ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΜΕΤΑΠΤΥΧΙΑΚΟ ΔΙΠΛΩΜΑ
ΕΙΔΙΚΕΥΣΗΣ (MSc)
στα ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

"ΚΙΟΤ: Εκτέλεση συναρτήσεων και βελτιστοποίηση υλοποίησης σε Java"

Γεώργιος Στυλιανόπουλος
ΜΜ4160014

Επιβλέπων: Γεώργιος Ξυλωμένος, Αν . Καθηγητής

ΑΘΗΝΑ, ΦΕΒΡΟΥΑΡΙΟΣ 2020

Περίληψη

Σκοπός της παρούσας διπλωματικής εργασίας είναι η επέκταση του πρωτοκόλλου δρομολόγησης KIOT και η βελτίωση της υφιστάμενης υλοποίησης σε Java.

Στο σύγχρονο κόσμο των διασυνδεδεμένων συσκευών (IoT) είναι απαραίτητη η επικοινωνία μεταξύ των κόμβων για την αποστολή και λήψη δεδομένων. Καθώς το πλήθος των συσκευών αυξάνεται, τόσο εντός σπιτιού όσο και εκτός, η επικοινωνία με τις συσκευές πραγματοποιείται κάνοντας χρήση της διεύθυνσης IP τους, γεγονός το οποίο καθιστά πρακτικά δύσκολη τη δημιουργία ιεραρχικών δεδομένων για την πραγματοποίηση επερωτήσεων αναφορικά με τα δεδομένα των συσκευών.

Το πρωτόκολλο δρομολόγησης KIOT αναπτύχθηκε για τη δρομολόγηση μηνυμάτων μεταξύ IOT συσκευών με κύρια χαρακτηριστικά τη δρομολόγηση με βάση τα “tags” στα δεδομένα των κόμβων που συμμετέχουν σε ένα δίκτυο και τη χρήση αυτών για την αποστολή και λήψη δεδομένων. Κατά αυτό τον τρόπο καθίσταται ευκολότερη η πραγματοποίηση αιτημάτων προς τα δεδομένα όπως επίσης και η δρομολόγηση των πακέτων.

Για την επέκταση του πρωτοκόλλου δρομολόγησης μπορεί πλέον να υποστηριχθεί η πραγματοποίηση πράξεων επί των δεδομένων ενός κόμβου. Έτσι για παράδειγμα το δίκτυο KIOT είναι σε θέση να απαντήσει ερωτήσεις που αφορούν το μέγιστο, το ελάχιστο και το μέσο όρο κάποιας ποσότητας (πχ θερμοκρασία) στα δεδομένα που αποθηκεύονται εντός κάποιου κόμβου.

Για τη βελτιστοποίηση της υπάρχουσας υλοποίησης πραγματοποιήθηκε ανασχεδιασμός του μηχανισμού καταγραφής του συστήματος καθώς αφορά μία πολυνηματική αρχιτεκτονική η οποία είναι από κατασκευής δύσκολη στην αποσφαλμάτωση. Επιπλέον, το σύστημα πλέον δεν είναι εξαρτώμενο από κάποιο εξομοιωτή δικτύου (Mininet) για την εκτέλεσή του, αλλά υποστηρίζει και την εκτέλεση πολλών κόμβων στο ίδιο μηχάνημα. Τέλος, η αρχικοποίηση του συστήματος και των κόμβων βελτιώθηκε και το μεγαλύτερο μέρος των τιμών που απαιτούνται μπορούν πλέον να δοθούν παραμετρικά.

Πίνακας Περιεχομένων

Πίνακας Εικόνων

1. Εισαγωγή

Στις μέρες μας, όλο και περισσότερο βλέπουμε συστήματα και συσκευές, ακόμη και τις πιο παράδοξες, να αποκτούν τον ορισμό «έξυπνες». Πέρα από τα κινητά τηλέφωνα και τους ηλεκτρονικούς υπολογιστές πλέον έξυπνες συσκευές είναι και οι συσκευές καθημερινής οικιακής χρήσης, μέσα μεταφοράς και πολλά άλλα. Όλες οι συσκευές αυτές είναι κεντρικά διαχειριζόμενες από μία εφαρμογή ή ένα κινητό, αλληλοεπιδρούν μεταξύ τους, έχουν τα δεδομένα τους στο cloud, και κάνουν ακόμη και αναβαθμίσεις αυτόματα στο λογισμικό τους και προβλέψεις με σκοπό την καλύτερη εξυπηρέτηση του χρήστη.

Ένα παράδειγμα είναι ότι ανεξάρτητα από την απόσταση που έχει ένας χρήστης από το έξυπνο σπίτι του, την ώρα αλλά και την περιοχή, τα φώτα και η θερμοκρασία του σπιτιού ρυθμίζονται αυτόματα. Ένα ακόμη παράδειγμα για την μετακίνηση είναι ότι από τον τρόπο και την ταχύτητά οδήγησης σε σχέση με το κανονικό, το αυτοκίνητο καταλαβαίνει αν έχει επηρεαστεί ο οδηγός από αλκοόλ και του αφαιρεί τον έλεγχο του αμαξιού και τον σταματά. Αυτά είναι μικρά αλλά πολύ δυνατά παραδείγματα της εποχής Internet of Things (IoT), όπου όλο και περισσότερες καθημερινές συσκευές διασυνδέονται και αυτοματοποιούνται στο πάτημα ενός κουμπιού και το μόνο που χρειάζεται είναι η σύνδεσή τους στο διαδίκτυο.

1.1. Information Centric Networking (ICN)

Ένας τρόπος προσέγγισης και υλοποίησης του IOT είναι το Information Centric Networking (γνωστό ως ICN) [1]. Η συγκεκριμένη προσέγγιση θεωρείται μία από τις πιο υποσχόμενες υλοποιήσεις για τον τρόπο που θα δομηθεί το μελλοντικό διαδίκτυο. Κύρια έμπνευση του ICN αποτελεί ότι το διαδίκτυο χρησιμοποιείτε κυρίως για μετάδοση πληροφοριών και όχι τόσο για την επικοινωνία μεταξύ των κόμβων του δικτύου.

Το ICN στοχεύει στο να αντικατοπτρίζει τόσο τις σημερινές όσο και τις μελλοντικές ανάγκες από της υπάρχουσα αρχιτεκτονική του Διαδικτύου. Αυτό το επιτυγχάνει με την ονομασία πληροφοριών σε επίπεδο δικτύου και έτσι το ICN ευνοεί την ανάπτυξη προσωρινής αποθήκευσης (in-network caching) και γενικά της αποθήκευσης, καθώς και του μηχανισμού ταυτόχρονης μετάδοσης (multicast). Έτσι διευκολύνει την λειτουργία του IoT αλλά και την έγκαιρη και έγκυρη παροχή πληροφορίων στους τελικούς χρήστες. Επίσης τα δεδομένα ανεξαρτητοποιούνται από την τοποθεσία τους, την αποθήκευση αλλά και τα μέσα μεταφοράς τους μέσω της προσωρινής αποθήκευσης που αναφέραμε παραπάνω. Τέλος έχοντας τα παραπάνω ως δεδομένα, αυξάνεται και η αποδοτικότητα στην επικοινωνία μεταξύ κόμβων που αποτελούν πρόκληση.

Παρόλα αυτά η προσέγγιση ICN, είναι πολύ παραπάνω από την διανομή πληροφοριών. Με σχετική έρευνα θα μπορούσε η πληροφορία που συλλέγεται να χρησιμοποιηθεί σαν μέσο αντιμετώπισης πρόσθετων περιορισμών στην τρέχουσα αρχιτεκτονική του διαδικτύου. Ένας τέτοιος περιορισμός θα ήταν η κινητικότητα στην αρχιτεκτονική του διαδικτύου και η διαχείριση και επιβολή της ασφάλειας σε αυτό. Με αυτό το τρόπο θα μπορούσαν να λυθούν κρίσιμα ζητήματα στο μελλοντικό Διαδίκτυο.

1.2. Named Data Networking (NDN)

Μία προτεινόμενη και πολύ διαδεδομένη λύση της προσέγγισης Information Centric Networking είναι η Named Data Networking (NDN) [2]. Το NDN έχει τις ρίζες του σε μια προηγούμενη υλοποίηση η οποία ονομάζεται Content-Centric Networking ή αλλιώς (CCN). Η υλοποίηση CCN έχει παρουσιαστεί για πρώτη φορά από τον Van Jacobson το 2006. Δίνει έμφαση στο περιεχόμενο, καθιστώντας το δίκτυο άμεσα προσπελάσιμο και διαθέσιμο καθώς τα τελικά σημεία επικοινωνούν με βάση ονόματα δεδομένων αντί για διευθύνσεις IP. Το CCN χαρακτηρίζεται από τη βασική ανταλλαγή μηνυμάτων αίτησης περιεχομένου (αποκαλούμενα “Interests”) και μηνυμάτων επιστροφής περιεχομένου (που ονομάζονται “Data”).

Το σχέδιο NDN ερευνά την προτεινόμενη εξέλιξη του Jacobson από την αρχική αρχιτεκτονική δικτύου κεντρικού υπολογιστή σε μία κεντρική βάση δεδομένων NDN. Η πεποίθηση είναι ότι αυτή η εννοιολογικά απλή μετατόπιση θα έχει εκτεταμένες συνέπειες για τον τρόπο με τον οποίο οι άνθρωποι σχεδιάζουν, αναπτύσσουν και χρησιμοποιούν δίκτυα και εφαρμογές.

Το NDN έχει τρεις βασικές έννοιες που το διακρίνουν από άλλες αρχιτεκτονικές δικτύου. Το πρώτο σημείο που το ξεχωρίζει είναι τα ονόματα των δεδομένων των εφαρμογών αλλά και τα ονόματα των δεδομένων χρησιμοποιούνται άμεσα στην προώθηση πακέτων δικτύου. Έτσι οι εφαρμογές καταναλωτών ζητούν τα επιθυμητά δεδομένα από το όνομά τους, επομένως οι επικοινωνίες στο NDN είναι καθοδηγούμενες από τον καταναλωτή.

Το δεύτερο σημείο που διακρίνει την υλοποίηση NDN από τις υπόλοιπες είναι ότι οι επικοινωνίες είναι προστατευμένες. Αποστέλλονται σαν δεδομένα όπου κάθε ένα κομμάτι τους (πακέτο δεδομένων) θα υπογραφεί κρυπτογραφικά από τον παραγωγό του και το ευαίσθητο ωφέλιμο φορτίο ή τα συστατικά του ονόματος μπορούν επίσης να κρυπτογραφηθούν για λόγους προστασίας της ιδιωτικής ζωής. Με αυτόν τον τρόπο, οι καταναλωτές μπορούν να επαληθεύσουν το πακέτο ανεξάρτητα από τον τρόπο λήψης του πακέτου.

Τρίτον το NDN υιοθετεί ένα stateful forwarding πρωτόκολλο όπου αυτοί που προωθούν τα μηνύματα μέσα στο δίκτυο κρατάνε την κατάσταση του πακέτου για κάθε αίτημα (που ονομάζεται πακέτο ενδιαφέροντος, Interest) και θα διαγράψει την κατάσταση αυτή όταν θα επιστρέψει ένα αντίστοιχο πακέτο δεδομένων. Η στρατηγική αυτή επιτρέπει έξυπνη προώθηση και μειώνει σημαντικά τις επαναλήψεις στο δίκτυο.

Η προϋπόθεση του είναι ότι το Διαδίκτυο χρησιμοποιείται πρωτίστως ως δίκτυο διανομής πληροφοριών, το οποίο δεν είναι μια καλή αντιστοιχία για την IP και ότι η το μέλλον του Internet πρέπει να βασίζεται σε δεδομένα και όχι σε αριθμητικά διευθυνόμενους κεντρικούς υπολογιστές.

Η βασική αρχή είναι ότι ένα δίκτυο επικοινωνίας θα πρέπει να επιτρέπει σε ένα χρήστη να επικεντρώνεται στα δεδομένα που χρειάζεται, ονομαζόμενο περιεχόμενο, αντί να χρειάζεται να αναφερθεί μια συγκεκριμένη, φυσική τοποθεσία, όπου τα

δεδομένα αυτά πρόκειται να ανακτηθούν από, ονομαζόμενους κεντρικούς υπολογιστές. Το κίνητρο γι 'αυτό προέρχεται από το γεγονός ότι η συντριπτική πλειοψηφία της τρέχουσας χρήσης του Διαδικτύου αποτελείται από τη διάδοση δεδομένων από μια πηγή σε έναν αριθμό χρηστών.

Το NDN έχει ευρύ φάσμα πλεονεκτημάτων όπως η προσωρινή αποθήκευση περιεχομένου για τη μείωση της κυκλοφοριακής συμφόρησης και η βελτίωση της ταχύτητας παράδοσης, η απλούστερη διαμόρφωση των συσκευών δικτύου και η ασφάλεια στο δίκτυο σε επίπεδο δεδομένων.

1.3. KIOT

Στην αρχιτεκτονική KIOT [3] χρησιμοποιείται μία πιο ευέλικτη ονοματοδοσία η οποία βασίζεται στο TagNet. Η συγκεκριμένη ονοματοδοσία διαφοροποιείται στην σχέση αντιστοίχισης, η οποία πλέον ορίζεται ως μια σχέση υπερσυνόλου μεταξύ ενός συνόλου λέξεων-κλειδιών μεταξύ των κόμβων.

Η δρομολόγηση λαμβάνει χώρα σε ένα δέντρο κόμβων, όπου ο κόμβος ρίζας είναι ο σύνδεσμος προς το δέντρο δρομολόγησης. Ακόμα, όλα τα αιτήματα προς το δέντρο προέρχονται από τον κόμβο ρίζας και οι τελικές απαντήσεις ταξιδεύουν μέχρι να φτάσουν σε αυτόν. Συνεπώς κάθε κόμβος οφείλει να γνωρίζει μόνο τις συνδέσεις που οδηγούν προς τον κόμβο ρίζας για να κατευθύνει τις απαντήσεις.

Για τον εντοπισμό κάθε αντικειμένου και του περιεχόμενού του, χρησιμοποιείται ένα Bloom Filter το οποίο υποδεικνύει τις λέξεις – κλειδιά που σχετίζονται με αυτό. Στην συνέχεια σε κάθε κόμβο του δέντρου διατηρείται ένας κατάλογος με τους συνδέσμους προς τους κόμβους του κατώτερου επιπέδου, δηλαδή μία λίστα με τα Bloom Filters η οποία δείχνει ποια αντικείμενα είναι διαθέσιμα μέσω ενός συνδέσμου.

Σε αντίθεση με το TagNet ένα αίτημα στη συγκεκριμένη αρχιτεκτονική ταιριάζει με ένα αντικείμενο περιεχομένου όταν το Bloom Filter του είναι υποσύνολο του Bloom Filter του αντικειμένου. Αυτό επιτρέπει τη συγχώνευση των Bloom Filters προς την κατεύθυνση του κόμβου ρίζας, με το κόστος της δημιουργίας false positives.

2. Εκτέλεση Συναρτήσεων KIOT

Στα υπάρχοντα συστήματα με βάση το cloud, τα δεδομένα απαιτείται να ταξιδέψουν από όλους τους IoT κόμβους προς τον cloud – server για την εκτέλεση των συναρτήσεων (π.χ. Average). Στην υλοποίηση του KIOT ο κατάλληλος κόμβος για την εκτέλεση των συναρτήσεων είναι ο root κόμβος του δικτύου ακολουθώντας την «Naive» στρατηγική [3].

Οι συναρτήσεις που υλοποιήθηκαν για την τρέχουσα εργασία αφορούν τα ακόλουθα:

- MIN: εύρεση ελαχίστου μεταξύ ενός συνόλου τιμών
- MAX: εύρεση μεγίστου μεταξύ ενός συνόλου τιμών
- AVERAGE: εύρεση μέσης τιμής μεταξύ ενός συνόλου τιμών

Επιπλέον η υλοποίηση θα μπορούσε να επεκταθεί για την υποστήριξη ερωτημάτων στη διάσταση του χρόνου. Για παράδειγμα «Ποια είναι η μέση θερμοκρασία μιας αίθουσας για το μήνα Αύγουστο;».

Η ροή εκτέλεσης μίας συνάρτησης ξεκινάει από τον πελάτη (client) ο οποίος απευθύνει ένα ερώτημα προς το δίκτυο με το αποτέλεσμα εκτέλεσης της συνάρτησης να επιστρέφει σε αυτόν. Συγκεκριμένα ακολουθούνται τα κάτωθι βήματα:

- Δημιουργία ενός πακέτου τύπου «Interest» από τον client και αποστολή του προς τον κόμβο ρίζας του δικτύου.
- Το πακέτο «MSG_INTEREST» περιέχει ένα Bloom Filter με τα tags που περιλαμβάνονται στο query καθώς επίσης και την πράξη που πρέπει να πραγματοποιηθεί πάνω στα δεδομένα. Για παράδειγμα, ένα query θα μπορούσε να είναι {building1, floor1, temperature},{AVERAGE}.
- Ο κόμβος ρίζας προωθεί το πακέτο «MSG_INTEREST» προς τους κόμβους του δικτύου όπου το Bloom Filter τους ταιριάζει με αυτό του query.
- Το πακέτο «Interest» τελικά δρομολογείται προς τους κόμβους – φύλλα του χαμηλότερου επιπέδου του δικτύου (sensors).
- Οι τελικοί κόμβοι απαντούν ο καθένας με ένα πακέτο «MSG_DATA» το οποίο περιλαμβάνει τα ζητούμενα δεδομένα προς την κατεύθυνση του root κόμβου.
- Ο κόμβος ρίζας συλλέγει όλα τα πακέτα δεδομένων και εκτελεί την πράξη υπολογισμού.
- Το αποτέλεσμα της πράξης επιστρέφεται στον client στη μορφή αποτελέσματος.

Στην ακόλουθη Εικόνα 1 φαίνεται ο κώδικας στον root κόμβο που αναλαμβάνει την εκτέλεση της πράξης πάνω στα δεδομένα.

```java
synchronized (data)
{
    OperationType opcode = record.getOperation();
    Address requestAddr = record.getAddress();
    switch (opcode)
    {
        case NOTHING:
            for (int value : data)
            {
                new Sender(new IntData(value), requestAddr.getIp(), requestAddr.getPort(), 1000).start();
                Logger.log("Sent data " + value + " to the guest node: " + requestAddr.getIp());
            }
            break;
        case AVG:
            new Sender(new FloatData(processor.computeMean(data)), requestAddr.getIp(), requestAddr.getPort(), 1000).start();
            Logger.log("Sent the mean to the guest node: " + processor.computeMean(data) + " " + requestAddr.getIp());
            break;
        case MAX:
            new Sender(new IntData(processor.findMax(data)), requestAddr.getIp(), requestAddr.getPort(), 1000).start();
            Logger.log("Sent the max to the guest node: " + requestAddr.getIp());
            break;
        case MIN:
            new Sender(new IntData(processor.findMin(data)), requestAddr.getIp(), requestAddr.getPort(), 1000).start();
            Logger.log("Sent the min to the guest node: " + requestAddr.getIp());
            break;
    }
```

```java
public float computeMean(List<Integer> values)
{
    float mean = 0.0f;
    if (values.size() == 0)
        return mean;
    for (int value : values)
        mean += value;
    return mean/values.size();
}

public int findMax(List<Integer> values)
{
    return Collections.max(values);
}

public int findMin(List<Integer> values)
{
    return Collections.min(values);
}
```

Εικόνα 1: Κώδικας Εκτέλεσης Συναρτήσεων

2.1. Πρόσθετες Στρατηγικές Εκτέλεσης Συναρτήσεων

Κάνοντας την υπόθεση ότι όλοι οι κόμβοι έχουν τη δυνατότητα για τον υπολογισμό συναρτήσεων μία άλλη στρατηγική ονομάζεται «Minimum Transfer» κατά την οποία η εκτέλεση της συνάρτησης πραγματοποιείται στο χαμηλότερο επίπεδο κόμβων στον οποίο τα αιτήματα αρχίζουν και διακλαδώνονται για την αναζήτηση δεδομένων προς του κόμβους – φύλλα. Πλεονέκτημα της συγκεκριμένης στρατηγικής είναι η μείωση του bandwidth καθώς ελαχιστοποιείται η μεταφορά δεδομένων στο

δίκτυο, όμως οι κόμβοι που βρίσκονται κοντά στους sensors θα έχουν λιγότερα διαθέσιμα resources για την εκτέλεση υπολογισμών (π.χ. Raspberry PI).

Μία διαφοροποιημένη στρατηγική είναι η «Least Congested» όπου η εκτέλεση των πράξεων πραγματοποιείται στον κοντινότερο κόμβο προς τους sensors, ο οποίος ωστόσο έχει το μικρότερο φόρτο εργασίας. Η συγκεκριμένη μετρική υπολογίζεται καθώς το query διατρέχει τους κόμβους του δικτύου και σκοπό έχει τη μείωση υπερφόρτωσης των κόμβων.

3. Βελτιστοποίηση Υλοποίησης

Η υπάρχουσα υλοποίηση αποτελείται από δύο εφαρμογές σε Java: η πρώτη εκτελείται σε όλους τους κόμβους του δικτύου (συσκευές IoT) και η δεύτερη προσομοιώνει έναν guest κόμβο ο οποίος κατευθύνει ερωτήματα προς το δίκτυο KIOT. Η guest εφαρμογή συνδέεται απευθείας πάνω στον κόμβο ρίζας του δικτύου με την οποία επικοινωνεί για να απευθύνει τα ερωτήματα προς το δίκτυο [4].

Κατά την εκκίνηση λειτουργίας της εφαρμογής κάθε κόμβος βρίσκει αυτόματα τη θέση του στο δίκτυο ανάλογα με ένα προ-αποφασισμένο σύνολο παραμετροποίησης το οποίο ορίζει σε ποιο επίπεδο βρίσκεται. Στη συνέχεια καθορίζει τον πίνακα δρομολόγησής του και ετοιμάζει το χειρισμό των πακέτων δρομολόγησης από και προς τον ίδιο. Μόνο ο guest κόμβος και οι κόμβοι – φύλλα του δικτύου γνωρίζουν το περιεχόμενο των keywords που χρησιμοποιούνται, καθώς απαιτείται να κωδικοποιηθούν σε BFs. Η ρίζα και όλοι οι εσωτερικού κόμβοι χειρίζονται τα BFs για να κατευθύνουν τα πακέτα προς την ορθή κατεύθυνση του δικτύου.

3.1. Προσομοίωση Δικτύου

Για την προσομοίωση του δικτύου KIOT χρησιμοποιείται το λογισμικό Mininet. Το Mininet δημιουργεί ένα ρεαλιστικό εικονικό δίκτυο, που τρέχει εφαρμογές σε επίπεδο kernel, switch και εφαρμογής, σε ένα μόνο εικονικό μηχάνημα (VM). Το συγκεκριμένο δίκτυο αναλαμβάνει να δημιουργεί το σύνολο των κόμβων που απαρτίζουν το δίκτυο εκτελώντας ένα instance της εφαρμογής και τέλος εκτελεί ένα στιγμιότυπο της guest εφαρμογής. Ένα σύνολο από scripts σε Python χρησιμοποιείται για την παραμετροποίηση των κόμβων και την εκκίνηση λειτουργίας του δικτύου.

Συγκεκριμένα:

- **Script δημιουργίας κόμβων:** Το script λαμβάνει ως όρισμα το φάκελο που θα περιέχει τα αρχεία παραμετροποίησης και δημιουργεί ένα σύνολο αρχείων κάθε ένα από τα οποία ορίζει τα ακόλουθα:
  - **Node Id:** Το αναγνωριστικό του κόμβου
  - **Level:** Η θέση του κόμβου στο δίκτυο
  - **Type:** Ο τύπος του κόμβου, με βάση τον οποίο καθορίζονται και τα δεδομένα που θα περιέχει (πχ αισθητήρας θερμοκρασίας)
  - **Output File:** Το όνομα του αρχείου όπου θα καταγράφονται οι επικοινωνίες του κόμβου με άλλους
  - **Port:** Η πόρτα την οποία θα χρησιμοποιεί ο κόμβος για τις επικοινωνίες
  - **Tags:** Οι λέξεις – κλειδιά για τα δεδομένα του κόμβου, στην περίπτωση που είναι κόμβος στο χαμηλότερο επίπεδο του δικτύου
- **Script εκτέλεσης εφαρμογών:** Το script λαμβάνει ως είσοδο το φάκελο με τις παραμετροποιήσεις των κόμβων και στη συνέχεια χρησιμοποιώντας τις βιβλιοθήκες του Mininet εκκινεί ένα νέο στιγμιότυπο της εφαρμογής του KIOT και του Guest.

Η υλοποίηση της εφαρμογής βασίζεται σε πολυνηματική αρχιτεκτονική η οποία αποτελείται κατά βάση από δύο νήματα που αναλαμβάνουν το ρόλο λήψης και αποστολής πακέτων. Επιπρόσθετα, έχει υλοποιηθεί ένα ακόμα νήμα το οποίο

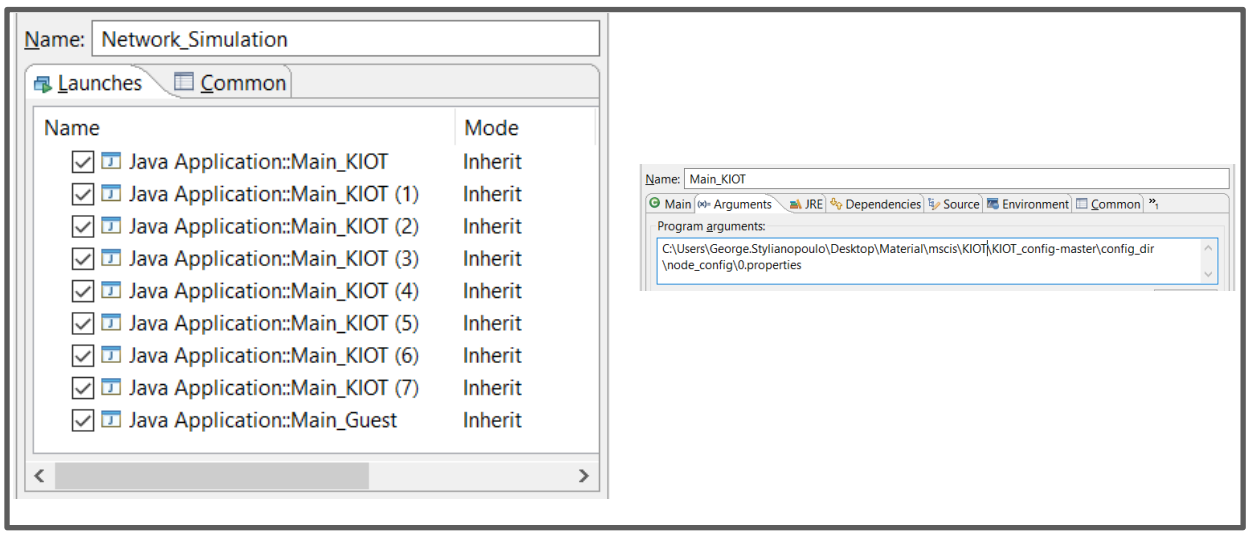
χρησιμοποιείται για την κατανάλωση και επεξεργασία των δεδομένων.

Η συγκεκριμένη υλοποίηση προσδίδει ένα μεγάλο αριθμό πλεονεκτημάτων για την υποστήριξη του πρωτοκόλλου KIOT, όπως είναι η υποστήριξη πολλαπλών παράλληλων λειτουργιών σε ένα κόμβο χωρίς την ύπαρξη καθυστερήσεων και η ασύγχρονη εκτέλεση ταυτόχρονων ενεργειών για την απάντηση αιτημάτων από εξωτερικούς clients. Επιπρόσθετα, η χρήση του λογισμικού Mininet επιτρέπει την παράλληλη εκτέλεση πολλών στιγμιοτύπων της εφαρμογής δημιουργώντας ένα εικονικό δίκτυο, το οποίο θα ήταν δύσκολο να δοκιμαστεί με πραγματικά μηχανήματα. Μπορούν επίσης να δοκιμαστούν διαφορετικές τοπολογίες και χαρακτηριστικά δικτύου.

Από την άλλη πλευρά, το μεγαλύτερο μειονέκτημα της συγκεκριμένης υλοποίησης είναι η δυσκολία αποσφαλμάτωσης και ελέγχου της ορθής λειτουργίας του συστήματος. Ακόμα, η αναγκαία μεταφορά των εκτελέσιμων αρχείων στο περιβάλλον του Mininet δημιουργεί ένα επιπλέον φορτίο δυσκολίας καθώς θα πρέπει τα περιβάλλοντα ανάπτυξης και εκτέλεσης να είναι συγχρονισμένα ως προς τις εκδόσεις λογισμικού που χρησιμοποιούν (π.χ. έκδοση Java). Δηλαδή, ο προγραμματιστής και η εικονική μηχανή του Mininet πρέπει να χρησιμοποιούν τις ίδιες βιβλιοθήκες.

Για την αντιμετώπιση της συγκεκριμένης δυσκολίας αρχικά αντικαταστάθηκε το εικονικό δίκτυο Mininet με την εκτέλεση του συνόλου των εφαρμογών σε ένα μόνο μηχάνημα, στο οποίο πραγματοποιήθηκε και η ανάπτυξη. Για αυτό το σκοπό χρησιμοποιήθηκαν οι δυνατότητες που προσφέρει ο IDE Eclipse με τη δημιουργία πολλαπλών παραμετροποιήσεων εκτέλεσης της ίδιας εφαρμογής και στη συνέχεια την ομαδοποίησή τους σε ένα group.

Για τους σκοπούς δοκιμαστικής εκτέλεσης της εφαρμογής δημιουργήθηκαν αρχικά οι παραμετροποιήσεις των κόμβων του δικτύου χρησιμοποιώντας το python script και στη συνέχεια κάνοντας χρήση του εργαλείου εννιά παραμετροποιήσεις στις οποίες δόθηκαν ως ορίσματα κατά την εκκίνηση τα αρχεία. Στο επόμενο βήμα οι κόμβοι ομαδοποιήθηκαν ώστε να επιτραπεί η ταυτόχρονη εκκίνησή τους, όπως φαίνεται στην Εικόνα 2.



Εικόνα 2: Ομαδοποίηση κόμβων

Αποτέλεσμα της συγκεκριμένης αλλαγής ήταν η βελτιστοποίηση της αποσφαλμάτωσης του κώδικα όπως επίσης και ανίχνευση των αιτημάτων που προκαλούσαν blocks στην εκτέλεση.

3.2. Παραμετροποίηση Συστήματος

Για να είναι εφικτή η ανωτέρω αλλαγή στην τοπολογία του δικτύου και να μην υπάρχουν σφάλματα στην επικοινωνία των κόμβων ήταν απαραίτητη η αλλαγή στον τρόπο που εκκινούν τη λειτουργία τους οι εφαρμογές. Συγκεκριμένα, καθώς πλέον όλοι οι κόμβοι βρίσκονται στο ίδιο μηχάνημα η διεύθυνση επικοινωνίας τους απέκτησε ένα επιπλέον χαρακτηριστικό, εκτός από την IP, που είναι η πόρτα, όπως φαίνεται στην Εικόνα 3.

Κατά την εκκίνηση λειτουργίας της εφαρμογής δεσμεύεται μία πόρτα του τοπικού μηχανήματος η οποία χρησιμοποιείται για να ανοίξει ένα socket μέσω του οποίου πραγματοποιείται το σύνολο μετάδοσης πακέτων από και προς το συγκεκριμένο κόμβο. Το socket αυτό παραμένει δεσμευμένο καθ' όλη τη διάρκεια λειτουργίας της εφαρμογής και χρησιμοποιείται για την αποστολή και λήψη των πακέτων από άλλους κόμβους.

```
NodeID = root
Level = 1
Type = 1
OutputFile = root_log.txt
LoggingMode = 2
Port = 9000
```

```
try {
    prop.load(new FileInputStream(config_file));
    nodeId = prop.getProperty("NodeID");
    level = Integer.parseInt(prop.getProperty("Level"));
    type = Integer.parseInt(prop.getProperty("Type"));
    outputFile = prop.getProperty("OutputFile");
    logMode = Short.parseShort(prop.getProperty("LoggingMode"));
    PORT = Integer.parseInt(prop.getProperty("Port"));
    socket = new DatagramSocket(PORT);
} catch (Exception ex) {
    ex.printStackTrace();
}
```

Εικόνα 3: Παραμετροποίηση Κόμβων

3.3. Καταγραφή

Σημαντικό κομμάτι για την ανάπτυξη της εφαρμογής αποτελεί η αποτύπωση των επικοινωνιών που πραγματοποιεί ο κάθε κόμβος με σκοπό τον εντοπισμό των πακέτων που φτάνουν και φεύγουν προς άλλους.

Το σύστημα υποστηρίζει δύο διαφορετικούς τύπους καταγραφής:

- Console: τα logs αποτυπώνονται στην κονσόλα εκτέλεσης
- Αρχεία: τα logs καταγράφονται σε αρχεία για κάθε κόμβο του συστήματος σύμφωνα με το όνομα που έχει οριστεί στην παραμετροποίηση

Η υποδομή καταγραφής της εφαρμογής βελτιώθηκε σημαντικά αντικαθιστώντας τα μονοπάτια των αρχείων που καταγράφει ο κάθε κόμβος, με μονοπάτια σχετικά ως προς το μονοπάτι εκτέλεσης της εφαρμογής, όπως φαίνεται στην Εικόνα 4.

```
public static String basePath = new File("").getAbsolutePath() + "\\..\\logs";
```

Εικόνα 4: Logging - Relative Path

Επιπλέον τοποθετήθηκαν σε καίρια σημεία στον κώδικα logs, τα οποία βοηθούν σημαντικά στην αναγνώριση κάποιου σφάλματος επικοινωνίας όπως επίσης και στη διαπίστωση σε σφάλματα που αφορούν τα δεδομένα που είναι αποθηκευμένα στον κάθε κόμβο.

4. Βιβλιογραφία

1. Xylomenos, G., et al., "A Survey of Information-Centric Networking Research," in IEEE Communications Surveys & Tutorials, vol. 16, no. 2, pp. 1024-1049, Second Quarter 2014.
2. Afanasyev, A., Burke, J., Refaei, T., Wang, L. and Zhang, B., " A Brief Introduction To Named Data Networking," IEEE MILCOM Conference, 2018.
3. Ascigil, O., Rene, S., Xylomenos, G., Psaras, I. and Pavlou, G., "A Keyword-Based ICN-IoT Platform,", ACM ICN Conference, 2017.
4. Xylomenos, G., Zafeiratos, E. and Prokopakis, M., "Keyword-Based Information Retrieval For The IoT,", ACM/IEEE HoTWoT Workshop, 2019.