ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Επιλογή και διαχείριση ετερογενών πηγών μεγάλων δεδομένων σε πραγματικό χρόνο

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΑΝΑΓΝΩΣΤΟΠΟΥΛΟΥ ΠΑΝΑΓΙΩΤΗ

Επιβλέπων : Βαρβαρίγου Θεοδώρα
Καθηγήτρια Ε.Μ.Π.

Αθήνα, Απρίλης 2018

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Επιλογή και διαχείριση ετερογενών πηγών μεγάλων δεδομένων σε πραγματικό χρόνο

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΑΝΑΓΝΩΣΤΟΠΟΥΛΟΥ ΠΑΝΑΓΙΩΤΗ

Επιβλέπων : Βαρβαρίγου Θεοδώρα
Καθηγήτρια Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 30 Απριλίου 2018.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Θ. Βαρβαρίγου	Ε. Βαρβαρίγος	Σ. Παπαβασιλείου
Καθηγήτρια Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Απρίλης 2018

(Υπογραφή)

.....

ΠΑΝΑΓΙΩΤΗΣ ΑΝΑΓΝΩΣΤΟΠΟΥΛΟΣ

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

Ευχαριστίες

Καταρχάς, θα ήθελα να ευχαριστήσω την επιβλέπουσα καθηγήτρια Θεοδώρα Βαρβαρίγου που μου ανέθεσε αυτή την εργασία, δίνοντάς μου τη δυνατότητα να ασχοληθώ με ένα πολύ ενδιαφέρον αντικείμενο.

Επίσης, ευχαριστώ θερμά τον υποψήφιο Διδάκτορα του Ε.Μ.Π., Βρεττό Μουλό και τον Αχιλλέα Μαρινάκη, για το χρόνο που μου αφιέρωσαν, τη βοήθεια, τις οδηγίες και κατευθύνσεις που μου έδιναν σε όλα τα στάδια εκπόνησης της εργασίας.

Ένα μεγάλο ευχαριστώ στους συμφοιτητές και φίλους μου, την Ανθούσα, το Ζαχαρία και τον Τάσο, για αυτά τα πολύ όμορφα φοιτητικά χρόνια που μου χάρισαν.

Το μεγαλύτερο ευχαριστώ, όμως, οφείλω προπαντός στην οικογένειά μου, για την υπομονή και τη στήριξή τους κατά τη διάρκεια των σπουδών μου.

Παναγιώτης Θ. Αναγνωστόπουλος,

Αθήνα, Απρίλιος 2018

Περίληψη

Η ανάπτυξη εφαρμογών στο Internet of Things, αποτελεί έναν από τους ταχύτερα αναπτυσσόμενους τεχνολογικούς κλάδους. Ο γοργός αυτός ρυθμός της εξέλιξης του, ωστόσο, είναι και η αιτία για μία πληθώρα προβλημάτων.

Τα προβλήματα αυτά αυξάνονται με τη χρήση των Big Data, από τη στιγμή που οι εφαρμογές έρχονται αντιμέτωπες με μεγάλο όγκο δεδομένων, τα οποία θα πρέπει να διαχειριστούν με κατάλληλο τρόπο, ώστε να εκτελέσουν περαιτέρω ενέργειες. Το ζήτημα αυτό, γίνεται ακόμη πιο περίπλοκο, διότι βασικός μας στόχος είναι η διαχείριση αυτών των δεδομένων σε πραγματικό χρόνο, γεγονός που είναι βασικό χαρακτηριστικό του Internet of Things.

Στόχος της παρούσας διπλωματικής εργασίας είναι η δημιουργία μίας εφαρμογής, η οποία θα συμβάλλει στη διαχείριση και επιλογή κατάλληλης της πηγής δεδομένων για την διευκόλυνση του προγραμματισμού IoT με Big Data, καθώς και τη βέλτιστη διαχείριση τους σε πραγματικό χρόνο.

Λέξεις Κλειδιά: Internet of Things, Big Data

Abstract

Programming in Internet of Things sees a fast-growing development nowadays. However, this can be the cause of multiple problems. These problems are being increased with the use of Big Data, since applications are confronted with a large amount of data, which should be handled in an appropriate manner to perform further actions. This problem is becoming even more complex, because our primary goal is to manage these data in real time, a key feature of Internet of Things.

The aim of this diploma thesis is to create an application that will help manage and select the appropriate data source to facilitate IoT programming with Big Data, as well as optimize their management in real time.

Keywords: Internet of Things, Big Data

Πίνακας περιεχομένων

Πίνακας Εικόνων

1

Εισαγωγή

Το Internet of Things (IoT) αναπτύσσεται συνεχώς με όλο και περισσότερες συσκευές να παράγονται ετησίως. Για το λόγο αυτό, η δημιουργία εφαρμογών που λειτουργούν σε συσκευές IoT έχει γίνει ιδιαίτερα δημοφιλής τα τελευταία χρόνια, υπάρχουν, όμως, πολλά προβλήματα που ο προγραμματιστής καλείται να λύσει, αν θέλει να δημιουργήσει μία τέτοια εφαρμογή. Ένα από τα κυριότερα ζητήματα, είναι αυτό της διαχείρισης των δεδομένων σε πραγματικό χρόνο. Συγκεκριμένα, από τη στιγμή που ο όγκος των δεδομένων που συλλέγονται ανά πάσα στιγμή είναι πολύ μεγάλος (Big Data), το έργο του προγραμματιστή δυσκολεύει αρκετά, διότι η ταχύτητα της εφαρμογής αρχίζει να μειώνεται σημαντικά.

1.1 Αντικείμενο διπλωματικής

Το αντικείμενο της διπλωματικής είναι η δημιουργία ενός middleware, το οποίο θα βοηθάει τους προγραμματιστές που θέλουν να δημιουργήσουν εφαρμογές για συσκευές IoT, που θα ελαχιστοποιεί τον κώδικα που θα χρειάζεται να συντάξει μόνος του ο προγραμματιστής. Θα επικεντρωθούμε στον έλεγχο και στην διαχείριση ετερογενών πηγών δεδομένων, έτσι ώστε να διευκολύνουμε τη διαχείριση των Big Data σε πραγματικό χρόνο.

1.2 Οργάνωση κειμένου

Στα Κεφάλαια 2, 3 και 4 παρουσιάζεται το θεωρητικό υπόβαθρο που απαιτείται για την διπλωματική, Internet of Things, Big Data και Cloud Computing αντίστοιχα. Στο Κεφάλαιο 5 αναλύονται ορισμένες πλατφόρμες που χρησιμεύουν για την ανάπτυξη εφαρμογών για IoT συσκευές. Το Κεφάλαιο 6 συζητά τα προβλήματα που υπάρχουν στην δημιουργία εφαρμογών για IoT και στην λύση που προσφέρει η παρούσα εργασία. Στο Κεφάλαιο 7 αναλύονται οι τεχνολογίες που χρησιμοποιήθηκαν και στο κεφάλαιο 8 ο τρόπος υλοποίησης της διπλωματικής.

2

Internet of Things

2.1 Γενικά

Το Internet of Things (IoT) θα αλλάξει τα πάντα. Αυτό μπορεί να φαίνεται υπερβολικό αλλά ήδη το Internet έχει τεράστια επίδραση σε πολλούς τομείς, όπως εκπαίδευση, επικοινωνίες, επιστήμη, επιχειρήσεις και γενικά στην ανθρωπότητα. Προφανώς το Internet είναι μία από τις πιο σημαντικές δημιουργίες στην ιστορία της ανθρωπότητας. Το IoT είναι το επόμενο βήμα στην εξέλιξη του Internet, με μία τεράστια εξέλιξη στην ικανότητα του να μαζεύει δεδομένα, να τα αναλύει και να τα διαμοιράζει, τα οποία μπορούμε να τα χρησιμοποιήσουμε και να τα μετατρέψουμε σε πληροφορίες και γνώσεις.

Ήδη πολλά έργα IoT δημιουργούνται που φαίνεται ότι θα μειώσουν το μεγάλο χάσμα που υπάρχει μεταξύ των φτωχών και των πλούσιων, να διανέμουν καλύτερα τους παγκόσμιους πόρους και κυρίως σε αυτούς που τους χρειάζονται περισσότερο, και τελικά να μας βοηθήσουν να καταλάβουμε καλύτερα τον πλανήτη μας. Όμως υπάρχουν πολλά προβλήματα που πρέπει πρώτα να ξεπεραστούν τα οποία καθυστερούν την ανάπτυξη του IoT, περιλαμβανόμενος της μετάβασης από IPv4 σε IPv6, την ύπαρξη ενός κοινού σετ προτύπων, και την ανάπτυξη πηγών ενέργειας για να είναι εφικτό να τροφοδοτηθούν εκατομμύρια ή ακόμα και δισεκατομμύρια από αισθητήρες και διάφορες IoT συσκευές.

2.2 Τι είναι το Internet of Things

Το Internet of Things (IoT) αναφέρεται στην διασύνδεση καθημερινών συσκευών που έχουν την ικανότητα να αλληλοεπιδρούν και να αναγνωριστούν από άλλες συσκευές. Το IoT είναι σημαντικό γιατί τα αντικείμενα τα οποία μπορούν να εκπροσωπούν τον εαυτό τους ψηφιακά γίνονται κάτι παραπάνω από το ίδιο το αντικείμενο. Πλέον τα αντικείμενα δεν σχετίζονται μόνο με τον χρήστη τους αλλά είναι συνδεδεμένα με άλλα περιβάλλοντα αντικείμενα και με βάση

δεδομένων. Όταν πολλά αντικείμενα ενεργούν από κοινού, είναι γνωστά ως περιβαλλοντική νοημοσύνη.

Το Internet of Things είναι μία αρκετά περίπλοκη έννοια, ώστε να καθοριστεί ακριβώς. Υπάρχουν πολλές διαφορετικές ομάδες που έχουν καθορίσει τον όρο, αν και η αρχική χρήση έχει αποδοθεί στον Kevin Ashton, έναν ειδικό στην ψηφιακή καινοτομία. Κάθε ορισμός μοιράζεται την ιδέα ότι η πρώτη έκδοση του Internet ήταν για δεδομένα που δημιουργούνται από ανθρώπους, ενώ η επόμενη έκδοση είναι για δεδομένα που δημιουργούνται από αντικείμενα. Πολλοί άνθρωποι θεωρούν ότι η σύνδεση συσκευών αναφέρεται σε υπολογιστές, tablets και smartphones. Όμως το IoT περιγράφει έναν κόσμο όπου σχεδόν τα πάντα θα μπορούν να είναι συνδεδεμένα και να επικοινωνούν μεταξύ τους με έναν έξυπνο τρόπο. Με άλλα λόγια, με το IoT ο φυσικός κόσμος γίνεται ένα τεράστιο σύστημα πληροφοριών.

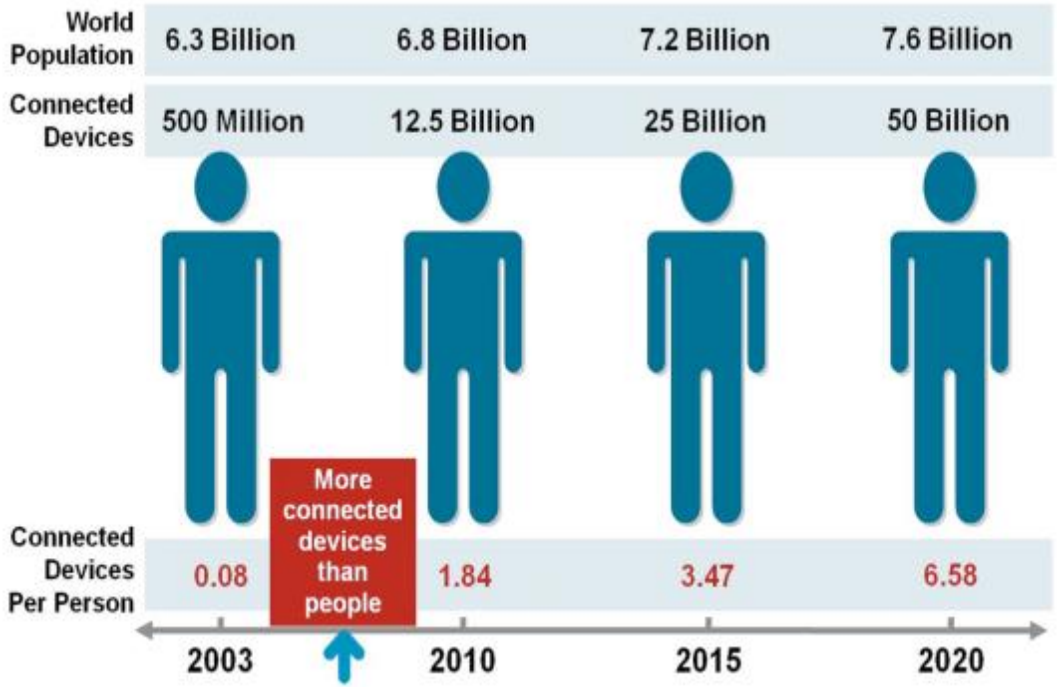
Εικόνα 1 Τα πάντα συνδέονται μέσω του Internet of Things (Sabin, 2017)

Ένας ορισμός που χρησιμοποιείται είναι: "Το Internet of Things είναι η διασυνδεδεμένη σφαίρα φυσικών συσκευών με το Internet και άλλων δικτύων, μέσω μοναδικών αναγνωρίσιμων IP διευθύνσεων, όπου δεδομένα συλλέγονται και μεταφέρονται μέσω ενσωματωμένων αισθητήρων, ηλεκτρονικών και λογισμικού".

Το IoT είναι ένα επιπλέον στρώμα πληροφοριών, αλληλεπίδρασης, συναλλαγής και δράσης που προστίθεται στο Internet εξ αιτίας των συσκευών, που είναι εξοπλισμένες με δυνατότητες ανίχνευσης δεδομένων, ανάλυσης και επικοινωνίας χρησιμοποιώντας τεχνολογίες του Internet. Το IoT γεφυρώνει περισσότερο την ψηφιακή και την φυσική πραγματικότητα και συμβάλλει στην αυτοματοποίηση, η οποία βασίζεται στις πληροφορίες και βελτιώσεις στο επίπεδο των επιχειρήσεων, της κοινωνίας και της ζωής των ανθρώπων.

Ένας άλλος ορισμός σύμφωνα με το Cisco Internet Business Solutions Group: IoT είναι απλά το σημείο στον χρόνο, όπου περισσότερα "αντικείμενα" είναι συνδεδεμένα στο Internet από τους ανθρώπους.

Το 2003 υπήρχαν περίπου 6,3 δισεκατομμύρια άνθρωποι στον πλανήτη και 500 εκατομμύρια συσκευές συνδεδεμένες στο Internet. Προφανώς φαίνεται ότι αν διαιρέσουμε τον αριθμό των συσκευών με τον αριθμό των ανθρώπων, θα δούμε ότι υπάρχουν λιγότερες από μία για κάθε άνθρωπο. Άρα σύμφωνα με αυτόν τον ορισμό το IoT δεν υπήρχε το 2003. Όμως η τεράστια ανάπτυξη των smartphones και των υπολογιστών εκτόξευσε τον αριθμό των συσκευών που είναι συνδεδεμένες στο Internet στα 12,5 δισεκατομμύριά το 2010, ενώ ο αριθμός των ανθρώπων έφτασε μόνο στα 6,8 δισεκατομμύρια, αυτό σημαίνει ότι υπήρχαν περισσότερες από μία συσκευή μέσο όρο ανά άνθρωπο.



Εικόνα 2 Αριθμός ανθρώπων και συνδεδεμένων συσκευών (CISCO IBSG, 2011)

Ο αριθμός των συσκευών ανά άνθρωπο μπορεί να φαίνεται χαμηλός, αλλά αυτό συμβαίνει επειδή υπολογίζεται όλος ο πληθυσμός της γης και ένα μεγάλο μέρος του δεν έχει πρόσβαση σε Internet.

2.3 Γιατί είναι σημαντικό το IoT

Αρχικά πρέπει να καταλάβουμε την διαφορά ανάμεσα στο web (World Wide Web) και στο Internet, όροι οι οποίοι χρησιμοποιούνται συχνά σαν να είναι όμοιοι. Το Internet είναι το φυσικό στρώμα ή δίκτυο φτιαγμένο από switches, routers και άλλο εξοπλισμό, και η κύρια λειτουργία του είναι να μεταφέρει πληροφορίες από ένα σημείο σε ένα άλλο. Το web είναι μία

εφαρμογή που λειτουργεί πάνω στο Internet, και ο κύριος ρόλος του είναι να παρέχει ένα interface ώστε να γίνεται ο διαμοιρασμός πληροφοριών μέσω του Internet ευκολότερος.

Το IoT είναι η πρώτη εξέλιξη του Internet. Το Internet πολλά χρόνια αναπτύσσεται και βελτιώνεται, αλλά δεν έχει εξελιχθεί πολύ. Ουσιαστικά κάνει το ίδιο πράμα που έκανε και όταν δημιουργήθηκε. Για αυτό το λόγο το IoT είναι πολύ σημαντικό, ως η πρώτη σημαντική εξέλιξη του Internet. Μία εξέλιξη η οποία θα οδηγήσει σε επαναστατικές εφαρμογές, οι οποίες έχουν τη δυνατότητα να βελτιώσουν δραστικά τον τρόπο που ζουν οι άνθρωποι, μαθαίνουν, δουλεύουν και διασκεδάζουν. Χάρις του IoT ήδη υπάρχουν διάφοροι αισθητήρες που συνδέονται στο Internet και οι πληροφορίες που συλλέγουν είναι διαθέσιμες αμέσως.

Επιπλέον το Internet μέσω του IoT, επεκτείνεται σε μέρη όπου πριν δεν ήταν εφικτό. Για παράδειγμα, ασθενείς καταπίνουν συσκευές συνδεδεμένος στο Internet για να βοηθήσουν τους γιατρούς να κάνουν την διάγνωση και να καθορίσουν τον λόγο πολλών ασθενειών. Ακόμα πολλοί μικροί αισθητήρες μπορούν να τοποθετηθούν σε φυτά, ζώα και γεωλογικές τοποθεσίες και να είναι συνδεδεμένοι στο Internet.

Η εξέλιξη των ανθρώπων συμβαίνει επειδή επικοινωνούμε. Για παράδειγμα όταν η φωτιά ανακαλύφθηκε δεν χρειαζόταν να ξανά ανακαλυφθεί, ή η ανακάλυψη της ελικοειδή δομή του DNA. Η αρχή του διαμοιρασμού της πληροφορίας και της εύρεσης ανακαλύψεων μπορεί να κατανοηθεί καλύτερα από την παρακάτω εικόνα:

Εικόνα 3 Η μεταμόρφωση των δεδομένων (CISCO IBSG, 2011)

Τα δεδομένα είναι ακατέργαστες ύλες τα οποία μπορούν να επεξεργαστούν και να γίνουν πληροφορίες. Μεμονωμένα δεδομένα από μόνα τους δεν έχουν κάποια ιδιαίτερη χρησιμότητα, αλλά πολλά δεδομένα μπορούν να διακρίνουν τάσεις και πρότυπα. Αυτά μετά και μαζί με άλλες πηγές πληροφορίας συνδυάζονται σε γνώσεις, το οποίο είναι απλά πληροφορίες, τις οποίες κάποιος αντιλαμβάνεται. Τέλος με τις γνώσεις και εμπειρία γεννιέται η σοφία, η οποία είναι διαχρονική και ξεκινάει από την συλλογή δεδομένων. Βλέπουμε ότι όσο περισσότερα δεδομένα δημιουργούνται, τόσο περισσότερη γνώση και σοφία μπορούν να αποκτήσουν οι άνθρωποι. Το IoT είναι σημαντικό γιατί αυξάνει δραματικά την ποσότητα των δεδομένων που είναι διαθέσιμα για να επεξεργαστούμε.

Το IoT είναι κρίσιμο για την πρόοδο της ανθρωπότητας. Όσο αυξάνεται ο πληθυσμός της γης, γίνεται ακόμα πιο σημαντικό οι άνθρωπο να ελέγχουν την γη και τους πόρους της. Επιπλέον οι άνθρωποι θέλουν να ζουν υγιής και άνετα. Έτσι συνδυάζοντας την ικανότητα του IoT να συλλέγει, να επεξεργάζεται και να διαμοιράζεται δεδομένα σε τεράστιες ποσότητες, οι άνθρωποι θα αποκτήσουν την γνώση και την σοφία να ευδοκιμήσουν τα επόμενα χρόνια και δεκαετίες.

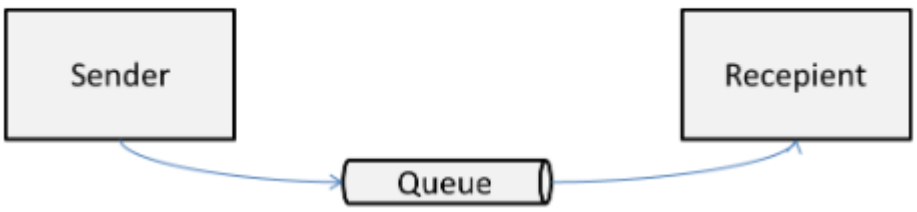
2.4 Τεχνολογίες και πρωτόκολλά επικοινωνίας IoT

2.4.1 Κύριοι τρόποι επικοινωνίας

Υπάρχουν διάφοροι τρόποι επικοινωνίας μεταξύ συστημάτων.

Client-Server: Βασίζεται σε μία βασική αλληλεπίδραση μεταξύ clients και server. Ο client στέλνει ένα αίτημα σε ένα server και περιμένει (non-blocking) μία απάντηση από το server. Ο server μπορεί να είναι stateful ή Stateless. Η διαφορά ανάμεσα τους είναι αν το σύστημα κρατά τις προηγούμενες αλληλεπιδράσεις με των client και έχει μία πεπερασμένη μηχανή καταστάσεων συσχετισμένη με τις αλληλεπιδράσεις.

Message Passing: Στο message passing τα γεγονότα μπαίνουν σε μία ουρά για να επεξεργαστούν μαζί με τα σχετικά δεδομένα.



Εικόνα 4 Message Passing (IEEE, 2015)

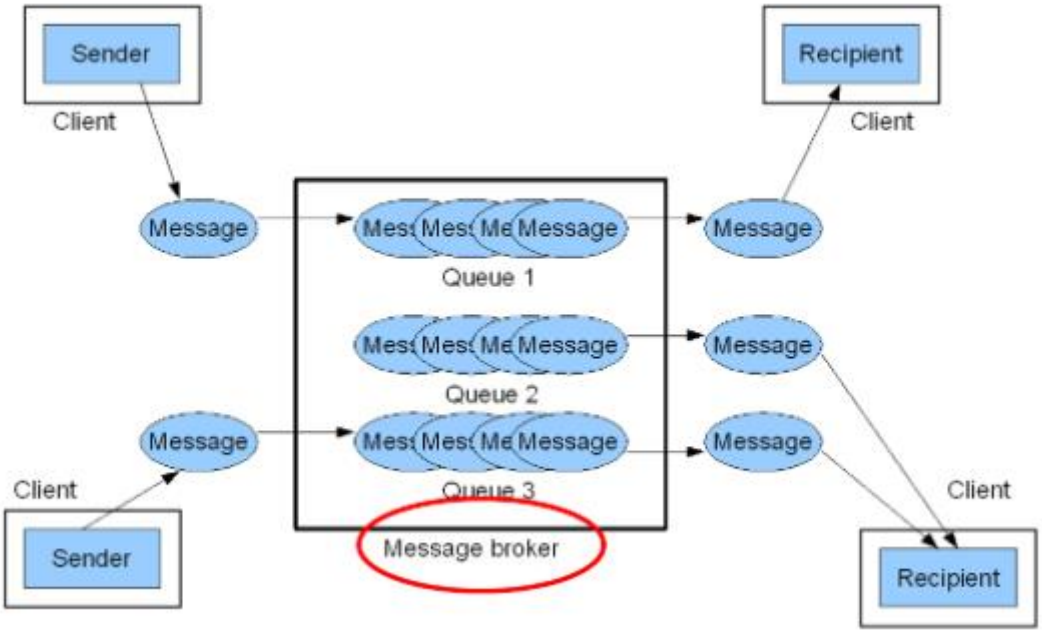
Όταν τα δεδομένα βγουν από την ουρά μπορούν να επεξεργαστούν. Πρέπει να υπάρχει μία σαφή μεταχείριση της επικοινωνίας και διαχείρισης δεδομένων. Η αποσυναρμολόγηση και η συναρμολόγηση των μηνυμάτων μπορεί να είναι πολύπλοκο.

Shared Memory: Ένα από τα πλεονεκτήματα αυτής της επικοινωνίας είναι ότι οι λεπτομέρειες μεταξύ προγραμμάτων και "shared memory" είναι κρυφά από τον προγραμματιστή, ελευθερώνοντας τους από το πρόβλημα να ασχολούνται με την επικοινωνία και με τη διαχείριση δεδομένων. Ο προγραμματιστής μπορεί να συγκεντρωθεί στον παραλληλισμό και στα προβλήματα. Όμως τα προγράμματα πρέπει να βρουν τρόπο να συγχρονιστούν με νέα δεδομένα και γεγονότα, επιπλέον η διαχείριση δεδομένων είναι βελτιστοποιημένη για ένα system optimization, και για αυτό συγκεκριμένες εφαρμογές που χρειάζονται άλλη βελτιστοποίηση θα δυσκολευτούν να εφαρμόσουν συγκεκριμένες τακτικές. Αυτό είναι ένα παράδειγμα το οποίο βοηθάει στον προγραμματισμό, αλλά δυσκολεύει μερικές εφαρμογές που θέλουν να κάνουν διαφορετικά πράγματα.

Task and Channels: Μπορεί να θεωρηθεί ως μία παραλλαγή του memory passing. Αυτός ο τρόπος επιβάλλει στον προγραμματιστή να ορίσει καλά τις έννοιές της επικοινωνίας ενός παράλληλου και διανεμημένου προγράμματος. Αυτό μπορεί να προσφέρει πλεονεκτήματα σε ένα μεγάλο διανεμημένο πρόγραμμα, αλλά επίσης έχει μεγαλύτερο overhead για τον έλεγχο ατομικών channels και για την δημιουργία σχέσεων μεταξύ channels και tasks.

Data Parallelism: Αυτός ο τρόπος μπορεί να οδηγήσει σε σημαντικά πλεονεκτήματα όταν οι ίδιες εντολές μπορούν να εφαρμοστούν σε διαφορετικά δεδομένα, αλλά είναι δύσκολη η εφαρμογή του. Το μεγάλο μειονέκτημα αυτού του τρόπου είναι ότι είναι περιορισμένο σε ένα μικρό σετ εφαρμογών.

Σε μεγάλα συστήματα Message Passing, οι senders και οι recipients μοιράζονται μία κοινή υποδομή φτιαγμένη από ουρές, το οποίο είναι οργανωμένο σαν message broker για να δέχεται και να στέλνει μηνύματα (ασύγχρονα) ανάμεσα σε πηγές και δέκτες.



Εικόνα 5 Message Passing σύστημα (IEEE, 2015)

Υπάρχουν τουλάχιστον τρία σημαντικά χαρακτηριστικά:

Routing: Το σύστημα μπορεί να δρομολογήσει τα μηνύματα ακόμα και αν ο sender δεν έχει διευκρινίσει το full path για τον προορισμό.

Conversion: Οι συναρτήσεις Broker μπορούν να κάνουν συμβατούς διάφορους τύπους μηνυμάτων και έτσι να βοηθούν την επικοινωνία μεταξύ διαφορετικών συστημάτων μηνυμάτων.

Το μοντέλο μπορεί να είναι σύγχρονο ή ασύγχρονο. Στο σύγχρονο ο sender περιμένει να λάβει ο client το μήνυμα, ενώ στο ασύγχρονο, όταν το μήνυμα σταλθεί ο sender μπορεί να συνεχίσει. Η επιλογή μεταξύ σύγχρονου και ασύγχρονου εξαρτάται από τις απαιτήσεις της εφαρμογής. Οι ουρές που αποθηκεύουν τα μηνύματα για να επεξεργαστούν αργότερα είναι μία περίπτωση ασύγχρονου μηχανισμού.

2.4.2 Πρωτόκολλα web service

Τα πρωτόκολλα web service κατηγοριοποιούνται σε δύο ομάδες, τα Simple Object Access Protocol (SOAP) και τα REpresentational State Transfer (REST). Τα IoT ευνοούν την αρχιτεκτονική REST λόγο της απλότητας του και την άνεση σε περιορισμένο περιβάλλον.

SOAP

Το SOAP βασίζεται αποκλειστικά σε XML για να παρέχει υπηρεσίες μηνυμάτων. Το SOAP είναι σχεδιασμένο να είναι επεκτάσιμο, αλλά χρησιμοποιεί μόνο τα κομμάτια που χρειάζεται για κάποια συγκεκριμένη εργασία. Τα XML που χρειάζονται για το SOAP μπορεί να γίνουν

ιδιαιτέρα περίπλοκα. Σε πολλές γλώσσες ο χρήστης πρέπει να τα φτιάξει μόνος του, το οποίο είναι προβληματικό, αλλά υπάρχουν και διάφορες γλώσσες που προσφέρουν συντομεύσεις.

Επίσης υπάρχει και το Web Service Description Language (WSDL). Αυτό είναι ένα αρχείο συσχετισμένο με το SOAP, το οποίο παρέχει έναν ορισμό του πως λειτουργεί η web υπηρεσία, έτσι ώστε όταν δημιουργεί κάποιος μία αναφορά σε αυτό, το IDE (integrated development environment) μπορεί να αυτοματοποιήσει την διαδικασία. Συνεπώς η δυσκολία του SOAP εξαρτάται σε μεγάλο βαθμό από την γλώσσα.

Ένα σημαντικό χαρακτηριστικό του SOAP είναι ενσωματωμένη διαχείριση error. Αν υπάρχει κάποιο πρόβλημα με το αίτημα, η απάντηση περιέχει πληροφορίες του error ώστε να διορθωθεί το λάθος. Ένα άλλο σημαντικό χαρακτηριστικό είναι ότι δεν απαιτείται αναγκαστικά χρήση του HyperText Transfer Protocol (HTTP).

REST

Το REST είναι ένας τύπος αρχιτεκτονικής λογισμικού για διανεμημένα συστήματα hypermedia. Οι αρχιτεκτονικές REST αποτελούνται συνήθως από clients και servers. Οι clients κάνουν αιτήματα σε servers, τα servers επεξεργάζονται τα αιτήματα και επιστρέφουν την κατάλληλη απάντηση.

Πολλοί προγραμματιστές βρίσκουν το SOAP δύσκολο στην χρήση. Για παράδειγμα, για τη χρήση SOAP με JavaScript απαιτείται πολύς κώδικας για αρκετά απλές εργασίες επειδή κάποιος πρέπει να δημιουργήσει ένα XML κάθε φορά. Το REST είναι μία lighter-weight εναλλακτική. Αντί να χρησιμοποιεί XML, βασίζεται στην χρήση απλών URL. Σε μερικές περιπτώσεις μπορεί να χρειάζεται κάποιος να στείλει παραπάνω πληροφορίες, αλλά συνήθως οι περισσότερες υπηρεσίες web που χρησιμοποιούν REST παίρνουν όλες τις πληροφορίες από το URL. Το REST μπορεί να χρησιμοποιήσει τέσσερεις HTTP διαδικασίες (GET, PUT, POST, DELETE). Αντίθετα από το SOAP το REST δεν επιστρέφει αναγκαστικά XML. Μπορεί να επιστρέψει τα δεδομένα σε Command Separated Value (CSV), JavaScript Object Notation (JSON) ή Really Simple Syndication (RSS). Το θέμα είναι ότι μπορεί κάποιος να αποκτήσει τα δεδομένα σε μία μορφή που είναι εύκολη η επεξεργασία από την γλώσσα της εφαρμογής του.

2.4.3 Πρωτόκολλα σχετικά με IoT

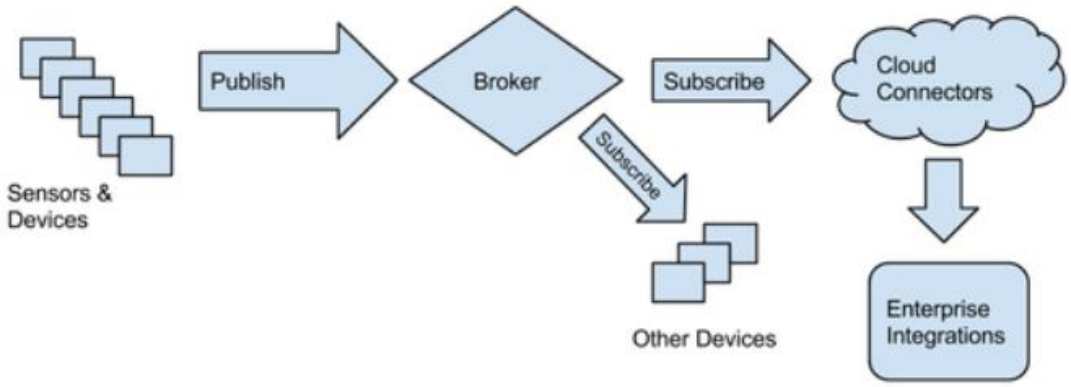
Η αλληλεπίδραση μεταξύ δύο endpoints IoT χρησιμοποιεί την έννοια M2M επικοινωνίας (machine to machine). Η επικοινωνία M2M μπορεί να βασίζεται σε γεγονότα ή να συμβαίνει σε τακτά χρονικά διαστήματα.

Για τις συσκευές IoT, είναι δύσκολο να εφαρμοστεί το πρωτόκολλο HTTP για διάφορους λόγους. Κυρίως επειδή το IoT αποτελείται από αφανείς συσκευές που απαιτούν πολύ λίγη

αλληλεπίδραση. Οι συσκευές IoT καταναλώνουν λίγη ενέργεια και συνήθως έχουν κακή σύνδεση Internet. Το HTTP είναι πολύ βαρύ για αυτού του είδους τις συσκευές.

Η κίνηση στο δίκτυο στο IoT αποτελείται από δύο κατηγορίες: telemetry και telecommand. Telemetry είναι η πράξη συλλογής telemetrics, ή αποστολής δεδομένων σε μεγάλες αποστάσεις. Συνήθως το telemetry περιλαμβάνει πολλούς "χαζούς" αισθητήρες, που στέλνουν δεδομένα σε ένα "έξυπνο" κεντρικό σημείο. Telecommand είναι η πράξη αποστολής εντολών σε ένα δίκτυο.

Τα περισσότερα πρωτόκολλα telemetry ακολουθούν την αρχιτεκτονική publish/subscribe. Αισθητήρες συνδέονται σε ένα broker και δημοσιεύουν περιοδικά τις μετρήσεις τους σε ένα topic. Ένα κεντρικό σύμπλεγμα από servers θα εγγραφεί στο topic και θα επεξεργάζεται τις μετρήσεις των αισθητήρων σε πραγματικό χρόνο.



Εικόνα 6 Αρχιτεκτονική συστημάτων telemetry (IEEE, 2015)

Ένα πολύ γνωστό telemetry πρωτόκολλο είναι το MQTT.

MQTT

Το MQTT είναι ένα publish/subscribe πρωτόκολλο που είναι light weight για χρήση πάνω από το TCP/IP πρωτόκολλο. Είναι σχεδιασμένο για συνδέσεις με απομακρυσμένες τοποθεσίες όπου χρειάζεται ένα ελαφρύ πρωτόκολλο και/ή η σύνδεση είναι περιορισμένη. Ο τρόπος publish/subscribe χρειάζεται ένα message broker. Ο broker ευθύνεται για την διανομή μηνυμάτων σε ενδιαφερόμενους clients του topic του μηνύματος.

Το MQTT είναι πολύ απλό και ελαφρύ πρωτόκολλο. Αυτά τα χαρακτηριστικά το κάνουν ιδανικό για χρήση σε περιορισμένα περιβάλλοντα όπου το εύρος ζώνης του δικτύου είναι χαμηλό και με συσκευές που έχουν περιορισμένες ικανότητες και μνήμη. Το MQTT ελαχιστοποιεί τις απαιτήσεις στο εύρος ζώνης του δικτύου και στους πόρους συσκευών ενώ προσπαθεί να εξασφαλίσει αξιοπιστία και παράδοση. Αυτό κάνει το MQTT πρωτόκολλο αρκετά κατάλληλο για την σύνδεση M2M, το οποίο είναι πολύ κρίσιμο κομμάτι του IoT.

CoAP

Το CoAP είναι ένα από τα πρωτόκολλα telecommand. Το telecommand είναι αντίστοιχο με το telemetry. Εκεί που το telemetry στέλνει δεδομένα από έναν αισθητήρα σε ένα κεντρικό σημείο, το telecommand στέλνει εντολές σε μία απομακρυσμένη συσκευή. Μαζί οι δύο αυτές κατηγορίες, αντιστοιχούν σε όλα τα πρωτόκολλα του IoT σε μία μορφή ή σε άλλη.

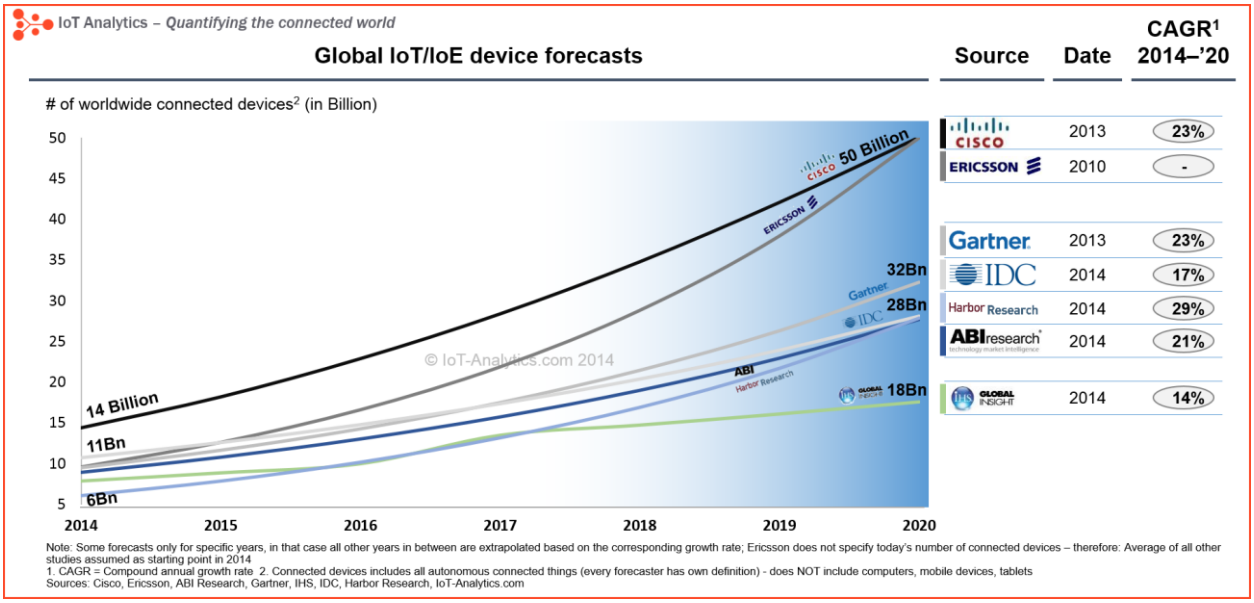
Εννοιολογικά το CoAP είναι κοντά στο HTTP εκτός του ότι το CoAP έχει σχεδιαστεί για συσκευές με περιορισμένους πόρους, όπως αισθητήρες και μικροελεγκτές, που έχουν περιορισμένη μνήμη και ικανότητες ζώνης εύρους. Είναι σχεδιασμένο να υποστηρίζει αρχιτεκτονικές RESTful, όποτε υποστηρίζει τις τέσσερεις κύριες μεθόδους και αναγνωρίζει πόρους μέσω URL. Όμως αντίθετα με το HTTP το CoAP είναι μία συμπαγή δυαδική μορφή. Μπορεί εννοιολογικά να μοιάζει πολύ με το HTTP, αλλά στην πραγματικότητα το πρωτόκολλο σύνδεσης είναι πολύ διαφορετικό. Το CoAP είναι σχεδιασμένο για περιορισμένες πλατφόρμες όπου κάθε bit και κάθε κύκλος επεξεργαστή αξίζει.

Η κύρια διαφορά μεταξύ του CoAP και του HTTP είναι ότι το CoAP αντί να έχει πολλούς αδύναμους clients και μερικούς δυνατούς servers, έχει πολλούς αδύναμους servers και μερικούς δυνατούς clients. Αυτό συμβαίνει γιατί στο HTTP οι servers ελέγχουν τους πόρους, ενώ στο CoAP οι αισθητήρες και οι ενεργοποιητές ελέγχουν τους πόρους. Για έναν ενεργοποιητή ο πόρος είναι για παράδειγμα η ικανότητα να κινήσει κάτι, όπως πόρτα.

2.5 Μέλλον του IoT

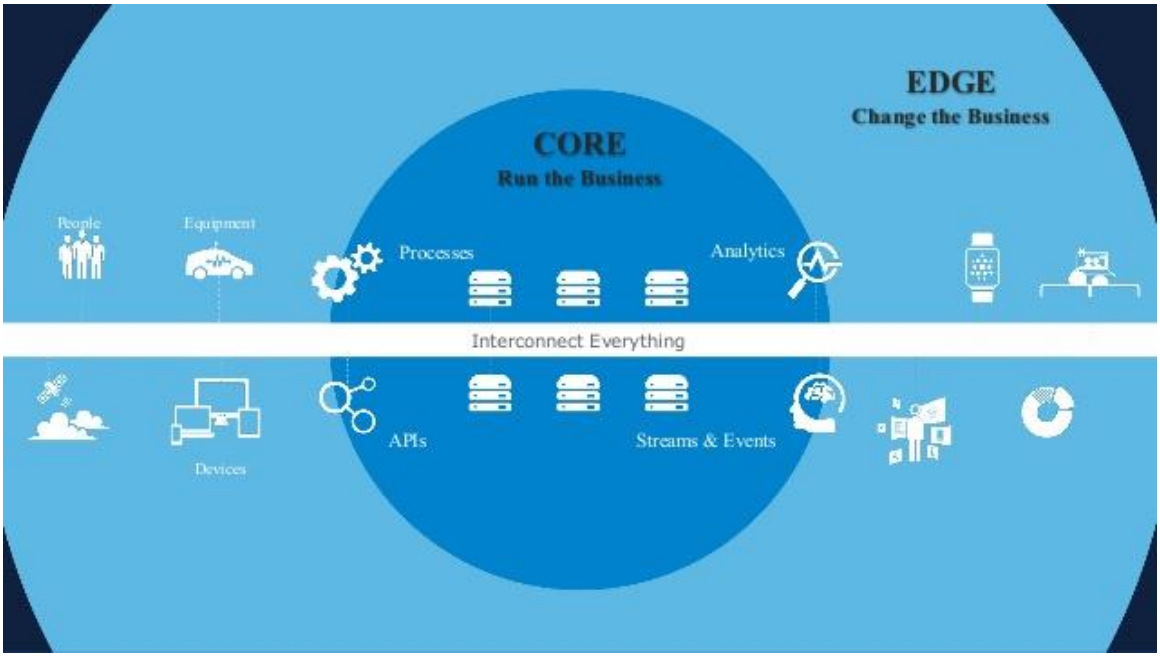
Το IoT είναι σε ένα σημείο όπου ξεχωριστά δίκτυα και πολλοί αισθητήρες πρέπει να ενωθούν κάτω από ένα κοινό σετ από πρότυπα. Αυτό απαιτεί επιχειρήσεις, κυβερνήσεις και διάφορους οργανισμούς προτύπων να συνεργαστούν για τον κοινό αυτό στόχο.

Η ανάπτυξη του Internet of Things συμβαίνει σε διαφορετικές ταχύτητες. Για παράδειγμα, οι επενδύσεις στην βιομηχανία κατασκευής σε IoT είναι κατά πολύ μεγαλύτερες από οποιαδήποτε άλλη βιομηχανία και από το χώρο καταναλωτών. Όμως παρά τις διάφορες προκλήσεις, τις διαφορετικές ταχύτητες και την ταχύτατη εξέλιξη την οποία θα δούμε κυρίως αρχές της επόμενης δεκαετίας, το IoT είναι ήδη εδώ. Υπάρχουν χιλιάδες περιπτώσεις χρήσης IoT σε επιχειρήσεις και βιομηχανίες αλλά και στο χώρο των καταναλωτών και θα συνεχίσουν να αναπτύσσονται ραγδαία. Σύμφωνα με τις περισσότερες προβλέψεις από αναλυτές το 2020 θα υπάρχουν περισσότερες από είκοσι δισεκατομμύρια συσκευές χωρίς να συμπεριλαμβάνονται τα smartphones, tablets και υπολογιστές.



Εικόνα 7 Προβλέψεις για τον αριθμό των IoT συσκευών ως το 2020 (Knud, 2014)

Μία πολύ σημαντική απαίτηση για την επιτυχία των IoT είναι το integration, γιατί χωρίς integration δεν υπάρχει Internet of Things. Γενικότερα, βλέπουμε ότι δημιουργείται ένας πυρήνας (που αποτελείται από τα τοπικά hardware και τα δημόσια clouds) που είναι integrated και εκτελεί τα κρίσιμα συστήματα, υπηρεσίες cloud και άλλα. Τώρα όμως υπάρχει ακόμα και το edge για αυτό τώρα μιλάμε για το IoT (αμάξια, smartphones, smartwatch, και οποιαδήποτε συσκευή που επικοινωνεί με κάποια άλλη). Για αυτό πρέπει να συνδέσουμε τα πάντα μεταξύ τους ώστε να διευκολύνεται η ανάπτυξη και η χρήση των IoT.



Εικόνα 8 Πυρήνας και Edge (Wähner, 2017)

Ακόμα για να γίνει το IoT αποδεκτό στον γενικό πληθυσμό, δεν πρέπει να αντιπροσωπεύει τις εξελίξεις της τεχνολογίας για χάρη της τεχνολογίας, αλλά πρέπει να επιδείξει ότι έχει αξία για

την ζωή των ανθρώπων. Συμπερασματικά το IoT αντιπροσωπεύει το επόμενο βήμα του Internet. Βάση του ότι οι άνθρωποι εξελίσσονται μετατρέποντας δεδομένα σε πληροφορίες, και τελικά σοφία, το IoT έχει την δυνατότητα να αλλάξει τον κόσμο όπως τον ξέρουμε σήμερα, για το καλύτερο.

3

Big Data

3.1 Ορισμός Big Data

Big Data είναι ένας όρος ο οποίος περιγράφει τον μεγάλο όγκο δεδομένων, δομημένα και μη, που πλημυρίζουν μία επιχείρηση σε καθημερινή βάση. Όμως, δεν είναι η ποσότητα των δεδομένων αυτό που έχει σημασία, αλλά το τι κάνουν οι οργανισμοί με αυτά. Αυτά τα δεδομένα μπορούν να αναλυθούν για πληροφορίες που θα οδηγήσουν σε καλύτερες αποφάσεις και στρατηγικές επιχειρηματικές κινήσεις.

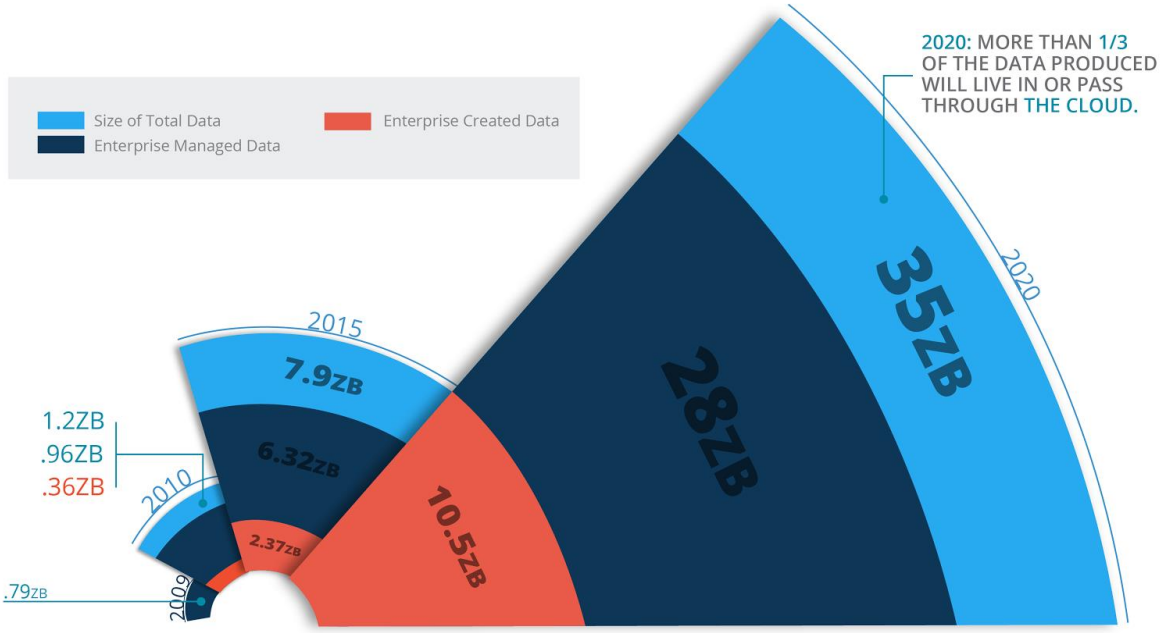
Τα Big Data υπάρχουν παντού και εμπλέκουν τους πάντες. Οι απλοί χρήστες του διαδικτύου παράγουν καθημερινά τεράστιο όγκο δεδομένων με τις καθημερινές δραστηριότητες τους. Οι επιχειρήσεις αποθηκεύουν δεδομένα που χρειάζονται, όπως τα στοιχεία των χρηστών και τις προτιμήσεις τους, ώστε να μπορούν να αναλύσουν καλύτερα τις ανάγκες τους. Όπως βλέπουμε, προέκυψε η ανάγκη ανάπτυξης νέων τεχνολογιών ώστε να είναι εφικτή η διαχείριση αυτών των δεδομένων, το οποίο θα ήταν αδύνατον με τις προηγούμενες τεχνολογίες.

Σύμφωνα με την Wikipedia: Big Data είναι ένας όρος για σετ δεδομένων τόσο μεγάλα και πολύπλοκα, όπου κλασικά λογισμικά επεξεργασίας δεδομένων είναι ανεπαρκή για να τα διαχειριστούν.

3.2 Χαρακτηριστικά των Big Data

Ο όρος Big Data είναι σχετικά καινούριος, όμως εδώ και πολλά χρόνια υπάρχει η πράξη της συλλογής και της αποθήκευση δεδομένων για ενδεχόμενη ανάλυση. Ο όρος έγινε πιο γνωστός αρχές του 2000 όταν ο αναλυτής βιομηχανίας Doug Laney έθεσε τον πλέον ευρέως γνωστό ορισμό του Big Data ως τα τρία Vs:

- Όγκος (Volume): Ο όγκος των δεδομένων είναι η ποσότητα των δεδομένων που έχουμε. Ό,τι στο παρελθόν μετριόταν σε Gigabytes τώρα μετριέται σε Zettabytes ή ακόμα και Yottabytes. Το IoT δημιουργεί μία εκθετική ανάπτυξη στην ποσότητα των δεδομένων.



Εικόνα 9 Ο ρυθμός αύξησης των Big Data (DeVan, 2016)

- Ταχύτητα (Velocity): Η ταχύτητα με την οποία τα δεδομένα είναι διαθέσιμα. Τα δεδομένα ρέουν με πρωτοφανή ταχύτητα και πρέπει να τα αντιμετωπίζουμε έγκαιρα. Οι ετικέτες RFID, οι αισθητήρες και τα έξυπνα συστήματα οδηγούν στην ανάγκη να αντιμετωπίσουμε τεράστια ποσότητα δεδομένων.
- Ποικιλία (Variety): Η ποικιλία είναι μία από τις μεγαλύτερες προκλήσεις του big data. Τα δεδομένα μπορεί να είναι αδόμητα, ή μπορεί να περιέχουν πολλούς διαφορετικούς τύπους από XML ή video ή SMS. Η οργάνωση των δεδομένων με έναν σημαντικό τρόπο δεν είναι απλό, ειδικά όταν τα δεδομένα αλλάζουν πολύ γρήγορα.

Σήμερα όμως αυτά τα τρία Vs δεν είναι αρκετά ώστε να περιγράψουν τα Big Data και για αυτό έχουν προστεθεί ακόμα τέσσερα Vs σε μερικούς ορισμούς:

- Μεταβλητότητα (Variability): Η μεταβλητότητα αφορά τα δεδομένα των οποίων η σημασία αλλάζει. Αν η σημασία αλλάζει συνεχώς μπορεί να υπάρχει σημαντική επίπτωση στην ομογενοποίηση των δεδομένων.
- Αξιοπιστία (Veracity): Veracity είναι η ορθότητα και η ακρίβεια των δεδομένων. Τα δεδομένα είναι σχεδόν άχρηστα αν δεν είναι ακριβή. Τα big data εμπεριέχουν ένα

μεγάλο βαθμό θορύβου με αποτέλεσμα να χρειάζονται μεγάλη προεργασία ώστε να παράγουμε ένα αξιόπιστο και ακριβές σύνολο δεδομένων πριν την ανάλυση τους.

- Οπτικοποίηση (Visualization): Η οπτικοποίηση είναι πολύ σημαντική στον σημερινό κόσμο. Η χρήση διαγραμμάτων και γραφικών για την παρουσίαση μεγάλης ποσότητας πολύπλοκων δεδομένων είναι πιο αποτελεσματικό από τα υπολογιστικά φύλλα και τις αναφορές γεμάτα με αριθμούς και φόρμουλες.
- Αξία (Value): Η αξία των δεδομένων είναι πολύ σημαντική τόσο για τις επιχειρήσεις όσο και για την κοινωνία.

4

Cloud Computing

4.1 Ορισμός

Σε απλούς όρους, cloud computing σημαίνει αποθήκευση και πρόσβαση σε δεδομένα και προγράμματα στο Internet αντί στον σκληρό δίσκο του υπολογιστή. Το cloud είναι απλά μία μεταφορά για το Internet. Ο όρος προέρχεται από τα παλιά διαγράμματα ροών και παρουσιάσεις όπου το Internet παρουσιαζόταν σαν ένα σύννεφο, όπου δέχεται συνδέσεις και μεταφέρει δεδομένα.

Ένας ορισμός σύμφωνα με το NIST (National Institute of Standards and Technologies) είναι: Cloud computing είναι ένα μοντέλο που επιτρέπει άνετη, on-demand πρόσβαση δικτύου σε ένα κοινό σετ από διαμορφωμένους υπολογιστικούς πόρους, οι οποίοι μπορούν να χορηγηθούν και να απελευθερωθούν με ελάχιστη προσπάθεια ή αλληλεπίδραση με τον πάροχο υπηρεσιών.

4.2 Γενικά

4.2.1 Τι διαφορετικό έχει το cloud computing

It's managed: Ένα από τα πιο σημαντικά χαρακτηριστικά είναι ότι η υπηρεσία που κάποιος χρησιμοποιεί είναι διαθέσιμη από κάποιον άλλον, ο οποίος και την διαχειρίζεται. Μία από τις βασικές αρχές του cloud computing είναι ότι δεν χρειάζεται ο χρήστης πλέον να ανησυχεί, πως η υπηρεσία που αγοράζει του παρέχεται, αλλά με υπηρεσίες βασισμένες στο web μπορεί να συγκεντρωθεί στην δουλειά που χρειάζεται να κάνει και να αφήσει αυτό το πρόβλημα σε κάποιον άλλον.

It's on-demand: Οι υπηρεσίες είναι διαθέσιμες on-demand και συνήθως αγοράζεται ως συνδρομή ή πληρωμή για ότι χρησιμοποιεί ο χρήστης. Άρα συνήθως οι υπηρεσίες αυτές

αγοράζονται όπως το ρεύμα ή υπηρεσίες τηλεφώνου ή πρόσβαση στο Internet. Μερικές φορές υπάρχουν και υπηρεσίες cloud computing που είναι δωρεάν ή πληρώνονται έμμεσα με κάποιον άλλον τρόπο, όπως για παράδειγμα διαφημίσεις. Όπως ακριβώς και με το ρεύμα μπορεί κάποιος να αγοράσει υπηρεσίες cloud computing ακριβώς όπως τις χρειάζεται από την μία μέρα στην επόμενη, το οποίο είναι πολύ σημαντικό αν οι ανάγκες του χρήστη αλλάζουν απρόβλεπτα.

It's public or private: Το cloud computing μπορεί να είναι ή δημόσιο ή ιδιωτικό. Για παράδειγμα υπηρεσίες email και δωρεάν υπηρεσίες που παρέχει η Google είναι από τα πιο γνωστά παραδείγματα δημόσιου cloud. Το ιδιωτικό cloud λειτουργεί όπως και το δημόσιο, αλλά η πρόσβαση γίνεται μέσω μίας ασφαλής σύνδεσης δικτύου.

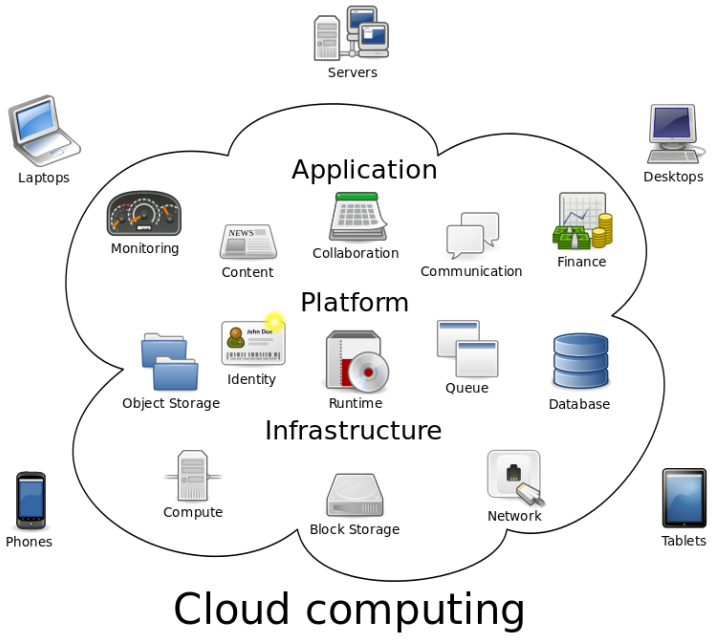
4.2.2 Είδη cloud computing

Υπάρχουν τρία είδη cloud computing, όμως υπάρχει ασάφεια για το πως αυτά ορίζονται και υπάρχει και ορισμένη επικάλυψη μεταξύ τους.

Infrastructure as a Service (IaaS): Αυτό σημαίνει ότι ο χρήστης αγοράζει πρόσβαση σε hardware μέσω του Internet, όπως servers και χώρο αποθήκευσης.

Software as a Service (SaaS): Σε αυτό το είδος αυτό που αγοράζεται είναι μία ολοκληρωμένη εφαρμογή σε σύστημα κάποιου άλλου. Τα πιο γνωστά παραδείγματα σε αυτό το είδος είναι οι υπηρεσίες email.

Platform as a Service (PaaS): Αυτό σημαίνει ότι ο χρήστης αναπτύσσει εφαρμογές χρησιμοποιώντας εργαλεία βασισμένα στο web ώστε να λειτουργούν σε λογισμικό συστήματος και hardware, τα οποία παρέχονται από κάποια εταιρία.



Εικόνα 10 Cloud Computing (Wikipedia)

4.3 Πλεονεκτήματα και μειονεκτήματα του cloud computing

4.3.1 Πλεονεκτήματα

Τα πλεονεκτήματα του cloud computing είναι αρκετά προφανή. Υπάρχει για τις εταιρίες μικρότερο κόστος εκ των προτέρων και μειωμένα κόστη υποδομής, αφού δεν χρειάζεται μία εταιρία να αγοράσει και να συντηρεί πολύπλοκα συστήματα, αφού μπορεί απλά να τα αγοράσει σαν cloud υπηρεσίες. Είναι αρκετά ευκολότερο να δημιουργήσει ο χρήστης μία εφαρμογή μέσω cloud computing. Οι υπηρεσίες cloud είναι αρκετά ευέλικτες σε αντίθεση με το local hardware, αφού είναι εφικτό να μειώσει ή να αυξήσει τι χρησιμοποιεί ο χρήστης πολύ εύκολα. Επιπροσθέτως ο χρήστης πληρώνει ότι χρησιμοποιεί μόνο άρα είναι βέλτιστο σε αυτό τον τομέα. Τέλος υπάρχουν και συνολικά πλεονεκτήματα στο περιβάλλον (όπως χαμηλότερες εκπομπές άνθρακα) αφού πολλοί χρήστης μοιράζονται αποδοτικά μεγάλα συστήματα.

4.3.2 Μειονεκτήματα

Η άμεση άνεση στην αγορά και στην χρήση cloud computing, έρχεται με κάποια τιμή. Αντί της αγοράς υπολογιστικών συστημάτων και λογισμικών μία φορά, το αρχικό κόστος γίνεται συνεχή έξοδα λειτουργίας. Αυτό μακροπρόθεσμα μπορεί να αποδειχτεί ακριβότερο. Επίσης η χρήση λογισμικών ως υπηρεσίες χρειάζεται και μία γρήγορη και αξιόπιστη χρήση Internet, το οποίο σε αρκετές χώρες ή μη αστικές περιοχές μπορεί να αποδειχτεί πρόβλημα.

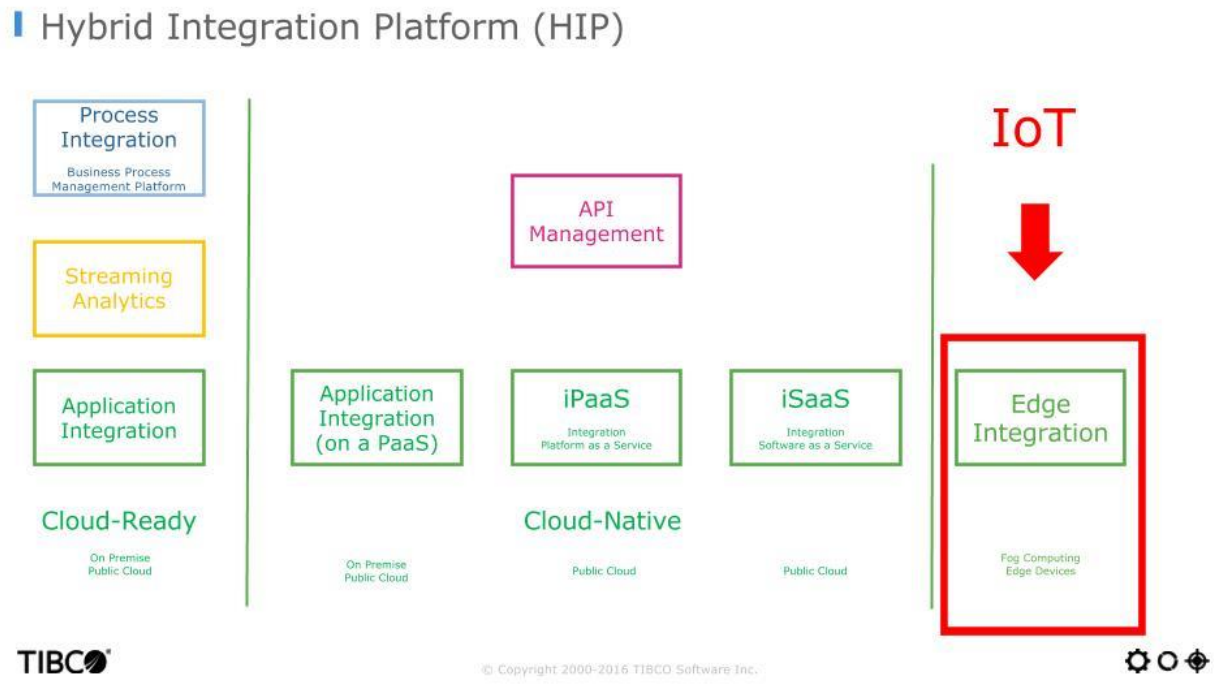
Ακόμα υπάρχει το ρίσκο να χρησιμοποιεί κάποιος πολύ τις υπηρεσίες κάποιου παρόχου, το οποίο μπορεί να κάνει πολύ δύσκολο την μεταφορά σε άλλο σύστημα ή σε κάποιο άλλο πάροχο υπηρεσιών αν χρειαστεί, όπως και το πρόβλημα, του τι θα συμβεί αν για οποιονδήποτε λόγο ο πάροχος αποφασίσει να σταματήσει να υποστηρίζει το σύστημα ή κάποιο προϊόν που κάποιος χρησιμοποιεί. Τέλος υπάρχουν και πιθανές παραβιάσεις της ιδιωτικότητας και προβλήματα ασφάλειας όταν κάποιος αποθηκεύει σημαντικά δεδομένα σε κάποιου άλλου το σύστημα σε κάποια άγνωστη τοποθεσία.

5

Πλατφόρμες για προγραμματισμό του Internet of Things

5.1 Γενικά

Όπως προαναφέρθηκε το integration είναι αναγκαίο για την επιτυχία των IoT αφού δεν υφίστανται χωρίς αυτό. Ωστόσο, δεν υπάρχει ακόμα κάποιο ενιαίο integration. Το integration των IoT θα είναι μέρος μίας υβριδικής αρχιτεκτονικής.



Εικόνα 11 Υβριδική πλατφόρμα integration (Wähner, 2017)

Το edge integration και η επεξεργασία γεγονότων γίνεται με τις λεγόμενες μηχανές επεξεργασίας. Αυτές, δεν εστιάζουν απλά στην απορρόφηση δεδομένων ή στο ETL, αλλά εφαρμόζουν επίσης λογική στο edge. Οι IoT μηχανές επεξεργασίας έχουν τα παρακάτω κοινά χαρακτηριστικά και δυνατότητες:

- Συνδεσιμότητα και ενορχήστρωση από διάφορες IoT πηγές δεδομένων και τεχνολογίες όπως MQTT, CoAP ή REST
- Διασύνδεση συσκευών, APIs και υπηρεσιών του internet
- Εκκαθάριση δεδομένων: μεταμόρφωση, φιλτράρισμα, δρομολόγηση, συσσωμάτωση, εμπλουτισμό δεδομένων
- Επεξεργασία δεδομένων: χειρισμό των σφαλμάτων, Re-Try, Re-Routing, αναμονή, επανάληψη
- Σύγχρονη και ασύγχρονη επικοινωνία
- Συνήθως επεξεργασία πραγματικού χρόνου
- Οπτική κωδικοποίηση, δοκιμές και debugging με συντάκτη ροών, αλλά επίσης και την επιλογή για σύνταξη πηγαίου κώδικα και επέκταση εξαρτημάτων χρησιμοποιώντας ανοιχτά SDKs και APIs
- Είναι δυνατόν να αναπτυχθούν στο edge, είτε σε ένα τοπικό IoT gateway ή απευθείας σε συσκευές και αισθητήρες

Υπάρχουν διάφορα frameworks που χρησιμεύουν για να αναπτυχθούν εφαρμογές που λειτουργούν σε IoT συσκευές και κάνουν ευκολότερη την χρήση τους και την διασύνδεση τους. Συγκεκριμένα θα δούμε αναλυτικά τα εξής: Eclipse Kura μαζί με Apache Camel, Node-Red και Flogo.

Τα κοινά χαρακτηριστικά των παραπάνω είναι τα ακόλουθα:

- Είναι όλα open source frameworks, δηλαδή είναι δυνατόν να χρησιμοποιηθούν από όλους δωρεάν.
- Επιτρέπουν τη σύνδεση με οποιαδήποτε τεχνολογία IoT.
- Διαθέτουν διεπαφή χρήστη στο web για σύνταξη κώδικα, δοκιμή και debugging.
- Είναι δυνατόν να αναπτυχθούν στο “edge”.
- Διαθέτουν επεκτάσιμα SDKs και APIs.

Παρόλο που τα παραπάνω frameworks εστιάζουν στο IoT edge integration, δημιουργήθηκαν για την εξυπηρέτηση διαφορετικών στόχων. Συνεπώς, αντί να υπάρχει ανταγωνισμός μεταξύ των τριών, συμπληρώνουν το ένα το άλλο.

5.2 Eclipse Kura με Apache Camel

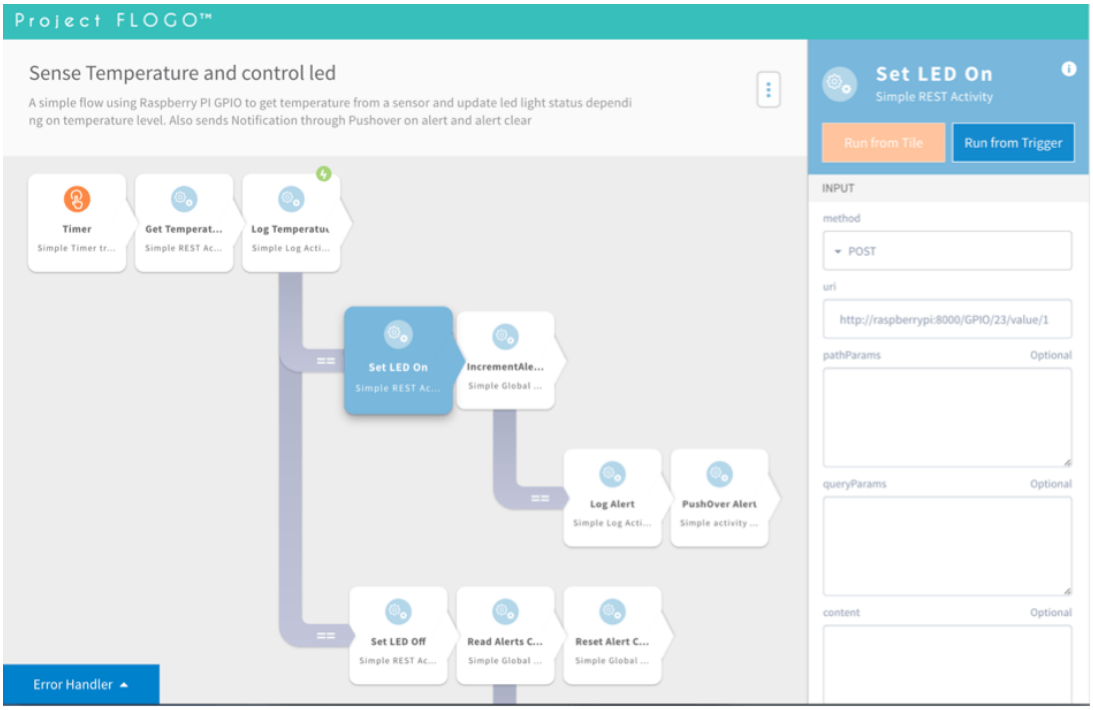
Ο συνδυασμός αυτών των δύο δημιουργεί μια πλατφόρμα η οποία είναι κατάλληλη για προγραμματισμό στο IoT gateway. Κυκλοφόρησε τον Δεκέμβριο του 2013, πράγμα που την καθιστά μία καλά χρονικά αναπτυγμένη πλατφόρμα. Είναι βασισμένη σε υπηρεσίες Java και OSGi (για I/O, δεδομένα, cloud και άλλες), και ο χρήστης μπορεί να το διαχειριστεί με τη βοήθεια γραφικού περιβάλλοντος, όπου όμως, επικεντρώνεται κυρίως σε σύνταξη κώδικα και όχι σε οπτική σχεδίαση του προγράμματος.

Το βασικό προσόν της πλατφόρμας αυτής, είναι ότι συνδυάζει τα σημαντικότερα πλεονεκτήματα των δύο λογισμικών. Το eclipse Kura παρέχει αρκετές χρήσιμες υπηρεσίες για το IoT gateway όπως το eclipse SmartHome, το Mosquitto, που είναι μία υλοποίηση για το πρότυπο MQTT server, και το eclipse Paho που συμβάλλει για τη συνδεσιμότητα μέσω MQTT client. Το Apache Camel από την άλλη πλευρά, παρέχει ένα integration framework, καθώς και τη δυνατότητα για οπτική κωδικοποίηση μέσω του JBoss ή του Talend (και τα δύο είναι open source).

Ακόμη, επειδή η πλατφόρμα είναι βασισμένη σε java, έχει τη δυνατότητα να εκτελεστεί σε πληθώρα συσκευών που μπορούν να τρέξουν java (on premise, cloud, edge devices, containers).

Παρόλα αυτά, η πλατφόρμα αυτή είναι ειδικά σχεδιασμένη για προγραμματιστές και integration specialists, γεγονός που σημαίνει ότι είναι βαρύ πρόγραμμα, που απαιτεί πολύπλοκες διαδικασίες για να εγκατασταθεί και να δημιουργηθεί η πρώτη IoT ροή. Επιπλέον, βασίζεται στη σύνταξη κώδικα, και επομένως απαιτεί περισσότερη δουλειά από τον προγραμματιστή.

5.3 *Flogo*



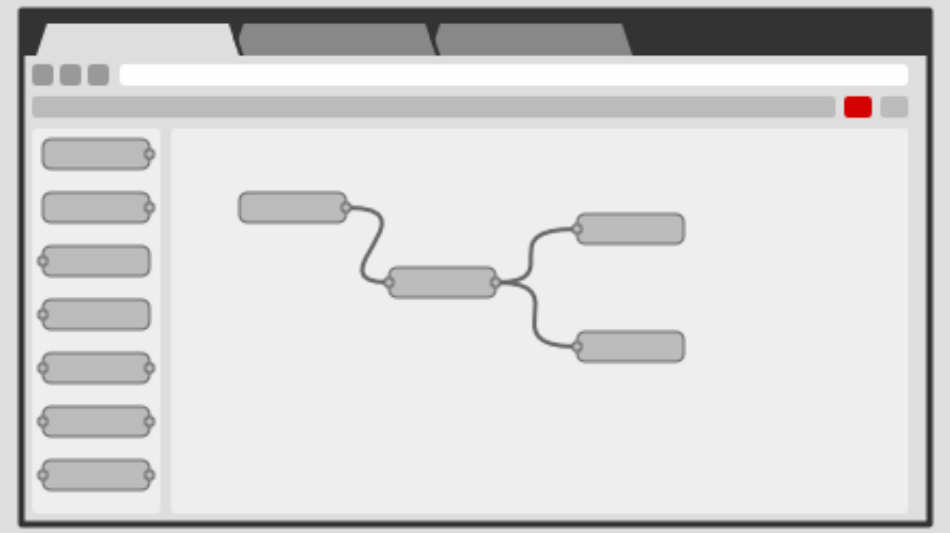
Εικόνα 12 Παράδειγμα Ροής Flogo (Middeljans, 2016)

https://rubenmiddeljans.wordpress.com/2016/11/16/project-flogo-is-it-really-happening/

Το Flogo επικεντρώνεται στο integration σε IoT Gateway και σε μικρές edge εφαρμογές. Είναι βασισμένο στη γλώσσα προγραμματισμού Go, και γι' αυτό διαθέτει μηδενικές εξαρτήσεις, και επομένως μηδενικό overhead. Είναι αρκετά απλό στο να εγκατασταθεί και να δημιουργηθεί η πρώτη IoT integration ροή. Επιπλέον, μας παρέχει τη δυνατότητα για οπτική κωδικοποίηση μέσω web, και οι ροές που συγγράφουμε μπορούν να διαμοιράζονται μέσω JSON, ενώ είναι ακόμη δυνατόν να δημιουργήσουμε δυαδικά αρχεία, μικρά σε μέγεθος, τα οποία μπορούν να εγκατασταθούν σε οποιαδήποτε συσκευή. Επιπρόσθετα, μπορεί να εκτελεστεί σε πολλές πλατφόρμες (on premise, cloud, συσκευές edge, container). Ωστόσο, είναι ακόμα αρκετά νέο λογισμικό, αφού κυκλοφόρησε μόλις τον Οκτώβριο του 2016.

Αξίζει επίσης να αναφερθεί, ένα πολύ σημαντικό πλεονέκτημα που διαθέτει το flogo, είναι το web-native step-back debugger. Μέσω αυτού ο προγραμματιστής διαθέτει την ικανότητα για debug της εφαρμογή του όσες φορές θέλει χωρίς να απαιτείται επανεκκίνηση. Μπορεί επίσης να κάνει διαδραστικά debug και να σχεδιάζει την πορεία του και να προσομοιώνει γεγονότα αισθητήρων. Τέλος, έχει τη δυνατότητα να εκμεταλλευτεί αυτήν την τεχνολογία για απομακρυσμένο debugging για ροές που έχουν αποτύχει να λειτουργήσουν.

5.4 *Node-Red*



Εικόνα 13 Node-Red

Όπως και οι άλλες δύο πλατφόρμες, το Node-Red επικεντρώνεται στο integration σε IoT Gateway. Αποτελεί ένα αρκετά αναπτυγμένο framework, αφού χρησιμοποιείται εδώ και τέσσερα χρόνια, και διαθέτει πολλά παραδείγματα και πλούσιο documentation,. Είναι εύκολο να εγκατασταθεί, και να δημιουργηθεί η πρώτη ροή σε IoT.

Βασικό του πλεονέκτημα είναι ότι διευκολύνει σημαντικά τη σύνδεση συσκευών, APIs και υπηρεσιών Internet. Μας παρέχει έναν browser-based flow editor, κάνοντας πιο εύκολη την σύνδεση ροών, χρησιμοποιώντας κόμβους, και οι ροές που δημιουργούνται, μπορούν να εκτελεστούν με ένα μόνο κλικ. Μπορούμε, επιπλέον, να συντάξουμε JavaScript συναρτήσεις με τη χρήση ενός rich text editor, και επίσης, μέσω ενσωματωμένων βιβλιοθήκων, έχουμε τη δυνατότητα να αποθηκεύσουμε χρήσιμες συναρτήσεις, ή και ολόκληρες ροές, για εύκολη επαναχρησιμοποίηση τους.

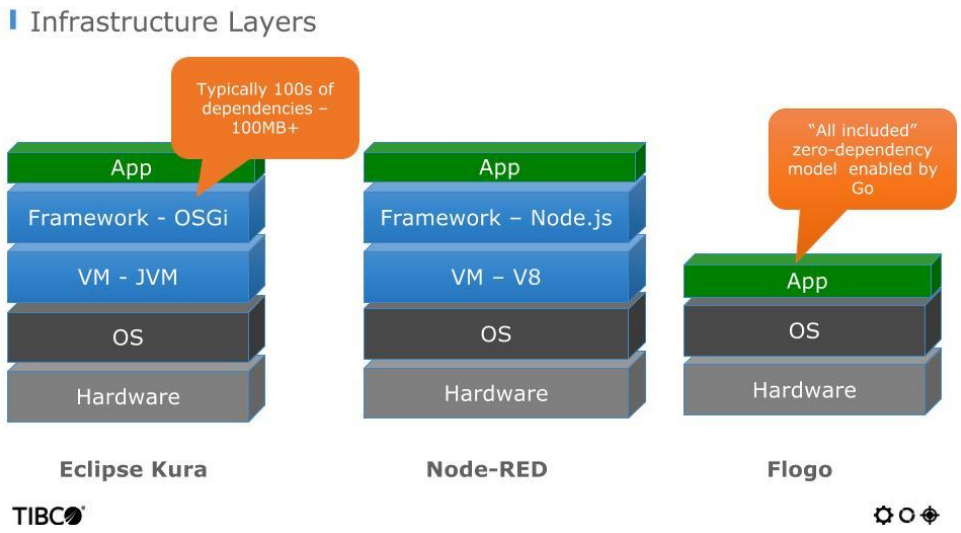
Το runtime του Node-Red είναι γραμμένο σε Node.js, και εκμεταλλεύεται πλήρως τα χαρακτηριστικά του. Αυτό το καθιστά ιδανικό για εκτέλεση σε φθηνές συσκευές όπως το Raspberry Pi ή στο cloud. Επιπλέον, μπορεί να εκτελεστεί σε διάφορες πλατφόρμες (on premise, cloud, συσκευές edge, container).

Οι ροές είναι αποθηκευμένες ως αρχεία JSON, τα οποία μπορούν εύκολα να εξαχθούν και να εισαχθούν, διευκολύνοντας έτσι το διαμοιρασμό τους. Ωστόσο, δεν υπάρχουν δυαδικά αρχεία για διαμοιρασμό και εγκατάσταση σε συσκευές που δεν εκτελούν το Node-Red, το οποίο σημαίνει ότι πιθανώς δεν είναι εφικτό να εκτελεστεί σε πολύ μικρές συσκευές που δεν διαθέτουν την δυνατότητα να εκτελέσουν το Node-red.

Αξίζει επίσης να αναφερθεί, ότι διαθέτει integration στην πλατφόρμα IBM Bluemix cloud και μπορεί να εκμεταλλευτεί εύκολα άλλες υπηρεσίες της Bluemix.

5.5 *Σύγκριση Eclipse Kura, Node-Red, Flogo*

Όπως φαίνεται κάθε ένα από αυτά τα frameworks επικεντρώνεται σε διαφορετικές χρήσεις IoT. Αυτό φαίνεται και από τα layers:



Εικόνα 14 Σύγκριση των Infrastructure layers που έχουν οι πλατφόρμες (Wähner, 2017)

Το Node-Red και το Kura χρησιμοποιούν μία τελείως διαφορετική πλατφόρμα (JavaScript και Java αντίστοιχα) πάνω από το λειτουργικό σύστημα και γι' αυτό έχουν πολλές εξαρτήσεις, γεγονός που τα καθιστά σημαντικά βαρύτερα από το Flogo όπως φαίνεται και στον παρακάτω πίνακα.

Resource Requirements

	Eclipse Kura	Node-RED	Flogo
VM	JVM	V8	Golang
Base Disk Space	59 MB	56 MB	~ 0 MB
Base Runtime Memory	~ 170 MB (with Open JDK) ~ 20 MB (with Oracle Embedded Java)	> 50 MB	~ 5 MB
Startup time	Slow (~8 sec)	Slow (~5 sec)	Fast (~1 sec)
Application Build	Slow	Not Applicable	Fast (~2 secs)

TIBCO

Εικόνα 15 Σύγκριση των πόρων που χρησιμοποιούν οι πλατφόρμες (Wähner, 2017)

Και τα τρία παραπάνω frameworks διαθέτουν αρκετά θετικά χαρακτηριστικά, και επομένως μία άμεση σύγκριση τους δεν είναι κατάλληλη. Ο κάθε προγραμματιστής πρέπει να επιλέγει το κατάλληλο, ανάλογα με τις απαιτήσεις της εφαρμογής του. Είναι πιθανό, η εφαρμογή να απαιτεί IoT Gateway και δυνατό integration σε πλατφόρμα Java, ή ένα καλό εργαλείο με πολλές IoT διεπαφές, με ένα ώριμο οπτικό IDE (integrated development environment), ή μία εφαρμογή πολύ ελαφριά για μία πολύ μικρή IoT συσκευή, και έτσι, ανάλογα με τις απαιτήσεις, γίνεται και η κατάλληλη επιλογή.

Επιλέξαμε να χρησιμοποιήσουμε το Node-Red διότι είναι πιο ελαφρύ από το eclipse Kura επειδή είναι βασισμένο σε node.js, αντιθέτως με το eclipse Kura που βασίζεται σε java που απαιτεί περισσότερους πόρους. Επιπλέον διαθέτει γραφικό περιβάλλον, στο οποίο γίνεται σχεδίαση ροών και δεν απαιτεί σύνταξη κώδικα σε μεγάλο βαθμό. Επιπρόσθετα, όπως προαναφέρθηκε, είναι αρκετά πιο απλό να εγκατασταθεί και να ξεκινήσουμε να δημιουργούμε ροές.

Ο λόγος που δεν επιλέξαμε το Flogo είναι επειδή είναι αρκετά καινούριο λογισμικό, με μόλις ένα χρόνο κυκλοφορίας, σε σχέση με το Node-Red, το οποίο κυκλοφόρησε περίπου πριν τέσσερα χρόνια, και αποτελεί επομένως ένα αρκετά ώριμο λογισμικό με πλούσιο documentation και με μεγάλη βιβλιοθήκη από έτοιμες ροές χρηστών που το χρησιμοποιούν. Στο Flogo θα απαιτούνταν περισσότερο συγγραφή κώδικα, επειδή δεν υπάρχουν αρκετοί έτοιμοι κόμβοι όπως συμβαίνει στο Node-Red, και επίσης η επίλυση πιθανών προβλημάτων θα απαιτούσε περισσότερο χρόνο, από τη στιγμή που ο αριθμός των χρηστών και των έτοιμων εφαρμογών σε αυτό είναι ελάχιστες σε σχέση με τις άλλες πλατφόρμες.

6

Προβλήματα και λύση

Σε αυτό το κεφάλαιο θα αναλύσουμε προκλήσεις που υπάρχουν στους τομείς των IoT, Big Data και cloud computing, και στην συνέχεια θα αναλύσουμε το πρόβλημα που λύνουμε εμείς στην διπλωματική μας.

6.1 Προκλήσεις στον τομέα του IoT

Σύμφωνα με το άρθρο “The Programming Challenges of IoT” (Esposito, 2014) υπάρχουν δύο τύπου προκλήσεων: Data and control και Information and business logic.

6.1.1 Data and control

Power

Το πρόβλημα αυτό είναι προφανής. Οι περισσότερες IoT συσκευές είναι ασύρματες και για αυτό το λόγο υπάρχει το πρόβλημα της ενέργειας. Η μία λύση είναι να δημιουργούνται οι αλγόριθμοι αποδοτικά και να μην χάνονται κύκλοι του επεξεργαστή χωρίς να κάνουν κάποια εργασία. Η δεύτερη λύση είναι πιο περίπλοκη, η οποία είναι να απενεργοποιούνται οι συσκευές όταν δεν χρειάζονται και να ξανά ενεργοποιούνται όταν πρέπει.

Latency

Το κύριο πρόβλημα για αυτό βρίσκεται στο hardware. Τα chips των IoT συσκευών είναι συνήθως πολύ μικρά, το οποίο σημαίνει ότι μπορούν να είναι τόσο δυνατά όσο επιτρέπει η τωρινή τεχνολογία των τρανζίστορ.

Ένας άλλος λόγος για αυτό το πρόβλημα είναι η υποδομή του δικτύου. Αφού όσο περισσότερες συσκευές IoT, τόσο λιγότερο διαθέσιμο bandwidth υπάρχει. Και αυτό γίνεται χειρότερο επειδή οι πλειοψηφία των IoT συσκευών μεταδίδουν συνέχεια, χωρίς κάποιος να τους το ζητήσει. Άρα υπάρχει μεγάλη πιθανότητα χαμένου bandwidth.

Unreliability

Οι συσκευές IoT είναι συνήθως φθηνές, και άρα δυσλειτουργούν πιο συχνά. Επίσης αφού δεν είναι πολύ αξιόπιστες συσκευές δεν παράγουν και πολύ αξιόπιστες πληροφορίες, και άρα στο επίπεδο της εφαρμογής πρέπει να γίνει περισσότερος έλεγχος.

6.1.2 Information and Business Logic

Vast and thin data

Η ποσότητα των δεδομένων που παράγουν τα IoT είναι τεράστια και δεν είναι πάντα εύκολο να επεξεργαστούν και να παραχθεί κάποιο χρήσιμο αποτέλεσμα. Ακόμα και να ξέρει κάποιος τι θέλει να κάνει τα δεδομένα από κάποια συσκευή, θα πρέπει να μάθει να χρησιμοποιεί νέους αλγορίθμους και δομές δεδομένων.

6.2 Προκλήσεις στα Big Data

Αντιμετώπιση των τεράστιων ποσών δεδομένων

Η πιο προφανής πρόκληση είναι απλά η αποθήκευση και η ανάλυση όλων αυτών των πληροφοριών. Σύμφωνα με έρευνες η ποσότητα των δεδομένων που αποθηκεύεται στα συστήματα σε όλο των κόσμο διπλασιάζεται ανά χρόνο. Πολλά από αυτά τα δεδομένα είναι χωρίς δομή. Έγγραφα, φωτογραφίες, αρχεία ήχου, βίντεο και άλλα δεδομένα χωρίς δομή είναι πολύ δύσκολα να αναλυθούν. Σύμφωνα με το IDG, η διαχείριση δεδομένων χωρίς δομή αυξάνεται, από 31% το 2015 σε 45% το 2016.

Για να αντιμετωπίσουν αυτήν την ανάπτυξη των δεδομένων, οργανισμοί στρέφονται σε διάφορες τεχνολογίες. Όσον αφορά την αποθήκευση, τεχνολογίες συμπίεσης, deduplication και tiering μπορούν να μειώσουν τον χώρο και το κόστος που σχετίζονται με την αποθήκευση Big Data. Για την διαχείριση και την ανάλυση, εταιρίες χρησιμοποιούν εργαλεία όπως βάσεις δεδομένων NoSQL, Hadoop, Spark, λογισμικά ανάλυσης Big Data, τεχνητή νοημοσύνη και machine learning για να τους βοηθήσουν να αναλύσουν τα Big Data και να βρουν πληροφορίες που χρειάζεται η εταιρία.

Δημιουργία πληροφοριών εγκαίρως

Οι επιχειρήσεις προφανώς δεν έχουν ως σκοπό απλά να αποθηκεύουν αυτά τα δεδομένα, αλλά να τα χρησιμοποιήσουν ώστε να επιτύχουν τους επιχειρηματικούς στόχους τους. Οι στόχοι αυτοί βοηθούν οργανισμούς να γίνουν πιο ανταγωνιστικοί, αλλά μόνο αν μπορούν να εξάγουν

πληροφορίες από τα Big Data και να δράσουν σε αυτές γρήγορα. Για αυτό το λόγο επιχειρήσεις επενδύουν σε λογισμικό με δυνατότητες real time ανάλυση.

Ενοποίηση διαφορετικών πηγών δεδομένων

Η ποικιλία που υπάρχει στα Big Data οδηγεί σε προκλήσεις ενσωμάτωσης των δεδομένων. Τα Big Data προέρχονται από διάφορα μέρη (εφαρμογές, social media, συστήματα email, και άλλα). Ο συνδυασμός όλων αυτών των δεδομένων ώστε να χρησιμοποιηθούν για να παραχθούν αναφορές είναι πολύ δύσκολος.

Επικύρωση Δεδομένων

Ένα ακόμα πρόβλημα είναι ότι τα δεδομένα από διάφορα συστήματα μπορεί να μην συμφωνούν μεταξύ τους. Η διαδικασία για να συμφωνήσουν όλα αυτά τα αρχεία, και επίσης να είναι σίγουρο ότι είναι ακριβής, χρήσιμα και ασφαλή, είναι πολύ περίπλοκη.

Ασφάλεια Δεδομένων

Η ασφάλεια είναι μία ανησυχία των επιχειρήσεων, αφού τα Big Data είναι ελκυστικοί στόχοι για hackers. Όμως οι περισσότεροι οργανισμοί θεωρούν ότι οι τωρινοί μέθοδοι ασφάλειας επαρκούν.

6.3 Προκλήσεις στο cloud computing

Ασφάλεια των δεδομένων και ιδιωτικότητα

Από την στιγμή που τα δεδομένα βρίσκονται στο δίκτυο μίας εταιρίας, τότε υπάρχει ρίσκο για την ιδιωτικότητα των δεδομένων αν οι πάροχοι των υπηρεσιών cloud δεν ακολουθούν τους κανονισμούς που υπάρχουν για την ιδιωτικότητα και την ασφάλεια των δεδομένων. Για αυτό είναι πολύ επικίνδυνο να μοιράζεται κάποιος απόρρητες πληροφορίες με κάποια εξωτερική εταιρία.

Έλλειψη τυποποίησης

Δεν υπάρχουν καθορισμένες οδηγίες, κάτω από τις οποίες λειτουργούν οι πάροχοι cloud. Το αποτέλεσμα αυτού είναι ότι διάφοροι πάροχοι λειτουργούν διαφορετικά, και άρα διαφέρουν και τα πρωτόκολλα ασφάλειας τους.

Ιδιοκτησία δεδομένων

Το θέμα της ιδιοκτησίας των δεδομένων είναι ακόμα και σήμερα μία πρόκληση του cloud computing. Όταν κάποιος δίνει τα δεδομένα του σε έναν πάροχο cloud, τότε χάνει την ιδιοκτησία των δεδομένων του. Τα περισσότερα συμβόλαια δηλώνουν ότι αυτό γίνεται έτσι ώστε να μπορούν να προστατέψουν καλύτερα τα δεδομένα, από την στιγμή που θα έχουν περισσότερη νομική προστασία.

6.4 *Στόχος διπλωματικής*

Εμείς στην διπλωματική λύνουμε το πρόβλημα ότι στα IoT και στα Big Data υπάρχουν πάρα πολλές διαφορετικές πηγές δεδομένων, και ως συνέπεια μία τεράστια ποσότητα δεδομένων. Η επιλογή μίας κατάλληλης πηγής δεδομένων για την εφαρμογή που θέλει να δημιουργήσει ένας χρήστης δεν είναι εύκολη διαδικασία, αφού υπάρχουν πολλές μεταβλητές που έχει να αναλύσει ο χρήστης για να βρει την καλύτερη για την εφαρμογή του. Επίσης διαφορετικές εφαρμογές έχουν διαφορετικές απαιτήσεις και άρα διαφορετική μέθοδο επιλογής.

Συγκεκριμένα κάποια εφαρμογή μπορεί να απαιτεί η πηγή να έχει πολύ γρήγορη ανταπόκριση στο 90% των περιπτώσεων και να μην την ενδιαφέρει σημαντικά αν στις υπόλοιπες περιπτώσεις υπάρχει πιθανότητα να υπάρξει κάποια καθυστέρηση. Άλλες εφαρμογές μπορεί να απαιτούν να έχει η πηγή που χρησιμοποιεί κάθε φορά την καλύτερη εφικτή ανταπόκριση.

Εμείς για να λύσουμε αυτό το πρόβλημα δημιουργήσαμε μία εφαρμογή η οποία μαζεύει στατιστικά από τις πηγές δεδομένων που θέλει ο χρήστης να χρησιμοποιήσει και στην συνέχεια βρίσκει ποια η καλύτερη.

6.5 *Τρόπος πρόσβασης αποτελεσμάτων*

Για να έχει πρόσβαση κάποιος στα αποτελέσματα μας, δηλαδή το ποια πηγή είναι καλύτερη, δημιουργήθηκαν δύο REST APIs, όπου το ένα επιστρέφει τα δεδομένα από την πηγή που επιλέχθηκε και το δεύτερο επιστρέφει το όνομα της πηγής που επιλέχθηκε. Το δεύτερο το προσθέσαμε γιατί μπορεί ο χρήστης για τον οποιονδήποτε λόγο να μην θέλει να του επιστρέψουμε εμείς τα αποτελέσματα απλά να θέλει να μάθει απλά ποια είναι η βέλτιστη πηγή.

Δεν χρησιμοποιήθηκε MQTT για την αποστολή των αποτελεσμάτων αντί REST με HTTP διαδικασίες, επειδή θέλουμε ο χρήστης να μπορεί να έχει πρόσβαση στα δεδομένα on-demand και όχι να αποστέλλονται συνεχώς, κάτι το οποίο θα ήταν το αποτέλεσμα αν χρησιμοποιούσαμε MQTT πρωτόκολλο. Το MQTT είναι βέλτιστο να χρησιμοποιείται από αισθητήρες τύπου συσκευές, οι οποίοι μαζεύουν συνεχώς δεδομένα και τα στέλνουν σε ένα κεντρικό σύστημα για να αποθηκευτούν και να επεξεργαστούν.

Η εφαρμογή μας δημιουργήθηκε κυρίως για τον έλεγχο πηγών δεδομένων, οι οποίες συνήθως είναι αισθητήρες ή διάφορες άλλες συσκευές IoT που συλλέγουν δεδομένα, αλλά μπορεί και να είναι και πιο περίπλοκα συστήματα, και όχι να τρέχει υποχρεωτικά σε πολύ αδύναμου hardware συσκευή. Για αυτό και η επιλογή του MQTT πρωτοκόλλου, που γενικά είναι προτιμότερο σε IoT συσκευές επειδή είναι κατάλληλο για αυτές, δεν είναι αναγκαία στην εφαρμογή μας.

6.6 Σημασία της εφαρμογής

Γενικά όπως αναφέρθηκε, τα δεδομένα είναι πολύ σημαντικά στην σημερινή εποχή, επειδή μπορούν να οδηγήσουν σε πολλά πλεονεκτήματα αν διαχειριστούν με σωστό τρόπο. Για αυτό και η επιλογή κατάλληλης πηγής δεδομένων, η οποία θα επιστρέφει δεδομένα γρήγορα και δεν θα καθυστερεί και επίσης θα λειτουργεί σωστά σε ένα αρκετά μεγάλο ποσοστό του χρόνου, είναι πολύ σημαντική, επειδή η ταχύτητα συλλογής των δεδομένων μπορεί να επηρεάσει σημαντικά τα πλεονεκτήματα που θα αποκτηθούν από αυτά.

Για αυτό θεωρούμε ότι η εφαρμογή που δημιουργήσαμε είναι πολύ σημαντική επειδή μπορεί εύκολα ο χρήστης και χωρίς ιδιαίτερη προσπάθεια από τον ίδιο, να την χρησιμοποιήσει και να βρίσκει κάθε φορά που είναι η βέλτιστη πηγή δεδομένων για να χρησιμοποιήσει. Επιπλέον είναι εύκολη η κλιμάκωση της σε περισσότερες πηγές δεδομένων, όταν και αν ο χρήστης αποφασίσει να επεκτείνει τον έλεγχο του.

Οι μετρικές που χρησιμοποιήθηκαν θεωρούμε ότι είναι σημαντικές επειδή με τον μέσο όρο και με έναν μεγάλο αριθμό μετρήσεων μπορούμε με πολύ μικρό σφάλμα να υπολογίσουμε τον χρόνο τον οποίο θα κάνει η κάθε πηγή δεδομένων να ανταποκριθεί όταν της ζητηθεί. Τέλος ο μέσος όρος με συντελεστές, αν επιθυμεί ο χρήστης να χρησιμοποιήσει αυτήν την μετρική, παίρνει υπόψιν του την τωρινή συμπεριφορά της πηγής δεδομένων, η οποία μπορεί να έχει αλλάξει σημαντικά.

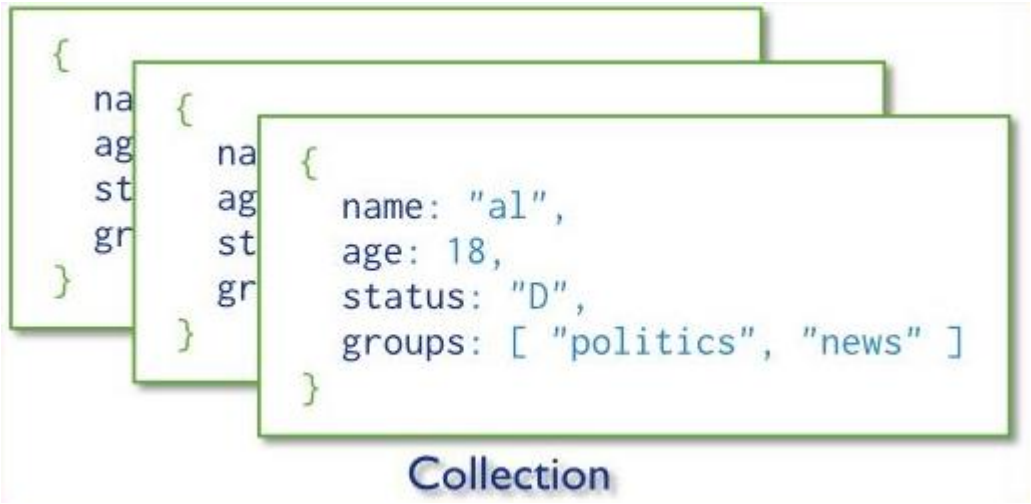
7

Τεχνολογίες της Διπλωματικής

7.1 MongoDB

Για τη βάση δεδομένων για την εφαρμογή μας χρησιμοποιήθηκε η MongoDB. Είναι NoSQL βάση δεδομένων, δηλαδή δεν είναι σαν τις συνηθισμένες σχεσιακές βάσεις δεδομένων. Η βάση αυτή έχει να κάνει με έγγραφα, και όχι με γραμμές, και είναι πολύ γρήγορη, επεκτάσιμη και εύκολη στη χρήση. Για να το επιτύχει αυτό η MongoDB έχασε ορισμένες λειτουργίες, το οποίο σημαίνει ότι η MongoDB δεν είναι ιδανική βάση για όλες τις καταστάσεις. Η MongoDB είναι πολύ καλή στην αποθήκευση σύνθετων δεδομένων και παρέχει μία πλούσια, προσανατολισμένη σε έγγραφα βάση και προσφέρει ταχύτητα και επεκτασιμότητα. Επιπλέον μπορεί να εκτελεστεί σε οποιοδήποτε σύστημα χρειαστεί, Windows, Linux, και Mac.

Η εγκατάσταση της MongoDB μπορεί να έχει πολλές βάσεις δεδομένων. Η κάθε βάση δεδομένων αποτελείται από συλλογές και κάθε συλλογή αποτελείται τελικά από έγγραφα. Το έγγραφο είναι ένα σύνολο από ζεύγη κλειδιού-τιμής. Τα έγγραφα έχουν δυναμικό σχήμα, δηλαδή τα έγγραφα της ίδιας συλλογής δεν είναι ανάγκη να έχουν τα ίδια πεδία ή δομή, ή και τα κοινά πεδία μίας συλλογής εγγράφων μπορεί να περιέχουν διαφορετικούς τύπους δεδομένων. Για παράδειγμα η δομή μίας συλλογής φαίνεται παρακάτω.



Εικόνα 16 Παράδειγμα δομής συλλογής της MongoDB

Η MongoDB αποθηκεύει τα έγγραφα σε φόρμα σειριοποίησης BSON. Τα BSON είναι μία δυαδική αναπαράσταση των JSON εγγράφων, αλλά περιέχει πολλούς περισσότερους τύπους δεδομένων από το JSON. Η τιμή ενός πεδίου μπορεί να είναι οποιοσδήποτε τύπος δεδομένων BSON, συμπεριλαμβανόμενων και άλλων εγγράφων, πινάκων ή και πινάκων από έγγραφα.

Στα έγγραφα το πρώτο πεδίο είναι πάντα το _id που δημιουργείται αυτόματα από την MongoDB, για το οποίο δημιουργεί ένα μοναδικό ευρετήριο κατά τη δημιουργία της συλλογής. Αυτό είναι μοναδικό και θέτεται αυτόματα αυξανόμενο για κάθε έγγραφο εκτός αν το θέσει ο χρήστης κατά την εισαγωγή του εγγράφου. Ο χρήστης για να έχει πρόσβαση στα στοιχεία ενός εγγράφου ή ενός πίνακα χρησιμοποιεί την τελεία, δηλαδή “document.field”.

Εμείς επιλέξαμε MongoDB όχι μόνο γιατί είναι αρκετά γρήγορη βάση για έγγραφα αλλά ακόμα επειδή υπάρχουν έτοιμοι κόμβοι για το Node-Red που μπορεί ο χρήστης να εισάγει στην ροή του και να κάνει queries στη βάση.

Αυτός ο κόμβος βρίσκεται στο πακέτο node-red-node-mongodb το οποίο πρέπει να εγκατασταθεί για να χρησιμοποιηθεί. Η εγκατάσταση γίνεται πολύ απλά μέσα από το διαδραστικό UI του Node-Red, πατώντας στις επιλογές manage palette. Ή αλλιώς γίνεται τρέχοντας την εντολή “npm install node-red-node-mongodb” στον φάκελο του Node-Red.

Αυτό το πακέτο χρησιμεύει για αποθήκευση και ανάκτηση δεδομένων από μία τοπική βάση MongoDB. Περιέχει έναν κόμβο για είσοδο και έναν για έξοδο.

Είσοδος

Καλεί μία μέθοδο σε μία συλλογή MongoDB ανάλογα με την επιλογή. Υπάρχουν τρεις επιλογές:

- Find: Εκτελεί query στην συλλογή χρησιμοποιώντας το msg.payload ως statement για την συνάρτηση .find(). Προαιρετικά υπάρχουν και μερικές ακόμα επιλογές που πρέπει να τεθούν από κόμβο συνάρτησης ως πεδία του msg. Με το projection γίνεται να

περιοριστούν τα πεδία που θα επιστραφούν, με το sort επιλέγεται τρόπος ταξινόμησης, με το limit μπορεί να τεθεί όριο στο πόσα έγγραφα θα επιστραφούν και με το skip μπορεί να προσπεραστεί ένας αριθμός.

- Count: Επιστρέφει τον αριθμό των εγγράφων σε μία συλλογή ή των αριθμό των εγγράφων που ταιριάζουν σε ένα query που τίθεται στο msg.payload.
- Aggregate: Παρέχει πρόσβαση στο aggregation pipeline χρησιμοποιώντας το msg.payload ως πίνακα pipeline.

Η συλλογή καθορίζεται είτε στις ρυθμίσεις του κόμβου είτε από έναν κόμβο συνάρτησης στο πεδίο collection του msg. Αν καθοριστεί από τις ρυθμίσεις τότε δεν θα έχει σημασία αν υπάρχει πεδίο collection στο msg. Το αποτέλεσμα επιστρέφεται στο msg.payload.

Έξοδος

Υπάρχουν τέσσερις επιλογές:

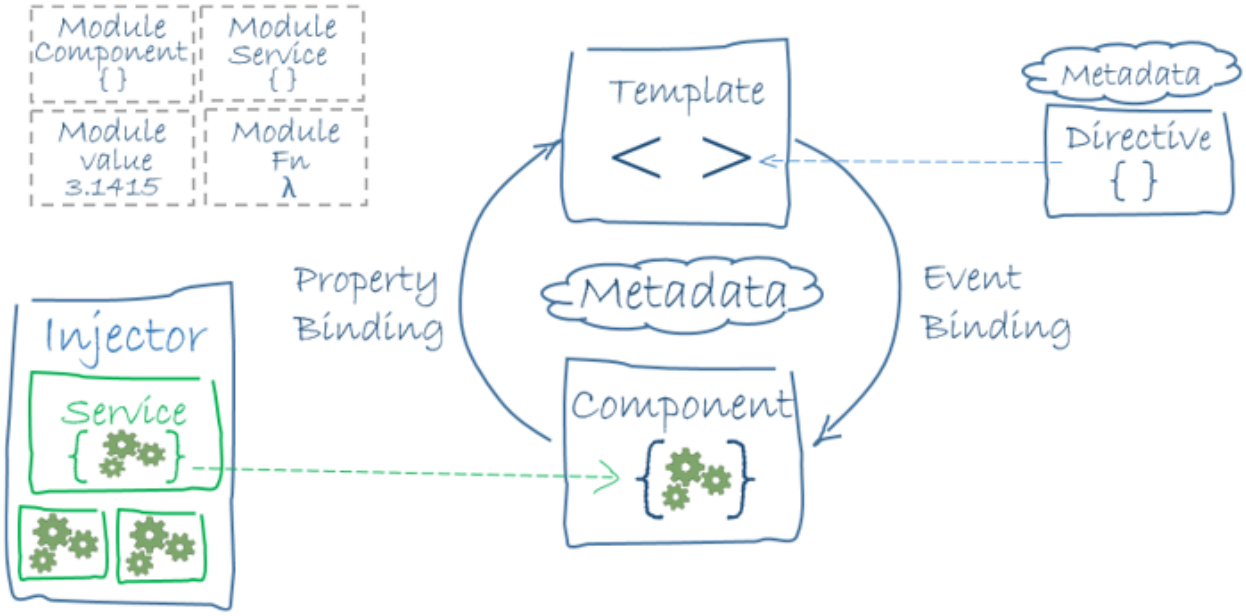
- Save: Θα κάνει update ένα αντικείμενο που ήδη υπάρχει ή θα εισάγει ένα νέο αντικείμενο αν δεν υπάρχει
- Insert: Εισαγωγή νέου αντικειμένου.
- Update: Θα τροποποιήσει ένα υπάρχον αντικείμενο ή αντικείμενα. Για την επιλογή των αντικειμένων που θα τροποποιηθούν χρησιμοποιείται το πεδίο query στο msg και η τροποποίηση στο στοιχείο χρησιμοποιεί το πεδίο payload. Το Update μπορεί και να εισάγει ένα αντικείμενο αν δεν υπάρχει.
- Remove: θα αφαιρέσει αντικείμενα που ταιριάζουν με το query στο πεδίο payload. Ένα κενό query θα διαγράψει όλα τα αντικείμενα στη συλλογή.

Οι επιλογές save και insert μπορούν να αποθηκεύσουν είτε το msg είτε το msg.payload. Αν το msg.payload είναι επιλεγμένο τότε πρέπει να περιέχει ένα αντικείμενο αλλιώς δημιουργηθεί ένα αντικείμενο με το όνομα payload.

Η συλλογή καθορίζεται όπως στην είσοδο. Προκαθορισμένα η MongoDB δημιουργεί ένα πεδίο _id ως το κύριο κλειδί άρα επαναλαμβανόμενες εισαγωγές του ίδιου msg θα έχει σαν αποτέλεσμα πολλές καταχωρήσεις στη βάση. Αν δεν είναι αυτό επιθυμητή συμπεριφορά, για παράδειγμα μπορεί να θέλει ο χρήστης επαναλαμβανόμενες εισαγωγές να αντικαθίστανται, τότε πρέπει να τεθεί ένα _id πεδίο στο msg για να είναι σταθερό σε έναν προηγούμενο κόμβο συνάρτησης. Αν αποθηκεύεται μόνο το msg.payload τότε το _id πρέπει να είναι πεδίο του payload.

7.2 AngularJs

Η AngularJs είναι ένα framework για την δημιουργία εφαρμογών σε HTML και/ή σε JavaScript ή μία γλώσσα σαν TypeScript η οποία γίνεται compile σε JavaScript. Υπάρχουν πολλές βιβλιοθήκες, μερικές από αυτές είναι κύριες και μερικές προαιρετικές. Για τη σύνταξη μίας Angular εφαρμογής συντάσσονται HTML πρότυπα με σημάνσεις Angular, και γράφονται κλάσεις component για την διαχείριση αυτών των προτύπων, προσθέτοντας λογική σε υπηρεσίες, και πακετάροντας components και υπηρεσίες σε modules.



Εικόνα 17 Επισκόπηση αρχιτεκτονικής της AngularJs

Η εφαρμογή ξεκινάει με bootstrapping στο root module. Η Angular παίρνει τον έλεγχο, παρουσιάζοντας την εφαρμογή σε ένα browser και ανταποκρίνεται σε αλληλεπιδράσεις με τον χρήστη σύμφωνα με τις οδηγίες που δόθηκαν.

Οι εφαρμογές της Angular είναι modular και έχει το δικό της σύστημα που καλείται NgModules. Κάθε εφαρμογή έχει τουλάχιστον ένα NgModule, το root module που καλείται AppModule. Μικρές εφαρμογές μπορεί να έχουν μόνο αυτό το module, αλλά μεγαλύτερες εφαρμογές έχουν πολύ περισσότερα

Ένα component ελέγχει ένα κομμάτι της οθόνης που καλείται view. Η λογική του component, τι κάνει για να υποστηρίξει το view, θέτεται μέσα σε μία κλάση. Η κλάση αλληλοεπιδρά με το view μέσω ενός API από ιδιότητες και μεθόδους. Η Angular δημιουργεί, ανανεώνει και καταστρέφει components όπως κινείται ο χρήστης στην εφαρμογή. Η εφαρμογή μπορεί να κάνει μία ενέργεια οποιαδήποτε στιγμή στον κύκλο ζωής μέσω προαιρετικών lifecycle hooks.

Ο καθορισμός του view ενός component γίνεται με το δικό του template. Ένα template είναι μία φόρμα HTML που λέει στην Angular πως να κάνει render το component. Αυτό το template μοιάζει με συνηθισμένη HTML, εκτός από μερικές διαφορές. Για παράδειγμα:

src/app/hero-list.component.html

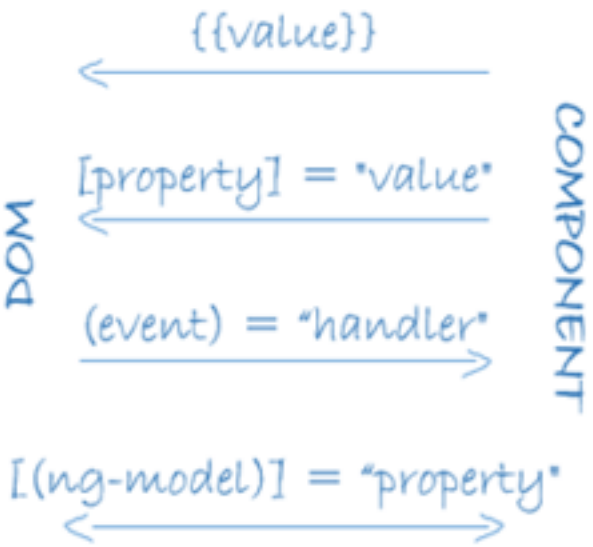
```
<h2>Hero List</h2>

<p><i>Pick a hero from the list</i></p>
<ul>
  <li *ngFor="let hero of heroes" (click)="selectHero(hero)">
    {{hero.name}}
  </li>
</ul>

<hero-detail *ngIf="selectedHero" [hero]="selectedHero"></hero-detail>
```

Εικόνα 18 Παράδειγμα ενός template της AngularJs

Βλέπουμε ότι χρησιμοποιούνται πολλά τυπικά στοιχεία της HTML, αλλά υπάρχουν και μερικές διαφορές οι οποίες χρησιμοποιούν σύνταξη της Angular. Και συγκεκριμένα το <hero-detail> tag είναι ένα προσαρμοσμένο στοιχείο το οποίο αντιπροσωπεύει ένα καινούριο component. Τα προσαρμοσμένα στοιχεία συνδυάζονται τέλεια με την HTML στις ίδιες διατάξεις.

Η Angular χρησιμοποιεί metadata τα οποία την πληροφορούν πως να επεξεργαστεί μία κλάση. Ένα component είναι απλά μία κλάση και δεν είναι component μέχρι να ενημερωθεί η Angular για αυτό. Αυτό γίνεται συνδέοντας στην κλάση metadata. Τα template, metadata και component μαζί περιγράφουν ένα view. Το σημαντικό είναι ότι πρέπει να προστεθούν metadata στον κώδικα ώστε η Angular να ξέρει τι να κάνει.

Χωρίς κάποιο framework, ο προγραμματιστής θα ήταν υπεύθυνος για να στέλνει τιμές δεδομένων στην HTML και να μετατρέπει τις αποκρίσεις του χρήστη σε ενέργειες και αναβαθμίσεις των τιμών. Η σύνταξη μίας τέτοιας λογικής είναι κουραστική, επιρρεπή σε λάθη, και δύσκολη να διαβαστεί. Η Angular υποστηρίζει δέσμευση δεδομένων, ένας μηχανισμός, ο οποίος συντονίζει τμήματα ενός template με τμήματα ενός component. Για να γνωρίζει η



Εικόνα 19 Τρόποι σύνταξης της δέσμευσης δεδομένων

Angular πως να συνδέσει και τις δύο πλευρές προστίθενται σήματα δέσμευσης. Όπως φαίνεται στο διπλανό διάγραμμα υπάρχουν τέσσερεις συντάξεις της δέσμευσης δεδομένων.

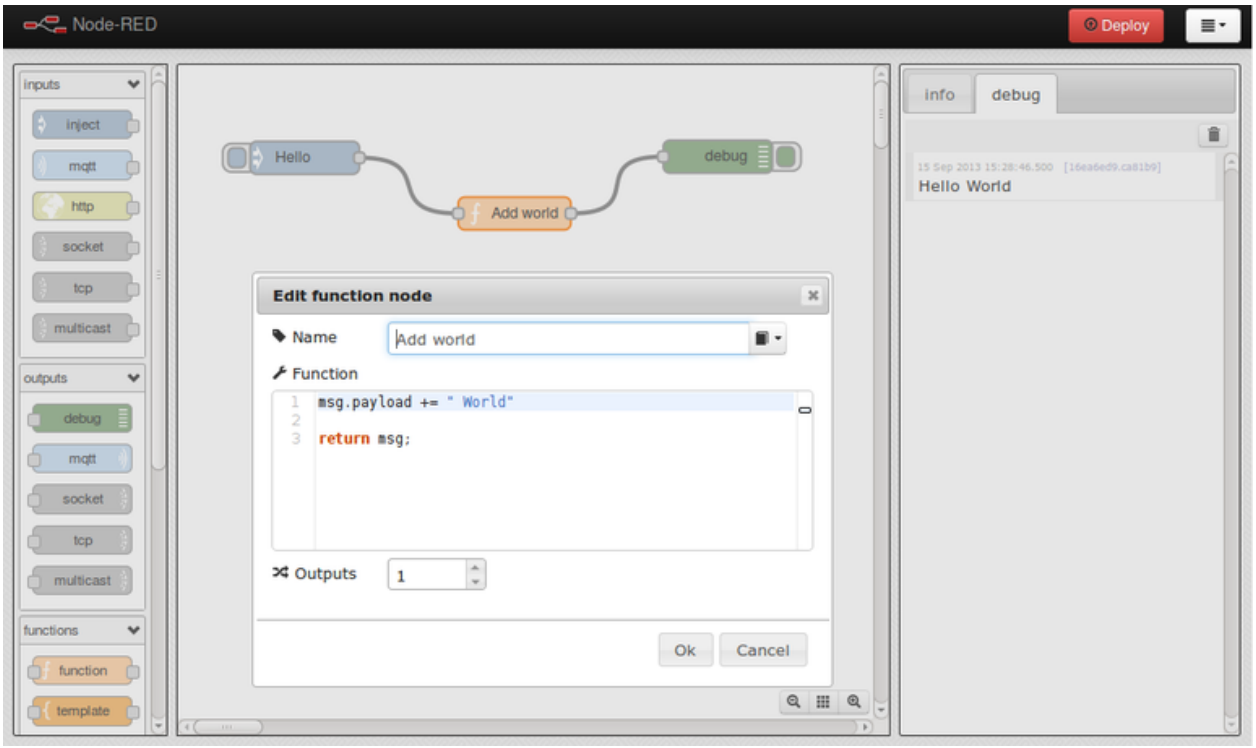
Η αμφίδρομη δέσμευση είναι σημαντική μορφή, η οποία συνδυάζει property και event δέσμευση σε μία μόνο σημείωση, χρησιμοποιώντας το ngModel. Στην αμφίδρομη δέσμευση, μία τιμή δεδομένων ρέει στο κουτί εισόδου από το component όπως στην δέσμευση property. Οι αλλαγές του χρήστη ρέουν πίσω στο component επαναφέροντας το property στην τελευταία τιμή, όπως στην δέσμευση event. Η Angular επεξεργάζεται όλες τις δεσμεύσεις μία φορά για κάθε κύκλο γεγονότος JavaScript από το root της εφαρμογής μέχρι και όλα τα παιδιά components. Η δέσμευση παίζει σημαντικό ρόλο στην επικοινωνία μεταξύ ενός template και του component του, όπως και στην επικοινωνία μεταξύ πατέρα και παιδιού component.

Εμείς επιλέξαμε AngularJs για τον σχεδιασμό των διαγραμμάτων ώστε να είναι δυνατόν να ανανεωθεί αυτόματα και να εμφανίζονται τα διαγράμματα όταν παράγονται τα δεδομένα που χρειαζόμαστε για να το σχεδιάσουμε. Η Angular μπορεί να ελέγχει αυτόματα αν τα δεδομένα παράχθηκαν χωρίς καμία ενέργεια του χρήστη και να παράγει κατευθείαν τα διαγράμματα. Για την ενσωμάτωση με το Node-Red χρησιμοποιήθηκε ένας κόμβος template με τον οποίο έγινε ανακατεύθυνση στο URL που είναι η εφαρμογή της Angular. Όταν ο χρήστης εισέλθει στο url ώστε να εμφανιστεί το διάγραμμα γίνεται αυτόματη ανακατεύθυνση στην Angular και ταυτόχρονα το Node-Red ξεκινάει να παράγει τα δεδομένα. Η Angular περιμένει και ελέγχει τη στιγμή που τα δεδομένα θα γίνουν διαθέσιμα και τότε εμφανίζει αυτόματα το διάγραμμα.

7.3 Node-Red

Το Node-Red παρέχει ένα γραφικό περιβάλλον χρήστη, όπου οι χρήστες μπορούν να προσθέσουν κόμβους μέσω drag-and-drop, οι οποίοι αντιπροσωπεύουν εξαρτήματα ενός μεγαλύτερου συστήματος, συνήθως τις συσκευές, πλατφόρμες λογισμικού και υπηρεσίες Internet, που θα συνδεθούν μεταξύ τους. Περισσότεροι κόμβοι μπορούν να προστεθούν αναμεταξύ σε αυτά τα εξαρτήματα για να αντιπροσωπεύσουν συναρτήσεις οι οποίες

μετασχηματίζουν τα δεδομένα στην μεταφορά. Παρακάτω φαίνεται η αναπαράσταση ενός προγράμματος “Hello World” με Node-Red:



Εικόνα 20 Hello World σε Node-Red (Heath, 2014)

https://www.techrepublic.com/article/node-red/

Στην εικόνα φαίνεται το γραφικό περιβάλλον του Node-Red και πως η πλατφόρμα σπάει συστήματα στα συστατικά τους μέρη. Κάθε ένα από αυτά τα blocks που φαίνονται είναι ένας κόμβος, ο οποίος είναι μία οπτική αναπαράσταση ενός block από κώδικα JavaScript σχεδιασμένο να εκτελεί μία συγκεκριμένη εργασία.

Ο μπλε κόμβος είναι ένας κόμβος inject, ο οποίος κάνει inject μία μεταβλητή στην ροή. Στο συγκεκριμένο παράδειγμα το string “Hello”. Στην συνέχεια υπάρχει ένας κόμβος συνάρτησης ο οποίος προσθέτει το string “ World” σε οποιοδήποτε μήνυμα λάβει. Αυτοί οι δύο κόμβοι ενώνονται μεταξύ τους. Τέλος ενώνεται ένας κόμβος debug, ο οποίος τυπώνει το μήνυμα που λαμβάνει στο παράθυρο debug. Τελικά βλέπουμε το string “Hello World” στο παράθυρο debug. Αυτό το πρόγραμμα είναι ένα παράδειγμα ροής στο Node-Red.

Το ενδιαφέρον του Node-Red είναι ότι μπορεί να κάνει πολύ περισσότερα από το να τυπώνει ένα απλό μήνυμα, καθώς μπορεί επίσης να ενώσει υπηρεσίες Internet και hardware, πράγμα στο οποίο είναι ιδιαίτερα αποτελεσματικό. Ένα παράδειγμα τέτοιας χρήσης του node-red, είναι αυτό ενός φωτογράφου και υπάλληλου της IBM, του Dom Bramley, ο οποίος φωτογραφίζει τον νυχτερινό ουρανό, και ήθελε να βρει λύση στο πρόβλημα του, ότι ο εξοπλισμός του δεν είναι αδιάβροχος αλλά έπρεπε να μείνει έξω την νύχτα. Με την βοήθεια του Node-Red, έφτιαξε ένα προσωπικό σύστημα ειδοποίησης καιρού. Δημιούργησε μία ροή, η οποία έπαιρνε δεδομένα

από το forecast.io κάθε τρία λεπτά για να ελέγχει για προβλέψεις βροχής. Ένας δεύτερος κόμβος έπαιρνε τα δεδομένα που επέστρεφε το forecast.io και υπολόγιζε πόσα λεπτά μέχρι να φτάσει η βροχή σπίτι του. Τέλος χρησιμοποιούσε λάμπες στο σπίτι του για να ειδοποιηθεί για την βροχή.

Αυτό το παράδειγμα επιδεικνύει πως το Node-Red επιτρέπει στους δημιουργούς να βασιστούν σε εργασίες άλλων δημιουργών. Για παράδειγμα, για να πάρει δεδομένα από το forecast.io ο Bramley χρησιμοποίησε έναν υπάρχον κόμβο, ο οποίος απαιτεί χρήση HTTP get requests, αλλάζοντας μόνο το URL και προσθέτοντας ένα string, το οποίο ήταν το query του. Με την σειρά του, το σύστημα ειδοποίησης καιρού του Bramley είναι διαθέσιμο για άλλους να το εισάγουν στο Node-Red και να το χρησιμοποιήσουν όπως κρίνουν κατάλληλα, και ακόμα και αν δεν χρειάζονται ολόκληρη την ροή, μπορούν απλά να χρησιμοποιήσουν κομμάτια αυτής.

Νέοι κόμβοι για αλληλεπίδραση με hardware, λογισμικό και υπηρεσίες Internet προστίθενται συνεχώς. Επίσης η εγγραφή ενός κόμβου δεν είναι ιδιαίτερα δύσκολη, καθώς ένας κόμβος είναι απλά κώδικας JavaScript που εκτελείται σε περιβάλλον node.js. Υπάρχει ήδη μία ενεργή κοινότητα που παράγει νέους κόμβους και η πλατφόρμα Node-Red είναι ένα open source λογισμικό, όποτε ο καθένας έχει τη δυνατότητα να συνεισφέρει. Είναι επίσης εφικτό να μετατραπεί μία ροή σε έναν κόμβο ώστε να είναι ευκολότερο να χρησιμοποιηθεί.

Το Node-Red είναι μία εφαρμογή βασισμένη σε node.js, μία server side πλατφόρμα JavaScript που χρησιμοποιεί ένα μοντέλο εισόδου/εξόδου μη αποκλεισμού που βασίζεται σε συμβάντα, κατάλληλο για την δημιουργία εφαρμογών real time με πολλά δεδομένα που εκτείνονται σε κατανεμημένες συσκευές. Άρα για να είναι εφικτό να λειτουργήσει το Node-Red σε κάποια συσκευή απαιτείται μία βιβλιοθήκη node.js ή ένα module ικανό να αλληλοεπιδράσει με αυτό. Ωστόσο, ο αριθμός των συμβατών υπηρεσιών και συσκευών είναι εκτενής, εν μέρη χάρις την τάση για πολλές μοντέρνες πλατφόρμες να εκθέτουν τα δεδομένα και την λειτουργικότητα μέσω ανοιχτών APIs, τα οποία αποδέχονται HTTP requests. Τέλος το Node-Red, επειδή είναι βασισμένο στο node.js μπορεί να τρέξει εύκολα ακόμα και σε λιγότερο δυνατές συσκευές, όπως το Raspberry Pi.

8

Πρότυπη υλοποίηση και τεχνικές επίλυσης

8.1 Στόχος της Διπλωματικής

Ο στόχος της διπλωματικής μας είναι να δημιουργήσουμε μία εφαρμογή που θα κάνει εύκολη και χωρίς να χρειάζεται καθόλου ή να χρειάζεται ελάχιστη σύνταξη κώδικα από τον χρήστη την εύρεση της καλύτερης πηγής δεδομένων (από άποψη ταχύτητα ανταπόκρισης) από πολλές πηγές (μπορεί γενικά να χρησιμοποιηθεί για την εύρεση του ταχύτερου συστήματος ή εφαρμογής ανάμεσα σε δύο ή περισσότερα).

Εμείς δημιουργήσαμε δύο APIs σε μορφή REST με HTTP κόμβους σε Node-Red και παρέχουμε εύκολα τα δεδομένα από την καλύτερη πηγή αν επιθυμεί ο χρήστης και ακόμα απλά το ποια προκύπτει ότι είναι η καλύτερη πηγή. Εμείς για παράδειγμα χρησιμοποιήσαμε δύο πηγές δεδομένων καιρού να συγκρίνουμε.

Για την επιλογή της κατάλληλης πηγής χρησιμοποιήθηκαν δύο μετρικές, ο μέσος όρος και ο μέσος όρος πρόσφατων μετρήσεων με συντελεστές βαρύτητας.

8.1.1 Μέσος όρος

Η μία μετρική που χρησιμοποιήθηκε είναι ο μέσος όρος της πηγής από όλες τις μετρήσεις που έχουμε μαζέψει και έχουμε αποθηκεύσει στη βάση δεδομένων. Μέσω αυτού μπορούμε να συμπεράνουμε ποια είναι η καλύτερη πηγή στην πλειοψηφία των περιπτώσεων. Βέβαια δεν λαμβάνει υπόψιν του τις αποκλίσεις, οι οποίες μπορεί να είναι αρκετά μεγάλες και να κάνουν την πηγή να μην αξίζει να χρησιμοποιηθεί.

8.1.2 Μέσος όρος πρόσφατων μετρήσεων με συντελεστές βαρύτητας

Η δεύτερη μετρική που αποφασίστηκε να χρησιμοποιηθεί είναι ο μέσος όρος πρόσφατων μετρήσεων με συντελεστές βαρύτητας. Αποφασίσαμε να χρησιμοποιήσουμε τέσσερεις

συντελεστές βαρύτητας, ο κάθε ένας για κάθε ένα τέταρτο των μετρήσεων. Οι πιο πρόσφατες μετρήσεις έχουν τον μεγαλύτερο συντελεστή βαρύτητας και μειώνεται όσο πηγαίνουμε στις πιο παλιές.

Αυτή η μετρική μας δείχνει πόσο καλά ανταποκρίνεται η πηγή δεδομένων των τελευταίο καιρό, κάτι το οποίο είναι σημαντικό, γιατί μπορεί για οποιονδήποτε λόγο η πηγή να μην λειτουργεί όσο καλά λειτουργούσε στο παρελθόν, ή να λειτουργεί αρκετά καλύτερα. Αυτό δεν μπορούμε να το καταλάβουμε εύκολα από τον μέσο όρο όλων των μετρήσεων, γιατί αν έχουμε έναν πολύ μεγάλο αριθμό μετρήσεων, δεν θα αλλάξει σημαντικά από τις καινούριες.

8.1.3 Availability

Αξίζει να σημειωθεί ότι το availability είναι μία πολύ σημαντική μετρική μίας πηγής δεδομένων, αφού από μία πηγή απαιτούμε να λειτουργεί σχεδόν πάντα όταν χρειαζόμαστε δεδομένα. Το availability μας δείχνει σε τι ποσοστό ένα σύστημα λειτουργεί όταν του ζητηθεί.

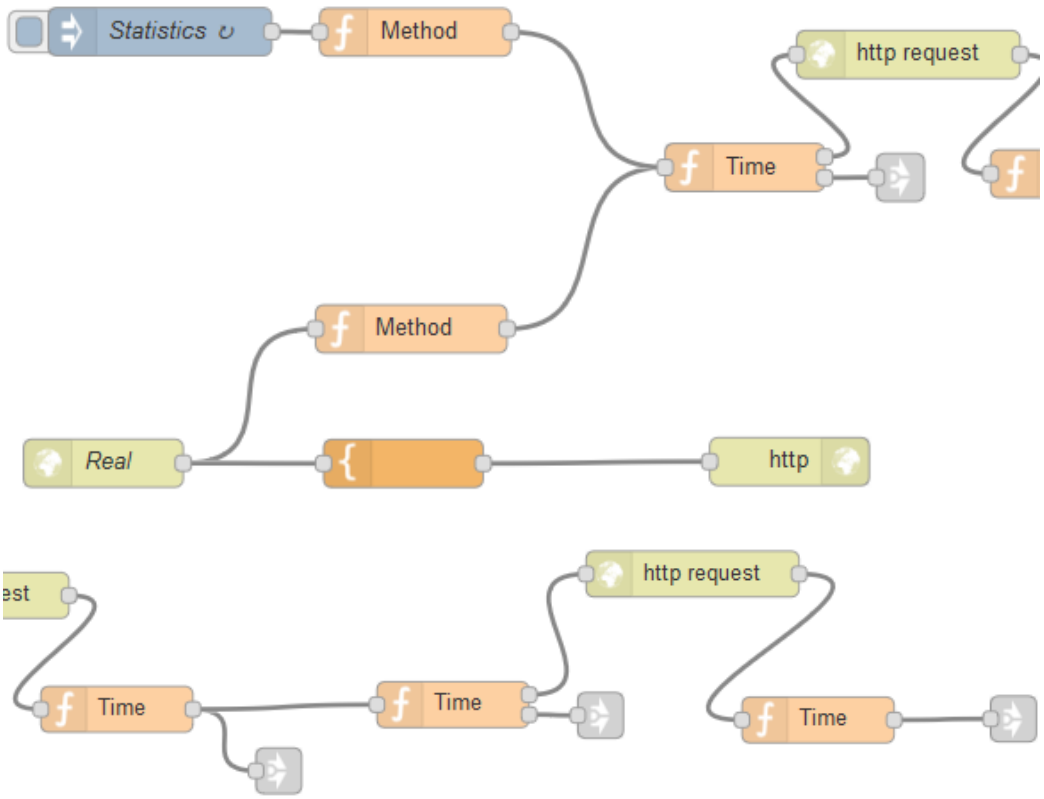
Εμείς αποφασίσαμε τελικά να μην χρησιμοποιήσουμε το availability, επειδή από τις μετρήσεις που μαζέψαμε μας προέκυψε ότι οι πηγές λειτουργούσαν πάντα, δηλαδή επέστρεφαν αποτέλεσμα στο 100% των περιπτώσεων, και άρα δεν θα βοηθούσε στο να αποφανθούμε ποια πηγή είναι καλύτερη (προφανώς το availability δεν είναι 100%, αλλά είναι πολύ μεγάλο ώστε με τις μετρήσεις που συλλέξαμε να μην φαίνεται ότι η πηγή αποτυγχάνει).

Όμως για να προσπεράσουμε το πρόβλημα με το availability, αφού δεν το παίρνουμε υπόψιν μας, βάλαμε όταν μία πηγή καθυστερεί αρκετά ή δεν επιστρέφει αποτελέσματα ή επιστρέφει λανθασμένα αποτελέσματα (συντακτικά όχι νοηματικά), να καλείται η δεύτερη πηγή σαν εναλλακτική.

8.2 Συλλογή μετρήσεων

Για την εύρεση της καλύτερης πηγής δεδομένων συλλέξαμε αρκετούς χρόνους απόκρισης και από τις δύο πηγές και τις αποθηκεύαμε σε μία βάση δεδομένων MongoDB. Αυτό υλοποιήθηκε με δύο ροές οι οποίες θα εξηγηθούν παρακάτω.

8.2.1 *Ροή συλλογής μετρήσεων*



Εικόνα 21 Ροής συλλογής μετρήσεων

Για την συλλογή μετρήσεων χρησιμοποιήθηκε το πάνω κομμάτι αυτής της ροής που ξεκινάει από τον κόμβο inject με όνομα “Statistics”. Το κομμάτι που ξεκινάει από τον κόμβο HTTP “Real” θα εξηγηθεί στην συνέχεια.

Ο κόμβος inject ανά τακτά χρονικά διαστήματα ξεκινάει την ροή ώστε να συλλέγουμε τιμές. Στην συνέχεια ο χρόνος για τον κόμβο που επιθυμούμε να μετρήσουμε υπολογίζεται με τον εξής τρόπο:

Προσθέσαμε δύο επιπλέον κόμβους συνάρτησης όπως φαίνεται και στις εικόνες, έναν αμέσως πριν τον κόμβο πηγής και έναν αμέσως μετά, και σε αυτούς τους κόμβους βρίσκουμε την ακριβή ημερομηνία εκείνης της στιγμής σε ms, και μετά υπολογίζουμε την διαφορά αυτών των δύο τιμών, που είναι και ο χρόνος (σε ms) για τον οποίο εκτελείται ο ενδιάμεσος κόμβος. Οι κόμβοι έχουν δηλαδή τον παρακάτω κώδικα:

Function

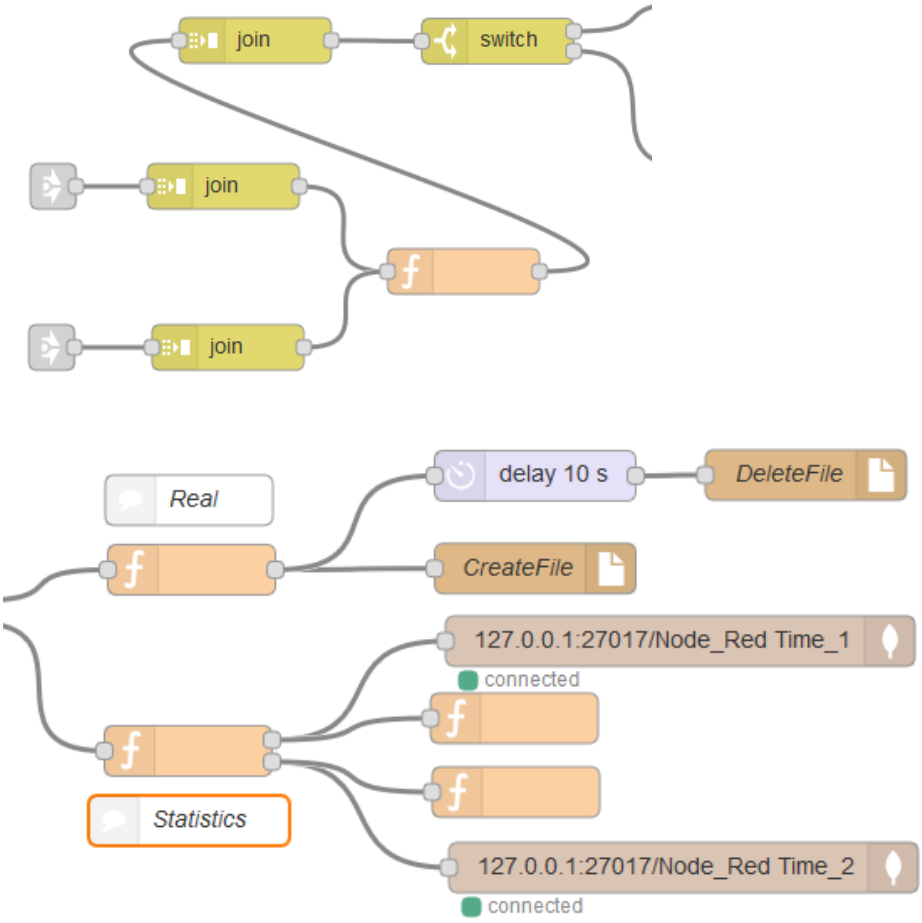
```
1
2  var time = {payload: {Time: new Date().valueOf()}};
3
4  return [msg, time];
```

Εικόνα 22 Κώδικας του κόμβου Time

Οι γκρι κόμβοι μετά τους κόμβους συνάρτησης “Time” είναι τύπου link και στέλνουν τις τιμές στην επόμενη ροή για να υπολογιστεί η διαφορά και να αποθηκευτεί. Ο πρώτος κόμβος “Time” έχει τον κώδικα που αναφέρθηκε παραπάνω. Ο δεύτερος έχει την διαφορά ότι προσθέτει την τιμή που επιστρέφει ο HTTP κόμβος στο payload καθώς και ένα όνομα για να ξέρουμε σε ποιο κόμβο αντιστοιχεί κάθε τιμή. Ο HTTP κόμβος είναι ο κόμβος που χρησιμοποιούμε για να μετρήσουμε την ανταπόκριση του. Οι κόμβοι “Method” χρησιμοποιούνται απλά για να ξέρουμε αν η ροή ξεκίνησε από τον κόμβο “Statistics” ή από τον “Real”.

Ο χρόνος των δύο πηγών θα μπορούσε να υπολογίζεται και παράλληλα αντί σειριακά, αλλά αποφασίσαμε να τα κάνουμε σειριακά για να υπολογίζεται ο χρόνος όσο πιο ακριβής γίνεται και να μην υπάρχει καμία καθυστέρηση λόγο διαμοιρασμό πόρων του υπολογιστή σε διαφορετικές εργασίες.

8.2.2 Ροή αποθήκευσης μετρήσεων



Εικόνα 23 Ροή αποθήκευσης μετρήσεων

Για να υπολογιστεί ο χρόνος, όταν είναι έτοιμες και οι δύο τιμές, χρησιμοποιήθηκαν κόμβοι τύπου join (όπως φαίνεται και στην εικόνα), οι οποίοι περιμένουν να παραλάβουν στην είσοδο τους ένα συγκεκριμένο αριθμό μηνυμάτων.

Όταν και οι δύο τιμές φτάσουν στον κόμβο join η ροή συνεχίζει, και μετά με έναν ακόμα κόμβο συνάρτησης υπολογίζεται η διαφορά. Ο επόμενος κόμβος join περιμένει να υπολογιστούν οι τιμές και για τις δύο πηγές. Ο κόμβος switch στην συνέχεια ελέγχει αν η ροή ξεκίνησε από τον κόμβο "Statistics" ή από τον κόμβο "Real" για να συνεχίσει στο αντίστοιχο κομμάτι. Το κάτω κομμάτι είναι για την αποθήκευση των μετρήσεων στην βάση. Οι κόμβοι που βρίσκονται στο ίδιο επίπεδο με τις βάσεις δεδομένων είναι οι κόμβοι που ανανεώνουν τον μέσο όρο (θα εξηγηθεί παρακάτω). Ο κόμβος πριν από αυτά, μετατρέπει απλώς τα δεδομένα στη μορφή που χρειάζεται ώστε να αποθηκευτούν στη βάση δεδομένων. Για τους κόμβους βάσεις βάλαμε τα παρακάτω στοιχεία: Node_Red είναι το όνομα της βάσης μας και Time_1 το όνομα της συλλογής για τις τιμές του πρώτου κόμβου που μετράμε (ανάλογα Time_2 για τον δεύτερο).

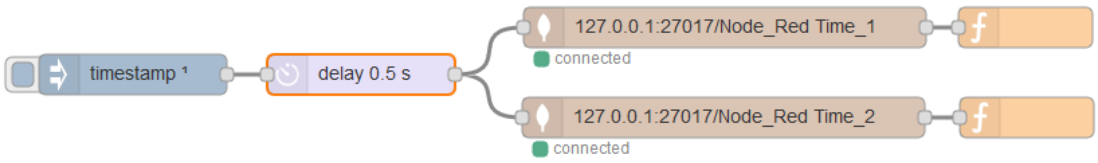
8.3 *Εύρεση στατιστικών*

Για την εύρεση την καλύτερης πηγής χρησιμοποιήσαμε δύο διαφορετικούς τρόπους. Ο πρώτος τρόπος χρησιμοποιεί σαν μετρική το μέσο όρο από όλες τις τιμές που υπάρχουν στην βάση και όλες τις καινούριες που συλλέγονται και τελικά η πηγή με τον χαμηλότερο θεωρείται η καλύτερη.

Ο δεύτερος τρόπος χρησιμοποιεί σαν μετρική τον μέσο όρο των τελευταίων 1000 μετρήσεων (χρησιμοποιήσαμε 1000 γιατί δεν είχαμε πολύ μεγάλο αριθμό μετρήσεων, με έναν μεγαλύτερο αριθμό μετρήσεων θα χρησιμοποιούταν μεγαλύτερος αριθμός όπως για παράδειγμα οι μετρήσεις του τελευταίου μήνα) και βάζουμε στις τιμές συντελεστές με τις πιο πρόσφατες τιμές να έχουν μεγαλύτερο συντελεστή. Χρησιμοποιήθηκαν τέσσερεις συντελεστές και χωρίσαμε τις μετρήσεις σε τέσσερα ίσα σετ σε αυτούς τους τέσσερεις. Οι τιμές των συντελεστών είναι από τις πιο παλιές μετρήσεις στις πιο πρόσφατες: $w_1 = 0{,}4$, $w_2 = 0{,}3$, $w_3 = 0{,}2$, $w_4 = 0{,}1$

8.3.1 *Μέσος όρος*

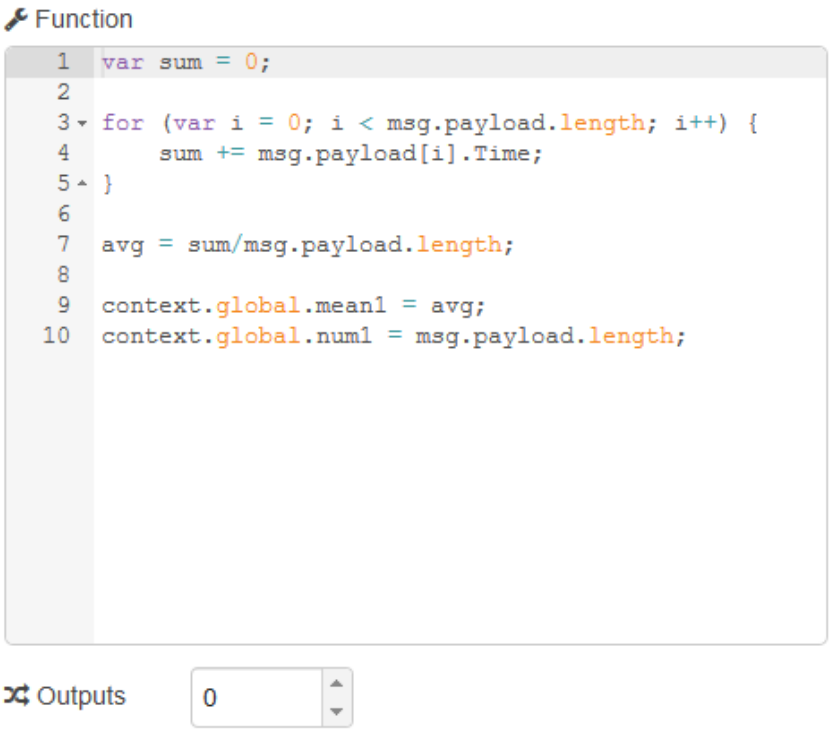
Για να βρούμε τον μέσο όρο χρησιμοποιούμε την παρακάτω ροή:



Εικόνα 24 Ροή εύρεσης πρώτη φορά του συνολικού μέσου όρου

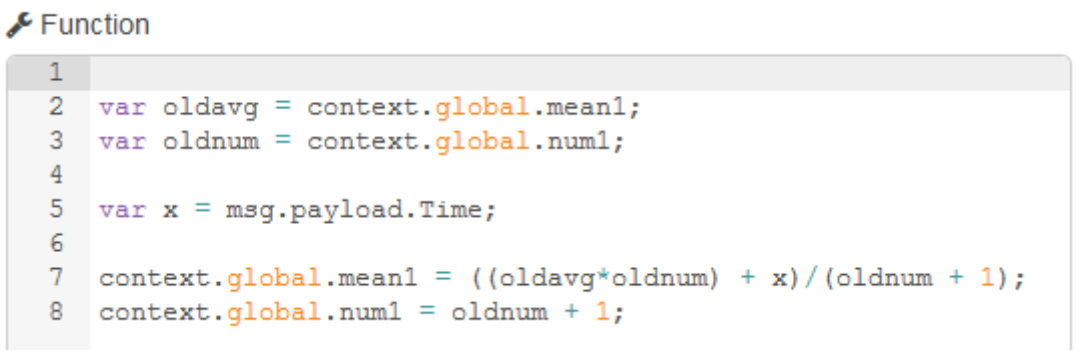
Η ροή αυτή χρησιμεύει ώστε να υπολογίσει μία φορά στην αρχή τον συνολικό μέσο όρο από την βάση δεδομένων τη στιγμή που ξεκινάει το Node-Red. Δίνουμε καθυστέρηση 0,5 δευτερολέπτων έτσι ώστε οι κόμβοι που χρησιμοποιούνται να έχουν αρκετό χρόνο να λάβουν

τις αρχικές τους τιμές. Οι συναρτήσεις μετά τους κόμβους της βάσης Mongo είναι οι παρακάτω:



Εικόνα 25 Κώδικας κόμβου συνάρτησης υπολογισμού του μέσου όρου

Η συνάρτηση αυτή δεν επιστρέφει κάποια έξοδο, αλλά την χρησιμοποιούμε απλά για να θέσουμε μεταβλητές global (δηλαδή μεταβλητές που μπορούν να χρησιμοποιηθούν από οποιονδήποτε κόμβο σε οποιαδήποτε ροή). Η global μεταβλητή mean1 είναι ο μέσος όρος και num1 είναι ο αριθμός των μετρήσεων που έχουμε. Η συνάρτηση αυτή χρησιμοποιείται για την συλλογή ένα, και αντίστοιχα για την συλλογή δύο, με τη μόνη διαφορά ότι στις μεταβλητές global, αντί για τον αριθμό ένα, έχουμε τον αριθμό δύο. Η τιμή του μέσου όρου ανανεώνεται κάθε φορά που λαμβάνουμε μία νέα μέτρηση και δεν απαιτείται εκ νέου πρόσβαση σε ολόκληρες τις βάσεις. Η συνάρτηση βρίσκεται στο σημείο όπου πραγματοποιείται η αποθήκευση στις βάσεις (αναφέρθηκε και προηγουμένως) και είναι η παρακάτω:

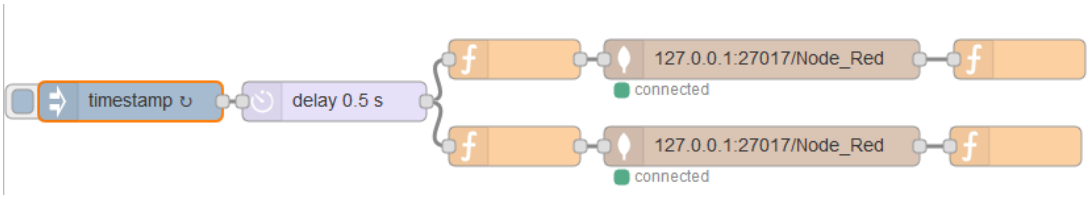


Εικόνα 26 Κώδικας ανανέωσης του μέσου όρου με την νέα τιμή

Παρατηρούμε ότι ούτε αυτή η συνάρτηση δεν επιστρέφει κάποια έξοδο, απλά ανανεώνει τις global μεταβλητές.

8.3.2 Μέσος όρος με συντελεστές βαρύτητας

Για την εύρεση της μετρικής του μέσου όρου των πρόσφατων μετρήσεων με χρήση συντελεστών βαρύτητας, χρησιμοποιείται η παρακάτω ροή:



Εικόνα 27 Ροή υπολογισμού του μέσου όρου με βάρη

Η καθυστέρηση υπάρχει για τον ίδιο λόγο που αναφέρθηκε προηγουμένως, ώστε δηλαδή οι κόμβοι να έχουν χρόνο να λάβουν αρχικές τιμές. Οι συναρτήσεις πριν τους κόμβους για τις βάσεις, χρησιμοποιούνται για να θέσουμε επιλογές για την αναζήτηση στην βάση, όπως το όριο των τιμών που θα επιστραφούν και τον τρόπο ταξινόμησης. Επίσης, χρησιμοποιούνται και για να τεθεί σε ποια συλλογή θα εκτελεστεί το query. Ο κώδικας είναι ο παρακάτω:

```
1
2  msg.collection = "Time_1";
3  msg.sort = {_id: -1};
4  msg.limit = 1000;
5
6  return msg;
```

Εικόνα 28 Κόμβος συνάρτησης με τις ρυθμίσεις για το query στην βάση

Οι συναρτήσεις μετά τους κόμβους βάσης χρησιμοποιούνται για την εύρεση της τιμής του μέσου όρου με συντελεστές βαρύτητας. Ο κώδικας φαίνεται παρακάτω:

Function

```
1  var sum = 0;
2  var L = msg.payload.length;
3
4  for (var i = 0; i < L; i++) {
5      sum += msg.payload[i].Time*(0.5 - (0.1*Math.floor((i + (L/4))/(L/4))));
6  }
7
8  avg = sum/(L/4);
9
10 context.global.weighted1 = avg;
```

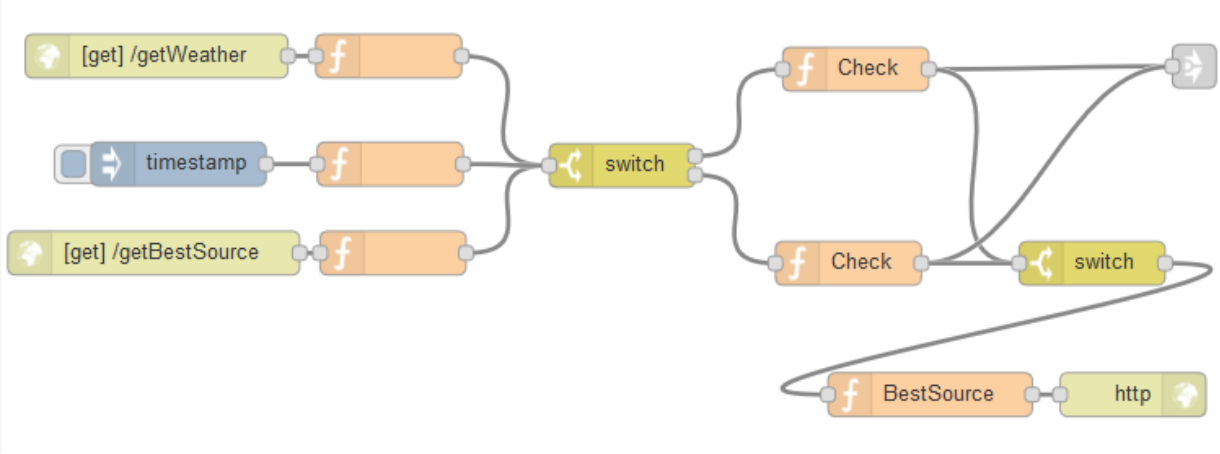
Εικόνα 29 Κώδικας κόμβου συνάρτησης υπολογισμού μέσου όρου με βάρη

Το Math.floor υπάρχει ώστε να γίνεται αλλαγή συντελεστή για τις διαφορετικές τιμές. Αφαιρούμε κάθε φορά από το 0,5 την τιμή της παρένθεσης (για μικρές τιμές του i, δηλαδή για πρόσφατες τιμές, ο πολλαπλασιαστής είναι 0,4 και ανά L/4 μειώνεται κατά 0,1). Το άθροισμα

διαιρείται τελικά με το άθροισμα των συντελεστών που προκύπτει ίσο με L/4 (επειδή έχουμε τέσσερεις διαφορετικούς συντελεστές με άθροισμα 1 και ο καθένας αντιστοιχεί σε L/4 τιμές) όπου το L είναι ο συνολικός αριθμός τιμών που πήραμε από την βάση. Με τους παραπάνω τρόπους υπολογίζονται οι μετρικές που χρησιμοποιούνται στην ροή η οποία επιλέγει την καλύτερη πηγή.

8.4 *Δημιουργία των APIs*

Για να ελέγξουμε ποιος από τους δύο τρόπους θα χρησιμοποιηθεί, χρησιμοποιείται κόμβος switch, ο οποίος, ανάλογα με κάποιο κριτήριο, στέλνει την συνέχεια της ροής σε κάποια έξοδο του. Χρησιμοποιήθηκε η παρακάτω ροή:



Εικόνα 30 Ροή εύρεσης της καλύτερης πηγής

Ο κόμβος getWeather χρησιμοποιείται για την δημιουργία του API που επιστρέφει τον καιρό από την πηγή που προτιμήθηκε, ανάλογα με την μέθοδο ελέγχου, και ο κόμβος getBestSource χρησιμοποιείται για την δημιουργία του API για να επιστραφεί το όνομα της καλύτερης πηγής, ανάλογα πάλι με την μέθοδο ελέγχου.

Οι κόμβοι συνάρτησης αμέσως μετά, που δεν έχουν όνομα έχουν την παρακάτω μορφή:

Function

```
1
2  context.global.return = "Weather";
3
4  return msg;
```

Εικόνα 31 Κώδικας του κόμβου συνάρτησης

Η μεταβλητή return χρησιμοποιείται έτσι ώστε, στη συνέχεια να γίνει έλεγχος για το τι να επιστρέψει, στο αίτημα get που πραγματοποιήθηκε. Μετά τον κόμβο inject η μεταβλητή return παίρνει τιμή “nothing” έτσι ώστε να μην επιστρέψει τίποτα. Ο κόμβος switch, όπως αναφέρθηκε, ελέγχει την μέθοδο και συνεχίζει ανάλογα την ροή. Οι κόμβοι check κάνουν τον

έλεγχο για να εντοπίσουν, ανάλογα με τον τρόπο που χρησιμοποιείται, ποια είναι η καλύτερη πηγή, και έπειτα θέτουν το URL στην ανάλογη πηγή. Παρακάτω φαίνεται ο κώδικας:

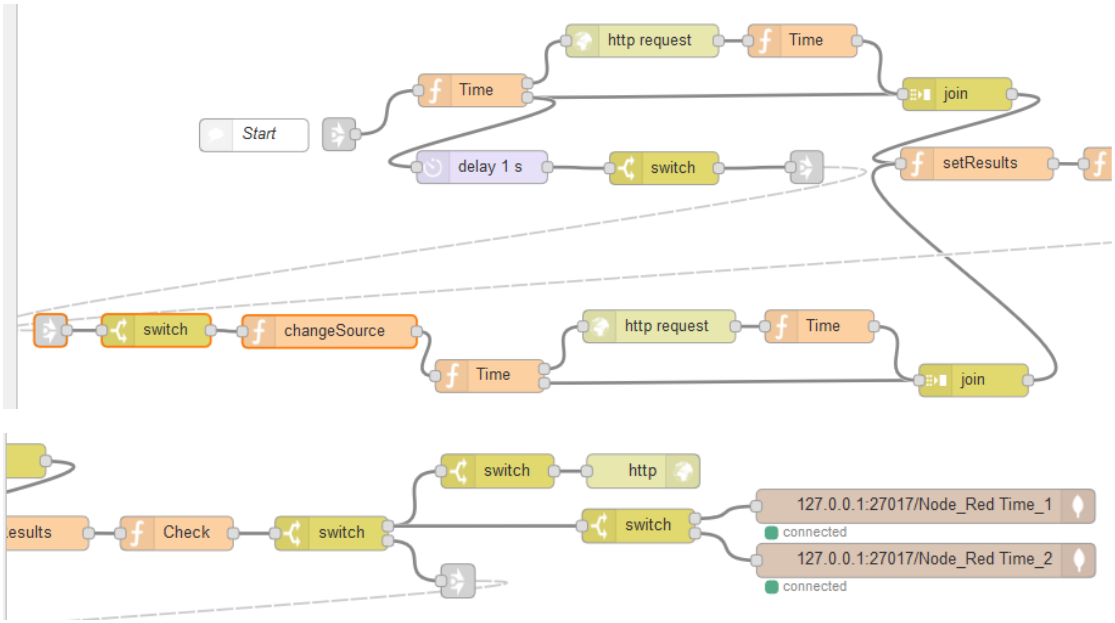
Εικόνα 32 Κώδικας του κόμβου συνάρτησης Check

Ανάλογα, ο δεύτερος κόμβος check ελέγχει για τον δεύτερο τρόπο (δηλαδή αλλάζουν οι μεταβλητές mean).

Ο δεύτερος κόμβος switch υπάρχει μόνο για να ελέγχει την τιμή της μεταβλητής return, και αν είναι weather συνεχίζει, αλλιώς σταματάει η ροή. Τέλος ο κόμβος συνάρτησης BestSource απλά βρίσκει και επιστρέφει το όνομα της καλύτερης πηγής, και ο HTTP out το επιστρέφει εκεί που πραγματοποιήθηκε get.

Το κομμάτι της ροής από τον κόμβο HTTP “getBestSource” ως τον κόμβο HTTP out δίπλα στην συνάρτηση “BestSouce” επαρκεί για το API που επιστρέφει το όνομα της καλύτερης πηγής.

Από τον γκρι κόμβο link συνεχίζει η ροή και καλείται η πηγή που επιλέχτηκε. Στην συνέχεια πραγματοποιείται το HTTP request στην επιλεγμένη πηγή. Υπολογίζεται ο χρόνος όπως και προηγουμένως. Αν υπάρχει μεγάλη καθυστέρηση, τότε καλείται η εναλλακτική πηγή ώστε να μην καθυστερεί η εφαρμογή (θέσαμε για παράδειγμα τιμή 1 δευτερόλεπτο). Στο τέλος ελέγχεται αν η πηγή επέστρεψε όντως σωστά αποτελέσματα (δηλαδή τον καιρό), και αν ναι, τότε επιστρέφεται η τιμή με HTTP out, αν η ροή ξεκίνησε με HTTP get στο getWeather, διαφορετικά απλά αποθηκεύεται η τιμή στην βάση μας. Αν η πηγή δεν επέστρεψε λογικά αποτελέσματα, τότε καλείται η εναλλακτική πηγή. Η ροή φαίνεται παρακάτω:



Εικόνα 33 Ροή εκτέλεσης της επιλεγμένης πηγής

Η ροή ξεκινάει από τον κόμβο link δίπλα στο σχόλιο Start. Οι κόμβοι "Time" είναι όμοιοι με προηγουμένως ενώ επιτελούν ορισμένους παραπάνω ελέγχους. Το link κάτω αριστερά υπάρχει για την περίπτωση που χρειαστεί να αλλάξει η πηγή και το switch ελέγχει αν έγινε ξανά αλλαγή ώστε να μην συνεχίσει (για την αλλαγή πηγής χρησιμοποιείται ξεχωριστή ροή, επειδή αν καθυστερήσει αρκετά η πρώτη, μπορεί να υπάρξει πρόβλημα στον κόμβο HTTP request). Το switch, μετά την καθυστέρηση ενός δευτερολέπτου, ελέγχει αν η πηγή μας επέστρεψε αποτελέσματα ή αν καθυστερεί. Το "setResults" θέτει απλώς τις τιμές στο msg.payload με τον τρόπο που επιθυμούμε. Το check εκτελεί έλεγχο για το αν επιστράφηκε σωστό αποτέλεσμα (δηλαδή ο καιρός) και για να αποφανθεί αν θα αλλάξει η πηγή ή όχι. Το switch πριν τον κόμβο HTTP out χρησιμεύει, όπως και προηγουμένως, για να ελέγξει αν η ροή ξεκίνησε από τον κόμβο HTTP "getWeather". Τέλος, το switch πριν τους κόμβους της βάσης δεδομένων, χρησιμεύει ώστε να ελεγχθεί ποια πηγή επέστρεψε αποτέλεσμα για να αποθηκευτεί στην ανάλογη συλλογή.

Το κομμάτι από την προηγούμενη ροή από τον κόμβο HTTP "getWeather" ως τον κόμβο HTTP out στην ροή της εικόνας 24 αποτελεί τον τρόπο που επιστρέφεται στο API ο καιρός.

Τα APIs δημιουργήθηκαν ώστε να είναι εύκολα διαθέσιμα τα αποτελέσματα της εφαρμογής μας να χρησιμοποιηθούν και να ενταχθούν σε άλλες εφαρμογές, και να μην χρειάζεται πολύπλοκος προγραμματισμός.

8.5 Οπτικοποίηση αποτελεσμάτων

Επιπρόσθετα, παρέχεται η δυνατότητα στο χρήστη να δει σε διάγραμμα τις τιμές σε πραγματικό χρόνο. Για το διάγραμμα χρησιμοποιήθηκε AngularJs με το πακέτο FusionCharts. Αυτό

χρησιμεύει για την δημιουργία διαγραμμάτων JavaScript για το web ή για κινητά. Το πακέτο αυτό διατίθεται δωρεάν για μη εμπορική χρήση με watermark. Όπως αναφέρθηκε προηγουμένως για να ενώσουμε το Node-Red με την AngularJs χρησιμοποιήθηκε απλά ένας κόμβος template με HTML, ο οποίος κάνει ανακατεύθυνση κατευθείαν στο URL της Angular όπως φαίνεται παρακάτω:

Template Syntax Highlight: HTML

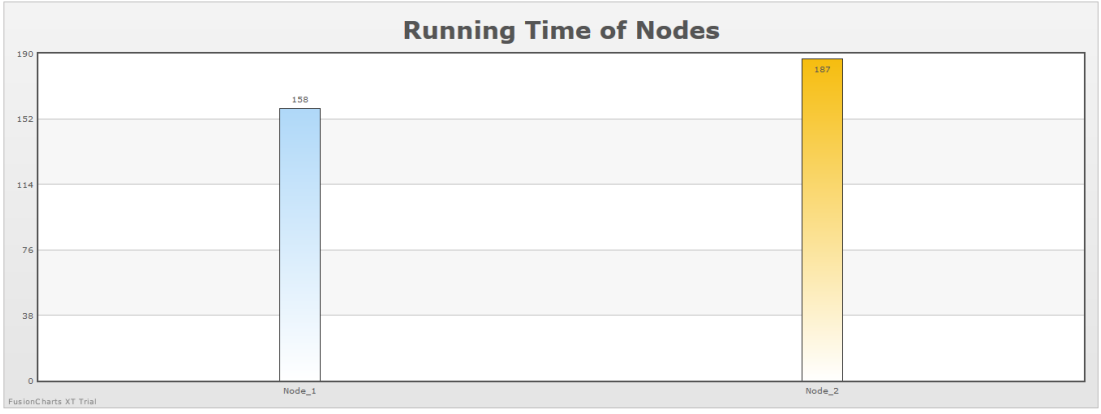
```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta http-equiv="Refresh" content="0; url=http://localhost:8000">
5      </head>
6
7      <body>
8      </body>
9  </html>
```

Εικόνα 34 HTML για ανακατεύθυνση στην AngularJs

Αυτό είναι το κομμάτι της ροής στην εικόνα 21 που ξεκινάει από το “Real”.

Η angular περιμένει να υπολογιστούν οι τιμές εκείνη την στιγμή ελέγχοντας αν δημιουργήθηκε ένα αρχείο με τα στοιχεία στον φάκελο που είναι εγκατεστημένη. Αυτό πραγματοποιείται στο πάνω κομμάτι του switch στην ροή της εικόνας 23.

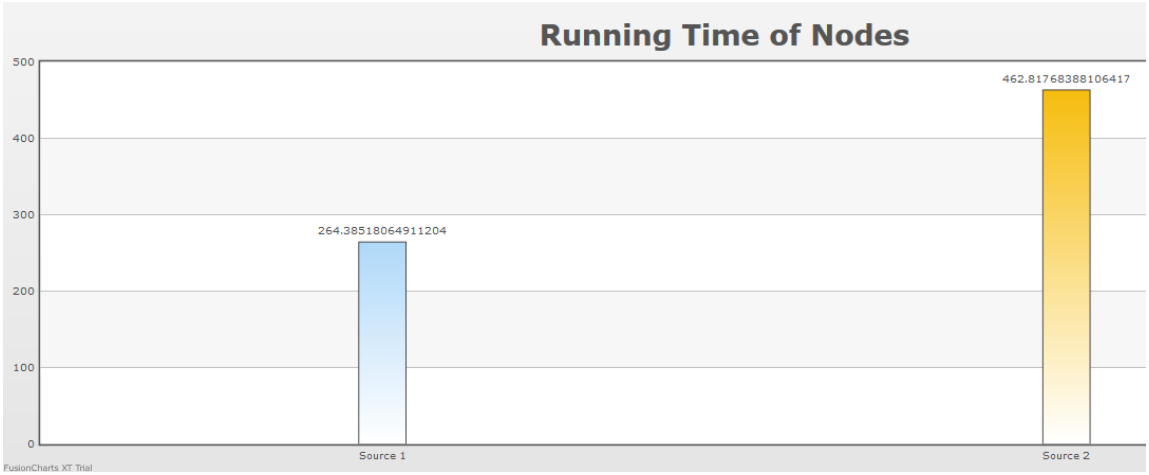
Ο κόμβος συνάρτησης πριν το CreateFile υπάρχει για να μετατρέπει τα δεδομένα στη μορφή που απαιτείται από το πακέτο FusionCharts. Ο κόμβος CreateFile απλά δημιουργεί το αρχείο στον φάκελο της Angular και ο κόμβος DeleteFile διαγράφει αυτό το αρχείο, έτσι ώστε όταν χρειαστεί να δημιουργηθεί ξανά το διάγραμμα να μην υπάρχουν άλλα παλιά δεδομένα. Το delay χρησιμεύει στο να δώσει χρόνο στην angular να διαβάσει το αρχείο. Το διάγραμμα έχει την παρακάτω μορφή:



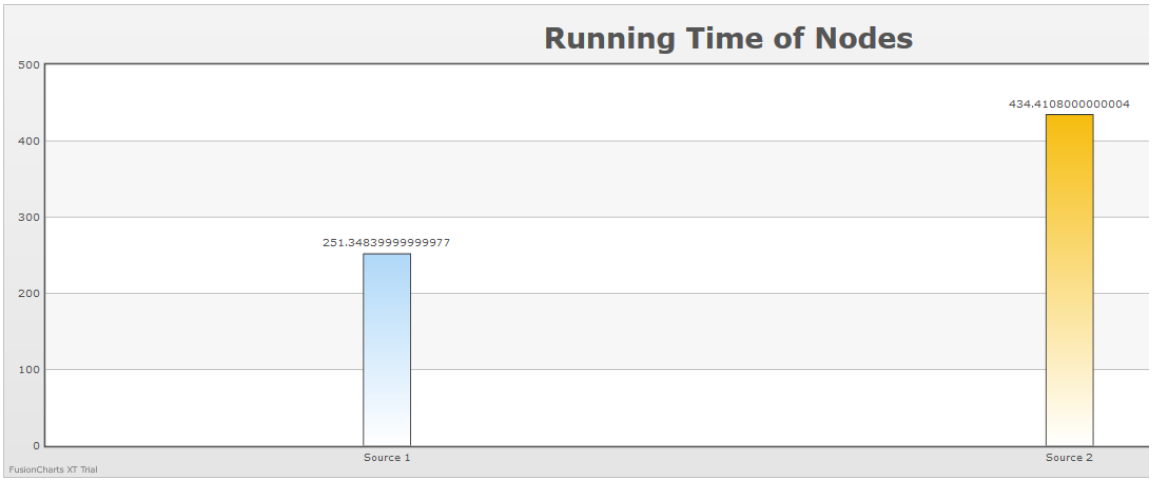
Εικόνα 35 Διάγραμμα χρόνου εκτέλεσης των δύο πηγών σε real time

Χρησιμοποιήθηκε επίσης η ίδια εφαρμογή Angular, για δημιουργία γραφικών παραστάσεων για την αναπαράσταση των δύο μετρικών:

Μέσος Όρος:

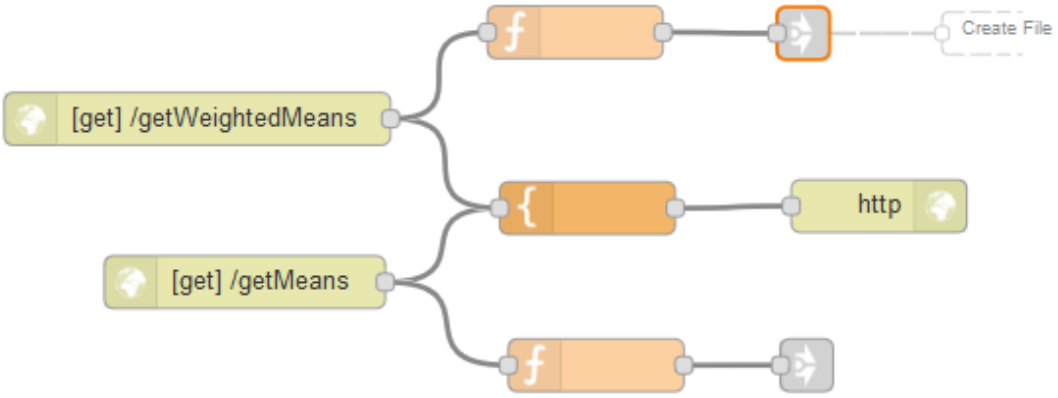


Μέσος όρος πρόσφατων μετρήσεων με συντελεστές βαρύτητας:



Βλέπουμε ότι οι δύο τιμές από τις δύο μετρικές είναι αρκετά κοντά και των δύο τιμών αλλά δεν είναι ίδιες. Αυτό δεν είναι ανάγκη να ισχύει πάντα και μπορεί να υπάρχουν μεγάλες αποκλίσεις.

Οι δύο αυτές γραφικές παραστάσεις δημιουργήθηκαν με την παρακάτω ροή:

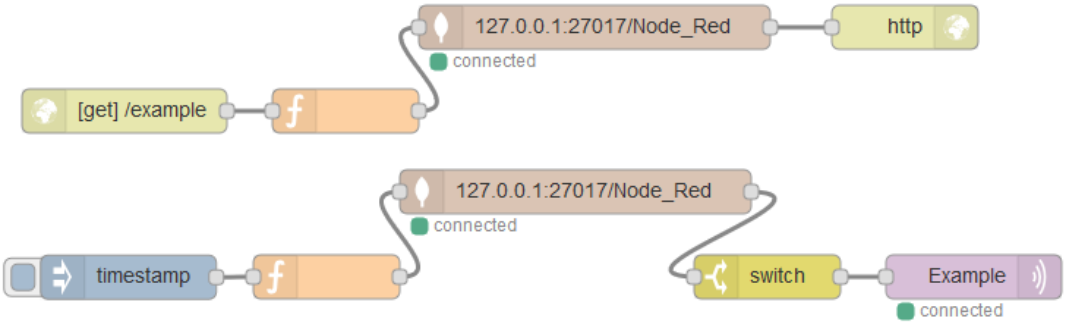


Η λογική είναι η ίδια και με την γραφική για τιμές σε πραγματικό χρόνο. Οι κόμβοι link οδηγούν στην ροή της εικόνας 23 στους κόμβους "CreateFile" και delay. Οι κόμβοι συναρτήσεων απλά θέτουν τα δεδομένα που θέλουμε για τις γραφικές παραστάσεις στην μορφή που χρειάζεται το FusionCharts.

Τα διαγράμματα χρησιμοποιήθηκαν, για να έχει την επιλογή ο χρήστης αν θέλει να δει τα αποτελέσματα, επειδή είναι ωραιότερη και βολικότερη η εμφάνισή τους σε διαγράμματα από το να επιστρεφόντουσαν απλά αριθμοί σαν μεταβλητές, κυρίως αν οι πηγές ήταν περισσότερες και έτσι οι αριθμοί θα ήταν πολλοί.

8.6 *Εναλλακτικός τρόπος αποστολής μέσω MQTT*

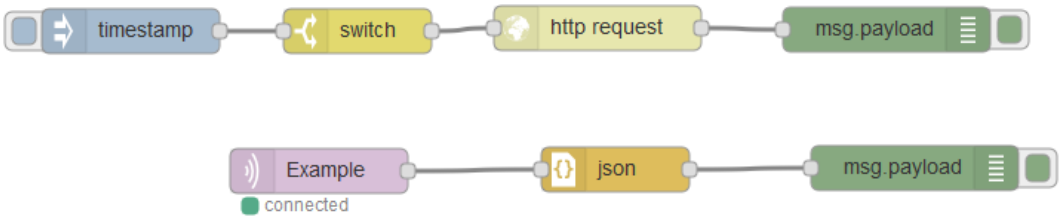
Τέλος, δοκιμάσαμε επίσης να στέλνουμε δεδομένα και με MQTT αντί με HTTP κόμβους. Χρησιμοποιήθηκαν οι παρακάτω ροές:



Εικόνα 36 Ροές αποστολής δεδομένων με HTTP και MQTT

Αυτές οι ροές χρησιμεύουν για την αποστολή των δεδομένων. Θέσαμε, για παράδειγμα, να αποστέλλεται το τελευταίο στοιχείο της συλλογής Time_1 από την βάση δεδομένων. Το switch πριν τον κόμβο MQTT χρησιμεύει ώστε να ελέγχει αν είναι επιλεγμένος ο αντίστοιχος τρόπος αποστολής.

Για το HTTP ο έλεγχος πραγματοποιείται στον client, όπως φαίνεται και παρακάτω:



Εικόνα 37 Ροές προσομοίωσης του client που δέχεται τα δεδομένα

Για να προσομοιώσουμε τον client τρέχουμε ένα δεύτερο Node-Red σε μία διαφορετική port με τον παρακάτω τρόπο:

node-red --settings C:\users\panag\Desktop\Node\settings.js --userDir C:\users\panag\Desktop\Node flows.json

Για να το κάνουμε αυτό δημιουργήσαμε ένα νέο φάκελο, ώστε να αποθηκεύονται εκεί τα δεδομένα του νέου Node-Red και αντιγράψαμε το αρχείο settings.js από το Node-Red. Σε αυτό αλλάξαμε το port σε 1890 αντί 1880:

```
module.exports = {
    // the tcp port that the Node-RED web server is listening on
    uiPort: process.env.PORT || 1890,

    // By default, the Node-RED UI accepts connections on all IPv4 interfaces.
    // The following property can be used to listen on a specific interface. For
    // example, the following would only allow connections from the local machine.
```

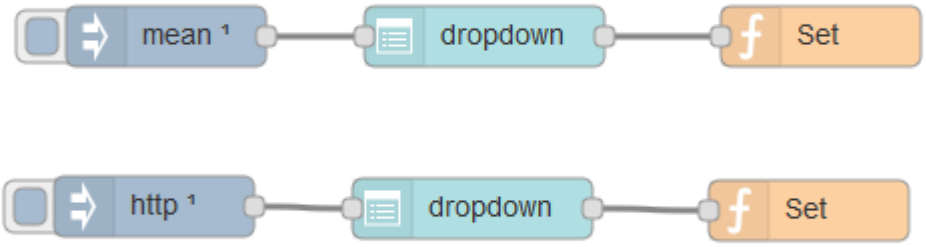
Εικόνα 38 Αλλαγή πόρτας στις ρυθμίσεις για το δεύτερο Node-Red

Θεωρούμε ότι ο client γνωρίζει τον τρόπο που του αποστέλλονται τα δεδομένα κάθε στιγμή, ώστε να ελέγχεται στο switch. Εμείς το αλλάζαμε και στα δύο Node-Red με user interface.

Όπως βλέπουμε η επιλογή για την αποστολή των δεδομένων με MQTT είναι επιλογή της πηγής, ενώ για την αποστολή μέσω REST η επιλογή είναι του client. Αυτό φαίνεται και από την χρήση των switch κόμβων αναγκαστικά σε εκείνα τα σημεία που είναι. Ένας κόμβος switch στην πηγή δεν θα είχε ιδιαίτερο νόημα στην χρήση HTTP κόμβων γιατί απλά θα εμπόδιζε τον client από το να πάρει τα δεδομένα όταν προσπαθούσε να κάνει get. Από την άλλη ένας κόμβος switch μετά τον MQTT κόμβο στον client θα είχε νόημα, γιατί θα μπορούσε να ελέγξει ο πελάτης πότε θα δέχεται τα δεδομένα, αλλά και πάλι οι χρόνοι αποστολής θα ήταν επιλογή της πηγής. Άρα η επιλογή μας να μεταφέρουμε εύκολα τα αποτελέσματα μας μέσω REST HTTP κόμβους είναι λογική, επειδή εμείς θέλουμε τα αποτελέσματα να μπορεί να τα χρησιμοποιήσει ο χρήστης όποτε αυτός επιθυμεί.

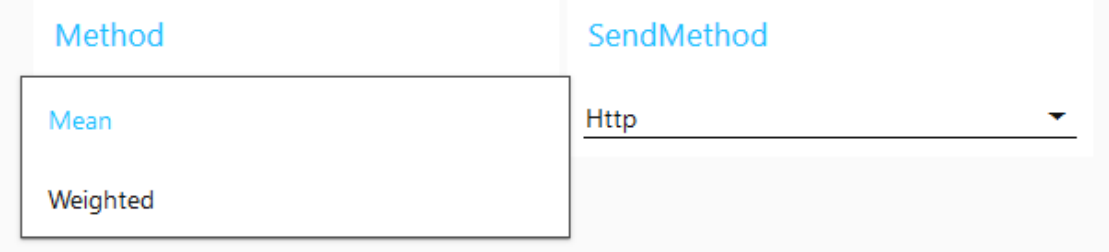
8.7 User interface

Για να είναι εύκολη η εναλλαγή μεθόδων, και για τον τρόπο εύρεσης καλύτερης πηγής δεδομένων και στον τρόπο αποστολή υλοποιήθηκε ένα user interface με τις παρακάτω ροές:



Εικόνα 39 Ροές υλοποίησης user interface

Το user interface που προκύπτει φαίνεται παρακάτω:



Εικόνα 40 User interface

Αυτό το μενού υλοποιήθηκε με κόμβο τύπου dashboard, οι οποίοι κόμβοι χρησιμοποιούνται για την δημιουργία ενός user interface ώστε να είναι ευκολότερη η σύνδεση με τον χρήστη.

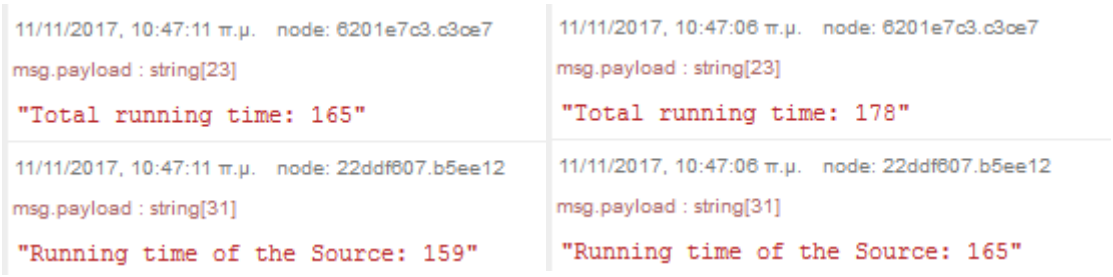
Οι κόμβοι inject υπάρχουν ώστε να θέτουν μία αρχική επιλογή και με τις συναρτήσεις "Set" θέτουμε τις global μεταβλητές στις επιλογές που επιστρέφουν τα dropdown μενού. Οι global μεταβλητές χρησιμοποιούνται ώστε να ελέγχονται τα αντίστοιχα switch.

8.8 Συγκριτική μελέτη

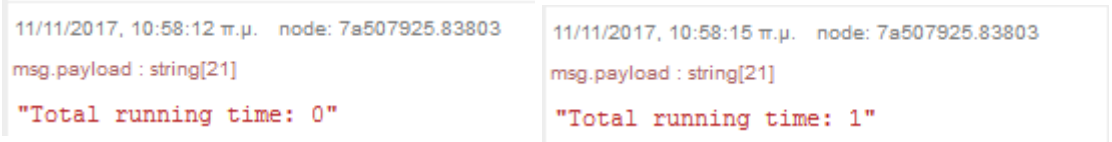
Μετά από πολλές εκτελέσεις του API που επιστρέφει τον καιρό παρατηρούμε ότι η εφαρμογή μας λειτουργεί πολύ γρήγορα, και δεν καθυστερεί σχεδόν καθόλου την λήψη των δεδομένων από την πηγή, σε σχέση με το να γινόταν η πρόσβαση κατευθείαν στην πηγή. Παρατηρούμε ότι ο παραπάνω χρόνος που χρειάζεται για να γίνει ο έλεγχος από την εφαρμογή μας και να γίνει πρόσβαση στην επιλεγμένη πηγή και να επιστρέψει τα αποτελέσματα στον χρήστη είναι περίπου 10ms.

Παρακάτω φαίνονται μερικά παραδείγματα:

```
11/11/2017, 10:46:51 π.μ.   node: 6201e7c3.c3ce7
msg.payload : string[23]
"Total running time: 173"
11/11/2017, 10:46:51 π.μ.   node: 22ddf607.b5ee12
msg.payload : string[31]
"Running time of the Source: 165"
```



Από την άλλη παρατηρούμε ότι αν ο χρήστης επιθυμεί να μάθει μόνο το όνομα της καλύτερης πηγής τα αποτελέσματα επιστρέφονται σχεδόν απευθείας, με μηδαμινή καθυστέρηση. Συγκεκριμένα βλέπουμε μετά από αρκετές επαναλήψεις ότι ο χρόνος που χρειάζεται είναι το πολύ 1ms. Παρακάτω φαίνονται παραδείγματα χρήσης:



Ο χρόνος που χρειάζεται ώστε να υπολογιστούν οι μετρικές δεν επηρεάζουν τον χρόνο που χρειάζεται για να ανταποκριθεί η εφαρμογή αν την χρειαστεί ο χρήστης. Αν και ο μέσος όρος υπολογίζεται πολύ γρήγορα ταυτόχρονα με κάθε νέα μέτρηση, πέρα από την πρώτη φορά που θα χρειαστεί μία πρόσβαση στην βάση δεδομένων. Από την άλλη ο μέσος όρος με συντελεστές υπολογίζεται λίγο πιο αργά καλώντας την βάση δεδομένων, αλλά χρησιμοποιεί μόνο πρόσφατες μετρήσεις και για αυτό δεν χρειάζεται υπερβολικό χρόνο. Όμως και πάλι δεν επηρεάζεται ο χρήστης από αυτήν την πρόσβαση γιατί γίνεται στο επίπεδο της εφαρμογής και δεν επηρεάζει την λειτουργία της.

9

Επίλογος

9.1 Σύνοψη και συμπεράσματα

Στην διπλωματική μας δημιουργήσαμε μία εφαρμογή, η οποία μπορεί να βρει την καλύτερη πηγή δεδομένων ανάμεσα σε όσες της δώσει ο χρήστης, και να κάνει αυτόματα την καλύτερη επιλογή χωρίς κάποια επέμβαση του χρήστη, πέρα από το να θέσει τις επιλογές που θέλει να ελεγχθούν, ενώ τα αποτελέσματα είναι άμεσα διαθέσιμα μέσω APIs. Η επιλογή για την καλύτερη πηγή δεδομένων γίνεται με δύο διαφορετικούς τρόπους για να μπορεί να επιλεχθεί ο κατάλληλος ανάλογα με τις απαιτήσεις της εκάστοτε εφαρμογής.

9.2 Μελλοντικές επεκτάσεις

Η εφαρμογή που δημιουργήσαμε, είναι εφικτό να επεκταθεί με τη χρήση διαφορετικών μετρικών, όπως το availability που αναφέρθηκε ξανά, ή και άλλες, οι οποίες θα είναι πιο περίπλοκες και θα δίνουν καλύτερα αποτελέσματα, σε σχέση με αυτές που χρησιμοποιήσαμε. Επίσης, είναι δυνατόν να γίνει χρήση του OpenWhisk μαζί με το Node-Red, το οποίο είναι μία severless πλατφόρμα που δεν χρησιμοποιεί πόρους όταν δεν εκτελεί κάποια ενέργεια, και επίσης παρέχει την δυνατότητα χρήσης άλλων γλωσσών, πέρα από JavaScript που χρησιμοποιείται από το Node-Red. Με αυτό τον τρόπο μπορούμε να κάνουμε την εφαρμογή περισσότερο οικονομική, καθώς και να εκτελέσουμε περίπλοκες λειτουργίες που διαφορετικά δε θα μπορούσαμε.

10

Βιβλιογραφία

[1] IEEE Internet Initiative, Towards a definition of the Internet of Things (IoT), May 2015 from https://iot.ieee.org/images/files/pdf/IEEE_IoT_Towards_Definition_Internet_of_Things_Revision1_27MAY15.pdf

[2] Dave Evans, The Internet of Things How the Next Evolution of the Internet Is Changing Everything, April 2011 from https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

[3] https://www.techopedia.com/definition/28247/internet-of-things-iot

[4] https://www.sas.com/en_us/insights/big-data/what-is-big-data.html

[5] Ashley DeVan, The 7 V's of Big Data, April 2017 from https://www.impactradius.com/blog/7-vs-big-data/

[6] https://www.i-scoop.eu/internet-of-things-guide/#What_is_the_Internet_Of_Things_IoT

[7] Kai Wähner, IoT trends, March 2017 from https://www.voxxed.com/2017/03/iot-trends/

[8] MongoDB site https://www.mongodb.com/

[9] AngularJs site https://angularjs.org/

[10] Node-Red site https://nodered.org/

[11] Cynthia Harvey, Big Data Challenges, June 2017, from https://www.datamation.com/big-data/big-data-challenges.html

[12] NIST: National Institute of Standards and Technology, The NIST Definition of Cloud Computing from http://faculty.winthrop.edu/domanm/csci411/Handouts/NIST.pdf

[13] John Esposito, The Programming Challenges of IoT, August 2014 from https://dzone.com/articles/programming-challenges-iot

[14] Cloud Computing Challenges In 2017 from http://www.opencirrus.org/cloud-computing-challenges-2017/

[15] https://seekingalpha.com/article/4069233-general-electric-internet-things-awesome-match

[16] https://iot-analytics.com/iot-market-forecasts-overview/