορές από την επιλογή ανάμεσα σε σύνολο 200 συσκευών.

Τέλος, λαμβάνοντας υπόψη το χρόνο καθώς και τους υπολογιστικούς πόρους που απαιτούνται για την επιλογή του miner, καταλήγουμε πως όσο μεγαλώνει ο αριθμός των συσκευών ενός cluster τόσο πιο αποδοτικό γίνεται το μοντέλο διότι δεν προκύπτει η ανάγκη για αλλαγή miner τόσο συχνά. Βέβαια σε πολύ μεγάλα cluster ενδέχεται να δημιουργηθούν καθυστερήσεις αν αναλογιστούμε πως κάθε miner θα επιβαρύνεται με

τον έλεγχο πολλών εισερχόμενων συναλλαγών. Σε κάθε περίπτωση αυτό είναι κάτι που χρήζει επιπλέον έρευνας.

7.2 ΠΡΟΤΑΣΕΙΣ ΓΙΑ ΣΥΝΕΧΙΣΗ ΤΗΣ ΕΡΕΥΝΑΣ

Το μοντέλο που παρουσιάσαμε είναι νέο και προσπαθεί να δώσει λύσεις στην ανάγκη για ασφάλεια δεδομένων και μεταφοράς αυτών διαμέσου διαφόρων συσκευών. Έγινε μια πρώτη προσέγγιση αλλά το σίγουρο είναι ότι πολλά ακόμη μπορούν και πρέπει να γίνουν προτού εφαρμοστεί σε πραγματικά δεδομένα και σε πραγματικές συνθήκες. Μερικές από τις προτάσεις και κατευθύνσεις για τη συνέχιση της έρευνας είναι οι παρακάτω:

1) Για να είναι πιο αξιόπιστο το μοντέλο θα πρέπει να λαμβάνονται υπόψη επιπλέον παράγοντες για το χαρακτηρισμό μιας συσκευής ως αξιόπιστης ή όχι. Να λαμβάνονται υπόψη περισσότερες από μια καταγραφές για τον έλεγχο του επιπέδου αξιοπιστίας του αποστολέα με ειδικά ίσως βάρη και στον όγκο των δεδομένων που έχει στείλει κάθε φορά. Επιπλέον να λαμβάνεται υπόψη και η αξιοπιστία του δέκτη όσον αφορά στη βαθμολογία με την οποία κρίνει τον αποστολέα.
2) Να αναπτυχθεί και ο τρόπος με τον οποίο οι συσκευές που έχουν επίπεδο αξιοπιστίας κάτω από το όριο που θέτουμε κάθε φορά να μπορούν να το αυξήσουν ώστε να συμμετέχουν ξανά ισότιμα στο δίκτυο της μεταφοράς δεδομένων. Μια σχετική πρόταση, που προϋποθέτει την επιβάρυνση του miner, έχει να κάνει με την αποστολή μικρού όγκου δεδομένων από τη συσκευή αυτή στο miner ο οποίος θα τα αξιολογεί και θα αυξάνει σταδιακά το επίπεδο αξιοπιστίας της συσκευής.
3) Σε πραγματικές συνθήκες οι συσκευές θα ανταλλάσουν δεδομένα χωρίς απαραίτητα να ανήκουν στο ίδιο cluster. Στην περίπτωση αυτή θα πρέπει να αναπτυχθεί μια διαδικασία για τον έλεγχο αυτών. Θα ελέγχονται από miner που βρίσκονται μέσα στα clusters ή θα υπάρχουν miners με πιο εποπτικό ρόλο?
4) Τέλος, θα πρέπει να γίνεται σταδιακά εφαρμογή σε πραγματικές συνθήκες με πραγματικές συσκευές. Αυτό είναι ίσως το πιο σημαντικό βήμα για την ανάπτυξη

του μοντέλου διότι μόνο έτσι θα εμφανιστούν όποια προβλήματα δε μπορούμε να προβλέψουμε στη θεωρητική ανάπτυξη του.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Ericcson Mobility Report, “On The Pulse Of The Networked Society”, November 2015

[2] Jure Leskovec, Anand Rajaraman, “Clustering Algorithms”, Stanford University

[3] Malhotra , Agarwal and Jaiswal, “Virtualization in Cloud Computing”, Malhotra et al., J Inform Tech Softw Eng 2014, 4:2

[4] K C Gouda, Anurag Patro, Dines Dwivedi , Nagaraj Bhat, “Virtualization Approaches in Cloud Computing”, International Journal of Computer Trends and Technology (IJCTT) – volume 12 Issue 4–June 2014

[5] Christopher Olive, “Cloud Computing Characteristics Are Key”, General Physics Corporation

[6] Sabrina Zimara, “The Five Essential Characteristics of Cloud Computing” July 12, 2013

[7] “Introduction to Cloud Computing architecture”, Sun Microsystems, 1st edition, June 2009

[8] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica, Matei Zaharia, “Above the Clouds: A Berkeley View of Cloud Computing”, Electrical Engineering and Computer Sciences University of California at Berkeley, February 10, 2009

[9] Dave LeClair, “The Edge of Computing: It's Not All About the Cloud”, Inovation Insights,July,2014

[10] Mobile-Edge Computing, Introductory Technical White Paper, September 2014

[11] Patrick McGary, “Why Edge Computing Is Here To Stay: Five Use Cases”, RTInsights.com

[12] “Internet of Things – from Wikipedia”, https://en.wikipedia.org/wiki/Internet_of_things

[13] Salvatore Gaglio, Giuseppe Lo Re, Gloria Martorella, Daniele Peri, and Salvatore Davide Vassallo, “Development of an IoT Environmental Monitoring Application with a Novel Middleware for Resource Constrained Devices”, University of Palermo

[14] Thanasis Tsikogias, “Energy-Efficient Infrastructure Management in the IoT era”, Schneider Electric

[15] A guide to healthcare IoT possibilities and obstacles, http://searchhealthit.techtarget.com

[16] Carl Weinschenk, "Energy Management: The Internet of Things Changes Everything", Energy Manager Today, November 2011

[17] "Building an Intelligent Transportation System with the Internet of Things (IoT)", Intel

[18] Tyler Reed, "The Internet of Things and home automation", https://www.control4.com

[19] **"**Cloud Computing and Security – A Natural Match", Trusted Computing Group, April 2010

[20] Jaydip Sen, "Security and Security and Privacy Issues in Cloud Computing", Innovation Labs, Tata Consultancy Services Ltd

[21] David Naylor , Alessandro Finamore , Ilias Leontiadis , Yan Grunenberger , Marco Mellia , Maurizio Munafò , Konstantina Papagiannaki , and Peter Steenkiste, "The Cost of the "S" in HTTPS", Carnegie Mellon University, Politecnico di Torino, Telefónica Research

[22] Jon Brodkin, "5 problems with SaaS security", http://www.networkworld.com/

[23] Devi T , Ganesan R*,* "Platform-as-a-Service (PaaS): Model and Security Issues*",* TELKOMNIKA Indonesian Journal of Electrical Engineering, Vol. 15, No. 1, July 2015, pp. 151 ~ 161

[24] Deb Shinder, " Security Consideration for Infrastructure as a Service Cloud Computing Model", WindowsSecurity.com, June 2011

[25] Sam Curry, Jon Darbyshire, Douglas W. Fisher, Bret Hartman, Dr. Stephen Herrod," Infrastructure Security: Getting to the Bottom of Compliance in the Cloud", RSA Security Brief, March, 2010

[26] Satoshi Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System"

[27] https://bitcoin.org/el/faq#how-does-bitcoin-work

[28] Choon Hoong Ding, Sarana Nutanong, and Rajkumar Buyya, "Peer-to-Peer Networks for Content Sharing", Grid Computing and Distributed Systems Laboratory, Department of Computer Science and Software Engineering, The University of Melbourne, Australia

[29] http://www.coindesk.com/information/how-bitcoin-mining-works

[30] https://en.bitcoin.it/wiki/Block_hashing_algorithm

[31] M. Tamer Ozsu, Patrick Valduriez, “Distributed and Parallel Database Systems”, Department of Computing Science University of Alberta

ΠΑΡΑΡΤΗΜΑ

Στο παράρτημα παρατίθεται ο κώδικας που αναπτύχθηκε για την υλοποίηση του μοντέλου:

1) MainProcess.java

```
package gr.cloud.trust;

    import java.nio.file.Path;
    import java.nio.file.Paths;

    public class MainProcess {

            public static void main(String[] args) throws Exception {

                    final Path NODES_PATH = Paths.get("tmp\\nodes.txt");
                    final int NUMBER_OF_NODES = 200;
                    final int NUMBER_OF_CLUSTER_NODES = 30;
                    final Path TRANSACTIONS_PATH = Paths.get("tmp\\trans.txt");
                    final double TRUST_LEVEL_THRESHOLD = 0.55;
                    final double INITIAL_TRANSACTIONS_STANDARD_DEVIATION = 0.15;
                    final double INITIAL_TRANSACTIONS_MEAN = 0.7;

                    Utils.CreateNodes(NODES_PATH, NUMBER_OF_NODES);

                    Utils.CreateCluster(NODES_PATH, NUMBER_OF_CLUSTER_NODES);

                    Utils.CreateInitialTransactions(TRANSACTIONS_PATH, NODES_PATH,
                    INITIAL_TRANSACTIONS_STANDARD_DEVIATION,
                    INITIAL_TRANSACTIONS_MEAN);

                    Utils.findNewMiner(TRANSACTIONS_PATH, NODES_PATH, null);


                    for(int i=1;i<=100;i++){

                    Transaction newTransaction = Utils.CreateNewTransaction(NODES_PATH);

                    Utils.CheckTrustLevelAndVoting(newTransaction, TRANSACTIONS_PATH,
                    NODES_PATH, TRUST_LEVEL_THRESHOLD, null);
            }

    }

}
```

2) Utils.java

```
package gr.cloud.trust;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.Collection;
import java.util.Collections;
import java.util.Date;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import java.util.Random;

public class Utils {

	//Create the file with all nodes
	public static void CreateNodes(Path nodesFilePath, int numberOfNodes) throws IOException {

		System.out.println("Creating Nodes File...");

		if (!Files.exists(nodesFilePath.getParent())){
			File folder = new File(nodesFilePath.getParent().toUri());
			folder.mkdirs();
		}


		if (Files.exists(nodesFilePath)){
			FileOutputStream writer = new
FileOutputStream(nodesFilePath.toAbsolutePath().toString());
			writer.write((new String()).getBytes());
			writer.close();
		} else {
			File file = new File(nodesFilePath.toUri());
			file.createNewFile();
		}

		StringBuilder fileContent = new StringBuilder();
		for(int i=1;i<=numberOfNodes;i++){
			fileContent.append(String.valueOf(i)+";"+"\n");
		}
		FileOutputStream writer = new FileOutputStream(nodesFilePath.toAbsolutePath().toString());
		writer.write(fileContent.toString().getBytes());
		writer.close();
	}

	//Build the cluster from all nodes
	public static void CreateCluster(Path nodesFilePath, int numberOfClusterNodes) throws Exception {
		System.out.println("Creating Cluster...");

		if (!Files.exists(nodesFilePath)){
			System.out.println("The file "+nodesFilePath.toUri().toString()+" does not exist.");
			throw new Exception();
		}

		InputStream in = new FileInputStream(new File(nodesFilePath.toAbsolutePath().toString()));
	BufferedReader reader = new BufferedReader(new InputStreamReader(in));
```

```
	String line;
	StringBuilder currentFile = new StringBuilder();
			List<String> nodeList = new ArrayList<String>();
			while ((line = reader.readLine()) != null) {
				currentFile.append(line+"\n");
		String[] lineTokens = line.split(";");
		nodeList.add(lineTokens[0]);
	}
	reader.close();

			Collections.shuffle(nodeList);
		List<String> clusterNodeList = nodeList.subList(0, numberOfClusterNodes);

		for(String clusterNode : clusterNodeList){
			currentFile = new StringBuilder(currentFile.toString().replace(";\n"+clusterNode+";\n",
";\n"+clusterNode+";1;\n"));
		}
		FileOutputStream writer = new FileOutputStream(nodesFilePath.toAbsolutePath().toString());
			writer.write(currentFile.toString().getBytes());
			writer.close();
	}

	//Create a sample of previous transaction, at least one for every node
	public static void CreateInitialTransactions(Path transactionsFilePath, Path nodesFilePath, double
standardDeviation, double mean) throws Exception {
			System.out.println("Creating Initial Transactions...");

			if (Files.exists(transactionsFilePath)){
				FileOutputStream writer = new
FileOutputStream(transactionsFilePath.toAbsolutePath().toString());
				writer.write((new String()).getBytes());
				writer.close();
			} else {
				File file = new File(transactionsFilePath.toUri());
				file.createNewFile();
			}

			if (!Files.exists(nodesFilePath)){
				System.out.println("The file "+nodesFilePath.toUri().toString()+" does not exist.");
				throw new Exception();
			}

			InputStream in = new FileInputStream(new File(nodesFilePath.toAbsolutePath().toString()));
	BufferedReader reader = new BufferedReader(new InputStreamReader(in));
	String line;
			List<String> nodeList = new ArrayList<String>();
			while ((line = reader.readLine()) != null) {
		String[] lineTokens = line.split(";");
		nodeList.add(lineTokens[0]);
	}
	reader.close();

	StringBuilder fileContent = new StringBuilder();
	Date date = new Date();
			for(String nodeId : nodeList){

				String timestamp = String.valueOf(date.getTime());

				String senderId = nodeId;

				List<String> copy = new ArrayList<String>(nodeList);
				Collections.copy(copy, nodeList);
				Collections.shuffle(copy);
				String receiverId = copy.subList(0, 1).get(0);

				Random rng = new Random();
```

```
//                              double trustLevel = rng.nextGaussian()*standardDeviation+mean;

                                double trustLevel= (double)Math.round((rng.nextGaussian()*standardDeviation+mean) *
1000) / 1000;
                                if(trustLevel<0){
                                        trustLevel=0.0;
                                } else if (trustLevel>1){
                                        trustLevel=1.0;
                                }

                                fileContent.append(timestamp+";"+senderId+";"+trustLevel+";"+receiverId+";\n");
                        }
                        FileOutputStream writer = new FileOutputStream(transactionsFilePath.toAbsolutePath().toString());
                        writer.write(fileContent.toString().getBytes());
                        writer.close();
                }

                //Choose miner according trust level
                public static String findNewMiner(Path transactionsFilePath, Path nodesFilePath, String nodeIdExcluded)
throws Exception{
                        System.out.println("Finding New Minner...");

                        if (!Files.exists(nodesFilePath)){
                                System.out.println("The file "+nodesFilePath.toUri().toString()+" does not exist.");
                                throw new Exception();
                        }

                        if (!Files.exists(transactionsFilePath)){
                                System.out.println("The file "+transactionsFilePath.toUri().toString()+" does not exist.");
                                throw new Exception();
                        }

                        HashMap<String, Transaction> transactions = new HashMap<String, Transaction>();
                        HashSet<String> clusterNodes = new HashSet<String>();

                        //Find all the nodes of the cluster
                        InputStream in = new FileInputStream(new File(nodesFilePath.toAbsolutePath().toString()));
        BufferedReader reader = new BufferedReader(new InputStreamReader(in));
        String line;
                        while ((line = reader.readLine()) != null) {
           String[] lineTokens = line.split(";");
           if(lineTokens.length>1 && lineTokens[1].equals("1")){
             clusterNodes.add(lineTokens[0]);
           }
        }
        reader.close();


        //Sorting of the transaction list based in timestamp and sender ID
        in = new FileInputStream(new File(transactionsFilePath.toAbsolutePath().toString()));
        reader = new BufferedReader(new InputStreamReader(in));
                        while ((line = reader.readLine()) != null) {
           String[] lineTokens = line.split(";");
           if(lineTokens.length!=4){
             System.out.println("The transactions file is not properly constructed.");
             reader.close();
             throw new Exception();
           }
           Transaction t = new Transaction(Long.valueOf(lineTokens[0]),lineTokens[1], Double.valueOf(lineTokens[2]),
lineTokens[3]) ;
           if (!clusterNodes.contains(t.getSenderId())) continue;
           if(transactions.containsKey(lineTokens[1])){
             if(transactions.get(lineTokens[1]).getTimestamp()<t.getTimestamp()){
                        transactions.put(lineTokens[1], t);
             }
           } else {
```

```
            transactions.put(lineTokens[1], t);
          }
      }
                  reader.close();
                  List<Transaction> sortedTransactions = new ArrayList<Transaction>((Collection<Transaction>)
transactions.values());
                  Collections.sort(sortedTransactions);

                  //thelei elegxo gia ton arithmo twn nodes
                  String newMiner = sortedTransactions.get(0).getSenderId();
                  if(nodeIdExcluded!=null && newMiner.equals(nodeIdExcluded)){
                        newMiner = sortedTransactions.get(1).getSenderId();
                  }

                  //remove old miner if exists
                  in = new FileInputStream(new File(nodesFilePath.toAbsolutePath().toString()));
      reader = new BufferedReader(new InputStreamReader(in));
      StringBuilder currentFile = new StringBuilder();
      String oldMiner = null;
                  while ((line = reader.readLine()) != null) {
                        currentFile.append(line+"\n");
          String[] lineTokens = line.split(";");
          if(lineTokens.length==3){
            oldMiner = lineTokens[0];
          }
      }
      reader.close();

      if(oldMiner!=null){
            currentFile = new StringBuilder(currentFile.toString().replace(";\n"+oldMiner+";1;MINER;\n",
";\n"+oldMiner+";1;\n"));
      }
      //define new miner
      currentFile = new StringBuilder(currentFile.toString().replace(";\n"+newMiner+";1;\n",
";\n"+newMiner+";1;MINER;\n"));
            FileOutputStream writer = new FileOutputStream(nodesFilePath.toAbsolutePath().toString());
                  writer.write(currentFile.toString().getBytes());
                  writer.close();

                  return newMiner;
            }

            //Creation of new transaction (choose sender and receiver ID)
            public static Transaction CreateNewTransaction(Path nodesFilePath) throws Exception {
                  System.out.println("Creating New Transaction...");

                  if (!Files.exists(nodesFilePath)){
                        System.out.println("The file "+nodesFilePath.toUri().toString()+" does not exist.");
                        throw new Exception();
                  }

                  Long timestamp =  new Date().getTime();
                  Double trustLevel = 0.0;

                  List<String> clusterNodes = new ArrayList<String>();

                  InputStream in = new FileInputStream(new File(nodesFilePath.toAbsolutePath().toString()));
      BufferedReader reader = new BufferedReader(new InputStreamReader(in));
      String line;
                  while ((line = reader.readLine()) != null) {
          String[] lineTokens = line.split(";");
          if(lineTokens.length>1 && lineTokens[1].equals("1")){
            clusterNodes.add(lineTokens[0]);
          }
      }
      reader.close();
```

```
	Collections.shuffle(clusterNodes);
	String senderId = clusterNodes.get(0);
	String receiverId = clusterNodes.get(1);

			Transaction newTransaction = new Transaction(timestamp, senderId, trustLevel, receiverId);
			return newTransaction;
		}


		public static void CheckTrustLevelAndVoting(Transaction trans,Path transactionsFilePath, Path
nodesFilePath, double trustLevelThreshold, UseCaseLogger logger) throws Exception {
			System.out.println("Checking Trust Level And Voting...");

			if (!Files.exists(nodesFilePath)){
				System.out.println("The file "+nodesFilePath.toUri().toString()+" does not exist.");
				throw new Exception();
			}

			if (!Files.exists(transactionsFilePath)){
				System.out.println("The file "+transactionsFilePath.toUri().toString()+" does not exist.");
				throw new Exception();
			}

			HashMap<String, Transaction> transactions = new HashMap<String, Transaction>();
			HashSet<String> clusterNodes = new HashSet<String>();
			String miner = new String();


			InputStream in = new FileInputStream(new File(nodesFilePath.toAbsolutePath().toString()));
	BufferedReader reader = new BufferedReader(new InputStreamReader(in));
	StringBuilder currentNodesFile = new StringBuilder();
	String line;
			while ((line = reader.readLine()) != null) {
				currentNodesFile.append(line+"\n");

	   String[] lineTokens = line.split(";");
	   if(lineTokens.length>1 && lineTokens[1].equals("1")){
	     clusterNodes.add(lineTokens[0]);
	     if(lineTokens.length>2 && lineTokens[2].equals("MINER")){
			miner = lineTokens[0];
	     }
	   }
	}
	reader.close();

	if(logger==null){
				logger = new UseCaseLogger();
				 logger.setNodesFileInitial(currentNodesFile);
			}

	in = new FileInputStream(new File(transactionsFilePath.toAbsolutePath().toString()));
	reader = new BufferedReader(new InputStreamReader(in));
	StringBuilder prevTransFile = new StringBuilder();
			while ((line = reader.readLine()) != null) {
				prevTransFile.append(line+"\n");

	   String[] lineTokens = line.split(";");
	   if(lineTokens.length!=4){
	     System.out.println("The transactions file is not properly constructed.");
	     reader.close();
	     throw new Exception();
	   }

	   //double trustLevelTmp = Math.round(Double.valueOf(lineTokens[2]) * 1000) / 1000;
```

```
        Transaction t = new Transaction(Long.valueOf(lineTokens[0]),lineTokens[1], Double.valueOf(lineTokens[2]),
lineTokens[3]) ;
        if (!clusterNodes.contains(t.getSenderId())) continue;
        if(transactions.containsKey(lineTokens[1])){
          if(transactions.get(lineTokens[1]).getTimestamp()<t.getTimestamp()){
                transactions.put(lineTokens[1], t);
          }
        } else {
          transactions.put(lineTokens[1], t);
        }
      }
                reader.close();

                logger.setTransFileInitial(prevTransFile);

                if(miner.equals(trans.getSenderId())){
                        logger.setPrevMiner(miner);
                        String newMiner = Utils.findNewMiner(transactionsFilePath, nodesFilePath, miner);
                        logger.setCurrenMiner(newMiner);
                        CheckTrustLevelAndVoting(trans, transactionsFilePath, nodesFilePath,
trustLevelThreshold, logger);
                } else {
                        if(logger.getCurrenMiner()==null){
                                logger.setCurrenMiner(miner);
                        }
                        logger.setPrevTransaction(transactions.get(trans.getSenderId()));
                        if(transactions.get(trans.getSenderId()).getTrustLevel()>=trustLevelThreshold){
                                System.out.println("Successful transaction
"+transactions.get(trans.getSenderId()).getTrustLevel());
                                //double trustLevel = Utils.generateRate();
                                double trustLevel = (double)Math.round((Utils.generateRate()) * 1000) / 1000;
                                logger.setRate(String.valueOf(trustLevel));

                                trustLevel = transactions.get(trans.getSenderId()).getTrustLevel()+trustLevel;

                                //Check that trust level won't be over 1
                                if (trustLevel>1){
                                        trustLevel=1;
                                }

                        StringBuilder currentTransFile = new StringBuilder(prevTransFile);

currentTransFile.append(trans.getTimestamp()+";"+trans.getSenderId()+";"+trustLevel+";"+trans.getReceiverId()+";\n"
);
                                FileOutputStream writer = new
FileOutputStream(transactionsFilePath.toAbsolutePath().toString());
                                writer.write(currentTransFile.toString().getBytes());
                                writer.close();

                                logger.setTransFileFinal(currentTransFile);
                                trans.setTrustLevel(trustLevel);
                                logger.setCurrentTransaction(trans);

                        } else {
                                logger.setCurrentTransaction(trans);
                                System.out.println("Unsuccessful transaction
"+transactions.get(trans.getSenderId()).getTrustLevel());
                        }

                        in = new FileInputStream(new File(nodesFilePath.toAbsolutePath().toString()));
                reader = new BufferedReader(new InputStreamReader(in));
                currentNodesFile = new StringBuilder();
                        while ((line = reader.readLine()) != null) {
                                currentNodesFile.append(line+"\n");
                }
                reader.close();
```

```
                logger.setNodesFileFinal(currentNodesFile);
                        Utils.logTransaction(logger);
                }

        }

        //Generate the reciever's rating
        public static double generateRate(){
                Random rng = new Random();
                //double trustLevel = ((rng.nextDouble()-0.5)*0.1);
                double trustLevel = (double)Math.round(((rng.nextDouble()-0.5)*0.1) * 1000) / 1000;
                return trustLevel;
        }

        //Choose the case
        public static UseCase identifyUseCase(UseCaseLogger logger){
                if(logger.getPrevMiner()!= null && !logger.getPrevMiner().equals(logger.getCurrenMiner())){
                        return UseCase.CASE_4;
                } else if (logger.getCurrentTransaction() != null &&
logger.getCurrentTransaction().getTrustLevel()<0.55 && logger.getRate()==null) {
                        return UseCase.CASE_2;
                } else {
                        if(logger.getCurrentTransaction().getTrustLevel()>=0.55){
                                if(Double.valueOf(logger.getRate())>=0.0){
                                        return UseCase.CASE_1_1;
                                } else {
                                        return UseCase.CASE_1_2;
                                }
                        } else {
                                return UseCase.CASE_3;
                        }
                }
        }

        //building the cases
        public static void logTransaction(UseCaseLogger logger) throws Exception {
                UseCase useCase = Utils.identifyUseCase(logger);
                Path folderPath;
                File folder;
                FileOutputStream writer;
                System.out.println(logger.getCurrentTransaction().getTimestamp());
                switch (useCase){
                case CASE_1_1:
                        folderPath = Paths.get(UseCase.CASE_1_1.getUseCaseFolder().toString(),
logger.getCurrentTransaction().getTimestamp().toString());
                        folder = new File(folderPath.toUri());
                        if(!folder.exists()){
                                folder.mkdirs();
                        }

                        writer = new
FileOutputStream(folderPath.toAbsolutePath().toString()+"//nodesListInitial.txt");
                        writer.write(logger.getNodesFileInitial().toString().getBytes());
                        writer.close();

                        writer = new
FileOutputStream(folderPath.toAbsolutePath().toString()+"//nodesListFinal.txt");
                        writer.write(logger.getNodesFileFinal().toString().getBytes());
                        writer.close();

                        writer = new FileOutputStream(folderPath.toAbsolutePath().toString()+"//transInitial.txt");
                        writer.write(logger.getTransFileInitial().toString().getBytes());
                        writer.close();

                        writer = new FileOutputStream(folderPath.toAbsolutePath().toString()+"//transFinal.txt");
                        writer.write(logger.getTransFileFinal().toString().getBytes());
```

```
                        writer.close();

                        StringBuilder transactionDetails = new StringBuilder();
                        transactionDetails.append("Initial Miner:"+logger.getPrevMiner()+"\n");
                        transactionDetails.append("Final Miner:"+logger.getCurrenMiner()+"\n");
                        transactionDetails.append("Sender
Id:"+logger.getCurrentTransaction().getSenderId()+"\n");
                        transactionDetails.append("Receiver
Id:"+logger.getCurrentTransaction().getReceiverId()+"\n");
                        transactionDetails.append("Successful transaction\n");
                        transactionDetails.append("Initial trust
level:"+logger.getPrevTransaction().getTrustLevel()+"\n");
                        transactionDetails.append("Rating:"+logger.getRate()+"\n");
                        transactionDetails.append("Final trust
level:"+logger.getCurrentTransaction().getTrustLevel()+"\n");
                        writer = new
FileOutputStream(folderPath.toAbsolutePath().toString()+"//transactionDetails.txt");
                        writer.write(transactionDetails.toString().getBytes());
                        writer.close();

                        break;
                case CASE_1_2:
                        folderPath = Paths.get(UseCase.CASE_1_2.getUseCaseFolder().toString(),
logger.getCurrentTransaction().getTimestamp().toString());
                        folder = new File(folderPath.toUri());
                        if(!folder.exists()){
                                folder.mkdirs();
                        }

                        writer = new
FileOutputStream(folderPath.toAbsolutePath().toString()+"//nodesListInitial.txt");
                        writer.write(logger.getNodesFileInitial().toString().getBytes());
                        writer.close();

                        writer = new
FileOutputStream(folderPath.toAbsolutePath().toString()+"//nodesListFinal.txt");
                        writer.write(logger.getNodesFileFinal().toString().getBytes());
                        writer.close();

                        writer = new FileOutputStream(folderPath.toAbsolutePath().toString()+"//transInitial.txt");
                        writer.write(logger.getTransFileInitial().toString().getBytes());
                        writer.close();

                        writer = new FileOutputStream(folderPath.toAbsolutePath().toString()+"//transFinal.txt");
                        writer.write(logger.getTransFileFinal().toString().getBytes());
                        writer.close();

                        transactionDetails = new StringBuilder();
                        transactionDetails.append("Initial Miner:"+logger.getPrevMiner()+"\n");
                        transactionDetails.append("Final Miner:"+logger.getCurrenMiner()+"\n");
                        transactionDetails.append("Sender
Id:"+logger.getCurrentTransaction().getSenderId()+"\n");
                        transactionDetails.append("Receiver
Id:"+logger.getCurrentTransaction().getReceiverId()+"\n");
                        transactionDetails.append("Successful transaction\n");
                        transactionDetails.append("Initial trust
level:"+logger.getPrevTransaction().getTrustLevel()+"\n");
                        transactionDetails.append("Rating:"+logger.getRate()+"\n");
                        transactionDetails.append("Final trust
level:"+logger.getCurrentTransaction().getTrustLevel()+"\n");
                        writer = new
FileOutputStream(folderPath.toAbsolutePath().toString()+"//transactionDetails.txt");
                        writer.write(transactionDetails.toString().getBytes());
                        writer.close();
                        break;
                case CASE_2:
```

```
                folderPath = Paths.get(UseCase.CASE_2.getUseCaseFolder().toString(),
logger.getCurrentTransaction().getTimestamp().toString());
                folder = new File(folderPath.toUri());
                if(!folder.exists()){
                        folder.mkdirs();
                }

                writer = new
FileOutputStream(folderPath.toAbsolutePath().toString()+"//nodesListInitial.txt");
                writer.write(logger.getNodesFileInitial().toString().getBytes());
                writer.close();

                writer = new
FileOutputStream(folderPath.toAbsolutePath().toString()+"//nodesListFinal.txt");
                writer.write(logger.getNodesFileFinal().toString().getBytes());
                writer.close();

                writer = new FileOutputStream(folderPath.toAbsolutePath().toString()+"//transInitial.txt");
                writer.write(logger.getTransFileInitial().toString().getBytes());
                writer.close();

                writer = new FileOutputStream(folderPath.toAbsolutePath().toString()+"//transFinal.txt");
                writer.write(logger.getTransFileFinal().toString().getBytes());
                writer.close();

                transactionDetails = new StringBuilder();
                transactionDetails.append("Initial Miner:"+logger.getPrevMiner()+"\n");
                transactionDetails.append("Final Miner:"+logger.getCurrenMiner()+"\n");
                transactionDetails.append("Sender
Id:"+logger.getCurrentTransaction().getSenderId()+"\n");
                transactionDetails.append("Receiver
Id:"+logger.getCurrentTransaction().getReceiverId()+"\n");
                transactionDetails.append("Unsuccessful transaction\n");
                transactionDetails.append("Initial trust
level:"+logger.getPrevTransaction().getTrustLevel()+"\n");
//              transactionDetails.append("Rating:"+logger.getRate()+"\n");
//              transactionDetails.append("Final trust
level:"+logger.getCurrentTransaction().getTrustLevel()+"\n");
                writer = new
FileOutputStream(folderPath.toAbsolutePath().toString()+"//transactionDetails.txt");
                writer.write(transactionDetails.toString().getBytes());
                writer.close();

                break;
        case CASE_3:
                folderPath = Paths.get(UseCase.CASE_3.getUseCaseFolder().toString(),
logger.getCurrentTransaction().getTimestamp().toString());
                folder = new File(folderPath.toUri());
                if(!folder.exists()){
                        folder.mkdirs();
                }

                writer = new
FileOutputStream(folderPath.toAbsolutePath().toString()+"//nodesListInitial.txt");
                writer.write(logger.getNodesFileInitial().toString().getBytes());
                writer.close();

                writer = new
FileOutputStream(folderPath.toAbsolutePath().toString()+"//nodesListFinal.txt");
                writer.write(logger.getNodesFileFinal().toString().getBytes());
                writer.close();

                writer = new FileOutputStream(folderPath.toAbsolutePath().toString()+"//transInitial.txt");
                writer.write(logger.getTransFileInitial().toString().getBytes());
                writer.close();
```

```
                writer = new FileOutputStream(folderPath.toAbsolutePath().toString()+"//transFinal.txt");
                writer.write(logger.getTransFileFinal().toString().getBytes());
                writer.close();

                transactionDetails = new StringBuilder();
                transactionDetails.append("Initial Miner:"+logger.getPrevMiner()+"\n");
                transactionDetails.append("Final Miner:"+logger.getCurrenMiner()+"\n");
                transactionDetails.append("Sender
Id:"+logger.getCurrentTransaction().getSenderId()+"\n");
                transactionDetails.append("Receiver
Id:"+logger.getCurrentTransaction().getReceiverId()+"\n");
                transactionDetails.append("Successful transaction\n");
                transactionDetails.append("Initial trust
level:"+logger.getPrevTransaction().getTrustLevel()+"\n");
                transactionDetails.append("Rating:"+logger.getRate()+"\n");
                transactionDetails.append("Final trust
level:"+logger.getCurrentTransaction().getTrustLevel()+"\n");
                writer = new
FileOutputStream(folderPath.toAbsolutePath().toString()+"//transactionDetails.txt");
                writer.write(transactionDetails.toString().getBytes());
                writer.close();

                break;
            case CASE_4:
                folderPath = Paths.get(UseCase.CASE_4.getUseCaseFolder().toString(),
logger.getCurrentTransaction().getTimestamp().toString());
                folder = new File(folderPath.toUri());
                if(!folder.exists()){
                        folder.mkdirs();
                }

                writer = new
FileOutputStream(folderPath.toAbsolutePath().toString()+"//nodesListInitial.txt");
                writer.write(logger.getNodesFileInitial().toString().getBytes());
                writer.close();

                writer = new
FileOutputStream(folderPath.toAbsolutePath().toString()+"//nodesListFinal.txt");
                writer.write(logger.getNodesFileFinal().toString().getBytes());
                writer.close();

                writer = new FileOutputStream(folderPath.toAbsolutePath().toString()+"//transInitial.txt");
                writer.write(logger.getTransFileInitial().toString().getBytes());
                writer.close();

                writer = new FileOutputStream(folderPath.toAbsolutePath().toString()+"//transFinal.txt");
                writer.write(logger.getTransFileFinal().toString().getBytes());
                writer.close();

                transactionDetails = new StringBuilder();
                transactionDetails.append("Initial Miner:"+logger.getPrevMiner()+"\n");
                transactionDetails.append("Final Miner:"+logger.getCurrenMiner()+"\n");
                transactionDetails.append("Sender
Id:"+logger.getCurrentTransaction().getSenderId()+"\n");
                transactionDetails.append("Receiver
Id:"+logger.getCurrentTransaction().getReceiverId()+"\n");
                if(logger.getCurrentTransaction().getTrustLevel()==0.0){
                        transactionDetails.append("Unsuccessful transaction\n");
                        transactionDetails.append("Initial trust
level:"+logger.getPrevTransaction().getTrustLevel()+"\n");
                } else {
                        transactionDetails.append("Successful transaction\n");
                        transactionDetails.append("Initial trust
level:"+logger.getPrevTransaction().getTrustLevel()+"\n");
                        transactionDetails.append("Rating:"+logger.getRate()+"\n");
```

```
                                        transactionDetails.append("Final trust
level:"+logger.getCurrentTransaction().getTrustLevel()+"\n");
                                }
                                writer = new
FileOutputStream(folderPath.toAbsolutePath().toString()+"//transactionDetails.txt");
                                writer.write(transactionDetails.toString().getBytes());
                                writer.close();

                                break;
                        }

                }

}
```

3) Transaction.java

```
package gr.cloud.trust;

public class Transaction implements Comparable<Transaction>{

        private String senderId;
        private String receiverId;
        private double trustLevel;
        private Long timestamp;

        public Transaction(Long timestamp, String senderId, double trustLevel, String receiverId){
                this.senderId=senderId;
                this.receiverId=receiverId;
                this.trustLevel=trustLevel;
                this.timestamp=timestamp;
        }

        public String getSenderId() {
                return senderId;
        }

        public void setSenderId(String senderId) {
                this.senderId = senderId;
        }

        public String getReceiverId() {
                return receiverId;
        }

        public void setReceiverId(String receiverId) {
                this.receiverId = receiverId;
        }

        public double getTrustLevel() {
                return trustLevel;
        }

        public void setTrustLevel(double trustLevel) {
                this.trustLevel = trustLevel;
        }

        public Long getTimestamp() {
                return timestamp;
        }

        public void setTimestamp(Long timestamp) {
                this.timestamp = timestamp;
        }
```

```
	@Override
	public int compareTo(Transaction t) {
		if(this.getTrustLevel()>t.getTrustLevel()){
			return -1;
		} else if (this.getTrustLevel()<t.getTrustLevel()){
			return 11;
		} else {
			return 0;
		}
	}

}
```

4) UseCase.java

```
package gr.cloud.trust;

import java.nio.file.Path;
import java.nio.file.Paths;

public enum UseCase {

	CASE_1_1(Paths.get("C:\\Users\\nikos\\workspace\\TrustInCloud\\Snapshots\\CASE_1_1")),
	CASE_1_2(Paths.get("C:\\Users\\nikos\\workspace\\TrustInCloud\\Snapshots\\CASE_1_2")),
	CASE_2(Paths.get("C:\\Users\\nikos\\workspace\\TrustInCloud\\Snapshots\\CASE_2")),
	CASE_3(Paths.get("C:\\Users\\nikos\\workspace\\TrustInCloud\\Snapshots\\CASE_3")),
	CASE_4(Paths.get("C:\\Users\\nikos\\workspace\\TrustInCloud\\Snapshots\\CASE_4"));

	Path useCaseFolder;

	private UseCase(Path useCaseFolder) {
		this.useCaseFolder = useCaseFolder;
	}


	public Path getUseCaseFolder() {
		return useCaseFolder;
	}
}
```

5) UseCaseLogger.java

```
package gr.cloud.trust;

public class UseCaseLogger {
	StringBuilder nodesFileInitial = new StringBuilder();
	StringBuilder transFileInitial = new StringBuilder();
	StringBuilder nodesFileFinal = new StringBuilder();
	StringBuilder transFileFinal = new StringBuilder();
	Transaction prevTransaction;
	Transaction currentTransaction;
	String rate;
	String prevMiner;
	String currenMiner;

	public UseCaseLogger() {}

	public StringBuilder getNodesFileInitial() {
		return nodesFileInitial;
	}
```

```
public void setNodesFileInitial(StringBuilder nodesFileInitial) {
	this.nodesFileInitial = nodesFileInitial;
}

public StringBuilder getTransFileInitial() {
	return transFileInitial;
}

public void setTransFileInitial(StringBuilder transFileInitial) {
	this.transFileInitial = transFileInitial;
}

public StringBuilder getNodesFileFinal() {
	return nodesFileFinal;
}

public void setNodesFileFinal(StringBuilder nodesFileFinal) {
	this.nodesFileFinal = nodesFileFinal;
}

public StringBuilder getTransFileFinal() {
	return transFileFinal;
}

public void setTransFileFinal(StringBuilder transFileFinal) {
	this.transFileFinal = transFileFinal;
}

public Transaction getPrevTransaction() {
	return prevTransaction;
}

public void setPrevTransaction(Transaction prevTransaction) {
	this.prevTransaction = prevTransaction;
}

public Transaction getCurrentTransaction() {
	return currentTransaction;
}

public void setCurrentTransaction(Transaction currentTransaction) {
	this.currentTransaction = currentTransaction;
}

public String getRate() {
	return rate;
}

public void setRate(String rate) {
	this.rate = rate;
}

public String getPrevMiner() {
	return prevMiner;
}

public void setPrevMiner(String prevMiner) {
	this.prevMiner = prevMiner;
}

public String getCurrenMiner() {
	return currenMiner;
}

public void setCurrenMiner(String currenMiner) {
	this.currenMiner = currenMiner;}}
```