ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

"ΠΛΗΡΟΦΟΡΙΚΗ"

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Κατανεμημένη Διαχείριση Εργασιών στις Παρυφές του Δικτύου

Δημήτρης Χ. Φλουρής

Επιβλέπων: **Χατζηευθυμιάδης Ευστάθιος,** Καθηγητής Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών

ΑΘΗΝΑ

ΟΚΤΩΒΡΙΟΣ 2020

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Κατανεμημένη Διαχείριση Εργασιών στις Παρυφές του Δικτύου

(Distributed Task Management at the Edge of the Network)

Δημήτρης Χ. Φλουρής
Α.Μ.: cs2180023

ΕΠΙΒΛΕΠΩΝ: **Χατζηευθυμιάδης Ευστάθιος,** Καθηγητής Εθνικού Καποδιστριακού Πανεπιστημίου Αθηνών

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ: **Κολομβάτσος Κωνσταντίνος,** Επίκουρος Καθηγητής Πανεπιστημίου Θεσσαλίας
Σαράντης Πασκαλής, Εργαστηριακό Διδακτικό Προσωπικό Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών

Οκτώβριος 2020

ΠΕΡΙΛΗΨΗ

Παρατηρούμε ότι, όλο και περισσότερες συσκευές του Διαδικτύου των Πραγμάτων (ΔτΠ) γίνονται απαραίτητες στην καθημερινότητα μας. Η αυξημένη χρήση τέτοιου είδους συσκευών οδηγεί στην παραγωγή μεγάλου όγκου δεδομένων. Ως εκ τούτου, υφίσταται σήμερα πληθώρα εφαρμογών για τη διαχείριση αυτών των δεδομένων. Πέραν από τις συσκευές οι οποίες επεξεργάζονται και αξιοποιούν δεδομένα καταλήγοντας σε συμπεράσματα, υπάρχουν και οι συσκευές στο πεδίο του ΔτΠ οι οποίες είναι αναγκαίο να λαμβάνουν ορθές αποφάσεις ακαριαία και σε πραγματικό χρόνο. Οι επιπτώσεις της καθυστέρησης λήψης αποφάσεων σε πραγματικό χρόνο από τέτοια συστήματα είναι καθοριστικές. Χαρακτηριστικό είναι το παράδειγμα των αυτόνομων οχημάτων στην περίπτωση των οποίων έρχονται αντιμέτωπα με άγνωστα γεγονότα για τα οποία θα πρέπει να ζητήσουν δεδομένα από το σύστημα για να ανταποκριθούν ανάλογα. Αν το σύστημα καθυστερήσει, υπάρχει περίπτωση ο χρήστης, απλώς να αναμένει την ανταπόκριση του συστήματος. Υπάρχει ωστόσο και η περίπτωση, η καθυστέρηση του συστήματος να οδηγήσει σε τραγική κατάληξη. Έτσι λοιπόν, το επιχείρημα που δημιουργήθηκε είναι ότι κατανέμοντας σωστά τα δεδομένα έχουμε πρότερη γνώση για το που φιλοξενούνται. Συνεπώς γνωρίζονται το τι δεδομένα υπάρχουν στον κάθε κόμβο, έτσι μπορούμε να κατευθύνουμε αποδοτικά τα ερωτήματα που εισέρχονται στο δίκτυο. Ακόμη έχοντας τα δεδομένα κοντά στις πηγές τους, δεν χρειάζεται να ταξιδεύουν μέχρι το υπολογιστικό νέφος οπότε μειώνεται και η καθυστέρηση του δικτύου. Μια τέτοια προσέγγιση είναι και αυτή του Edge Computing. Στην παρούσα εργασία, λαμβάνοντας υπόψιν την υφιστάμενη βιβλιογραφία, δημιουργήσαμε και εκτελέσαμε την προσομοίωση του δικτύου και προτείνουμε έναν αλγόριθμο διαμοιρασμού δεδομένων σε κόμβους. Στόχος μας είναι να διαμοιραστούν όσο το δυνατόν πιο γρήγορα αλλά και ομοιόμορφα τα δεδομένα στους κόμβους. Ανάλογα με τις απαιτήσεις της συγκεκριμένης συσκευής χρησιμοποιούμε κάποιες μεθόδους ομαδοποίησης κόμβων για να μειώσουμε το εύρος αναζήτησης. Τέλος, χρησιμοποιώντας μετρικές συναρτήσεις, υπολογίζουμε τον κατάλληλο κόμβο για να εισάγουμε τις συγκεκριμένες τιμές. Η μελέτη αυτή περιέχει την περιγραφή του αλγορίθμου που προτείνουμε και την αξιολόγησή του μέσα από ένα πλήθος προσομοιώσεων, που υιοθετούν πειραματικά σενάρια.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Διαδίκτυο των Πραγμάτων

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: διαχείριση διεργασιών, μεγάλος όγκος δεδομένων, αλγόριθμοι ομαδοποίησης, μετρικές συναρτήσεις, κόμβοι δικτύου, κατανομή δεδομένων, δίκτυο.

ABSTRACT

It becomes apparent that more and more devices that belong in the Διαδίκτυο των Πραγμάτων (IoT) are evolving into an imperative part of our everyday lives. The increased use of this type of devices has as a result the production of large volumes of data and consequently the need for a plethora of applications to control these data. Despite there being devices, where by using the aforementioned data, manage to reach to their own specific conclusions, there are certain devices under the network of the IoT, which should have the ability to make accurate and instantaneous decisions in real time. The consequences of a potential delay to respond are serious especially in a scenario where an autonomous vehicle encounters a situation which requires it to respond using the system's data. In case of a delayed response, the best-case scenario would have the user waiting patiently for the device to respond, while the worst-case scenario could involve severely tragic consequences. The need for better data allocation was created for this exact purpose: The efficient data allocation is also targeting to collect 'similar' data to the same repositories, thus, to have the opportunity to build effective query management plans beforehand. The concept of edge computing aims to achieve this goal. This research will go beyond the analysis of the relevant bibliography, by creating a network simulation where data will be distributed into edge nodes, suggesting a data distribution algorithm into those nodes. Our goal is to reach the fastest and most consistent way of achieving data distribution into edge nodes. Depending on the requirements of each specific device, certain methods of clustering edge nodes will be used in order to decrease the range of search while through using clustering algorithms, we will attempt to calculate the appropriate edge node in order to allocate the specific data. This research describes our suggested algorithm and evaluates its performance using a plethora of simulations which implement experimental scenarios.

SUBJECT AREA: Διαδίκτυο των Πραγμάτων

KEYWORDS: distributed task management, big data, classification, clustering, standard deviation, edge computing, edge nodes, distributed data, network

Την παρούσα διπλωματική θα ήθελα να την αφιερώσω στους γονείς μου για την στήριξη που είχα όλα αυτά τα χρόνια μέχρι την περάτωση των σπουδών μου.

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα μελέτη συνιστά τη διπλωματική μου εργασία στο πλαίσιο του μεταπτυχιακού προγράμματος «Πληροφορική», του τμήματος Πληροφορικής και Τηλεπικοινωνιών. Με την περάτωση της παρούσας διπλωματικής εργασίας θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή, κ. Κολομβάτσο Κωνσταντίνο, για τις πολύτιμες υποδείξεις και συμβουλές του, καθώς και για την προθυμία του να με βοηθήσει να εμπλουτίσω τις γνώσεις μου καθ' όλη την διάρκεια των φοιτητικών μου χρόνων. Επίσης ένα μεγάλο ευχαριστώ για την άψογη συνεργασία, την εμπιστοσύνη και εκτίμηση που μου έδειξε, σε όλη τη διάρκεια της μελέτης μου.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΠΡΟΛΟΓΟΣ

Η παρούσα διπλωματική εργασία εκπονήθηκε στο πλαίσιο του μεταπτυχιακού προγράμματος σπουδών με τίτλο «Πληροφορική», του τμήματος Πληροφορικής και Τηλεπικοινωνιών, του Εθνικού Καποδιστριακού Πανεπιστημίου Αθηνών, υπό την επίβλεψη του καθηγητή κ. Κολομβάτσου Κωνσταντίνου. Ο σκοπός της έρευνας ήταν η ευρύτερη μελέτη και εμβάθυνση στο πεδίο του Διαδικτύου των Πραγμάτων (ΔτΠ), αλλά και η ανάδειξη των ζητημάτων που ανακύπτουν λόγω της παραγωγής μεγάλου όγκου δεδομένων. Η ραγδαία αύξηση των δεδομένων, έγκειται στην αντίστοιχη αύξηση του αριθμού των συσκευών στο ΔτΠ, τα τελευταία χρόνια και με τις εκτιμήσεις να πολλαπλασιάζονται αναφορικά με το μέλλον. Η προστιθέμενη αξία της παρούσας μελέτης βασίζεται στην προσπάθεια επίλυσης του προβλήματος της καθυστέρησης ανταπόκρισης των υφιστάμενων προσεγγίσεων, ακολουθώντας τη μέθοδο διαμοιρασμού δεδομένων σε κόμβους Edge Computing. Η παρούσα μελέτη προτείνει την αξιοποίηση συγκεκριμένου αλγορίθμου διαμοιρασμού δεδομένων σε κόμβους, δίνοντας την ενδεδειγμένη λύση στο πρόβλημα. Επισημαίνεται ωστόσο ότι, υπάρχουν περιθώρια βελτίωσης του αλγορίθμου που χρησιμοποιείται στην εκπόνηση της παρούσας εργασίας και είναι αναγκαίο να συνεχιστεί η ερευνητική προσπάθεια που καταβάλλεται προς επίλυση των ζητημάτων που ανακύπτουν από την ευρεία χρήση συσκευών του ΔτΠ.

1. ΕΙΣΑΓΩΓΗ

1.1 Περιγραφή Προβλήματος.

Στις μέρες μας θεωρείται δεδομένη και αναγκαία η χρήση συσκευών που αξιοποιούνται στο πλαίσιο του Διαδικτύου των Πραγμάτων, όπως είναι για παράδειγμα τα smartphones, smartwatches, συσκευές τύπου smart home, κάμερες, αισθητήρες μέτρησης, αυτόνομα οχήματα. Η συνεχής και καθημερινή αλληλεπίδραση του ανθρώπου με την τεράστια γκάμα των συσκευών του ΔτΠ και η αδιάκοπη συλλογή δεδομένων, έχει ως αποτέλεσμα την ραγδαία αύξηση του όγκου των δεδομένων. Ωστόσο, η επεξεργασία του όγκου αυτού των δεδομένων παρουσιάζει κάποιες αδυναμίες. Ενδεικτικά, όσον αφορά τις εφαρμογές που παρουσιάζουν ευαισθησία στο χρόνο, ήταν επιτακτική ανάγκη να δημιουργηθεί μια διαφορετική προσέγγιση ενός μοντέλου διαχωρισμού και συνεπώς, επεξεργασίας των δεδομένων.

Το Edge Computing αποτελεί μια ικανοποιητική λύση στην κατανομή των δεδομένων. Η γενική του ιδέα, βασίζεται στον τρόπο που κατανέμω τα δεδομένα μου σε ένα δίκτυο από κόμβους (Edge Nodes) λαμβάνοντας υπόψιν κάποια κριτήρια. Με το Edge Computing ο χρόνος απόκρισης και επεξεργασίας των δεδομένων μειώνεται αισθητά νοούμενου ότι η επεξεργασία γίνεται πιο κοντά στις πηγές των δεδομένων. Στην παρούσα εργασία, έχουμε εστιάσει στην αποθήκευση των δεδομένων σε ένα δίκτυο με Edge κόμβους και στο πως μειώνεται ο χρόνος απόκρισης ενός συστήματος και απάντησης στα αιτήματα των χρηστών-συσκευών. Στο πλαίσιο λειτουργίας του συγκεκριμένου Edge δικτύου που δημιουργήσαμε, τα όμοια δεδομένα ομαδοποιούνται και αναλόγως της συχνότητας χρήσης τους αποθηκεύονται τοπικά σε κάθε κόμβο. Αυτό έχει ως αποτέλεσμα όχι μόνο τη μείωση του φόρτου εργασίας του κάθε κόμβου αλλά και τη μείωση της καθυστέρησης ολόκληρου του δικτύου αφού μπορούμε πλέον να κατευθύνουμε τα ερωτήματα μόνο στους κόμβους που ταιριάζουν ελαχιστοποιώντας και τη κίνηση μέσα στο δίκτυο. Ακολούθως θα προβούμε σε ανάλυση της αρχιτεκτονικής και του τρόπου εφαρμογής του Edge Computing μέσα σε ένα δίκτυο.

1.2 Σύντομη περιγραφή του αλγορίθμου

Στη μελέτη που παρουσιάζουμε, προσομοιώνουμε ένα δίκτυο από κόμβους (nodes) με τη χρήση της γλώσσας προγραμματισμού Python. Οι κόμβοι θα δέχονται πολυδιάστατα δεδομένα. Εστιάζοντας αρχικά σε έναν κόμβο, ο οποίος λαμβάνει μια εγγραφή από

δεδομένα καλείται να αποφασίσει αν θα αποθηκεύσει την εν λόγω εγγραφή τοπικά, ή αν θα τη στείλει σε κάποιον άλλο κόμβο του δικτύου. Στόχος είναι να κρατηθεί σε υψηλά επίπεδα η ομοιομορφία, του συνόλου δεδομένων στον κάθε κόμβο. Τη στιγμή που ο αρχικός κόμβος αποφασίσει ότι θα πρέπει να στείλει τα δεδομένα σε άλλο κόμβο, ο κάθε κόμβος του δικτύου, θα στείλει μια αναφορά με τα στατιστικά του ενσωματώνοντας σε αυτήν και τη διαδικασία Feature Learning. Το Feature Learning αφορά τη διαλογή των σημαντικών χαρακτηριστικών των δεδομένων μας και θα αναλυθεί διεξοδικά στη συνέχεια. Τα κριτήρια επιλογής του κατάλληλου κόμβου είναι τα εξής: (i) η ομοιότητα του συνόλου των δεδομένων του κάθε κόμβο σε σχέση με την εισερχόμενη εγγραφή, (ii) το κόστος μεταφοράς των δεδομένων από τον αρχικό κόμβο και (iii) η χρονική σειρά έκδοσης της αναφοράς των στατιστικών για τον κάθε κόμβο. Στη συνέχεια, γίνεται ομαδοποίηση κόμβων, έτσι ώστε να ελεγχθούν μόνο οι σημαντικοί κόμβοι. Ακολούθως χρησιμοποιώντας ένα μηχανισμό ανταμοιβής δίνουμε πόντους στους κόμβους βάσει των εξής κριτηρίων: (i) υπολογίζοντας την πιθανότητα να ταιριάζουν τα δεδομένα σε αυτόν (ii) το κόστος μεταφοράς και (iii) το χρόνο έκδοσης της αναφοράς. Χρησιμοποιώντας μια αντίστροφη σιγμοειδή συνάρτηση (Sigmoid Function), υπολογίζεται το αποτέλεσμα για κάθε κόμβο και επιλέγεται ο κόμβος με το μεγαλύτερο σκορ. Στη συνέχεια, αποθηκεύεται η συγκεκριμένη εγγραφή δεδομένων στον επιλεχθέντα κόμβο. Τέλος, εκτελούνται μετρικές φόρμουλες για την εξαγωγή στατιστικών στοιχείων, όπως είναι για παράδειγμα το Standard Deviation Formula, ο χρόνος που χρειάζεται το σύστημα να αποφασίσει, ο ιδανικός αριθμός cluster και η Incluster απόσταση.

2. ΔΙΑΔΙΚΤΥΟ ΤΩΝ ΠΡΑΓΜΑΤΩΝ (ΔΙΑΔΊΚΤΥΟ ΤΩΝ ΠΡΑΓΜΆΤΩΝ)

2.1 Ορισμός

Το Διαδίκτυο των Πραγμάτων, όπως είναι εμφανές και από την ονομασία του, είναι το σύνολο των ετερογενών συστημάτων δικτύου και επικοινωνίας τα οποία παρέχουν ανοικτή πρόσβαση σε συγκεκριμένες βάσεις δεδομένων με σκοπό την ανάπτυξη και παροχή ψηφιακών υπηρεσιών. Ουσιαστικά είναι η διασύνδεση καθημερινών συσκευών, οχημάτων, κτιρίων και άλλων αντικειμένων με ενσωματωμένα ηλεκτρονικά συστήματα, λογισμικό, αισθητήρες, προσομοιωτές και διαδικτυακή συνδεσιμότητα, που επιτρέπει στα αντικείμενα αυτά να συλλέγουν και να ανταλλάσουν δεδομένα. Η έννοια Διαδίκτυο των Πραγμάτων [33] επινοήθηκε στα τέλη τις δεκαετίας του 90’ από τον Kevin Ashton [2] που μαζί με την ομάδα του κατάφεραν να συνδέσουν συσκευές στο διαδίκτυο μέσω μιας RFID ετικέτας [1]. Η τεχνολογία αυτή θεωρείται από τις μεγαλύτερες ανακαλύψεις του 21ου αιώνα, δεδομένου ότι όλο και περισσότεροι οργανισμοί και κυβερνήσεις μετατρέπουν πλήρως τις υπηρεσίες τους σε ψηφιακή μορφή χρησιμοποιώντας τεχνολογίες του Διαδίκτυο των Πραγμάτων, μεγιστοποιώντας παράλληλα και τα έσοδα τους. Αυτός είναι και ο λόγος για τον οποίο ο αριθμός των συσκευών που είναι συνδεδεμένες με το διαδίκτυο είναι αυξάνεται ραγδαία. Ενδεικτικά, οι ειδικοί εκτιμούν ότι κάθε δευτερόλεπτο 127 νέες συσκευές συνδέονται στο ΔτΠ [2]. Παρατίθεται ο αριθμός των συσκευών που βρίσκονται συνδεδεμένες και τα αντίστοιχα έσοδα της αγοράς του Διαδικτύου των Πραγμάτων, ανά έτος, και οι αναμενόμενοι αριθμοί για τα επόμενα έτη [2]:

- Το **2018** ξοδεύτηκαν περίπου $815 δισεκατομμύρια και υπήρχαν 7 δισεκατομμύρια συνδεδεμένες συσκευές.
- Το **2019** ξοδεύτηκαν περίπου $1.1 τρισεκατομμύρια και ο αριθμός των συσκευών του ΔτΠ σχεδόν τετραπλασιάστηκε σε σχέση με πέρσι.
- Μέχρι το τέλος του **2020** οι ειδικοί υπολογίζουν να έχουμε 31 δισεκατομμύρια «έξυπνες» συσκευές συνδεδεμένες με συνολικό κόστος $1.29 τρισεκατομμύρια.
- Η εκτιμώμενη τιμή για το **2021** αναμένετε να ανέβει στα 35 δισεκατομμύρια παγκοσμίως.
- Το **2025** ο αριθμός αυτός θα δεκαπλασιαστεί με έσοδα γύρω στα $11.5 τρισεκατομμύρια.

Η υιοθέτηση της τεχνολογίας αυτής, με συσκευές και εφαρμογές στο κόσμο του Διαδίκτυο των Πραγμάτων, προκάλεσε την παραγωγή και μετάδοση τεράστιου όγκου δεδομένων. Δεδομένου ότι οι χρήστες και οι υπηρεσίες τις οποίες θα εξυπηρετεί ποικίλουν, κάθε συσκευή διαφέρει σε χαρακτηριστικά αναλόγως και των καθηκόντων της. Λαμβάνοντας υπόψιν τις λειτουργίες της, κάθε συσκευή κατασκευάστηκε για να συνδυάζεται με το αντίστοιχο λογισμικό. Οι επιλογές υλικού (Hardware) σε συνδυασμό με τις επιλογές λογισμικού (Software) ποικίλουν. Το γεγονός αυτό οφείλεται στο ότι αρκετές επιλογές βρίσκονται σε open-source περιβάλλον, όπου ο καθένας μπορεί να προσθέσει το δικό του κομμάτι αναλόγως της χρήσης της συσκευής.

2.2 Αρχιτεκτονική του Διαδίκτυο των Πραγμάτων

Η φιλοσοφία πίσω από το Διαδίκτυο των Πραγμάτων είναι πράγματι σπουδαία ωστόσο, είναι ιδιαίτερα πολύπλοκη. Για να επιτευχθεί η διασύνδεση των συσκευών, είναι αναγκαίο να προηγηθεί η μελέτη και η δημιουργία μιας αποδοτικής δομής, η οποία θα αποτελέσει το θεμελιακό στοιχείο του διαδικτύου. Δυστυχώς δεν έχει συμφωνηθεί σε παγκόσμιο επίπεδο μια καθολική αρχιτεκτονική. Έτσι, υφίστανται διάφορες τρόποι υλοποίησης. Πιο κάτω θα αναφερθούμε στην τελευταία και επικρατέστερη αρχιτεκτονική που περιέχει 7 επίπεδα (layers). Περεταίρω παρατίθεται το κάθε επίπεδο και η περιγραφή της χρηστικότητας του στο Διαδίκτυο των Πραγμάτων [4-5].

Layer 1 - The Things Layer

Σε αυτό το επίπεδο βρίσκονται τοποθετημένες οι συσκευές όπως πχ. Smartphones, Αισθητήρες, Μικρο-ελεγκτές ή πλακέτες. Οι συσκευές αυτές μπορούν να είναι συνδεδεμένες με καλώδιο ή ασύρματα και λειτουργούν σαν σημεία αναφοράς (end points) καταγράφοντας δεδομένα και συλλέγοντας πληροφορίες. Περαιτέρω πληροφορίες για τις συσκευές θα αναλυθούν στην πιο επόμενη ενότητα.

Layer 2 – Connectivity / Edge Computing Layer

Στο συγκεκριμένο επίπεδο ορίζονται τα πρωτόκολλα επικοινωνίας του δικτύου που θα χρησιμοποιηθούν για τη συνδεσιμότητα στο Edge Computing. Με αυτά τα πρωτόκολλα θα μεταφέρονται οι πληροφορίες και τα δεδομένα που καταγράφηκαν από τις συσκευές στο επόμενο επίπεδο Cloud.

Layer 3 – Global Infrastructure Layer

Εδώ στην ουσία, βρίσκονται οι υπηρεσίες cloud που έχουν αποθηκευμένα διάφορα εργαλεία, τα οποία αξιοποιώντας τα δεδομένα και τις πληροφορίες που

υπάρχουν μπορεί να παρέχει χρήσιμες επιλογές στις επιχειρήσεις που βασίζονται στο ΔτΠ αλλά και να δίνει σχετικές προτάσεις στους χρήστες-πελάτες.

Layer 4 – Data Ingestion Layer

Σε αυτό το επίπεδο υπάρχει αποκλειστικά ότι αφορά τα δεδομένα. Εδώ, υπόκειται σε επεξεργασία ο μεγάλος όγκος δεδομένων (Big Data), καθαρίζεται από τυχόν άχρηστες ή κενές τιμές χρησιμοποιώντας διάφορες τεχνικές και στο τέλος αποθηκεύονται.

Layer 5 – Data Analysis Layer

Στο Data Analysis θα χρησιμοποιηθούν όλες οι τεχνικές που εφαρμόζονται στα δεδομένα, είτε για να εξάγουν κάποια αποτελέσματα (data mining), είτε για να προβλέψουν κάτι παρεμφερές χρησιμοποιώντας τεχνικές μηχανικής μάθησης. Τέλος γίνεται η σύνταξη αναφορών με βάση τα παραπάνω.

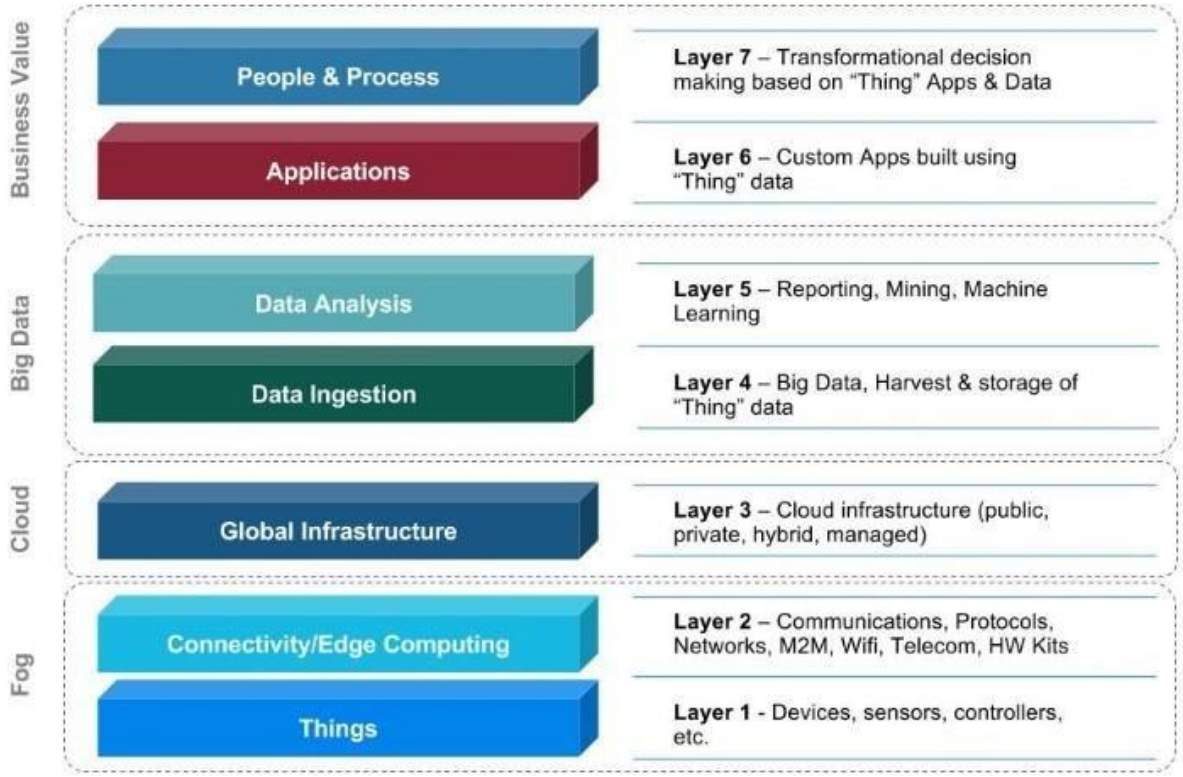
Layer 6 – The Application Layer

Σε αυτό το επίπεδο βρίσκονται οι εφαρμογές που αναπτύχθηκαν σε συγκεκριμένο λειτουργικό και ελέγχουν τις συσκευές που υπάρχουν στα άκρα του δικτύου και τα δεδομένα που παράγονται από αυτό.

Layer 7 – People and Process Layer

Σε αυτό το επίπεδο βρίσκονται οι χρήστες-πελάτες των IoT συσκευών, οι επιχειρήσεις πίσω από την υλοποίηση τους, τυχόν επιχειρησιακές συνεργασίες και αποφάσεις που λαμβάνονται σε σχέση με το Διαδίκτυο των Πραγμάτων.

Τέλος, αφού αναλύσαμε το κάθε επίπεδο ανάλογα με τη χρηστικότητα του, τα διακρίνουμε σε ομάδες όπως φαίνεται και πιο κάτω στην Εικόνα 1 [67]. Συνολικά υπάρχουν 4 κατηγορίες (Fog, Cloud, Big Data, Business Value) και τα πιο «συγγενικά» επίπεδα βρίσκονται στην ίδια ομάδα.



Εικόνα 1: Αρχιτεκτονική του Διαδικτύου των Πραγμάτων

2.3 Hardware και Software στο Διαδίκτυο των Πραγμάτων

Οι δυνατότητες του Διαδίκτυο των Πραγμάτων είναι απεριόριστες και γι' αυτό το λόγο υπάρχει πληθώρα στο υλικό το οποίο είναι διαθέσιμο και συμβατό με το δίκτυο. Σε συνδυασμό πάντα με το λογισμικό, έχουν στόχο να εκτελούν αποτελεσματικά τις λειτουργίες με βάση και τις αντίστοιχες απαιτήσεις του δικτύου. Πιο κάτω θα αναλύσουμε κάποιες επιλογές υλικού και λογισμικού που υπάρχουν.

2.3.1 Συσκευές

Smartphones

Τρανταχτό παράδειγμα των αμέτρητων δυνατοτήτων του ΔτΠ αποτελούν τα κινητά τηλέφωνα που ο καθένας μας έχει έστω και ένα στη κατοχή του. Ειδικότερα, από το 2010 και μετά, αναπτύχθηκε η τάση του να μπορείς να ελέγχεις, σχεδόν τα πάντα, μέσω της πλατφόρμας του κινητού σου. Κλασσικά παραδείγματα αποτελούν το να αποστείλεις μια παραγγελία φαγητού ή να σημειώσεις την τοποθεσία που έχεις σταθμεύσει το όχημα σου.

Wearable Συσκευές

Το επόμενο βήμα στην τεχνολογία, φέρνουν οι wearable συσκευές. Εμφανίστηκαν τα τελευταία χρόνια και παρέχουν πληροφορίες κυρίως, για την ανθρώπινη σωματική άσκηση και την ανάλυση της κατάστασης του ανθρώπινου οργανισμού. Είναι πιο μικρές και φορητές από ένα smartphone. Πιο συνηθισμένες συσκευές αυτού του είδους είναι τα smartwatches και ανερχόμενα είναι τα smart glasses, παρόλο που ακόμα υπάρχει μόνο σαν πρωτότυπο.

Αυτόνομα Οχήματα

Γύρω στο 2013 όπου έγιναν ευρέως γνωστά σε μας τα αυτόνομα οχήματα, ανέβασαν επίπεδο τις δυνατότητες που μας προσφέρουν συσκευές του ΔτΠ. Επίσης οι συγκεκριμένες συσκευές εκτός του ότι χρησιμοποιούν προϋπάρχουσες λειτουργίες του ΔτΠ είναι ένα υποσχόμενο δείγμα για το τι μπορεί να ακολουθήσει στο μέλλον.

Αισθητήρες

Οι αισθητήρες είναι συσκευές που κατά κύριο λόγο καταγράφουν αλλαγές στο περιβάλλον το οποίο θα τοποθετηθούν. Μερικά παραδείγματα αισθητήρων είναι θερμοκρασίας, υγρασίας, ήχου, φωτός, κίνησης και μπορεί να εφαρμοστούν σε επιστήμες όπως η ιατρική, ρομποτική, γεωπονία κτλ. Ανεξαρτήτως εφαρμογής, η

συγκεκριμένη συσκευή καταγράφει διάφορα γεγονότα και παράγει κάποια έξοδο από πληροφορίες. Υπήρχαν πολύ πριν δημιουργηθεί ο όρος ΔτΠ αλλά κατέχουν το μεγαλύτερο ποσοστό συσκευών του και θεωρούνται αναπόσπαστο κομμάτι του.

Ενεργοποιητές

Οι ενεργοποιητές συνήθως συνδυάζονται με αισθητήρες λόγω και της χρηστικότητας τους. Ανάλογα με τα δεδομένα που συλλέγουν οι αισθητήρες λαμβάνουν ένα αναγνωριστικό σήμα και ενεργοποιούν τη συσκευή που πρέπει να εκτελέσει κάποια ενέργεια.

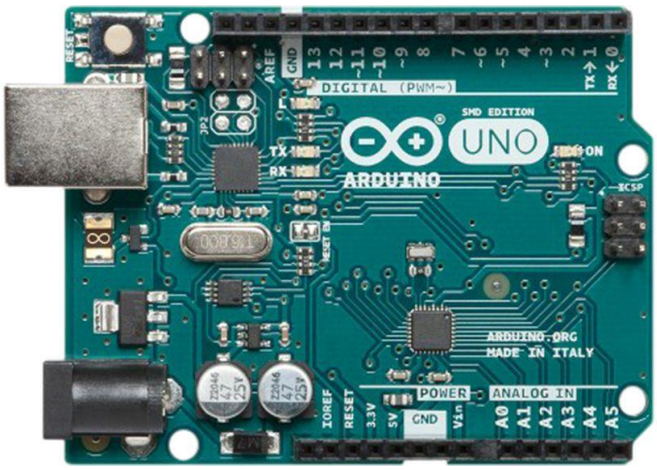
Ανεξάρτητα από τις συσκευές που προαναφέρθηκαν και λειτουργούν κυρίως στο Επίπεδο 1, υπάρχουν και οι ηλεκτρονικοί υπολογιστές όπως είναι οι πλακέτες, τα desktop και τα laptop. Αυτές οι συσκευές λειτουργούν ως εργαλεία [6] στα οποία υπάρχει το λογισμικό που χρησιμοποιείται για την ανάπτυξη των εφαρμογών του ΔτΠ. Με βάση τη χρήση που προορίζονται, συνήθως χρησιμοποιούνται πλακέτες τύπου SBC [7] (Single Board Computer) γιατί έχουν πάρα πολύ μικρό κόστος. Υπάρχει πληθώρα στην αγορά και είναι εύκολα αντικαταστάσιμες. Λαμβάνοντας υπόψιν ότι και οι προδιαγραφές που έχει το κάθε σύστημα ποικίλουν τις περισσότερες φορές οι πλατφόρμες που βασίζεται το ΔτΠ είναι open-source για να είναι εύκολα μετατρέψιμες από τον καθένα αναλόγως των απαιτήσεων τους. Στις δύο πιο κάτω ενότητες θα αναλύσουμε τις διαθέσιμες επιλογές που υπάρχουν, τόσο σε υλικό, όσο και σε λογισμικό.

2.3.2 Πλακέτες

Arduino Uno [9] [10]

Αναπτύχθηκε το 2005 στην Ιταλία από δύο καθηγητές στα πλαίσια εκπαίδευσης σε μαθητές. Το Arduino είναι μια απλή μητρική πλακέτα με ενσωματωμένους μικρο-ελεγκτές, εισόδους και εξόδους. Είναι συμβατό με τη γλώσσα προγραμματισμού Wiring που βασίζεται πάνω σε C++. Με σωστό προγραμματισμό της πλακέτας μπορείς να στείλεις τις ενέργειες που θες, και αυτό με τη σειρά του να εκτελέσει μια σειρά εντολών. Αγαπήθηκε και υποστηρίχτηκε για το λόγο ότι έχει μικρό κόστος μπορεί να τρέξει σε όλα τα περιβάλλοντα, είναι open-source και εύκολο στο προγραμματισμό του τόσο σε επίπεδο hardware όσο και σε software. Υποστηρίζεται από μεγάλη κοινότητα χρηστών και ειδικότερα των φοιτητών. Στην Εικόνα 2 βλέπουμε την πιο δημοφιλής πλακέτα, που είναι η Arduino Uno [11]. Έχει

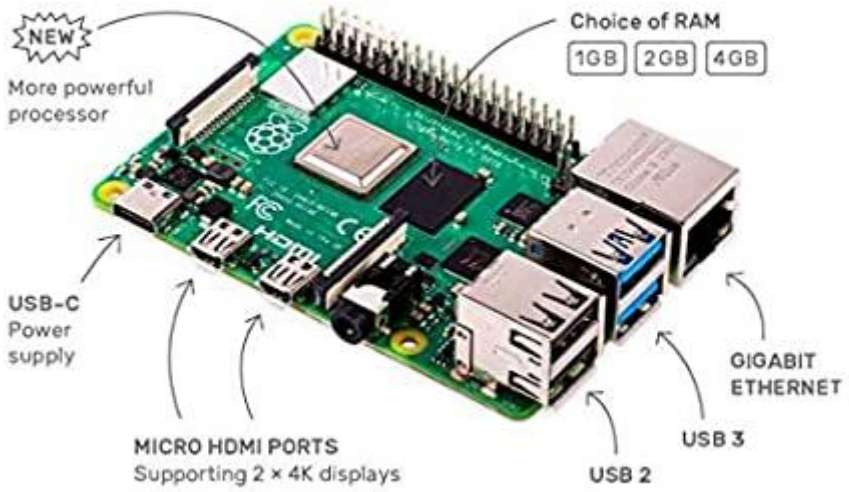
μέγεθος 68.6x53.4 mm και βάρος 25g. Διαθέτει ένα επεξεργαστή ATmega328P συγχρονισμένο στα 16MHz με χωρητικότητα 32KB όπου το 0.5KB χρησιμοποιούνται για το Bootloader και Ram στα 2KB.



Εικόνα 2: Πλακέτα Arduino UNO

Raspberry Pi [12] [13]

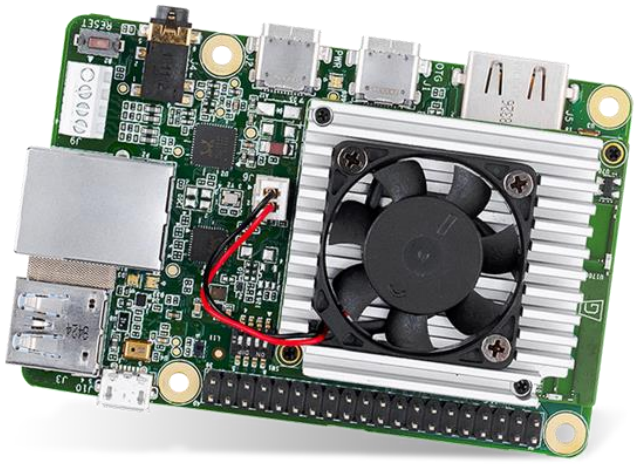
Αναπτύχθηκε το 2012 στο Ηνωμένο Βασίλειο και η ιδέα ήταν η ανάπτυξη ενός ηλεκτρονικού υπολογιστή στο μέγεθος πιστωτικής κάρτας. Προοριζόταν για χρήση κυρίως από μαθητές. Περιλαμβάνει ότι ακριβώς έχει ένας κανονικός υπολογιστής μνήμη RAM, επεξεργαστή, κάρτα γραφικών, ενσύρματη σύνδεση μέσω Ethernet και για αποθηκευτικό χώρο χρησιμοποιούνται κάρτες microSD. Από το 2015 και μετά η εταιρεία Raspberry υλοποίησε το δικό της λειτουργικό με όνομα Raspbian OS [14]. Η τελευταία πλακέτα που παρουσίασε η εταιρεία είναι το Raspberry Pi 4. Τα χαρακτηριστικά που μπορεί να βρει κάποιος σε μια πλακέτα Raspberry Pi είναι μνήμη RAM 1 έως 4 GB LPDDR4, Ένα 64bit επεξεργαστή Broadcom BCM2711 και πλέον διαθέτει ασύρματη σύνδεση στο διαδίκτυο σε 2.4GHz αλλά και 5GHz. Υποστηρίζει κωδικοποίηση βίντεο H.264 και H.265. Έχει 2 θύρες USB 2.0, 2 θύρες USB 3.0 και τροφοδοτείται μέσω μια θύρας USB-C. Στην Εικόνα 3 βλέπουμε μια φωτογραφία της συγκεκριμένης πλακέτας.



Εικόνα 3: Πλακέτα Raspberry Pi 4

Google Coral TPU [15] [16] [17]

Η συγκεκριμένη πλακέτα είναι και η πιο πρόσφατη υλοποίηση και δημιουργήθηκε από τη Google, το Μάρτιο του 2019. Επιτρέπει τη δημιουργία γρήγορων, αποτελεσματικών και ισχυρών τοπικά κατανεμημένων εφαρμογών διασφαλίζοντας έτσι την ιδιωτικότητα των δεδομένων. Υποστηρίζει και εκτελεί, σε γρήγορο χρονικό διάστημα, εφαρμογές μηχανικής μάθησης. Λόγω και της ιδιαιτερότητας της, χρησιμοποιείται σε εφαρμογές πραγματικού χρόνου, για αναγνώριση προσώπου και αντικειμένων, ή για να βρει ένα πιο αποτελεσματικό τρόπο εκτέλεσης μιας εργασίας. Μιας και η τεχνητή νοημοσύνη είναι για τα καλά πλέον στη ζωή μας και βάσει της συμβατότητας της με εφαρμογές μηχανικής μάθησης, η συγκεκριμένη πλακέτα εφαρμόζεται και σε υλοποιήσεις έξυπνων πόλεων «smart cities». Παραδείγματα αυτών των εφαρμογών είναι ο υπολογισμός της θέσης ενός οχήματος και ο χρόνος άφιξης σε συγκεκριμένο προορισμό. Δεν θα αναφερθούν όλα τα τεχνικά χαρακτηριστικά της εφόσον δεν παρατηρείται σημαντική διαφορά από τις πιο πάνω πλακέτες. Αυτό που την κάνει ξεχωριστή, είναι η χρήση του Edge TPU System on module (SOM) που είναι ο λόγος που επιτυγχάνονται γρήγορα και αποδοτικά οι υπολογισμοί των εφαρμογών. Επίσης, χρησιμοποιεί το Edge TPU ML accelerator coprocessor που διαχειρίζεται την κατανομή των δεδομένων για να πετύχει την ιδιωτικότητα. Στην Εικόνα 4 βλέπουμε την πλακέτα Google Coral.



Εικόνα 4: Coral TPU

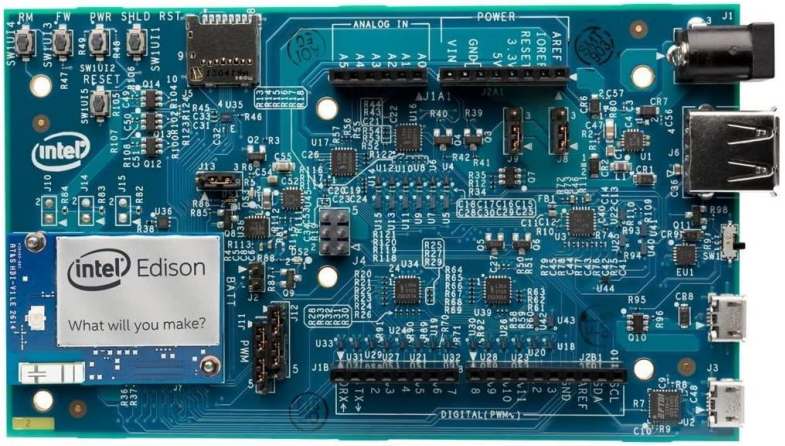
Intel Edison [18]

Η Intel ήταν ακόμη μια εταιρεία που από τον Ιανουάριο του 2014 αποφάσισε να εισέλθει στην συγκεκριμένη αγορά. Επιλέξαμε να δούμε τον Intel Edison γιατί διαφέρει από τις υπόλοιπες πλακέτες που προαναφέρθηκαν. Επικεντρώνεται κυρίως σε wearable συσκευές αν και μπορεί να χρησιμοποιηθεί εξίσου και σε άλλες ΔτΠ συσκευές. Λόγω και της χρηστικότητας που προορίζεται έχει μέγεθος όσο μια κάρτα μνήμης τύπου SD. Τα χαρακτηριστικά του ήταν πιο απλά σε σχέση με ότι είδαμε μέχρι τώρα. Έχει ένα επεξεργαστή συγχρονισμένο στα 400 MHz, σύνδεση Bluetooth και Wi-Fi.

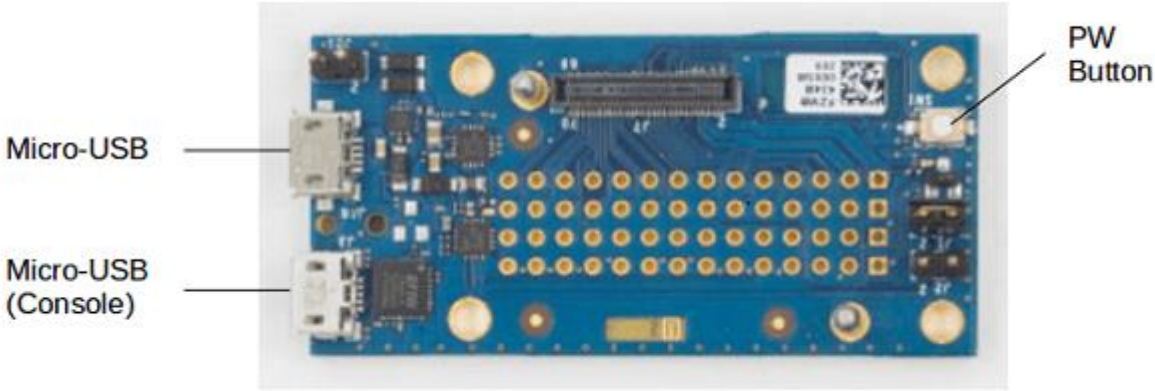
Εικόνα 5: Intel Edison

Αργότερα το Σεπτέμβριο του 2014 ανακοινώθηκαν δύο διαφορετικές προσεγγίσεις. Η μια ήταν λίγο μεγαλύτερη από την άλλη και στην ουσία κατασκεύασαν κάτω από τα ίδια πρότυπα την πλακέτα Arduino Uno και ενσωμάτωσαν πάνω τον Intel

Edison (Εικόνα 6). Η δεύτερη προσέγγιση (Intel Edison Break Out Board) που δημιούργησαν ήταν μια πιο συμπαγής πλακέτα με λιγότερες δυνατότητες σε σχέση με την πρώτη εστιάζοντας κυρίως στο μέγεθος. Μπορούμε να τη δούμε στην Εικόνα 7. Τρέχει RTOS και τα χαρακτηριστικά τους ήταν πανομοιότυπα με αυτά του Arduino.



Εικόνα 6: Πλακέτα τύπου Arduino Uno με ενσωματωμένο τον Intel Edison



Εικόνα 7: Intel Edison Break Out Board

Ο Intel Edison στην πιο πάνω πλακέτα τοποθετείται στο κέντρο.

Στη συνέχεια θα αναλύσουμε μερικά από τα πιο δημοφιλή λειτουργικά συστήματα που χρησιμοποιούνται σε πλακέτες και για εξυπηρέτηση των σκοπών του ΔτΠ.

2.3.3 Λογισμικό

Tiny OS [19][20]

Δημιουργήθηκε το 1999 και ήταν το πρώτο λειτουργικό σύστημα που σχεδιάστηκε για το ΔτΠ. Έγινε στο πανεπιστήμιο του Barkley στα πλαίσια του προγράμματος «DAPRA NEST». Χρησιμοποιήθηκε σε συσκευές που αφορούσαν το διάστημα και μάλιστα ενσωματώθηκε στο δορυφόρο ESTCube-1[21]. Είναι ιδανικό σε λειτουργίες χαμηλής ισχύος και ανταποκρίνεται σε πολύπλοκα αλλά και πολυνηματικά προγράμματα. Απαιτούνται μόλις 1KB RAM και 4KB ROM για το λειτουργικό. Γράφτηκε στη γλώσσα nesC που στην ουσία είναι η C με τη προσθήκη κάποιων επιπλέον βιβλιοθηκών για την υποστήριξη των απαιτήσεων. Κυκλοφόρησε δωρεάν ως open-source project και υποστηρίζεται μέχρι και σήμερα με τη τελευταία ενημέρωση να έγινε το 2012.

Contiki [22]

Δημιουργήθηκε το 2002 από τον Adam Dunkles και στη συνέχεια υποστηρίχθηκε από ομάδες προγραμματιστών και τεχνολογικούς κολοσσούς όπως είναι η Cisco. Βασίζεται στη γλώσσα προγραμματισμού C και στα αρχικά στάδια της ανάπτυξης του δοκιμάστηκε στον προσομοιωτή Cooja [68]. Υποστηρίζει πολλαπλές διεργασίες και κατά τη μεταφορά δεδομένων εντός των εφαρμογών του χρησιμοποιείται το πρωτόκολλο TCP. Λειτουργεί αποτελεσματικά με IPv4 και IPv6 και ήταν η αιτία που υιοθετήθηκε τόσο γρήγορα. Κυρίως χρησιμοποιείται σε ασύρματες συσκευές με εφαρμογές χαμηλής ισχύς και ο λόγος είναι ότι το πλήρες πακέτο του λειτουργικού χρειάζεται μόλις 10KB RAM και 30KB ROM για την εκτέλεση των διεργασιών. Υπάρχουν και οι πιο ελαφρές εκδόσεις που χρησιμοποιούν 2KB RAM και 30KB ROM. Πρόσφατα παρουσιάστηκε μια νέα έκδοση του λειτουργικού γνωστό ως Contiki-NG που εστιάζει σε συσκευές ΔτΠ για επόμενες γενεές.

RIOT [23] [24] [25] [26]

Ξεκίνησε να υλοποιείται από το 2008 σε συνεργασία με το ανοιχτό πανεπιστήμιο του Βερολίνου, το πανεπιστήμιο του Αμβούργου και το Εθνικό Ινστιτούτο Έρευνας στην Πληροφορική και στον Αυτοματισμό. Κυκλοφόρησε στο ευρύ κοινό το 2013. Είναι 32-bit σύστημα και αναπτύχθηκε πάνω στις γλώσσες C/C++. Μπορεί να υποστηρίξει εφαρμογές που γράφονται σε C/C++ και Rust μέσω της χρήσης ενός API. Χρειάζεται μόλις 1.5KB RAM και 5KB ROM και μπορεί να υποστηρίξει πολυνηματικές λειτουργίες σε πραγματικό χρόνο. Ένα από τα δυνατά του χαρακτηριστικά σε σχέση

με τον ανταγωνισμό είναι ότι είναι συμβατό με SSL/TLS πιστοποίηση και σε θέματα δικτύου υποστηρίζει IPv6, TCP και UPD πρωτόκολλα. Ο πηγαίος κώδικας του είναι διαθέσιμος στους προγραμματιστές μέσω της πλατφόρμας του GitHub.

Ubuntu Core [27] [28] [29]

Το συγκεκριμένο λειτουργικό είναι ήδη γνωστό και από το λειτουργικό των ηλεκτρονικών υπολογιστών Linux και είναι μια προσαρμοσμένη IoT έκδοση του Ubuntu. Δίνει αρκετή έμφαση στην ασφάλεια. Χρειάζεται τουλάχιστον 1MB RAM και 1MB ROM. Εκτός από τις όμοιες δυνατότητες που έχει με τα πιο πάνω παραδείγματα το συγκεκριμένο λειτουργικό έχει μια ιδιαιτερότητα στην αρχιτεκτονική που ακολουθήθηκε. Διαθέτει ένα πυρήνα που συνεχώς αναβαθμίζεται ώστε να μειώνονται στο ελάχιστο οι ευπάθειες – κενά ασφαλείας του συστήματος. Ως αποτέλεσμα παρέχει ευελιξία στους κατασκευαστές των ΔτΠ συσκευών αφού χρειάζεται μόνο να ανησυχούν για τη συμβατότητα της συσκευής με το λογισμικό και όχι και για την ασφάλεια του. Τον τελευταίο καιρό υιοθετείται όλο και περισσότερο αυτή η ιδέα όχι μόνο για συσκευές του ΔτΠ αλλά και σε άλλες κατηγορίες λειτουργικών όπως πχ το Android OS και Chrome OS αφού παρέχει ευελιξία.

Raspbian OS [14]

Είναι ένα open-source λειτουργικό και η επίσημη ονομασία του είναι Raspberry Pi OS. Η ιδέα ξεκίνησε να σχεδιάζεται το 2012 και κυκλοφόρησε για πρώτη φορά το 2015 και βασίζεται πάνω στο Debian. Προορίζεται για της πλακέτες της συγκεκριμένης εταιρείας και έρχεται με προ-εγκατεστημένα πάνω από 35.000 Debian πακέτα. Είναι πολύ φιλικό στην εγκατάσταση ως προς το χρήστη. Το λειτουργικό υποστηρίζεται συνεχώς από τη κοινότητα και υπάρχουν δυο εκδόσεις διαθέσιμες, το Raspbian Buster και το Raspbian Stretch.

Fuchsia OS [30]

Πίσω από το Fuchsia OS βρίσκεται η Google. Όπως και οι υπόλοιπες ιδέες έτσι και αυτή δημιουργήθηκε για να καλύψει τις ανάγκες του ΔτΠ και κυκλοφόρησε το 2016. Είναι από τα πιο «φρέσκα» λειτουργικά και είναι με τέτοιο τρόπο δομημένο που αποτελείται από πολλά κομμάτια κώδικα (components) που έχουν γραφτεί σε διάφορες γλώσσες όπως είναι η C/C++, Rust, GO, Python. Βασίζεται και αυτό σε ένα Linux πυρήνα όπως περιγράψαμε και στο Ubuntu Core. Ο κύριος στόχος του συγκεκριμένου λειτουργικού είναι να ενοποιήσει τα πάντα. Γι’ αυτό το λόγο γίνονται προσπάθειες και από την κοινότητα που υπάρχει και το υποστηρίζει. Τρέχει τόσο σε

ενσωματωμένα συστήματα τύπου ΔτΠ όσο και σε smartphones και tablets αλλά και σε ηλεκτρονικούς υπολογιστές πετυχαίνοντας έτσι την πλήρη συμβατότητα μεταξύ των συσκευών.

Amazon Free RT OS [31][32]

Όπως όλοι οι τεχνολογικοί κολοσσοί αναγκάστηκαν να μπουν στο παιχνίδι του ΔτΠ, έτσι και η Amazon έχει δημιουργήσει κάποιες συσκευές (Amazon Echo) αλλά και ψηφιακούς βοηθούς τεχνητής νοημοσύνης (Alexa). Είναι ένα λειτουργικό ανοιχτού κώδικα και συνδέει με ασφάλεια συσκευές του ΔτΠ μέσω κάποιων προ-εγκατεστημένων βιβλιοθηκών όπως το HTTP που και οι περισσότερες συνδέσεις χρησιμοποιούν, WiFi Mgmt, MQTT και για την ασφαλή μεταφορά των δεδομένων χρησιμοποιούνται πρωτόκολλα όπως το TLS. Μπορεί να συνδυαστεί άψογα με υπηρεσίες της Amazon Web Services γνωστό και από το ακρωνύμιο AWS και εκτελεί ΔτΠ εφαρμογές στο Cloud. Το συγκεκριμένο λειτουργικό υιοθετείται όλο και περισσότερο σε εφαρμογές πάνω σε μικροελεγκτές.

3. ΔΙΑΧΕΙΡΙΣΗ ΔΕΔΟΜΕΝΩΝ ΣΤΙΣ ΠΑΡΥΦΕΣ ΤΟΥ ΔΙΚΤΥΟΥ (Data Management at the Edge)

3.1 Ορισμός Edge Computing

Ο κόσμος του Διαδίκτυο των Πραγμάτων μεγαλώνει συνεχώς και οι απαιτήσεις που δημιουργούνται αυξάνονται εκθετικά. Πλέον εμφανίζονται τόσες πολλές εφαρμογές που απαιτούν γρήγορες αποκρίσεις, άρα και αποφόρτιση του δικτύου. Πάνω σε αυτές τις απαιτήσεις δημιουργήθηκε και η ιδέα του Edge Computing [34]. Υιοθετήθηκε γρήγορα από τις εταιρείες του ΔτΠ αφού με τον ερχομό της, μειώθηκε η καθυστέρηση στην επικοινωνία και στο εύρος ζώνης. Επίσης, έχει αυξηθεί η ιδιωτικότητα άρα και η ασφάλεια του δικτύου. Πως όμως επιτυγχάνεται αυτό; Ουσιαστικά, η διαφορετική προσέγγιση που έφερε το Edge Computing είναι η τοποθέτηση των διαθέσιμων υπολογιστών κοντά σε routers και gateways με αποτέλεσμα τα δεδομένα να αποθηκεύονται τοπικά στους υπολογιστές και οι διεργασίες να εκτελούνται στον κοντινότερο ηλεκτρονικό υπολογιστή από τις πηγές δεδομένων. Έχοντας την επεξεργασία των δεδομένων στους ηλεκτρονικούς υπολογιστές (κόμβοι) που βρίσκονται κοντά στις παρυφές του δικτύου (router), επιτυγχάνεται η μείωση της απόστασης των συσκευών του ΔτΠ με τις πηγές. Αυτό επιφέρει ταχύτερη απόκριση συστήματος. Μέσω των router, επικοινωνούν και αλληλοεπιδρούν άνθρωποι και συσκευές με το δίκτυο, παράγοντας ή καταναλώνοντας δεδομένα. Το Edge Computing είναι μια εναλλακτική προσέγγιση κατανομής του δικτύου, σε σχέση με το Cloud. Το τελευταίο συγκεντρώνει όλες τις διεργασίες και επεξεργασίες σε μια `κεντρική πηγή. Στη συνέχεια διαμοιράζονται τα δεδομένα-αποτελέσματα στις παρυφές του δικτύου έχοντας έτσι μεγαλύτερο εύρος ζώνης [35].

3.2 Πλεονεκτήματα Edge Computing

Μέχρι στιγμής η φιλοσοφία που υλοποιείται σε ένα δίκτυο ΔτΠ ήταν κεντροποιημένη, δηλαδή οι ηλεκτρονικοί υπολογιστές (κόμβοι) να βρίσκονται στο κέντρο του δικτύου. Αποτέλεσμα αυτής της αρχιτεκτονικής είναι τα δεδομένα που παράγουν οι συσκευές στις άκρες του δικτύου να μεταφέρονται στο κέντρο του – servers για να γίνει η επεξεργασία τους. Πιο πάνω αναλύσαμε τη γενική ιδέα αλλά και την εφαρμογή της υλοποίησης του Edge Computing. Σε αυτή τη ενότητα θα εξεταστούν τα συγκριτικά πλεονεκτήματα της αρχιτεκτονικής σε σχέση με υφιστάμενες προσεγγίσεις [36]. Θα

εντοπιστεί η αιτία για την οποία το Edge Computing έχει γίνει τόσο δημοφιλές στο χώρο και στις εφαρμογές του Διαδίκτυο των Πραγμάτων, που ορίστηκε ως το «Next Big Thing».

Ταχύτητα

Πλέον οι περισσότερες εφαρμογές που εμφανίζονται παρουσιάζουν ευαισθησία στο χρόνο, έτσι η ταχύτητα ήταν ο κύριος λόγος που έκανε το Edge Computing τόσο δημοφιλές. Με το τρόπο που υλοποιείται το δίκτυο, τοποθετώντας δηλαδή τους κόμβους κοντά στις πηγές παραγωγής δεδομένων, μπορεί να αυξήσει κατά πολύ την απόδοση του. Μειώνεται δραματικά η καθυστέρηση του δικτύου, αφού πλέον τα δεδομένα επεξεργάζονται, είτε τοπικά στο κάθε κόμβο, είτε στέλνονται στο πιο κοντινό. Ως εκ τούτου η κίνηση στο δίκτυο να είναι ελάχιστη, αφού δεν εκτελούνται μεταφορές μεγάλων αποστάσεων.

Ασφάλεια

Παρ’ όλο που η ύπαρξη περισσότερων συσκευών αυξάνει τις επιλογές κάποιου που θέλει να επιτεθεί στο δίκτυο, η αρχιτεκτονική του Edge Computing είναι από τις πιο ασφαλείς. Αν πάρουμε για παράδειγμα την αρχιτεκτονική Cloud, παρουσιάζει αδυναμίες, ειδικά σε επιθέσεις τύπου DDoS, ή ακόμα και σε μια απλή διακοπή ρεύματος. Αντίθετα, το Edge Computing κατανέμει στις «άκρες» του δικτύου διάφορους κόμβους, όπου θα αποθηκευτούν τα δεδομένα αλλά και θα εκτελούνται οι διεργασίες που θα αναλαμβάνει ο κάθε κόμβος. Ως εκ τούτου, είναι εξαιρετικά δύσκολο να καταφέρει κανείς να θέσει εκτός λειτουργίας όλο το δίκτυο. Ακόμη, αποθηκεύοντας τοπικά τα περισσότερα δεδομένα μειώνει την κίνηση στις ακμές του δικτύου. Επομένως, όσο λιγότερη κίνηση υπάρχει στο δίκτυο τόσο λιγότερα δεδομένα είναι εκτεθειμένα και ευάλωτα για υποκλοπή.

Επεκτασιμότητα – Ευελιξία

Οι ανάγκες που μπορεί να έχει ένα δίκτυο μπορεί να αλλάζουν χρόνο με το χρόνο. Η ανάγκη που προκύπτει για τις εταιρείες να δημιουργήσουν ένα νέο κέντρο δεδομένων έχει μεγάλο κόστος και λύνει το πρόβλημα προσωρινά. Με την αρχιτεκτονική που προσφέρει το Edge Computing, οποιαδήποτε εταιρεία όταν χρειαστεί μπορεί να επεκτείνει το δίκτυο προσθέτοντας εύκολα περισσότερους κόμβους. Αυτή η διαδικασία έχει πιο χαμηλό κόστος σε σχέση με ένα κέντρο δεδομένων. Επίσης, ένας κόμβος μπορεί να προστεθεί σε οποιοδήποτε μέρος του δικτύου στο οποίο απαιτείται χωρίς να

χρειαστεί η παραμικρή αλλαγή στο υπόλοιπο δίκτυο, επιτυγχάνοντας έτσι εύκολα και απλά την επεκτασιμότητα στην εμβέλεια του δικτύου.

Αξιοπιστία

Ένα ασφαλές σύστημα συνήθως θεωρείται και αξιόπιστο. Η χαμηλή κινητικότητα στις ροές του δικτύου ισοδυναμεί με την ικανοποίηση των απαιτήσεων του χρήστη – συσκευής. Σημαντική παράμετρο αποτελεί, επίσης, η διαθεσιμότητα επιλογής πολλών διαφορετικών κόμβων-εξυπηρετητών για τις τερματικές συσκευές αποφορτίζοντας έτσι επί μέρους τους κόμβους. Τέλος, ο τρίτος βασικότερος λόγος, που παρέχει αξιοπιστία στο δίκτυο είναι ότι ακόμα και στις περιπτώσεις που κάποιοι κόμβοι καταστραφούν το υπόλοιπο δίκτυο συνεχίζει να υπάρχει και να δουλεύει χωρίς ιδιαίτερα προβλήματα.

3.3 Διεργασίες και Υπολογισμοί που εκτελούνται στις παρυφές του δικτύου

Σε αυτή τη ενότητα θα εστιάσουμε στους κόμβους και στις διεργασίες που πρέπει να εκτελέσουν. Ο κύριος λόγος που εκτελούνται διεργασίες είναι για αποθήκευση, επεξεργασία ή και ανάλυση δεδομένων. Ακόμη μπορεί να εκτελέσει οποιαδήποτε εφαρμογή είναι διαθέσιμη και στο Cloud. Στη συνέχεια θα επεξηγήσω μερικές από τις σημαντικότερες και συχνότερες διεργασίες που μπορεί να συναντήσει κάποιος σε ένα δίκτυο ΔτΠ διαχωρίζοντας τις ανάλογα με το τύπο του αλγορίθμου τους.

Ταξινόμησης (Classification) [37]

Οι αλγόριθμοι αυτού του τύπου χρησιμοποιούνται κυρίως για την εξόρυξη δεδομένων. Η διαδικασία που ακολουθούν χωρίζεται σε δύο στάδια. Το πρώτο στάδιο είναι η εκμάθηση (train) του αλγορίθμου και το δεύτερο είναι η δοκιμή-πρόβλεψη (testing). Επομένως, σε αυτές τις περιπτώσεις χωρίζουμε τα δεδομένα που θα χρησιμοποιήσουμε για τον αλγόριθμο σε δύο σύνολα, το σύνολο εκπαίδευσης (train set) που είναι αισθητά μεγαλύτερο και το σύνολο δοκιμής (test set). Ποιο κάτω θα αναφέρουμε μερικούς τύπους αλγορίθμων ταξινόμησης και παραδείγματα αυτών.

- **Linear Classifier**: Η ελληνική ονομασία είναι Γραμμικοί Ταξινομητές. Ένας Γραμμικός Ταξινομητής προσπαθεί να εντοπίσει, από πολυδιάστατα δεδομένα, κάποια χαρακτηριστικά που επηρεάζουν την έξοδο που βλέπει. Λειτουργεί με βάση την εκμάθηση που του έχει γίνει. Μερικά παραδείγματα Linear Classifier είναι τα εξής:
    - **Logistic Regression** [38]: παίρνει πολυδιάστατα δεδομένα και αποφασίζει αν η έξοδος που θα δώσει είναι 0 ή 1.

- **Linear Support Vector Machines (Linear SVM)** [39]: Δέχεται δεδομένα και αφού τα απεικονίσει σε ένα σύστημα αξόνων, στη συνέχεια προσπαθεί να βρει τη βέλτιστη ευθεία που θα τα διαχωρίζει σε δύο ομάδες.
- **Perceptron** [40]: Πανομοιότυπος με τον Logistic Regression δέχεται δεδομένα πολλών διαστάσεων και βγάζει μια δυαδική έξοδο με βάση την είσοδο που είχε.

- **Bayesian** [41]: Γνώριμη και από τα μαθηματικά αυτή η διαδικασία, υπολογίζει τη πιθανότητα κάποιο δείγμα δεδομένων να έχει παραχθεί από τη συγκεκριμένη ομάδα. Ο υπολογισμός αυτής της πιθανότητας ορίζεται από την ομώνυμη εξίσωση του Naive Bayes.
- **Decision Trees** [42]: Η ελληνική ονομασία είναι Δέντρα Αποφάσεων. Είναι μια συγκεκριμένη δομή που ανάλογα με τη είσοδο που παίρνει επιστρέφει μια προκαθορισμένη έξοδο.
- **Neural Networks** [43]: Η ελληνική της ονομασία είναι Νευρωνικά Δίκτυα. Μια τεχνολογία που έγινε δημοφιλής τα τελευταία χρόνια και συνδυάζει τεχνικές σχεδόν από όλες τις προαναφερθείσες. Υπάρχουν στάδια που ακολουθούνται για να λειτουργήσει σωστά ένα Νευρωνικό Δίκτυο. Για αρχή, θα πρέπει να γίνει αναγνώριση τόσο και της εισόδου όσο και της εξόδου. Αφού γίνει η σωστή αναγνώριση, ακολούθως, δημιουργείται ένα δίκτυο με την κατάλληλη τοπολογία και πραγματοποιείται η επιλογή του συνόλου των δεδομένων, όπου και θα εκπαιδευτεί. Στη συνέχεια, θα τοποθετηθεί ένα άλλο σύνολο δεδομένων όπου θα γίνει μια περαιτέρω προσπάθεια ώστε να μεγιστοποιήσουμε τη δυνατότητα του δικτύου να αναγνωρίζει πρότυπα. Τέλος, χρησιμοποιώντας τα σύνολο δεδομένων για τη δοκιμή θα αξιολογηθεί η αξιοπιστία του δικτύου που δημιουργήθηκε.

<u>Κατηγοριοποίησης (Clustering)</u> [44]

Οι αλγόριθμοι κατηγοριοποίησης μας δίνουν τη δυνατότητα να μπορούμε να αναγνωρίσουμε κάποιο μοτίβο που ακολουθείτε στα δεδομένα. Συνήθως, αυτά έρχονται σε μορφή διανύσματος, έτσι ώστε ο διαχωρισμός τους σε ομάδες να δίνει τη δυνατότητα εντοπισμού των κοινών χαρακτηριστικών της κάθε ομάδας. Στη συνέχεια, έχοντας νέα δεδομένα στην είσοδο, εντοπίζεται η καλύτερη δυνατή αντιστοίχιση στις ομάδες. Στη συνέχεια, παρατίθενται οι δημοφιλέστεροι αλγόριθμοι ομαδοποίησης.

- **K-Means** [45]: Ίσως είναι και ο πιο διαδεδομένος αλγόριθμος ομαδοποίησης. Ανάλογα με το πόσες ομάδες επιθυμεί ο χρήστης ορίζει τόσα κέντρα. Όταν παίρνει για είσοδο ένα διάνυσμα δεδομένων υπολογίζει την απόσταση από το κάθε ένα κέντρο και το κατηγοριοποιεί στην ομάδα με τη μικρότερη απόσταση. Στη συνέχεια ξανά υπολογίζει τα κέντρα με βάση τη μέση τιμή των διανυσμάτων και επαναλαμβάνει ξανά την ίδια διαδικασία μέχρι να καταφέρει να έχει όσο το δυνατόν πιο όμοια δεδομένα σε κάθε ομάδα.
- **Mean-Shift** [46]: Είναι παρόμοιος με το K-Means, έχει ήδη τα δεδομένα και ξεκινά να κινείται πάνω σε ένα τυχαίο διάνυσμα με ακτίνα R σχηματίζοντας έτσι ένα κύκλο. Στη συνέχεια υπολογίζει το διάνυσμα της μέσης τιμής των δεδομένων, και μετακινεί το κύκλο. Συνεχίζει να υπολογίζει τη μέση τιμή και όσο εκτελείται η συγκεκριμένη διαδικασία, αυτή αλλάζει. Το ίδιο συμβαίνει και με το διάνυσμα όπου μετακινείται. Όσες περισσότερες φορές εκτελεστεί παρατηρείται ότι το διάνυσμα θα μετακινείται ελάχιστα έως καθόλου. Έτσι λοιπόν αναγνωρίζει ότι τα σημεία-δεδομένα που βρίσκονται εντός του κύκλου είναι όμοια μεταξύ τους.
- **Density Base Spatial Clustering of Application with Noise (DBSCAN)** [47]: Για αρκετούς στο κόσμο της επιστήμης των Δεδομένων θεωρείται ως η πιο εξελιγμένη έκδοση του Mean-Shift. Διασχίζει όλα τα δεδομένα αρχίζοντας από ένα τυχαίο σημείο για κέντρο του κύκλου με ακτίνα R. Στη συνέχεια μετακινεί τον κύκλο σε απόσταση ε και ξαναδημιουργεί ένα νέα κέντρο. Συνεχίζει αυτή τη διαδικασία μέχρι σχεδόν όλα τα σημεία να ανήκουν σε κάποια ομάδα. Αν τυχόν μερικά μείνουν εκτός ομάδας και το σύνολο είναι μικρότερης τάξης μεγέθους σε σχέση με τα υπόλοιπα τότε απλά τα θεωρούνται ως θόρυβος.
- **Expectation Maximization** [48]: Ο συγκεκριμένος αλγόριθμος παρέχει περισσότερη ευελιξία σε σχέση με τον K-Means. Έρχεται να λύσει το πρόβλημα του K-Means όταν τα διανύσματα-δεδομένα βρίσκονται τοποθετημένα σε κυκλική διάταξη. Ο συγκεκριμένος αλγόριθμος αρχίζει θέτοντας δύο clusters και στη συνέχεια υπολογίζει τη Gaussian κατανομή για το καθένα. Αφού τη βρει, παίρνει το κάθε διάνυσμα και υπολογίζει τη πιθανότητα να ανήκει στο συγκεκριμένο cluster. Θέτει ένα κατώφλι και τα δεδομένα που είναι κάτω από αυτό μένουν εκτός. Συνεχίζει με τα υπόλοιπα. Η

διαδικασία επαναλαμβάνεται έως ότου να μην υπάρχουν αλλαγές με αποτέλεσμα να έχουμε τα όμοια δεδομένα μαζί.

- **Agglomerative Hierarchical** [49]: Η φιλοσοφία αυτού του αλγορίθμου είναι λίγο διαφορετική. Θεωρεί το κάθε διάνυσμα ως ένα cluster και στη συνέχεια υπολογίζει την απόσταση μεταξύ του κάθε cluster-διανύσματος. Αυτά με τη μικρότερη απόσταση θεωρούνται όμοια. Ως εκ τούτου, τα μετατρέπει σε ένα cluster και επαναλαμβάνω την ίδια διαδικασία μαζί με τα υπόλοιπα που έχουν. Αν ο αλγόριθμος εκτελέσει τη διαδικασία μέχρι τέρμα θα μας δώσει ένα cluster που θα περιέχει όλα τα δεδομένα γι' αυτό και ονομάζεται ιεραρχικός. Οπότε επιλέγοντας από την αρχή πόσα cluster επιθυμούμε να έχουμε θέτουμε έτσι το τέλος της εκτέλεσης.

Εντοπισμός Ανώμαλων Τιμών (Outlier Detection)

Λόγω των πολλών συσκευών που υπάρχουν συνδεδεμένες στο ΔτΠ είναι συνεπακόλουθο να βλέπουμε μια τεράστια παραγωγή δεδομένων. Ένα συχνό φαινόμενο της μεγάλης παραγωγής είναι να εμφανίζονται ανωμαλίες στις τιμές που καταγράφονται. Τη λύση του προβλήματος έδωσαν οι αλγόριθμοι εντοπισμού είτε ελλειπών τιμών, είτε τιμές με μεγάλη απόκλιση απ' ότι αναμενόταν. Στη συνέχεια, προχωρούσαν στην αντικατάσταση τους, με μια νέα προσεγγιστική τιμή με βάση των γειτονικών τιμών τους. Μερικές αξιόλογες λύσεις είναι οι εξής:

- **k-Nearest Neighbors** [51]: Ο αλγόριθμος KNN όπως αποκαλείται χρησιμοποιείται για την εξόρυξη δεδομένων. Χρησιμοποιώντας στις ήδη υπάρχουσες τιμές. Μπορεί να υπολογίσει ξανά τιμές που για κάποιο λόγο παρουσίασαν κάποια ανωμαλία. Το k υποδηλώνει πόσες γειτονικές τιμές να λάβει υπόψιν. Χρησιμοποιώντας μετρικές αποστάσεων όπως είναι η Euclidean [52], Manhattan [53] ή Minkowski [54] για συνεχείς τιμές υπολογίζεται η τιμή στόχος.
- **Isolation Forest** [55]: Είναι αρκετά αποδοτική μέθοδος σε πολυδιάστατα δεδομένα. Η μέθοδος που χρησιμοποιείται είναι μέσω του αλγορίθμου Random Forest να δημιουργεί δέντρα αποφάσεων. Αφού δημιουργηθούν τα δέντρα διασχίζοντας τα προσπαθεί να αναγνωρίσει ανωμαλίες και όπως υποδηλώνει το όνομα του, να τις απομονώσει. Όσο πιο βαθιά προχωρεί σε ένα δέντρο τόσο πιο δύσκολο να απομονώσει κάποια ανωμαλία που βρίσκεται σε κόμβους

φύλλα. Όσο πιο κοντά στη ρίζα βρει ένα κόμβο φύλλο τόσο πιο εύκολα μπορεί να το απομονώσει.

- **Local Outlier Factor** [56]: Έχοντας μπροστά μας όλο το σύνολο δεδομένων προσπαθούμε να απομονώσουμε κάποιο διάνυσμα που φαίνεται να είναι διαφορετικό, άρα και απομακρυσμένο από τα υπόλοιπα. Στη συνέχεια αφού δηλώσουμε πόσους γείτονες επιθυμούμε να δούμε (k), βρίσκουμε την Local Reachability Density (LRD) για το σημείο που επιθυμούμε που είναι ο λόγος ένα ως προς το άθροισμα των μέγιστων αποστάσεων του σημείου και του γείτονα, επί των αριθμών των γειτόνων. Και τέλος αν υπολογίσω τη μέση τιμή του αθροίσματος LRD των γειτόνων ως προς το LRD του διανύσματος στόχου και είναι τάξης μεγαλύτερο της μονάδας τότε το σημείο θεωρείται outlier.

Μείωση Διαστάσεων σε Πολυδιάστατα Δεδομένα (Dimension Reduction)

Ειδικά τα τελευταία χρόνια ο όρος Big Data γίνεται όλο και πιο δημοφιλής. Τι εννοούμε με αυτό τον όρο όμως; Στην ουσία όταν λέμε Big Data τα δεδομένα σαν τιμές δεν έχουν καμία διαφορά από ότι ήδη ξέρουμε και έχουμε, απλά ο όγκος τους είναι τεράστιος με αποτέλεσμα για να τα επεξεργαστούμε να χρειάζονται ειδική μεταχείριση. Γι’ αυτό ακριβώς το λόγο προτάθηκαν και οι αλγόριθμοι που μειώνουν τις διαστάσεις των δεδομένων κρατώντας μόνο τις σημαντικές, ούτως ώστε να γίνεται πιο εύκολη και αποδοτική η επεξεργασία τους. Για να γίνει κατανοητή η διαδικασία της μείωσης διαστάσεων ας δούμε πως λειτουργεί η διαδικασία για μείωση από δύο σε μια διάσταση. Σκεφτείτε να έχουμε διασκορπισμένα σημεία σε άξονες δύο διαστάσεων. Αυτό που προσπαθεί να πετύχει η διαδικασία αυτή είναι να καταφέρει να βρει την απεικόνιση αυτών των σημείων πάνω σε μια ευθεία βρίσκοντας όσο το δυνατόν γίνεται την πιο κατάλληλη θέση της ευθείας ώστε να μην αλληλοκαλύπτονται τα σημεία και να είναι όσο το δυνατόν καλύτερα διασκορπισμένα πάνω στην ευθεία. Έτσι χωρίς να αλλοιώσουμε τα δεδομένα καταφέραμε από τις δύο διαστάσεις να πάμε σε μία. Αυτή είναι η φιλοσοφία των αλγορίθμων μείωσης των διαστάσεων. Προσθέτουμε μόνο κάποιες επιπλέον μετρικές όπως ο μέσος όρος των σημείων ως προς ένα άξονα, το Variance που είναι το Mean για κάθε διάσταση και το Co-Variance που είναι ο μέσος όρος της τιμής X στα διανύσματα των δεδομένων μας. Όσο πιο κοντά στο μηδέν βρίσκεται η τιμή του Co-Variance τόσο πιο διάσπαρτα είναι τα διανύσματα στο κέντρο των αξόνων. Στη συνέχεια θα δούμε αναφορικά μερικές από τις πιο πάνω μεθόδους.

- Principal Component Analysis (PCA) [57] [58]

- Linear Discriminant Analysis (LDA) [59]
- General Component Analysis (GCA) [60]

3.4 Προηγούμενες Μελέτες και Μοντέλα πάνω στο Edge Computing

Σε αυτή την ενότητα θα κάνουμε μια περιληπτική αναφορά σε προηγούμενες μελέτες που έχουν γίνει πάνω στις παρυφές του διαδικτύου και σε άλλες εφαρμογές που μπορεί να έχει η φιλοσοφία πίσω από το Edge Computing διαχωρίζοντας αποδοτικά τις διεργασίες που απαιτούνται για την ολοκλήρωση των απαιτήσεων ενός συστήματος.

- Κατανομή διεργασιών σε δυναμικά συστήματα

  Το άρθρο είναι μια μελέτη στα πλαίσια μιας διπλωματικής εργασίας με τίτλο «Decentralized Task Management for Dynamic Environments» [61]. Ο φοιτητής παρουσιάζει δύο υφιστάμενους άπληστους θεωρητικά αλγορίθμους και προτείνει και ένα καινούργιο πάνω στην ασύγχρονη κατανομή διεργασιών πάνω σε αποκεντροποιημένα συστήματα. Ο πρώτος αλγόριθμος που παρουσιάζει λέγεται Sequential Greedy Algorithm (SGA) και παρόλο που χρειάζεται αρκετές επαναλήψεις μέχρι να συγκλίνει στην απόφαση του είναι αρκετά γρήγορος πάνω σε αποκεντροποιημένα συστήματα. Ο δεύτερος ονομάζεται Consensus Based Bundle Algorithm (CBBA) που απαιτεί λιγότερες επαναλήψεις σε σχέση με τον πρώτο για να κατανέμει διεργασίες. Οι πιο πάνω αλγόριθμοι για να εφαρμοστούν πάνω σε αποκεντροποιημένα συστήματα βασίζονται πάνω σε καθολικούς συγχρονισμένους μηχανισμούς του συστήματος, δηλαδή ο κάθε κόμβος συγχρονίζεται κάτω από το «ρολόι» του συστήματος. Αυτό που προτείνει με την έρευνα του ο φοιτητής είναι ένα αλγόριθμο που τον ονόμασε «Asynchronous Consensus Based Bundle Algorithm» και στην ουσία είναι μια συνδυαζόμενη παραλλαγή των πιο πάνω που παρακάμπτει τους περιορισμούς για τους συγχρονισμένους μηχανισμούς και εφαρμόζει σε αποκεντροποιημένα συστήματα ασύγχρονους καθολικούς μηχανισμούς προσφέροντας και αυτονομία στους κόμβους αλλά και ευρωστία στο όλο σύστημα.

- Αναγνώριση διεφθαρμένων κόμβων ενός δικτύου

  Το άρθρο της πιο πάνω έρευνας έχει τίτλο «Efficient consensus algorithm for the accurate faulty node tracking with faster convergence rate in a distributed sensor network» [62]. Οι ερευνητές παρουσιάζουν ένα τρόπο που μπορεί να αναγνωρίζει τους διεφθαρμένους κόμβους μέσα σε ένα κατανεμημένο υπολογιστικό σύστημα. Για ένα τέτοιο σύστημα το να έχει κατεστραμμένους στην ουσία κόμβους μπορεί να

προκαλέσει σοβαρές επιπτώσεις στην αξιοπιστία του συστήματος αφού θα παρέχουν λανθασμένες πληροφορίες και παράλογα αποτελέσματα. Η λύση που δίνουν οι ερευνητές μπορεί να εφαρμοστεί και σε περιπτώσεις όπου κάποιος κόμβος έχει τεθεί εκτός δικτύου για οποιοδήποτε λόγο. Για την αναγνώριση των κόμβων χρησιμοποιούν δύο αλγόριθμους ομοφωνίας όπως ονομάζονται. Η φιλοσοφία πίσω από τη συγκεκριμένη πρόταση είναι ότι οι υγιείς κόμβοι εκτελούν κάποια διεργασία-φόρμουλα και επιστρέφοντας κάποια τιμή. Εάν όλοι οι υγιείς κόμβοι συμφωνούν στην ίδια τιμή τότε όποιος κόμβος επιστρέφει κάτι διαφορετικό θεωρείται διεφθαρμένος, έτσι αναγνωρίζεται και απομονώνεται. Η συγκεκριμένη μέθοδος μπορεί να εφαρμοστεί με επιτυχία και είναι πολύ αποδοτικότερη και με λιγότερο κόστος σε σχέση με ένα κεντροποιημένο σύστημα όπου θα χρειάζεται να έχουμε κάποιους κεντρικούς ελεγκτές που θα είναι αρμόδιοι να ελέγχουν ανά τακτά χρονικά διαστήματα ότι όλοι οι κόμβοι είναι εντάξει μέσα στο δίκτυο. Αυτό θα απαιτούσε πολύ χρόνο αλλά και θα κατανάλωνε άσκοπα τους πόρους του συστήματος.

- Κατανομή διεργασιών με περιορισμούς πάνω σε ένα σύστημα αποτελούμενο από ρομπότ

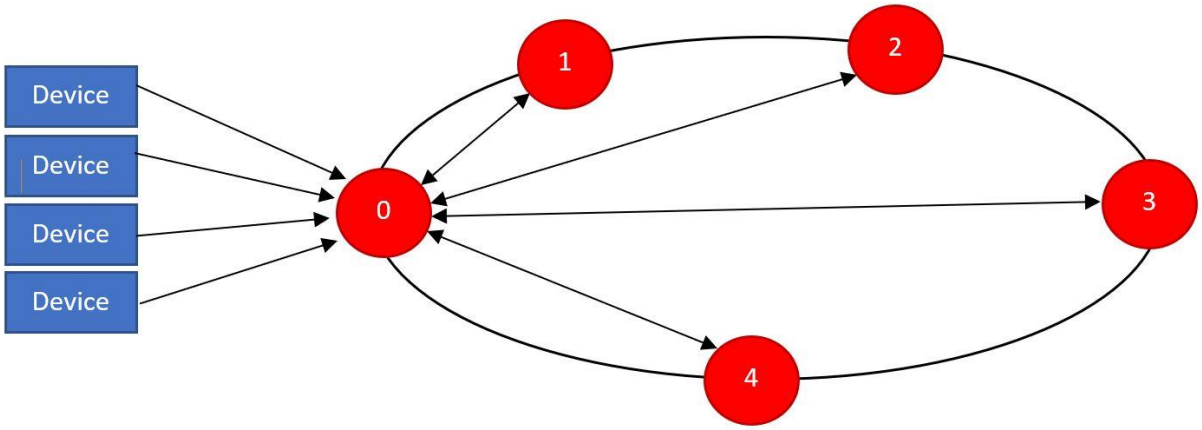
  Το άρθρο της έρευνας έχει τίτλο «A Distributed Algorithm for Constrained Multi-Robot Task Assignment for Group Tasks» [63] και βασίζεται πάνω σε ένα αλγόριθμο κατανομής διεργασιών για ένα ρομποτικό σύστημα. Συγκεκριμένα επεξηγείται ένας αλγόριθμος για το πως μπορούν να συνεργαστούν διάφορα ρομπότ με διαφορετικές ιδιότητες το καθένα, με σκοπό να πετύχουν ένα κοινό στόχο. Η εισήγηση των ερευνητών είναι ο τρόπος που θα ακολουθηθεί για να κατανέμουν τις διεργασίες στους πόρους του συστήματος (ρομπότ) προσπαθώντας να εγγυηθούν και την αποδοτικότητα του συστήματος κατά την εκτέλεση. Προηγούμενες υφιστάμενες εισηγήσεις υστερούσαν πάνω σε αυτό τον τομέα όταν είχαν να διαχειριστούν διεργασίες με περιορισμούς. Η αρχιτεκτονική του αλγορίθμου που ακολούθησαν είναι αποκεντροποιημένη ακριβώς δηλαδή όπως ένα Edge Computing δίκτυο. Η Επιλογή της παρουσίασης του άρθρου έχει να κάνει με την αποτύπωση της εφαρμογής της συγκεκριμένης φιλοσοφίας σε άλλα συστήματα το ίδιο αποδοτικά. Στη συνέχεια, οι συγγραφείς συγκρίνουν τη πρόταση τους με άλλες κεντροποιημένες προσεγγίσεις και καταλήγουν στο συμπέρασμα ότι κατάφεραν να παρέχουν βέλτιστη λύση O(n,ε) όπου n είναι ο αριθμός των διεργασιών που πρέπει να διαχειριστεί το σύστημα και ε μια παράμετρος που επιλέγουν οι ίδιοι κατά την εκτέλεση του αλγορίθμου.

4. ΠΡΟΤΕΙΝΟΜΕΝΟΣ ΑΛΓΟΡΙΘΜΟΣ

Στο παρόν κεφάλαιο θα παρουσιάσουμε το μοντέλο που αποτελεί την προσομοίωση ενός δικτύου στις προδιαγραφές του Διαδικτύου των Πραγμάτων. Ειδικότερα θα δοθεί περιγραφή με το τρόπο τον οποίο διαχειρίζεται τα δεδομένα και τις διεργασίες γι' αυτά στις παρυφές του δικτύου (nodes) και ποιες μέθοδοι ακολουθήθηκαν. Όλη η προσομοίωση υλοποιήθηκε στη γλώσσα προγραμματισμού Python.

4.1 Δομή δικτύου και ο τρόπος λειτουργίας του.

Στη παρούσα ενότητα δεν θα ασχοληθούμε με το κομμάτι του κώδικα αλλά με το τρόπο που προσομοιώσαμε το δίκτυο μας και πως λειτουργεί. Καταρχήν να πούμε ότι στο βρίσκεται συνδεδεμένο ένα πλήθος ΔτΠ συσκευών που παράγουν δεδομένα και ένα σύνολο από N edge κόμβους που αποτελούν και τον πυρήνα του δικτύου μας. Οι ΔτΠ συσκευές συλλέγουν πληροφορίες από το περιβάλλον στο οποίο βρίσκονται. Στη συνέχεια στέλνουν τα δεδομένα που έχουν σε ένα edge κόμβο, όπου συλλέγουν και αποθηκεύουν τις πληροφορίες που τους παρέχονται. Η πιο κάτω εικόνα μας δίνει τη γενική εικόνα και δομή του δικτύου μας.



Εικόνα 8: Δομή ΔτΠ δικτύου

Για αρχή πριν να ξεκινήσει η προσομοίωση παράγουμε κάποια δεδομένα για τον κάθε κόμβο ξεχωριστά. Στη συνέχεια αποθηκεύονται τοπικά στη μορφή δισδιάστατου πίνακα και αντιπροσωπεύουν το σύνολο των δεδομένων του κόμβου. Ο λόγος που γίνεται η συγκεκριμένη διαδικασία είναι για να αποκτήσει ο κάθε κόμβος κάποια ειδικά χαρακτηριστικά. Από την πλευρά του ο κόμβος θα λαμβάνει πολυδιάστατα δεδομένα από πολλές πηγές γι' αυτό και τα δεδομένα τα οποία στέλνονται έχουν τη μορφή πίνακα $[x] = <X_1, X_2, X_3,...,X_M>$ όπου M ο αριθμός των διαστάσεων-μεταβλητών μας. Ακολούθως ο κόμβος αφού λάβει τα δεδομένα από μια υποτιθέμενη συσκευή, θα

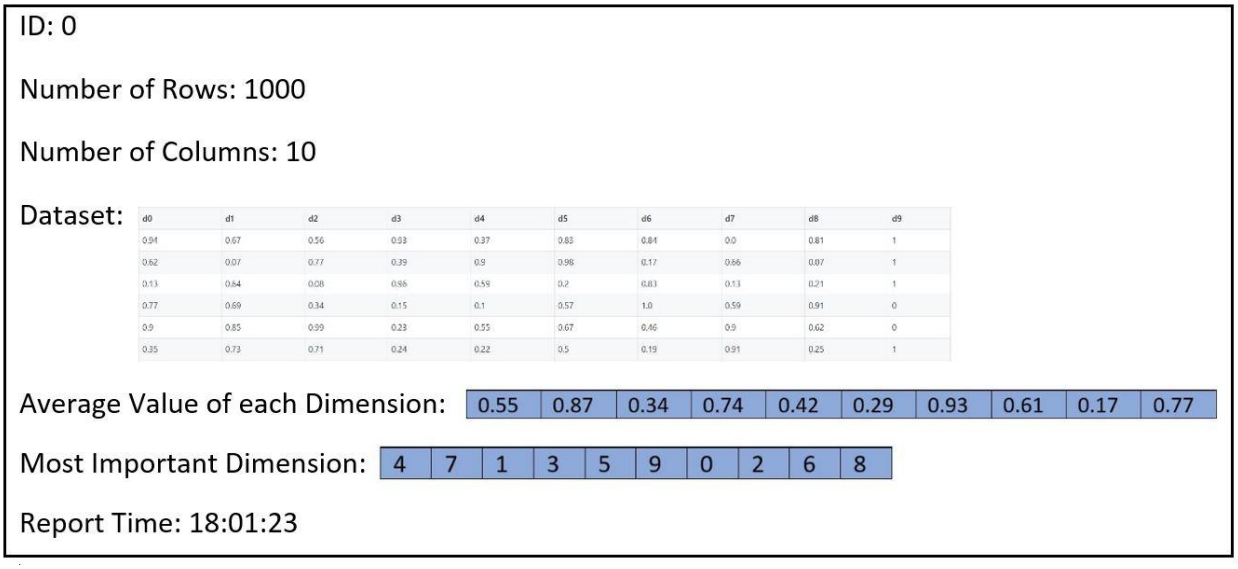
πρέπει να αποφασίσει τι ακριβώς πρέπει να κάνει. Για σκοπούς προσομοίωσης επιλέγουμε με βάση των πιθανοτήτων, την απόφαση του κάθε κόμβου αν θα αποθηκεύσει τα δεδομένα τοπικά (locally), ή αν θα επιλέξει να τα στείλει σε κάποιο άλλο κόμβο (remotely). Αν επιλέξει να τα αποθηκεύσει τοπικά ο τρέχων κόμβος θα προσθέσει στο σύνολο των δεδομένων που ήδη έχει την εγγραφή την οποία έλαβε και η πρώτη φάση τελειώνει εκεί. Αν όμως επιλεγεί ότι τα συγκεκριμένα δεδομένα θα πρέπει να αποθηκευτούν σε άλλο κόμβο τότε ο κόμβος χρησιμοποιώντας κάποιους μηχανισμούς που θα περιγράψουμε σε άλλη ενότητα, υπολογίζει τις πιο σημαντικές διαστάσεις, αυτές δηλαδή που έχουν τη μεγαλύτερη βαρύτητα για τη συγκεκριμένη εγγραφή που δέχτηκε σαν είσοδο. Στη συνέχεια ο κάθε κόμβος υπολογίζει τη μέση τιμή της κάθε διάστασης με βάση τα δεδομένα που έχει ήδη αποθηκευμένα σε αυτό. Για να μειωθεί το κόστος αλλά και ο χρόνος της προσομοίωσης του δικτύου εκτελούμε μια διαδικασία κατηγοριοποίησης κόμβων όπου με βάση τη μέση τιμή των σημαντικών διαστάσεων (που πήραμε από τα νέα δεδομένα) του κάθε κόμβου παίρνουμε την ομάδα κόμβων που μας ενδιαφέρει. Ως εκ τούτου μειώνεται το εύρος του συνόλου επιλογής του κόμβου. Τώρα από την πλευρά των υποψήφιων κόμβων που βρίσκονται στην ομάδα που μας ενδιαφέρει ετοιμάζουν και στέλνουν μια αναφορά στον τρέχων κόμβο όπου είναι έτοιμος να λάβει την απόφαση. Στη συνέχεια ο τρέχων κόμβος εκτελεί ένα μηχανισμό ανταμοιβής με βάση τα στοιχεία που έχει λάβει από την αναφορά του κάθε κόμβου και χρησιμοποιώντας μια αντίστροφη σιγμοειδής συνάρτηση υπολογίζει το συνολικό σκορ του κάθε κόμβου. Ο κόμβος με το μεγαλύτερο σκορ κερδίζει και η εγγραφή με τα δεδομένα αποστέλλεται σε αυτό. Τέλος υπολογίζουμε τη τυπική απόκλιση των δεδομένων για συμπερασματικούς σκοπούς για να δούμε πόσο συμπαγές κρατήσαμε τα δεδομένα.

4.1.1 Δομή ενός Edge κόμβου

Πριν μπούμε σε βάθος για την επεξήγηση των διαδικασιών που ακολουθείται για τις λειτουργίες του δικτύου, στη συγκεκριμένη ενότητα θα παρουσιάσουμε τη δομή που έχει ένας edge κόμβος στο δίκτυο μας. Κάθε κόμβος του δικτύου έχει:

- ένα μοναδικό αναγνωριστικό αριθμό
- τον αριθμό των διαστάσεων και των εγγραφών που έχει αποθηκευμένα στο σύνολο δεδομένων του ούτως ώστε να μπορεί να μας το επιστρέψει οποιαδήποτε στιγμή του ζητηθεί
- το σύνολο με τα δεδομένα του
- ένα μονοδιάστατο πίνακα, μήκους όσο και οι διαστάσεις των δεδομένων, που περιέχει την μέση τιμή για κάθε διάσταση
- ένα μονοδιάστατο πίνακα, μήκους όσο και οι διαστάσεις των δεδομένων, που περιέχει με σειρά σημαντικότητας της διαστάσεις των δεδομένων ξεκινώντας από τα αριστερά
- το χρόνο που έκδωσε την τελευταία του αναφορά για οποιοδήποτε κόμβο του δικτύου

Πιο κάτω μπορούμε να δούμε τη μορφή που έχει ένας edge κόμβος.



Εικόνα 9: Δομή ενός Edge κόμβου

4.2 Περιγραφή σημαντικών σταδίων του προτεινόμενου μοντέλου

Εδώ θα δούμε τόσο ονομαστικά αλλά όσο και λεπτομερώς με ποια διαδικασία εκτελούμε διάφορες λειτουργίες και μετρικές που χρησιμοποιούνται στη προσομοίωση που επεξηγήθηκε στη πιο πάνω ενότητα.

- Με ποιο τρόπο μοιράζονται τα δεδομένα στους κόμβους κατά την αρχή της προσομοίωσης;

  Με το που θα τρέξουμε τη προσομοίωση το πρόγραμμα παίρνει για είσοδο ένα σύνολο δεδομένων και μοιράζει τυχαία R εγγραφές σε κάθε κόμβο όπου R είναι ένας προκαθορισμένος ακέραιος αριθμός που δίνεται σαν παράμετρος στο σύστημα. Έτσι κάθε κόμβος έχει αρχικά κάποια δεδομένα αποθηκευμένα στο σύνολο του, με σκοπό να διαμορφώσει το δικό του χαρακτήρα κατά τη προσομοίωση.

- Πως υπολογίζεται η πιθανότητα επιλογής του τρόπου αποθήκευσης;

  Με το που έρθει μια νέα εγγραφή δεδομένων κατά τη προσομοίωση ο κόμβος παραλήπτης θα πρέπει να αποφασίσει αν θα αποθηκεύσει Locally ή Remotely τα δεδομένα. Η απόφαση αυτή για σκοπούς προσομοίωσης καθορίζεται από τη πιθανότητα Phi όπου δίνεται κατά την εκκίνηση του προγράμματος και μας δίνει Phi ο κόμβος να επιλέξει να αποθηκεύσει τα δεδομένα σε άλλο κόμβο του δικτύου (Remote Save) και 1-Phi ο κόμβος να αποφασίσει να αποθηκεύσει τοπικά τα δεδομένα (Local Save).

- Πως βρίσκουμε τις διαστάσεις με τη μεγαλύτερη βαρύτητα;

  Για να βρούμε ποιες διαστάσεις έχουν τη περισσότερη βαρύτητα κάθε φορά που έρχεται μια νέα εγγραφή δεδομένων, χρησιμοποιούμε τη συνάρτηση x^2 ή όπως ονομάζεται διεθνώς Chi-Square [64]. Θα αναλύσουμε πιο κάτω το τρόπο με τον οποίο λειτουργεί η συγκεκριμένη συνάρτηση για να δείξουμε με πιο τρόπο μας εξυπηρετεί στη συγκεκριμένη προσομοίωση. Πιο κάτω βλέπουμε ένα πίνακα με κάποια τυχαία δεδομένα από το σετ που χρησιμοποιήσαμε. Για σκοπούς δικής μας ευκολίας στην επεξήγηση, θα αρκεστούμε σε δύο χαρακτηριστικά.

Πίνακας 1: Κατηγορίες για Chi-Square συνάρτηση

	Θερμοκρασία αέρα > 55	Επίπεδα Υγρασίας < 60	
Δεν Έβρεξε	42 15	33 10	75
Έβρεξε	18 45	7 30	25
	70	30	100

Τώρα με βάση το πιο πάνω παράδειγμα με τελικό αποτέλεσμα το αν έβρεξει ή όχι στις επόμενες 100 μέρες προσπαθούμε να καταλάβουμε αν η διάσταση «Θερμοκρασία Αέρα» ή «Επίπεδα Υγρασίας» επηρεάζουν το τελικό αποτέλεσμα. Οπότε βάζοντας τις τιμές της κάθε διάστασης με την αναμενόμενη τιμή που βρίσκουμε (με κόκκινο χρώμα) βρίσκουμε το αποτέλεσμα της συνάρτησης. Χρησιμοποιώντας τη πιο κάτω συνάρτηση θα μας επιστρέψει τις *k* πιο σημαντικές διαστάσεις. Έχοντας τη συνάρτηση Chi-Square με τον εξής τύπο:

$$x^2 = \sum \frac{(< \text{τιμή που έχουμε πάρει ως δείγμα} > - < \text{αναμενόμενη τιμή} >)^2}{\text{αναμενόμενη τιμή}}$$

Η συνάρτηση βάζει τις τιμές δειγματοληψίας που ορίσαμε κατά την προσαρμογή του μοντέλου και ανάλογα με τις πιθανότητες που θα επιστρέψει το κάθε χαρακτηριστικό – διάσταση, η συνάρτηση θα μας δώσει ως έξοδο ένα μονοδιάστατο πίνακα με το σκορ της κάθε διάστασης αναλόγως σημαντικότητας της. Και μείς με τη σειρά μας θα πάρουμε τις πρώτες *k* διαστάσεις για να ασχοληθούμε μόνο με αυτές στις μετρήσεις μας. Έτσι πετυχαίνουμε τη μείωση των διαστάσεων στα δεδομένα μας χωρίς όμως να χάνουμε κάτι ουσιαστικό από τη σημαντικότητα των υπόλοιπων χαρακτηριστικών

- <u>Πως υπολογίζουμε τη μέση τιμή και την τυπική απόκλιση των διαστάσεων ενός κόμβου;</u>

  Όπως δείξαμε και στη δομή του edge κόμβου ο καθένας έχει αποθηκευμένη τη μέση τιμή για κάθε χαρακτηριστικό των δεδομένων του. Υπολογίζεται με το γνωστό τρόπο που ξέρουμε από τα μαθηματικά.

$$\overline{dn} = \frac{\sum_{t=0}^{R} X_T}{R}$$

όπου R ο αριθμός εγγραφών κάθε κόμβου. Αυτό γίνεται για όλες τις διαστάσεις M του σετ των δεδομένων του κάθε κόμβου.

Στη συνέχεια έχοντας τη μέση τιμή του κάθε κόμβου υπολογίζουμε την ομοιότητα με την εγγραφή δεδομένων που μόλις πήρα βρίσκοντας την τυπική απόκλιση μεταξύ της τιμής των σημαντικών χαρακτηριστικών των νέων δεδομένων και της μέσης τιμής των χαρακτηριστικών κάθε υποψήφιου κόμβου.

$$s = \sqrt{\frac{\left(d_0 - \overline{d_0}\right)^2 + \left(d_1 - \overline{d_1}\right)^2 + \cdots + \left(d_n - \overline{d_n}\right)^2}{n-1}}$$

Αφού βρω την τυπική απόκλιση για κάθε κόμβο που βρίσκεται στην ομάδα που μας ενδιαφέρει αποθηκεύω τα αποτελέσματα σε ένα μονοδιάστατο πίνακα μήκους όσο και ο αριθμός των κόμβων του δικτύου.

- <u>Πως γίνεται το clustering στους κόμβους;</u>

  Το clustering στους κόμβους γίνεται με βάση την μέση τιμή των διαστάσεων που έχει αποθηκευμένα ο κάθε κόμβος στο σύνολο των δεδομένων του. Για το clustering χρησιμοποιούμε το αλγόριθμο K-Means όπου ο αριθμός των cluster που επιθυμούμε να έχουμε δίνεται σαν παράμετρος. Αφού εκτελεστεί ο αλγόριθμος κάνουμε μια πρόβλεψη σε ποια ομάδα κόμβων μπορεί να ανήκει η νέα εγγραφή δεδομένων που μόλις πήραμε και τότε επιστρέφεται η ομάδα αυτή μειώνοντας έτσι το εύρος των κόμβων που πρέπει να ελέγξω.

- <u>Τι είναι μια αναφορά ενός κόμβου και πως υπολογίζονται τα στοιχεία που περιέχει;</u>

  Όταν ένας κόμβος αποφασίσει ότι θα στείλει τα δεδομένα του σε κάποιο άλλο κόμβο ζητάει από τον καθένα μια αναφορά με κάποια στατιστικά στοιχεία έτσι ώστε να επιλέξει τον κατάλληλο. Με το που ζητηθεί αυτή η αναφορά οι κόμβοι που βρίσκονται στο επικρατέστερο cluster αποστέλουν την αναφορά τους. Μέσα σε αυτήν υπάρχει ο μοναδικός αριθμός που υποδηλώνει την ταυτότητα του κάθε κόμβου, ο μέσος όρος της κάθε διάστασης τους, η τυπική απόκλιση με τα νέα δεδομένα που θα εισαχθούν, το κόστος που χρειάζεται για να σπαταλήσει ο κόμβος για να του σταλούν τα δεδομένα και την ώρα που ενημέρωσε για τελευταία φορά τα

στοιχεία αυτά. Στη συνέχεια με βάση αυτά τα στοιχεία υπολογίζεται και σημειώνεται στην αναφορά η πιθανότητα ο συγκεκριμένος κόμβος να έχει παράγει αυτός τα νέα δεδομένα. Ο υπολογισμός της πιθανότητας γίνεται πολλαπλασιάζοντας τη Gaussian πιθανότητα [64] για κάθε χαρακτηριστικό δεδομένου ότι έχει παραχθεί από το συγκεκριμένο κόμβο.

- Πως λειτουργεί ο μηχανισμός ανταμοιβής;

  Για να επιλεγεί κάποιος κόμβος εφαρμόζουμε ένα μηχανισμό ανταμοιβής. Παίρνουμε πρώτα απ' όλα την Gaussian πιθανότητα που υπολογίσαμε και αν είναι πάνω από ένα κατώφλι δίνουμε στο κόμβο 0.50 πόντους. Πιο κάτω φαίνεται η συνάρτηση που χρησιμοποιήσαμε για τον υπολογισμό αυτής της πιθανότητας.

$$\varphi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$$

  Στη συνέχεια παίρνουμε από την αναφορά που μας έστειλε ο κόμβος, το κόστος που χρειάζεται να δαπανήσουμε για να στείλουμε τα δεδομένα όπου υπολογίζεται βάση πιθανότητας από την ομοιόμορφη κατανομή να μας δώσει ένα οποιοδήποτε αριθμό από το 1 μέχρι το 75 και αν είναι κάτω από ένα κατώφλι του δίνουμε άλλους 0.15 πόντους.

  Τέλος παίρνουμε την τυπική απόκλιση από τα νέα δεδομένα με τη μέση τιμή των διαστάσεων του κάθε κόμβου και αν δεν ξεπερνάει το κατώφλι μας του δίνουμε έξτρα άλλους 0.35 πόντους. Πιο κάτω βλέπουμε τη συνάρτηση που χρησιμοποιήσαμε για να υπολογίσουμε τη τυπική απόκλιση όπου *n* είναι ο αριθμός των κόμβων.

  Ο τρόπος που επιλέγηκε ο διαμοιρασμός των πόντων έγινε μετά από πολλές δοκιμές που τρέξαμε στο μοντέλο προσομοίωσης.

- Πως υπολογίζεται το αντίστοιχο total reward;

  Το τελευταίο στάδιο για την διαδικασία της επιλογής κόμβου γίνεται βάζοντας όλους αυτούς τους συντελεστές πάνω σε μια αντίστροφη σιγμοειδή συνάρτηση όπου και θα μας δώσει το τελικό μας αποτέλεσμα που είναι και το σκορ του κάθε κόμβου. Πιο κάτω φαίνεται η συνάρτηση που χρησιμοποιείται για τον τελικό υπολογισμό της ανταμοιβής όπου α και b είναι παράμετροι που δίνονται πριν την εκτέλεση του προγράμματος.

$$total\_reward = \frac{1}{1 + e^{(a*report\_time + b*reward)}}$$

- Πως υπολογίζεται το solidity των δεδομένων;

  Για να πιστοποιήσουμε ότι διατηρούμε το solidity των δεδομένων μέσα στον κάθε κόμβο χρησιμοποιήσαμε δύο μετρικές. Στην αρχή υπολογίζουμε το Variance (απόκλιση) του συνόλου δεδομένων στον κάθε κόμβο. Αυτή η μετρική μας δίνει το πόσο μακριά βρίσκονται η κάθε μια από τις διαφορετικές τιμές των δεδομένων που έχω σε σχέση με τον μέσο όρο όλου το σετ για κάθε χαρακτηριστικό - διάσταση. Στη συνέχεια ξανά βρίσκουμε την τυπική απόκλιση (Standard Deviation) με τον ίδιο τύπο που δείξαμε παραπάνω. Όσο πιο μικρή είναι και η τυπική απόκλιση συνεπάγεται και το Variance τόσο πιο συμπαγές είναι το σύνολο των δεδομένων στον κόμβο μας.

5. ΠΕΙΡΑΜΑΤΙΚΗ ΑΠΟΤΙΜΗΣΗ

Στο παρόν κεφάλαιο θα παρουσιάσουμε τις μετρικές και οι συνδυασμοί των παραμέτρων που εφαρμόσαμε για να πετύχουμε όσο το δυνατόν καλύτερη απόδοση αλλά και αποτελεσματικότητα στην προτεινόμενη μέθοδο. Αρχικά θα δείξουμε ένα δείγμα από τα σύνολα των δεδομένων που βασιστήκαμε για τη δοκιμή της μεθόδου και στη συνέχεια θα περιγράψουμε τις παραμέτρους που δίνουμε κατά την εκτέλεση του μοντέλου προσομοίωσης που υλοποιήσαμε και ποια η χρηστικότητα τους. Τέλος θα παρουσιάσουμε κυρίως σε μορφή διαγραμμάτων τα αποτελέσματα των μετρήσεων της προσομοίωσης.

5.1 Σύνολα δεδομένων

Για να τρέξουμε την προσομοίωση υιοθετήθηκαν δεδομένα από δύο πηγές. Πήραμε ένα σύνολο δεδομένων μέσω τις ιστοσελίδας Kaggle [66] όπου αφορούσε για μετρήσεις ποιότητας αέρα για τη περιοχή San Diego της Καλιφόρνιας στις Ηνωμένες Πολιτείες Αμερικής. Στο συγκεκριμένο σετ οι μετρήσεις πάρθηκαν από αισθητήρες και αφορούσαν πίεση της ατμόσφαιρας, θερμοκρασία του αέρα, ταχύτητα και κατεύθυνση αέρα, επίπεδα υγρασίας και διάρκεια βροχής. Στη συνέχεια φτιάξαμε ένα άλλο πρόγραμμα που φαίνεται ο κύριος κώδικας του στο Παράρτημα ΙΙ όπου παίρνει τις πληροφορίες για την υγρασία και δημιουργεί για κάθε εγγραφή μια τιμή 1 αν τα επίπεδα της υγρασίας είναι πάνω από 0.50 αλλιώς 0 όπως φαίνεται στην Εικόνα 11. Στους πιο κάτω Πίνακες 1 και 2 φαίνονται μερικές από τις εγγραφές των δεδομένων που χρησιμοποιήθηκαν.

Πίνακας 2: Πραγματικά Δεδομένα από τις μετρήσεις για τη πόλη San Diego

air_pressure	air_temp	avg_wind_direction	avg_wind_speed	max_wind_direction	max_wind_speed	min_wind_direction	min_wind_speed	rain_accumulation	rain_duration	relative_humidity
912.3	64.76	97	1.2	106	1.6	85	1	0	0	60.5
912.3	63.86	161	0.8	215	1.5	43	0.2	0	0	39.9
912.3	64.22	77	0.7	143	1.2	324	0.3	0	0	43
912.3	64.4	89	1.2	112	1.6	12	0.7	0	0	49.5
912.3	64.4	185	0.4	260	1	100	0.1	0	0	58.8
912.3	63.5	76	2.5	92	3	61	2	0	0	62.6
912.3	62.78	79	2.4	89	2.7	62	2	0	0	65.6
912.3	62.42	86	2	92	2.4	75	1.8	0	0	65.2
912.3	62.24	105	1.4	125	1.9	82	1	0	0	65.8
912.3	62.24	93	0.4	126	0.7	14	0.2	0	0	58.6
912.3	62.24	144	1.2	167	1.8	115	0.6	0	0	38.5
912.2	63.14	105	1.6	126	2	92	0.9	0	0	42.6
912.2	64.04	116	1.8	143	2.7	104	1.1	0	0	45.3
912.2	64.4	142	1.1	200	1.9	93	0.7	0	0	36.1
912.2	64.94	150	1.3	173	2.1	117	0.8	0	0	33.2
912.2	65.48	90	1.5	100	1.9	80	1.3	0	0	45.2

Στη συνέχεια δημιούργησα ακόμα ένα πρόγραμμα σε γλώσσα προγραμματισμού Python (βλ. Παράρτημα ΙΙ), όπου παίρνει σαν παραμέτρους πόσες διαστάσεις επιθυμεί ο χρήστης να έχει, πόσες εγγραφές από αυτά τα δεδομένα και ένα διάστημα τιμών όπου θα παράγει τιμές με την ίδια πιθανότητα. Πιο κάτω φαίνεται ένας πίνακα με μερικές από τις εγγραφές που δημιουργήθηκαν μέσω αυτού του προγράμματος.

Πίνακας 3: Δεδομένα που παράχθηκαν μέσω ενός προγράμματος

d0	d1	d2	d3	d4	d5	d6	d7	d8	d9
0.94	0.67	0.56	0.93	0.37	0.83	0.84	0.0	0.81	1
0.62	0.07	0.77	0.39	0.9	0.98	0.17	0.66	0.07	1
0.13	0.64	0.08	0.96	0.59	0.2	0.83	0.13	0.21	1
0.77	0.69	0.34	0.15	0.1	0.57	1.0	0.59	0.91	0
0.9	0.85	0.99	0.23	0.55	0.67	0.46	0.9	0.62	0
0.35	0.73	0.71	0.24	0.22	0.5	0.19	0.91	0.25	1
0.8	0.53	0.97	0.54	0.93	0.28	0.6	0.98	0.65	1
0.71	0.7	0.7	0.11	0.47	0.76	0.12	0.56	0.71	1
0.78	0.29	0.7	0.93	0.11	0.97	0.85	0.68	0.7	1
0.72	0.25	0.19	0.86	0.67	0.17	1.0	0.75	0.77	0
0.68	0.64	0.14	0.57	0.49	0.01	0.81	0.97	0.96	0
0.1	0.24	0.35	0.21	0.44	0.88	0.66	0.81	0.95	0
0.83	0.37	0.33	0.27	0.1	0.79	0.11	0.29	0.47	0
0.84	0.66	0.85	0.75	0.93	0.57	0.96	0.23	0.05	0
0.5	0.79	0.65	0.49	0.22	0.01	0.25	0.27	0.81	0
0.22	0.73	0.81	0.75	0.2	0.62	0.82	0.35	0.46	1

5.2 Παράμετροι πειραμάτων και επιλογή τιμών τους

Σε αυτή τη ενότητα θα αναλύσουμε εκτενέστερα τις παραμέτρους που δίνουμε στην προσομοίωση και ποια η χρηστικότητα της κάθε μιας από αυτές. Συνολικά το σύστημα προσομοίωσης που υλοποιήσαμε παίρνει 13 παραμέτρους. Πιο αναλυτικά έχουμε:

- data_columns: Καθορίζει το πόσα χαρακτηριστικά διαθέτει ολόκληρο το σύνολο δεδομένων που θα χρησιμοποιήσουμε
- data_rows: Καθορίζει το πόσες εγγραφές διαθέτει ολόκληρο το σύνολο δεδομένων που θα χρησιμοποιήσουμε
- num_of_nodes: Ορίζουμε πόσους κόμβους θέλουμε να έχει η προσομοίωση του δικτύου που θα εκτελέσουμε.
- rows_per_node: Καθορίζει το πόσες εγγραφές θα διαμοιραστούν αρχικά σε κάθε κόμβο
- important_cols: Καθορίζει πόσες σημαντικές διαστάσεις θα λαμβάνονται υπόψιν από τα δεδομένα
- phi: Είναι η πιθανότητα κάποιος κόμβος να επιλέξει remote save
- update_stats: Σε κάθε πόσες νέες εγγραφές θα ενημερώνονται τα στατιστικά του κάθε κόμβου που τα περισσότερα χρησιμοποιούνται για την έκδοση της αναφοράς που στέλνουν μεταξύ τους οι κόμβοι.
- transfer_cost: Καθορίζει το πολλαπλασιαστή του κόστους από κόμβο σε κόμβο
- prob_thres: Κατώφλι πιθανότητας να έχουν παραχθεί τα δεδομένα από συγκεκριμένο κόμβο
- cost_thres: Κατώφλι κόστους να στείλουμε τα δεδομένα σε άλλο κόμβο
- AA: Συντελεστής που χρησιμοποιείται στη σιγμοειδή συνάρτηση και τις πλείστες φορές δίνουμε τη τιμή 1
- BB: Συντελεστής που χρησιμοποιείται στη σιγμοειδή συνάρτηση και συνήθως παίρνει τη τιμή 1
- num_of_clusters: Καθορίζει τον αριθμό των ομάδων (clusters) που θα χωριστούν οι κόμβοι

Για την πειραματική αποτίμηση και αξιολόγηση των αποτελεσμάτων θα πειραματιστούμε με διάφορες τιμές για τις ακόλουθες παραμέτρους.

- τον αριθμό των κόμβων που θα έχει το δίκτυο μας (num_of_nodes)
- τον αριθμό των ομάδων που θα χωριστούν οι κόμβοι (num_of_clusters)
- και τέλος πόσες διαστάσεις θεωρούμε σημαντικές (important_cols).

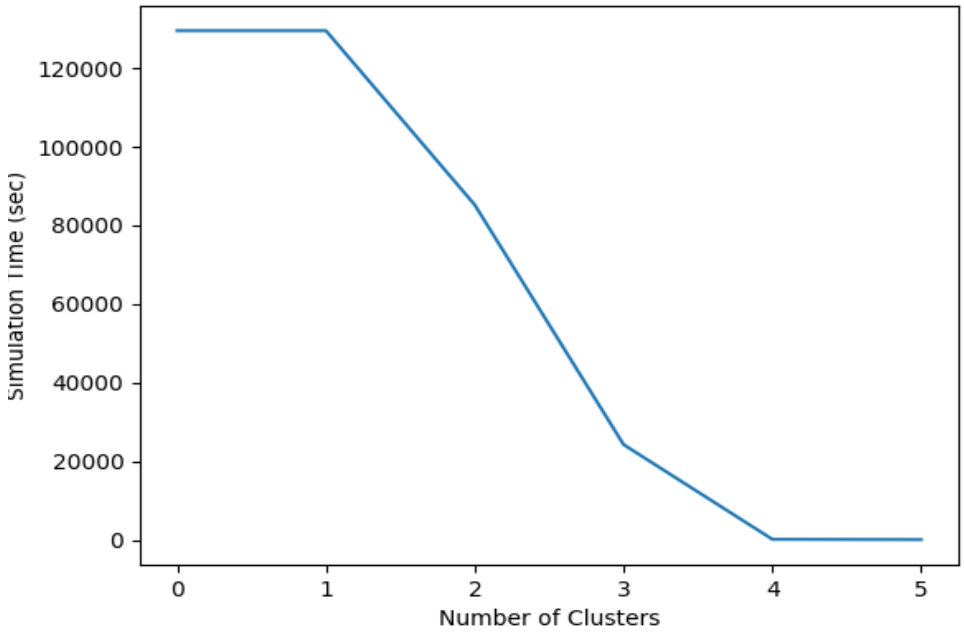
Οι υπόλοιπες παραμέτρους θα παραμείνουν με σταθερές τιμές για τις προσομοιώσεις.

5.3 Αποτελέσματα και αξιολόγηση

Για τις πειραματικές εκτελέσεις που κάναμε για την αξιολόγηση του μοντέλου μας χρησιμοποιήσαμε ένα σύνολο δεδομένων που το παρήγαγε το πρόγραμμα data_producer.py που μέρος του υπάρχει διαθέσιμο στο Παράρτημα ΙΙ. Το σετ έχει μέγεθος 25.000 εγγραφών και κάθε εγγραφή έχει 15 χαρακτηριστικά που παίρνουν τιμές στο διάστημα [0,1]. Σε όλες τις περιπτώσεις διαμοιράσαμε αρχικά από 100 εγγραφές από το σύνολο δεδομένων στους κόμβους έτσι ώστε οι κόμβοι να αποκτήσουν το δικό τους χαρακτήρα.

5.3.1 Εκτέλεση προσομοίωσης με τεχνική ομαδοποίησης κόμβων.

Πρώτα από όλα θέλουμε να επισημάνουμε τη διαφορά που κάνει στο χρόνο εκτέλεσης της προσομοίωσης του δικτύου μας η προσθήκη της τεχνικής ομαδοποίησης (clustering) που χρησιμοποιήσαμε. Όπως φαίνεται και στο πιο κάτω σχήμα ο χρόνος εκτέλεσης μειώνεται δραματικά με το που χρησιμοποιούμε clusters.



Σχήμα 1: Χρόνος εκτέλεσης προσομοίωσης με διαφορετικό αριθμό clusters

Οι διαθέσιμοι κόμβοι στο δίκτυο ήταν 7 και θεωρήσαμε σημαντικά τα 3 μεγαλύτερα σε σκορ χαρακτηριστικά. Στο Σχήμα 1 τώρα παρατηρούμε ότι για τους 7 κόμβους ο

ιδανικός αριθμός cluster είναι τα 4 αφού μετά σταματά να βελτιώνεται ο χρόνος προσομοίωσης. Στα 4 clusters οι κόμβοι χωρίζονται ομοιόμορφα σε ομάδες και έτσι το εύρος κόμβων που θα αναζητήσει το πρόγραμμα για να αποθηκεύει κάθε εγγραφή θα είναι σίγουρα μικρότερο αλλά και αν δούμε το μέγεθος του κάθε συνόλου κόμβων που δημιουργήθηκαν θα είναι ίσο. Αν δούμε για δύο clusters παρατηρούμε ότι και πάλι το σύνολο κόμβων που θα αναζητήσει ο αλγόριθμος μας είναι μεγάλο με αποτέλεσμα να κάνει μεν λιγότερο χρόνο αλλά να μην είναι τόσο αισθητό στο χρήστη που τρέχει τη προσομοίωση.

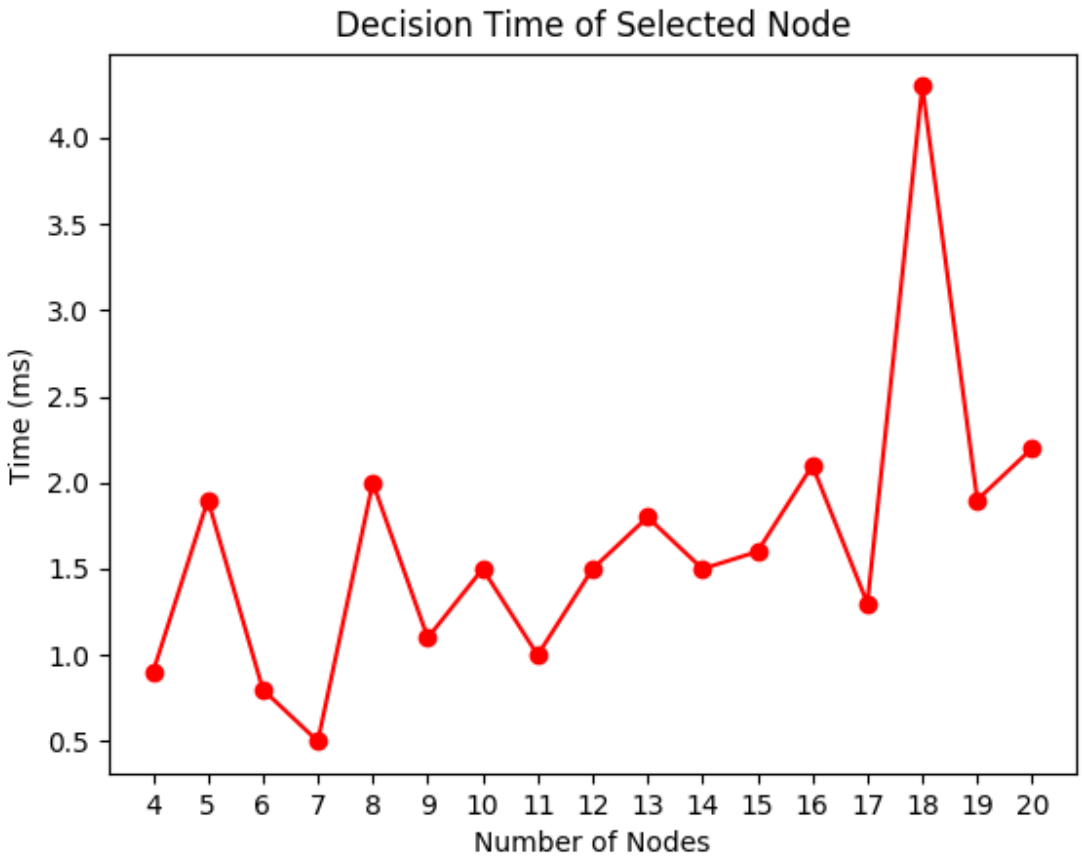
5.3.2 Χρόνος απόφασης επιλογής κόμβου αποθήκευσης

Ο συγκεκριμένος χρόνος αναφέρεται στη περίπτωση όπου επιλέγεται από το πρόγραμμα το Remote Save. Δηλαδή να στείλουμε τα δεδομένα που μας ήρθαν ως είσοδο σε ένα άλλο κόμβο γιατί ο κόμβος που τα έλαβε δεν ήταν ο κατάλληλος. Για τη συγκεκριμένη περίπτωση λάβαμε υπόψιν μας τρεις διαφορετικούς παράγοντες. Ο πρώτος είναι ο αριθμός των κόμβων που υπάρχουν στο δίκτυο, ο δεύτερος ο αριθμός των clusters που έχουμε διαθέσιμα και ο τρίτος το πόσες σημαντικές διαστάσεις θα λαμβάνουμε υπόψιν κατά την αποθήκευση των δεδομένων. Τέλος, τρέξαμε από τρεις φορές την κάθε περίπτωση που εξετάζαμε, και από τις μετρήσεις που παίρναμε υπολογίζαμε το μέσο όρο και τον καταγράφαμε ως τελικό χρόνο απόφασης με σκοπό να έχουμε όσο το δυνατό πιο αντικειμενική εικόνα για το σύστημα μας.

Αριθμός Κόμβων Δικτύου

Αρχικά εκτελέσαμε τη προσομοίωση για να ελέγξουμε πόσο επηρεάζει το χρόνο απόφασης ο αριθμός των κόμβων. Έτσι κρατήσαμε σταθερά καθ’ όλη τη προσομοίωση τον αριθμό των clusters σε 3 και τις σημαντικές διαστάσεις που θα λαμβάνουμε υπόψιν και αυτές σε 3. Στο Σχήμα 2 βλέπουμε την απόδοση του συστήματος για διαφορετικό αριθμό κόμβων και μπορούμε εύκολα να συμπεράνουμε ότι ο αριθμός των κόμβων παίζει ρόλο στο χρόνο απόφασης του συστήματος. Το οποίο είναι λογικό γιατί υπό κανονικές συνθήκες όσο πιο πολλούς κόμβους έχουμε τόσο πιο μεγάλο το εύρος αναζήτησης. Παρόλα αυτά όπως φαίνεται και στο σχήμα η γραφική μας παράσταση δεν είναι συνεχής. Αυτό συμβαίνει γιατί υιοθετείται ο αλγόριθμος ομαδοποίησης. Αν για παράδειγμα πάρουμε την περίπτωση να έχω 7 κόμβους στο δίκτυο μου το πρόγραμμα γίνεται αισθητά πιο αποδοτικό σε σχέση με

το να έχω 5 κόμβους. Άρα καταλήγουμε ότι ο αριθμός των κόμβων που έχω διαθέσιμους παίζει ρόλο στο χρόνο απόφασης για αποθήκευση.

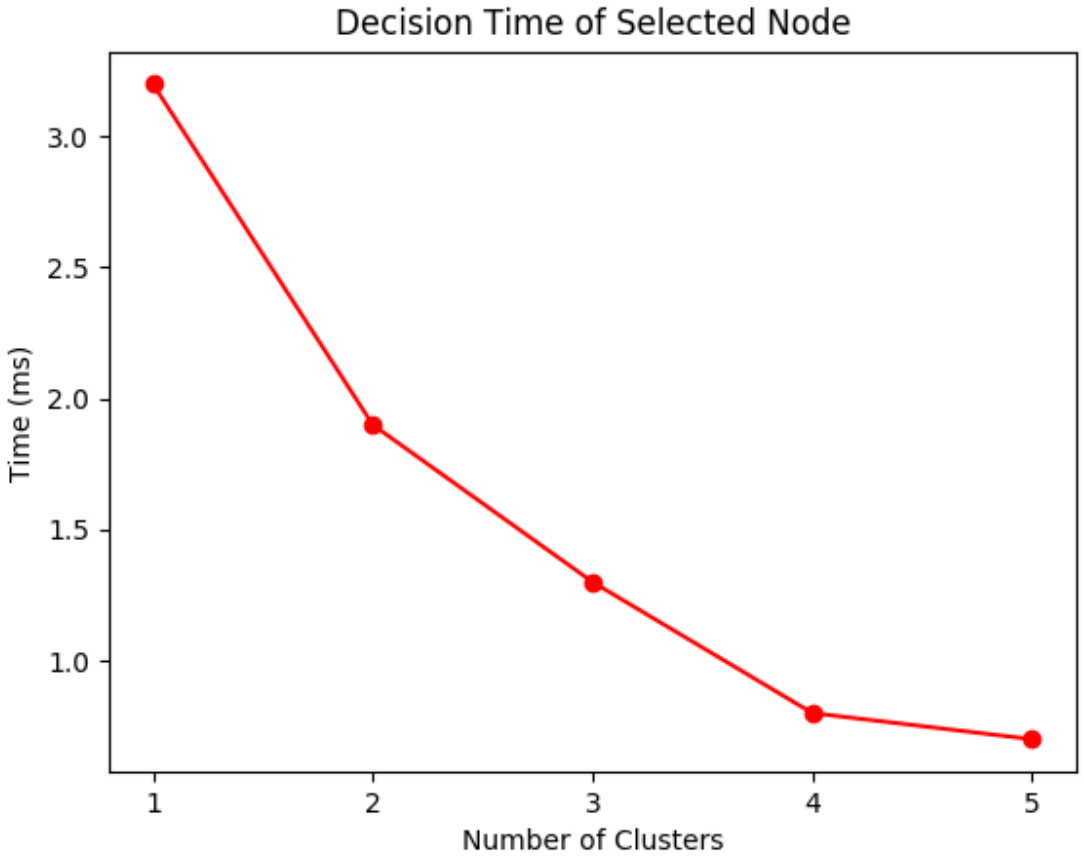


Σχήμα 2: Χρόνος απόφασης για επιλογή κόμβου αποθήκευσης για διαφορετικό αριθμό κόμβων στο δίκτυο

Αριθμός Ομάδων από Κόμβους (Clusters)

Για αυτή τη πειραματική αξιολόγηση επιλέγουμε να συνεχίσουμε με 3 σημαντικά χαρακτηριστικά και 11 διαθέσιμους κόμβους στο δίκτυο μας αντί για 7 που ήταν και η πιο αποδοτική επιλογή. Έτσι θα μπορούμε να έχουμε περισσότερη ευελιξία στο να εξετάσουμε διαφορετικές τιμές για τη συγκεκριμένη παράμετρο. Στο Σχήμα 3 φαίνονται τα αποτελέσματα που πήραμε εκτελώντας τις διάφορες περιπτώσεις. Παρατηρούμε εδώ ότι σε σχέση με το Σχήμα 2 έχουμε μια συνεχή πτώση στο χρόνο εκτέλεσης όσο αυξάνεται ο αριθμός των clusters. Το οποίο είναι λογικό αν σκεφτούμε ότι με το που έρχεται μια νέα εγγραφή δεδομένων στο σύστημα εκτελώντας ένα αλγόριθμο αναγνώρισης, όπου ο κώδικας του υπάρχει διαθέσιμος στο Παράρτημα Ι βρίσκει ποια ομάδα είναι η πιο κατάλληλη και στη συνέχεια ψάχνει να βρει το καταλληλότερο κόμβο από αυτή την ομάδα. Επομένως ο αριθμός των κόμβων που

θα χρειαστεί να στείλουν τη σχετική αναφορά είναι μικρότερος, αρά καταναλώνονται και λιγότεροι πόροι στο σύστημα, αλλά και οι επαναλήψεις που θα χρειαστεί να εκτελεστούν για να πάρει τη τελική του απόφαση είναι λιγότερες από τον συνολικό αριθμό των κόμβων. Άρα καταλήγουμε στο συμπέρασμα ότι και ο αριθμός των ομάδων που θα χωρίσουμε τους κόμβους παίζει ρόλο στο χρόνο απόφασης όμως όσο κοντεύουμε στον αριθμό των διαθέσιμων κόμβων σταματά η βελτίωση.

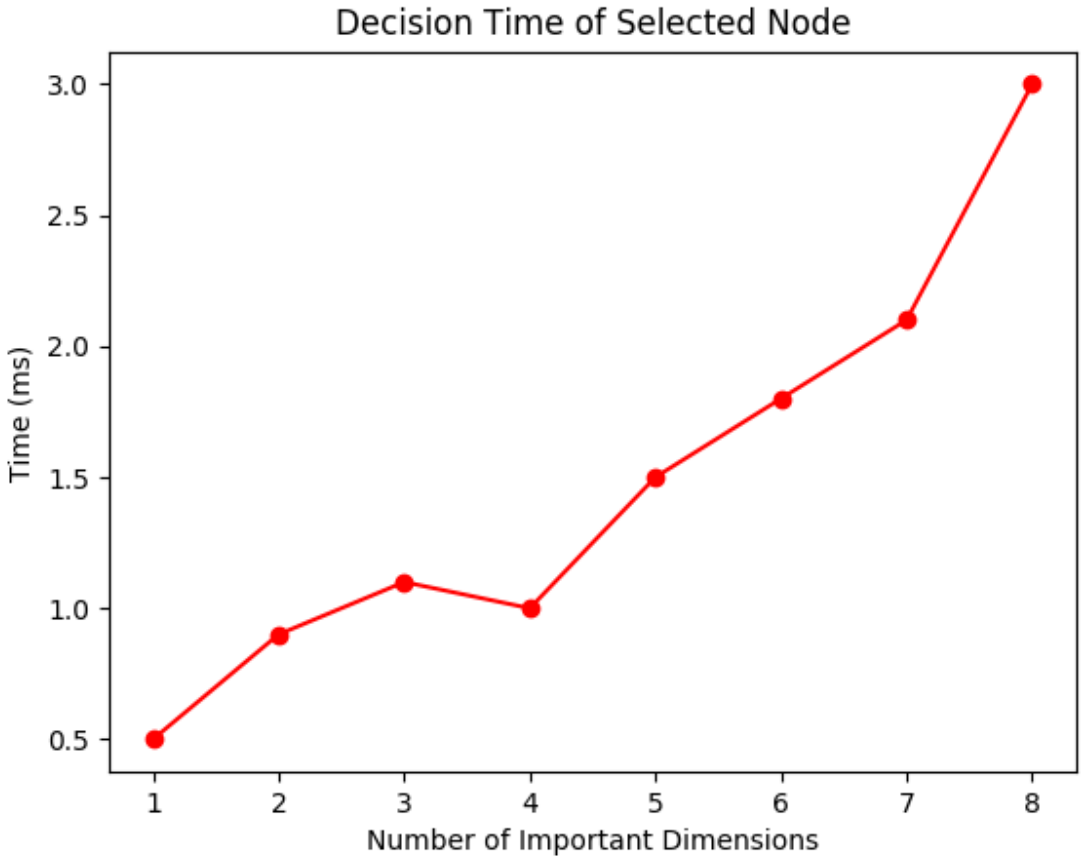


Σχήμα 3: Χρόνος απόφασης για επιλογή κόμβου αποθήκευσης για διαφορετικό αριθμό cluster

Αριθμός Σημαντικών Διαστάσεων

Όπως και στη πιο πάνω περίπτωση έτσι και εδώ επιλέγουμε να έχουμε 11 κόμβους στο δίκτυο μας και 5 διαθέσιμα clusters που ήταν και το πιο αποδοτικό από την πιο πάνω δοκιμή. Βλέποντας λοιπόν το Σχήμα 4 παρατηρούμε ότι όσο πιο πολλές διαστάσεις λάβουμε υπόψιν τόσο πιο πολλούς πόρους χρειάζεται να καταναλώσει το δίκτυο μας μέχρι να αποφασίσει που θα αποθηκεύσει τη νέα εγγραφή. Αν δούμε τη περίπτωση για 4 σημαντικές διαστάσεις παρατηρούμε ότι χρειάστηκε λιγότερο χρόνο να αποφασίσει σε πιο κόμβο θα στείλει τα δεδομένα γεγονός που έρχεται σε αντιπαράθεση με τα όσα είπαμε πιο πάνω. Ο λόγος που συμβαίνει αυτό είναι διότι

μπορεί να έτυχε για τις συγκεκριμένες 4 διαστάσεις η ομάδα κόμβων που επιλέγηκε να είχε μικρό εύρος αναζήτησης με αποτέλεσμα να αποφασίσει πιο γρήγορα παρά από ότι στις 3 σημαντικές διαστάσεις. Αυτό που θα είχε ενδιαφέρον να εξετάσουμε για τη συγκεκριμένη παράμετρο θα είναι το πόσες σημαντικές διαστάσεις θα πρέπει να λαμβάνουμε υπόψιν ώστε ναι μεν να είναι μικρός ο χρόνος απόφασης αλλά και να παραμένει συμπαγές το σύνολο δεδομένων του κάθε κόμβου.

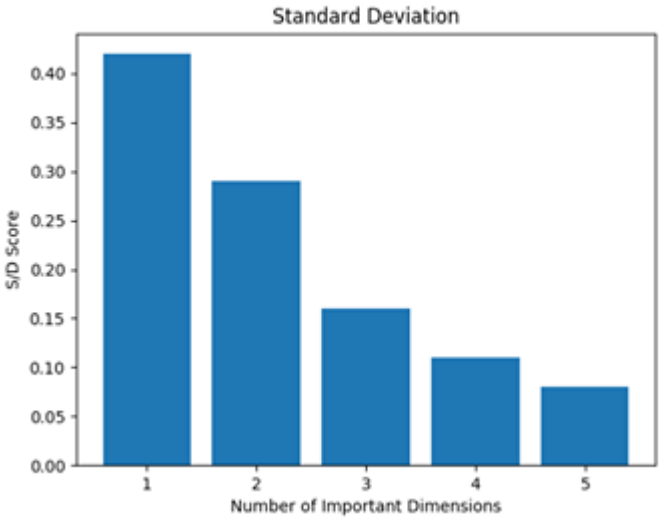


Σχήμα 4: Χρόνος απόφασης επιλογής κόμβου αποθήκευσης για διαφορετικό αριθμό σημαντικών διαστάσεων

5.3.3 Υπολογισμός Τυπικής Απόκλισης του συνόλου των δεδομένων

Θέλοντας να δούμε πόσο ρόλο παίζει ο αριθμός των σημαντικών διαστάσεων που λαμβάνουμε υπόψιν κατά την επιλογή ενός κόμβου αποθήκευσης στην συνολική ποιότητα του συνόλου των δεδομένων του συγκεκριμένου κόμβο υπολογίσαμε για διάφορες τιμές την τυπική απόκλιση του επιλεχθέντα κόμβου. Αναλυτικότερα πήραμε μια νέα εγγραφή από το αρχικό σύνολο δεδομένων που δημιουργήσαμε και αλλάζαμε κάθε φορά τον αριθμό των σημαντικών διαστάσεων της από 1 έως 5. Στη συνέχεια παίρναμε το κόμβο που επιλέγηκε για αποθήκευση αυτής της εγγραφής και

υπολογίζαμε αρχικά τη διακύμανση των τιμών του (variance) και στη συνέχεια την τυπική απόκλιση (standard deviation) αποκλειστικά για τις σημαντικές διαστάσεις που είχαμε κάθε φορά. Έτσι μπορούσαμε να αναγνωρίσουμε πόσο διάσπαρτες είναι η τιμές μέσα σε ένα σύνολο δεδομένων. Στις περιπτώσεις που είχαμε περισσότερες από μια διαστάσεις, παίρναμε το μέσο όρο. Όπως φαίνεται και στο πιο κάτω Σχήμα 5 παρατηρούμε ότι όσο πιο πολλές διαστάσεις λαμβάναμε υπόψιν τόσο πιο ακριβές και συμπαγές ήταν το σετ του κόμβου. Αυτό που πρέπει να διασφαλιστεί όμως είναι, η εξισορρόπηση ανάμεσα στην ακρίβεια του συνόλου των δεδομένων που εμπεριέχει ο κάθε κόμβος και ο φόρτος εργασίας που αντιμετωπίζει το συνολικό δίκτυο.



Σχήμα 5: Τυπική Απόκλιση δεδομένων επιλεχθέντα κόμβου

Θέλοντας λοιπόν να πετύχουμε τη χρυσή τομή στο χρόνο αποθήκευσης αλλά και στη ποιότητα που θα κρατήσουμε στα δεδομένα με βάση του αριθμού των σημαντικών διαστάσεων που θα πρέπει να λαμβάνουμε υπόψιν κατά την αποθήκευση μιας νέας εγγραφής δεδομένων τρέξαμε διάφορες περιπτώσεις μέχρι να βρούμε την πιο αποτελεσματική σε θέμα χρόνο κυρίως για 3 και 4 σημαντικές διαστάσεις αφού όπως είδαμε σε πιο πάνω μετρήσεις από 5 και πάνω ο χρόνος ξεφεύγει ενώ από 2 και κάτω η ακρίβεια χάνεται.

5.3.4 Τελικό Στάδιο Προσομοιώσεων

Λαμβάνοντας υπόψιν όλα τα συμπεράσματα που πήραμε από κάθε ένα από τους παραπάνω ελέγχους για το τι μας προσφέρει η κάθε παράμετρος δοκιμάσαμε διάφορους συνδυασμούς εκτελέσεων μέχρι να πετύχουμε τον καλύτερο δυνατό

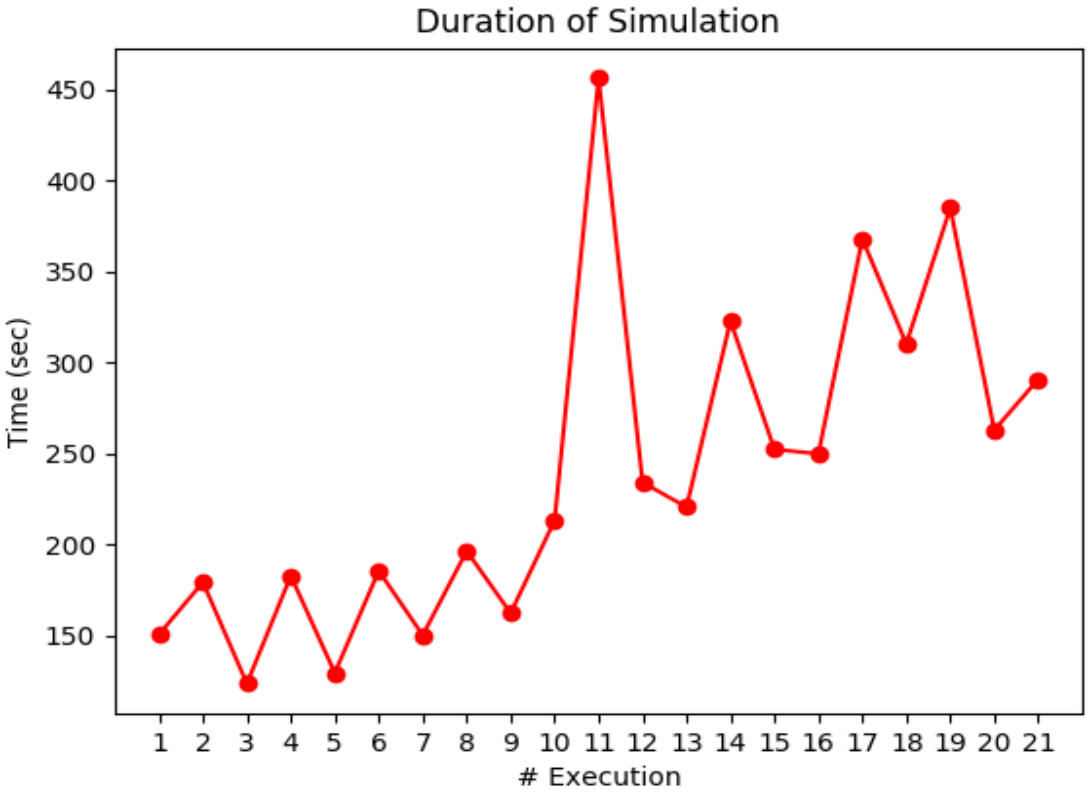
χρόνο. Στο πιο κάτω Πίνακα 4 βλέπουμε τις τιμές στις παραμέτρους που χρησιμοποιήθηκαν

Πίνακας 4: Παράμετροι Προσομοίωσης

A/A Execution	# Nodes	# Cluster	# Important Dimension	Execution Time (sec)
1	3	1	2	150.36
2	5	2	3	179.08
3	7	3	3	123.37
4	7	2	3	182.53
5	7	4	3	129.21
6	9	3	3	185.34
7	9	4	3	149.95
8	10	3	3	196.21
9	10	4	3	162.08
10	10	4	4	213.02
11	12	3	3	456.06
12	12	4	3	233.92
13	12	4	4	220.32
14	15	3	3	322.61
15	15	4	3	252.05
16	15	4	4	249.63
17	18	3	3	367.67

18	18	4	3	310.03
19	18	4	4	384.74
20	18	5	3	262.61
21	18	5	4	290.49

Στη συνέχεια στο Σχήμα 6 φαίνεται η γραφική παράσταση του χρόνου εκτέλεσης της κάθε περίπτωσης. Παρατηρούμε στο Σχήμα 6 ότι οι πιο καλές από θέμα χρόνου εκτελέσεις είναι οι περιπτώσεις 3, 5, 7 και 9. Αφού λάβαμε υπόψιν μας και τις πιο αποδοτικές τιμές στις παραμέτρους και από τον αριθμό των cluster, τον αριθμό των σημαντικών διαστάσεων ώστε να μην έχουμε διάσπαρτες τιμές στους κόμβους αλλά και αρκετές περιπτώσεις για τους διαθέσιμους κόμβους που θα έχουμε στο δίκτυο μου, πρέπει να καταλήξουμε σε μια προτεινόμενη ρύθμιση για την προσομοίωση. Έτσι λοιπόν στην επόμενη θα αξιολογήσουμε με βάση ακόμα μιας μετρικής την εικόνα που μας παρέχει το δίκτυο.



Σχήμα 6: Χρόνοι Προσομοίωσης

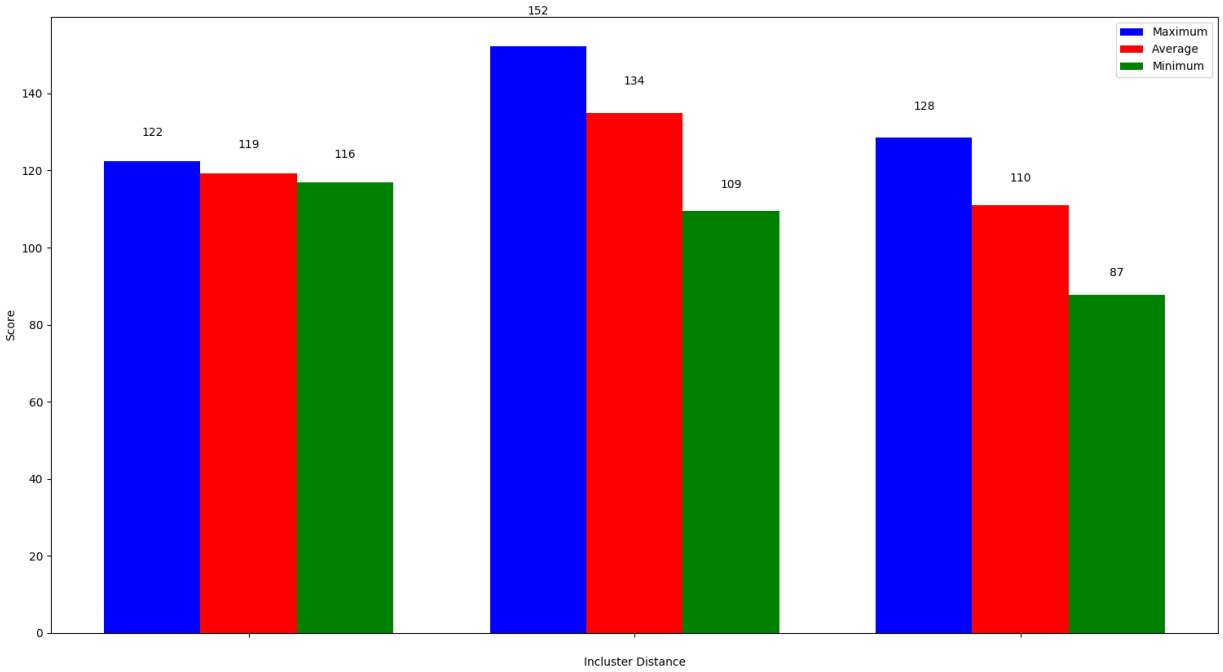
5.3.5 Incluster Απόσταση

Τέλος η τελευταία αξιολόγηση που κάναμε πάνω στο μοντέλο το οποίο δημιουργήσαμε ήταν να δούμε πόση απόσταση απέχουν μεταξύ τους, τα κέντρα των ομάδων από κόμβους που δημιουργήσαμε χρησιμοποιώντας τον αλγόριθμο K-Means. Όσο μεγαλύτερη απόσταση βρίσκουμε τόσο το καλύτερο, υποδηλώνοντας έτσι ότι οι κόμβοι που είναι όμοιοι μεταξύ τους βρίσκονται στην ίδια ομάδα. Για να πετύχουμε τη μέγιστη απόδοση του συστήματος προσομοίωσης για το Edge δίκτυο μας, είχαμε συνήθως από 3 έως 5 ομάδες αναλόγως και τον αριθμό των κόμβων. Με αποτέλεσμα να βρίσκουμε πολλές αποστάσεις ανάλογα και με το ποιες δύο ομάδες εξετάζαμε. Έτσι στο Πίνακα 5 παρουσιάζουμε τις τιμές των παραμέτρων που χρησιμοποιήσαμε για τον υπολογισμό της απόστασης.

Πίνακας 5: Τιμές παραμέτρων

A/A Execution	3	9	20
# Clusters	3	4	5
# Nodes	7	10	18

Πιο κάτω στο Σχήμα 7 δείχνουμε τα αποτελέσματα των τριών εκτελέσεων. Σε κάθε εκτέλεση παρουσιάζονται τρείς τιμές, η μέγιστη απόσταση, ο μέσος όρος των αποστάσεων αλλά και η ελάχιστη απόσταση μεταξύ των clusters.



Σχήμα 7: Incluster Απόσταση

Τέλος παρατηρούμε ότι ο καλύτερος συνδυασμός είναι αυτός της εκτέλεσης 9 και αν αναλογιστούμε ότι τα δεδομένα που χρησιμοποιήσαμε είναι από 0 έως 1 η τιμή της Incluster απόστασης είναι ικανοποιητική. Ο κώδικας του υπολογισμού της απόστασης φαίνεται στο Παράρτημα Ι.

6. ΣΥΜΠΕΡΑΣΜΑΤΑ

Η διαδικασία διαμοιρασμού των δεδομένων σε κόμβους του δικτύου αποτελεί μια από τις πιο σημαντικές διεργασίες που θα πρέπει να παρέχει αλλά και να μπορεί κάποιος να εκμεταλλευτεί από ένα Edge Computing δίκτυο προσφέροντας έτσι πιο γρήγορη απόκριση συνολικά στο δίκτυο. Η συγκεκριμένη υλοποίηση παρόλο που αξιολογήθηκε για σκοπούς αυτής της έρευνας σε δεδομένα της τάξης μεγέθους των 25 χιλιάδων εγγραφών δοκιμάστηκε και μπορεί να ανταπεξέλθει το ίδιο αποδοτικά σε αναλογία μεγέθους του 1.5 εκατομμυρίου εγγραφών και χωρίς να έχουμε οποιαδήποτε διακοπή ή φθορά στην προσομοίωση του δικτύου. Η υλοποίηση του Edge Computing ήρθε για να μείνει για τα καλά στη ζωή μας και ειδικότερα στο κόσμο του Διαδικτύου των Πραγμάτων και πιστεύω ότι είναι θέμα χρόνου να εξελιχθεί ακόμη περισσότερο και οι edge κόμβοι να μπορούν να υποστηρίζουν ακόμα πιο βαριές αλλά συχνές διεργασίες.

Σε αυτή την εργασία προτάθηκε ένα μοντέλο προσομοίωσης ενός δικτύου τύπου Edge Computing όπου έχουμε ένα σύνολο από ΔτΠ συσκευές που λειτουργούν ως πηγές δεδομένων και ένα σύνολο από edge κόμβους όπου ο καθένας ξεχωριστά βρίσκεται κοντά σε κάποιες από αυτές τις πηγές, αποθηκεύει δεδομένα που παράγουν οι πηγές και σιγά σιγά αποκτά το δικό του χαρακτήρα με αποτέλεσμα να αποθηκεύει μόνο συγγενικά με αυτόν νέα δεδομένα. Ο αλγόριθμος για το διαμοιρασμό των δεδομένων στους κόμβους βασίστηκε και χρησιμοποιεί κάποιους υφιστάμενους αλγορίθμους. Εν συντομία η διαδικασία που ακολουθεί ο προτεινόμενος αλγόριθμος είναι να ομαδοποιεί τους κόμβους χρησιμοποιώντας τον αλγόριθμο K-Means με βάση το μέσο όρο των σημαντικών διαστάσεων του κόμβου. Για να βρούμε ποιες είναι οι σημαντικές διαστάσεις χρησιμοποιούμε τη γνωστή και από τη θεωρία των πιθανοτήτων συνάρτηση Chi-Square όπου βλέποντας κάποια πληροφορία που μας δίνουν τα δεδομένα υπολογίζει ένα σκορ για το κάθε χαρακτηριστικό-διάσταση και στο τέλος επιστρέφει τις *n* πρώτες διαστάσεις. Στη συνέχεια ζητάει από τη πηγή νέα δεδομένα και βρίσκει πρώτα την ομάδα με τους υποψήφιους κόμβους που ανήκουν, ψάχνει αναλόγως τον κατάλληλο κόμβο για να αποθηκεύσει τα δεδομένα. Αυτό επιτυγχάνεται χρησιμοποιώντας ένα μηχανισμό ανταμοιβής που φτιάξαμε, που με βάση κάποιες συνθήκες δίνει πόντους στους υποψήφιους κόμβους. Ακολούθως βάζουμε ως συντελεστές τους πόντους που μάζεψε ο καθένας σε μια σιγμοειδή συνάρτηση και επιλέγεται ο κόμβος με τη μεγαλύτερη τιμή. Στο τέλος υπολογίζουμε τη τυπική απόκλιση για να διαπιστώσουμε πόσο συμπαγή είναι τα δεδομένα που βρίσκονται στο

κάθε κόμβο ξεχωριστά. Ο πιο πάνω μηχανισμός μας παρέχει ένα δίκτυο αποτελούμενο από Edge κόμβους και με ταξινομημένα τα δεδομένα που υπάρχουν σε αυτό.

Στο μέλλον θα γίνει μια προσπάθεια να επεκταθεί το υφιστάμενο μοντέλο έτσι ώστε να μπορεί να αναγνωρίζει το δίκτυο από μόνο του ανά πάσα στιγμή ποιος κόμβος να έχει τα δεδομένα που χρειάζεται αλλά και να μπορεί να εκτελεί στο κόμβο ερωτήματα τύπου Query για τα δεδομένα που έχει ο κόμβος στη κατοχή του.

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Ξενόγλωσσος όρος	**Ελληνικός Όρος**
Application	Εφαρμογή
Bandwidth	Εύρος Ζώνης
Big Data	Τεράστιος Όγκος Δεδομένων
Classification	Ταξινόμηση
Cloud	Νέφος Αποθήκευσης
Clusters	Δημιουργία ομάδων με όμοια αντικείμενα
Connectivity	Συνδεσιμότητα
Dimensional Reduction	Μείωση διαστάσεων σε πολυδιάστατα δεδομένα
Edge Computing	Διαχείριση Δεδομένων στις Παρυφές Διαδικτύου
Edge Nodes	Κόμβοι στις ακμές του δικτύου
Infrastructure	Υποδομή
Ingestion	Κατάποση
Internet of Things	Διαδίκτυο των Πραγμάτων
Latency	Καθυστέρηση
Layer	Επίπεδο
Node	Κόμβος Δικτύου
Open-Source	Πρόγραμμα ανοιχτού κώδικα
Outlier Detection	Εντοπισμός Ανώμαλων Τιμών
Standard Deviation	Τυπική Απόκλιση
Variance	Απόκλιση

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ

AWS	Amazon Web Services
ACCBA	Asynchronous Consensus Based Bundle Algorithm
CBBA	Consensus Based Bundle Algorithm
DBSCAN	Density Base Spatial Clustering of Application with Noise
DDoS	Distributed Denial of Service
GCA	General Component Analysis
INRIA	Institut National de Recherche en Informatique et en Automatique
IoT	Internet of Things
kNN	k-Nearest Neighbor
LDA	Linear Discriminant Analysis
LOF	Local Outlier Factor
LRD	Local Reachability Density
ML	Machine Learning
PCA	Principal Component Analysis
RFID	Radio Frequency Identification
SBC	Single Board Computer
SGA	Sequential Greedy Algorithm
SOM	System on Module
SVM	Support Vector Machines
TPU	Tensor Processing Unit
WiFi	Wireless Fidelity
ΔτΠ	Διαδίκτυο των Πραγμάτων
ΕΚΠΑ	Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

ΠΑΡΑΡΤΗΜΑ Ι

(Κώδικας Μοντέλου)

1. Υπολογισμός μέσου όρου που θα χρησιμοποιηθεί για την εύρεση και του Variance αλλά και του Standard Deviation.

```
def calc_avg(dn):
    res = dn.data.mean(axis=0).round(3)
    c = len(dn.data.columns)
    for i in range (c):
        dn.avg_[i] = res[i]
```

2. Ταξινόμηση και επιλογή κατάλληλων δεδομένων και στη συνέχεια εύρεση σημαντικών διαστάσεων.

```
def classify_df(dn):
    # Convert the data to be able to classify
    X = dn.data.iloc[:, :dn.noc-1]
    y = dn.data.iloc[:, dn.noc-1]

    # Extract the K most important columns
    topD = SelectKBest(score_func=chi2, k=important_cols)
    fit = topD.fit(X, y)
    dfscores = pd.DataFrame(fit.scores_)
    dfcolumns = pd.DataFrame(X.columns)
    featureScores = pd.concat([dfcolumns, dfscores], axis=1)
    featureScores.columns = ['Dimensions', 'Score']  # naming the dataframe columns
    temp = featureScores.nlargest(important_cols, 'Score')  # print n best features
    for i in range(important_cols):
        dn.dScore[i] = temp['Dimensions'].index[i]
```

3. Μηχανισμός ανταμοιβής υποψήφιου κόμβου

```
if Gauss_prob[node.id] > prob_thres:
    r += 0.5
if cost[node.id] > cost_thres:
    r += 0.15
if Similarity[node.id] < similar_thres:
    r += 0.35
rewards[node.id] = 1 / (1 + math.exp((AA*time_passed.total_seconds())+(BB*r)))
```

4. Ομαδοποίηση κόμβων και υπολογισμός incluster απόστασης.

```
def cluster_dns(cl_arr, pin, new_row):
    temp_cl_arr = []

    kmeans = KMeans(n_clusters=num_of_cluster, init='k-means++', max_iter=300, n_init=10, random_state=1)
    kmeans.fit(pin)  # data is of shape [1000,]
    labels = kmeans.predict(pin)  # labels of shape [1000,] with values 0<= i <= 9
    centroids = kmeans.cluster_centers_  # means of shape [10,]
    #incluster distance
    edist = euclidean_distances(kmeans.cluster_centers_)
    print "Incluster Distance = ", edist
    tri_dists = edist[np.triu_indices(3, 1)]
    max_dist, avg_dist, min_dist = tri_dists.max(), tri_dists.mean(), tri_dists.min()
    print max_dist, avg_dist, min_dist
    new_insert = kmeans.predict (new_row)

    for cl in range (0, num_of_cluster):
        temp_cl_arr = np.where(labels == cl)
        cl_arr.append(temp_cl_arr[0])
    return new_insert
```

ΠΑΡΑΡΤΗΜΑ ΙΙ

(Κώδικας βοηθητικών προγραμμάτων)

1. Πρόγραμμα παραγωγής δεδομένων.

```
for i in range(columns):
    header.append("d" + str(i))

with open('data.csv', 'wb') as file:
    writer = csv.writer(file, delimiter = ',')
    writer.writerow(header);
    for i in range(rows):
        rec = []
        for j in range(columns):
            if (j == columns-1):
                rec.append(random.choice([0, 1]))
            else:
                rec.append(round(random.random(), 2))
        writer.writerow(rec)
file.close()
```

ΑΝΑΦΟΡΕΣ – ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Επεξήγηση της ευρύτερης έννοιας του ΔτΠ μέσω της ψηφιακής εγκυκλοπαίδειας Wikipedia https://en.wikipedia.org/wiki/Internet_of_things [Προσπελάστηκε 11/08/20]

[2] Βιογραφικό για το Kevin Ashton μέσω της ψηφιακής εγκυκλοπαίδειας Wikipedia https://en.wikipedia.org/wiki/Kevin_Ashton [Προσπελάστηκε 11/08/20]

[3] Δημοσιευμένο άρθρο με τίτλο «The IoT Rundown for 2020: Stats, Risks and Solutions στο Security Today» από τον Gilad David Maayan, στις 13 Ιανουαρίου 2020 https://securitytoday.com/articles/2020/01/13/the-iot-rundown-for-2020.aspx [Προσπελάστηκε στις 11/08/20]

[4] Άρθρο με τίτλο «The Layers of IoT» από το ΔτΠ Sense στις 10 Ιουνίου 2018 https://iotsense.io/blog/the-layers-of-iot/ [Προσπελάστηκε 12/08/20]

[5] Δημοσιευμένο άρθρο στο Medium με τίτλο «Stages of IoT architecture explained in simple words» https://medium.com/datadriveninvestor/4-stages-of-iot-architecture-explained-in-simple-words-b2ea8b4f777f [Προσπελάστηκε 12/08/20]

[6] Επεξήγηση για τους μικρό ελεγκτές από τη ψηφιακή εγκυκλοπαίδεια Wikipedia https://en.wikipedia.org/wiki/List_of_common_microcontrollers#Microchip_Technology [Προσπελάστηκε 12/08/20]

[7] Επεξήγηση για τα Single Board Computer από τη ψηφιακή εγκυκλοπαίδεια Wikipedia https://en.wikipedia.org/wiki/Single-board_computer [Προσπελάστηκε 13/08/20]

[8] Άρθρο στο Circuit Digest από τον Emmanuel Odunlade δημοσιευμένο στις 22 Οκτωβρίου 2018 https://circuitdigest.com/article/top-hardware-platforms-for-internet-of-things-iot [Προσπελάστηκε 13/08/20]

[9] Επεξήγηση για το Arduino από τη ψηφιακή εγκυκλοπαίδεια Wikipedia https://el.wikipedia.org/wiki/Arduino [Προσπελάστηκε 14/08/20]

[10] Επίσημη ιστοσελίδα του Arduino και επεξήγηση της επιτυχής ιστορίας τους. https://www.arduino.cc/en/guide/introduction [Προσπελάστηκε 14/08/20]

[11] Τεχνικά χαρακτηριστικά Arduino Uno από την επίσημη ιστοσελίδα του καταστήματος. https://store.arduino.cc/usa/arduino-uno-rev3 [Προσπελάστηκε 14/08/20]

[12] Επεξήγηση για το Raspberry Pi από τη ψηφιακή εγκυκλοπαίδεια Wikipedia https://en.wikipedia.org/wiki/Raspberry_Pi [Προσπελάστηκε 14/08/20]

[13] Η ιστορία πίσω από το Raspberry Pi από την επίσημη ιστοσελίδα τους. https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/ [Προσπελάστηκε 14/08/20]

[14] Πληροφορίες για το Raspbian OS από τη ψηφιακή εγκυκλοπαίδεια Wikipedia https://en.wikipedia.org/wiki/Raspberry_Pi_OS [Προσπελάστηκε 14/08/20]

[15] Το Coral TPU Board από τη ψηφιακή εγκυκλοπαίδεια Wikipedia. https://en.wikipedia.org/wiki/Tensor_Processing_Unit [Προσπελάστηκε 16/08/20]

[16] Επίσημη ιστοσελίδα του Coral TPU και επεξήγηση της λογικής που χρησιμοποίησαν για την υλοποίηση. https://coral.ai/products/accelerator/ [Προσπελάστηκε 16/08/20]

[17] Επεξήγηση σε βάθος του Coral Edge TPU https://qengineering.eu/google-corals-tpu-explained.html [Προσπελάστηκε 16/08/20]

[18] Η ιστορία πίσω από το Intel Edison από τη ψηφιακή εγκυκλοπαίδεια Wikipedia. https://en.wikipedia.org/wiki/Intel_Edison [Προσπελάστηκε 16/08/20]

[19] Δημοσιευμένο άρθρο στο Πανεπιστήμιο του Barkeley με τίτλο «TinyOS: An Operating System for Sensor Networks» https://people.eecs.berkeley.edu/~pister/290Q/Papers/levis06tinyos.pdf [Προσπελάστηκε 17/08/20]

[20] Επεξήγηση για το TinyOS από τη ψηφιακή βιβλιοθήκη Wikipedia https://en.wikipedia.org/wiki/TinyOS [Προσπελάστηκε 17/08/20]

[21] ESTCude-1 από τη ψηφιακή βιβλιοθήκη Wikipedia https://en.wikipedia.org/wiki/ESTCube-1 [Προσπελάστηκε 17/08/20]

[22] Πληροφορίες για το Contiki από τη ψηφιακή εγκυκλοπαίδεια Wikipedia. https://en.wikipedia.org/wiki/Contiki [Προσπελάστηκε 17/08/20]

[23] Πληροφορίες για το RIOT OS από τη ψηφιακή εγκυκλοπαίδεια Wikipedia. https://en.wikipedia.org/wiki/RIOT_(operating_system) [Προσπελάστηκε 17/08/20]

[24] Επίσημη ιστοσελίδα του RIOT OS https://www.riot-os.org [Προσπελάστηκε 17/08/20]

[25] Δημοσιευμένο άρθρο στο IEEE Διαδίκτυο των Πραγμάτων Journal «RIOT: An Open Source Operating System for Low-End embedded devices in the IoT» https://www.riot-os.org/docs/riot-ieeeiotjournal-2018.pdf [Προσπελάστηκε 17/08/20]

[26] Δημοσιευμένο άρθρο στο INFOCOM για το RIOT OS με τίτλο «RIOTOS: Towards an OS for the Διαδίκτυο των Πραγμάτων» https://www.riot-os.org/docs/riot-infocom2013-abstract.pdf [Προσπελάστηκε 17/08/20]

[27] Επίσημη ιστοσελίδα του Ubuntu Core https://ubuntu.com/core [Προσπελάστηκε 18/08/20]

[28] Documentation για το Ubuntu Core https://core.docs.ubuntu.com/en/ [Προσπελάστηκε 18/08/20]

[29] Δημοσιευμένο άρθρο στο InfoWorld με τίτλο «Ubuntu Core has the keys to IoT Security» https://www.infoworld.com/article/3147793/ubuntu-core-has-the-keys-to-iot-security.html [Προσπελάστηκε 18/08/20]

[30] Πληροφορίες για το Fuchsia OS από τη ψηφιακή εγκυκλοπαίδεια Wikipedia. https://en.wikipedia.org/wiki/Google_Fuchsia [Προσπελάστηκε 18/08/20]

[31] Επίσημη ιστοσελίδα του Amazon Free RT OS https://aws.amazon.com/freertos/ [Προσπελάστηκε 18/08/20]

[32] Δημοσιευμένο άρθρο στην Amazon για τις διαδικτυακές υπηρεσίες που παρέχει και το συνδυασμό μαζί με το Free RT OS https://aws.amazon.com/freertos/ [Προσπελάστηκε 18/08/20]

[33] Δημοσιευμένο άρθρο στο Hindawi με τίτλο «Διαδίκτυο των Πραγμάτων: Architectures, Protocols and Applications» του Rajesh Khanna https://www.hindawi.com/journals/jece/2017/9324035/ [Προσπελάστηκε 12/08/20]

[34] Δημοσιευμένο άρθρο στη Forbes του Ralph Jennings με αφορμή τη έκθεση Computex Taipei και μια εισαγωγή στη τεχνολογία edge computing αναδημοσιευμένο στο Capital.gr https://www.capital.gr/forbes/3296736/to-edge-computing-apokentronei-tin-apothikeusi-dedomenon [Προσπελάστηκε 19/08/20]

[35] Επεξήγηση της τεχνολογίας Edge Computing και σύγκριση με τη τεχνολογία Cloud από την επίσημη ιστοσελίδα της εταιρείας IBM https://www.ibm.com/cloud/what-is-edge-computing [Προσπελάστηκε 19/08/20]

[36] Δημοσιευμένο άρθρο με τίτλο The Benefits and Potentials of Edge Computing της Kaylie Gyarmathy https://www.vxchnge.com/blog/the-5-best-benefits-of-edge-computing [Προσπελάστηκε 19/08/20]

[37] Δημοσιευμένο άρθρο με τίτλο Introduction to classification algorithms Upasana Priyadarshiny in Dzone https://dzone.com/articles/introduction-to-classification-algorithms [Προσπελάστηκε 20/08/20]

[38] Δημοσιευμένο άρθρο από το Towards Data Science με τίτλο Logistic Regression της Saishruthi Swaminathan https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc [Προσπελάστηκε 20/08/20]

[39] Δημοσιευμένο άρθρο από το Towards Data Science με τίτλο Support Vector του Machines Rushikesh Pupale https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989 [Προσπελάστηκε 20/08/20]

[40] Εκτενή επεξήγηση για το Classifier Perceptron https://deepai.org/machine-learning-glossary-and-terms/perceptron [Προσπελάστηκε 20/08/20]

[41] Δημοσιευμένο άρθρο στο Science Direct με τίτλο Object Classification Methods των Cheng-Jin Du και Da-Wen Sun https://www.sciencedirect.com/topics/computer-science/bayesian-classification [Προσπελάστηκε 20/08/20]

[42] Δημοσιευμένο άρθρο στο Towards Data Science στις 17 Μαιου 2017 με τίτλο Decision Trees in Machine Learning του Prashant Gupta https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052 [Προσπελάστηκε 20/08/20]

[43] Δημοσιευμένο άρθρο στις 14 Ιουλίου 2019 στο Medium με τίτλο Main Types of Neural Networks and its Applications των Patrik Shukla και Roberto Iriondo https://medium.com/towards-artificial-intelligence/main-types-of-neural-networks-and-its-applications-tutorial-734480d7ec8e [Προσπελάστηκε 20/08/20]

[44] Δημοσιευμένο άρθρο στο Science Direct με τίτλο Clustering Algorithm του Yaguo Lei https://www.sciencedirect.com/science/article/pii/B9780128115343000044 [Προσπελάστηκε 20/08/20]

[45] Πληροφορίες για τον αλγόριθμο K-Means από τη ψηφιακή εγκυκλοπαίδεια Wikipedia. https://en.wikipedia.org/wiki/K-means_clustering [Προσπελάστηκε 20/08/20]

[46] Δημοσιευμένο άρθρο στο Atomic Object με τίτλο Mean Shift Clustering του Matt Nedrich https://spin.atomicobject.com/2015/05/26/mean-shift-clustering/ [Προσπελάστηκε 20/08/20]

[47] Πληροφορίες για τον αλγόριθμο DBSCAN από τη ψηφιακή εγκυκλοπαίδεια Wikipedia. https://en.wikipedia.org/wiki/DBSCAN [Προσπελάστηκε 20/08/20]

[48] Πληροφορίες για τον αλγόριθμο Expectation Maximization από τη ψηφιακή εγκυκλοπαίδεια Wikipedia. https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm [Προσπελάστηκε 20/08/20]

[49] Επεξήγηση του αλγορίθμου Agglomeratie Hierarchical για Clustering από το Datanovia https://www.datanovia.com/en/lessons/agglomerative-hierarchical-clustering/ [Προσπελάστηκε 20/08/20]

[50] Δημοσιευμένο άρθρο στο Towards Data Science με τίτλο The five Clustering Algorithms Data Scientists Need to Know του George Seif https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68 [Προσπελάστηκε 20/08/20]

[51] Πληροφορίες για τον αλγόριθμο K-Nearest Neighbors από τη ψηφιακή εγκυκλοπαίδεια Wikipedia. https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm [Προσπελάστηκε 21/08/20]

[52] Πληροφορίες για την συνάρτηση υπολογισμού της Ευκλείδειας Απόστασης από τη ψηφιακή εγκυκλοπαίδεια Wikipedia. https://en.wikipedia.org/wiki/Euclidean_distance [Προσπελάστηκε 21/08/20]

[53] Πληροφορίες για την συνάρτηση υπολογισμού της απόστασης Μανχάταν από τη ψηφιακή εγκυκλοπαίδεια Wikipedia. https://en.wikipedia.org/wiki/Taxicab_geometry [Προσπελάστηκε 21/08/20]

[54] Πληροφορίες για την συνάρτηση υπολογισμού της Minkowski απόστασης από τη ψηφιακή εγκυκλοπαίδεια Wikipedia. https://en.wikipedia.org/wiki/Minkowski_distance [Προσπελάστηκε 21/08/20]

[55] Δημοσιευμένο άρθρο στο Medium στις 29 Ιανουαρίου 2019 με τίτλο Isolation Forest Algorithm for Anomaly Detection του Arpit. https://medium.com/@often_weird/isolation-forest-algorithm-for-anomaly-detection-f88af2d5518d [Προσπελάστηκε 21/08/20]

[56] Δημοσιευμένο άρθρο στο Medium στις 6 Δεκεμβρίου 2018 με τίτλο Local Outlier Factor for Anomaly Detection του Phillip Wenig https://towardsdatascience.com/local-outlier-factor-for-anomaly-detection-cc0c770d2ebe [Προσπελάστηκε 21/08/20]

[57] Πληροφορίες για την διαδικασία ανάλυσης και μείωσης των διαστάσεων στα διανύσματα των δεδομένων Principal Component Analysis μέσω της ψηφιακής εγκυκλοπαίδειας Wikipedia. https://en.wikipedia.org/wiki/Principal_component_analysis [Προσπελάστηκε 21/08/20]

[58] Δημοσιευμένο άρθρο πάνω στο PCA με τίτλο Co-Variance and Principal Component Analysis http://pmaweb.caltech.edu/~physlab/lab_21_current/Ph21_5_Covariance_PCA.pdf [Προσπελάστηκε 21/08/20]

[59] Δημοσιευμένο άρθρο στο Towards Data Science στις 28 Μαρτίου 2017 πάνω στο LDA με τίτλο Is LDA a dimensionality reduction technique or a classifier algorithm του Meigarom Lopes https://towardsdatascience.com/is-lda-a-dimensionality-reduction-technique-or-a-classifier-algorithm-eeed4de9953a [Προσπελάστηκε 21/08/20]

[60] Δημοσιευμένο άρθρο στο PlosOne στις 9 Ιουλίου 2018 με τίτλο General Component Analysis (GCA): A new approach to identify Chinese corporate bond market structures https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0199500 [Προσπελάστηκε 21/08/20]

[61] Μια δημοσιευμένη διπλωματική μελέτη με τίτλο «Decentralized Task Allocation for Dynamic Environments» του Luke B. Johnson http://acl.mit.edu/papers/Johnson_Masters12.pdf [Προσπελάστηκε 22/08/20]

[62] Μια δημοσιευμένη έρευνα με τίτλο «Efficient consensus algorithm for the accurate faulty node tracking with faster convergence rate in a distributed sensor network» και παρουσιάζεται μια πρόταση των ερευνητών Rajkin Hossain και Muhidul Islam Khan https://jwcn-eurasipjournals.springeropen.com/articles/10.1186/s13638-016-0698-x [Προσπελάστηκε 22/08/20]

[63] Μια δημοσιευμένη έρευνα με τίτλο «A Distributed Algorithm for Constrained Multi-Robot Task Assignment for Group Tasks» που παρουσιάζει την νέα πρόταση των ερευνητών Lingzhi Luo,

Nilanjan Chakraboty και Katia Sycana https://www.ri.cmu.edu/pub_files/2012/12/journal_icra11_techreport.pdf [Προσπελάστηκε 22/08/20]

[64] Εκτενέστερη επεξήγηση της Chi-square συνάρτησης από το Κολλέγιο Hobart and William Smith https://math.hws.edu/javamath/ryan/ChiSquare.html [Προσπελάστηκε 25/08/20]

[65] Πληροφορίες για την Gaussian Εξίσωση από τη ψηφιακή εγκυκλοπαίδεια Wikipedia https://en.wikipedia.org/wiki/Gaussian_function [Προσπελάστηκε 25/08/20]

[66] Μερικά από τα δεδομένα που χρησιμοποιήθηκαν τα πήραμε από την ανοιχτή συλλογή δεδομένων του Kaggle. https://www.kaggle.com/ [Προσπελάστηκε 13/03/20]

[67] Απεικόνιση των επιπέδων του Διαδικτύου των Πραγμάτων https://www.researchgate.net/figure/oT-represented-by-a-seven-layer-architecture-Source_fig3_329520432 [Προσπελάστηκε 12/08/20]

[68] Πληροφορίες για το προσομοιωτή Cooja από το πανεπιστήμιο της Νότιας Καλιφόρνιας https://anrg.usc.edu/contiki/index.php/Cooja_Simulator [Προσπελάστηκε 18/08/20]