ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

ΣΧΟΛΗ ΕΠΙΣΤΗΜΩΝ & ΤΕΧΝΟΛΟΓΙΑΣ ΤΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
SCHOOL OF INFORMATION SCIENCES & TECHNOLOGY

ΜΕΤΑΠΤΥΧΙΑΚΟ ΕΠΙΣΤΗΜΗ ΤΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
MSc IN COMPUTER SCIENCE

ΜΠΣ στην Επιστήμη Υπολογιστών
Διπλωματική Εργασία

Προγραμματισμός συστήματος δρομολόγησης με βάση τις λέξεις-κλειδιά στο Διαδίκτυο των Πραγμάτων

Διονύσιος Σκορδούλης
Επιβλέπων καθηγητής: Γεώργιος Ξυλωμένος

Αθήνα, Δεκέμβριος 2018

Περίληψη

Ο όρος το Διαδίκτυο των Πραγμάτων (Internet of Things, IoT) σημαίνει τη σύνδεση οποιασδήποτε συσκευής στο Internet, αποτελώντας ουσιαστικά ένα όραμα για τη διασύνδεση όλων των "αντικειμένων" στον κόσμο. Η έννοια "Things" (αντικείμενα) αναφέρεται σε μία ευρεία ποικιλία συσκευών εντελώς διαφορετικών μεταξύ τους, όπως αυτοκίνητα με ενσωματωμένους αισθητήρες, κάμερες, κλιματιστικά, φώτα, συστήματα ασφαλείας, smartwatches ακόμα και αυτοκίνητα των οποίων οι περίπλοκοι αισθητήρες εντοπίζουν αντικείμενα στην πορεία τους.

Η Πληροφοριοκεντρική Δικτύωση (Information-Centric Networking, ICN) είναι μια πολλά υποσχόμενη προσέγγιση για το IoT, καθώς η πρόσβαση στο περιεχόμενο στο ICN βασίζεται σε ονόματα δεδομένων και όχι σε διευθύνσεις IP. Η ονομαστική δικτύωση δεδομένων (Named data networking, NDN) είναι μια αρχιτεκτονική που μπορεί να ονομάσει μεμονωμένες τιμές δεδομένων, όπως όλες οι τιμές θερμοκρασίας σε ένα κτίριο, καθώς και ομάδες τιμών, χρησιμοποιώντας ένα σχήμα ιεραρχικής ονομασίας. Ωστόσο, δεν μπορεί να υποστηρίζει πολλές ιεραρχίες ονομάτων με οικονομικό τρόπο, επομένως τα ερωτήματα για τα δεδομένα περιορίζονται να ακολουθούν μία συγκεκριμένη σειρά ομαδοποίησης.

Η δρομολόγηση με βάση λέξεις-κλειδιά (keywords) για το IoT (KIOT) είναι ένα πρωτόκολλο που βασίζεται στην προσέγγιση ICN, και συγκεκριμένα στην αρχιτεκτονική NDN, η οποία βασίζεται στην αποστολή μηνύματος Ενδιαφέροντος (interest) για ονόματα δεδομένων και τη λήψη των αντίστοιχων μηνυμάτων Δεδομένων (data). Το σκεπτικό του KIOT είναι να χρησιμοποιεί σύνολα λέξεων-κλειδιών (keywords) τόσο για τις τιμές των ετικετών (gr, aueb, Antoniadou, floor2, room5, temperature), όσο και για να εκφράζει αιτήματα για δεδομένα, π.χ. (gr, aueb, temperature). Ο στόχος είναι ένα ερώτημα με συγκεκριμένες λέξεις-κλειδιά να εντοπίζει και να επιστρέφει όλες τις τιμές δεδομένων που χαρακτηρίζονται από ένα υπερσύνολο των κλειδιών αυτών.

Η παρούσα εργασία ξεκινά από μία υπάρχουσα υλοποίηση του συστήματος KIOT σε Java και Python, και έχει ως στόχο την αναλυτική τεκμηρίωση της υλοποίησης, την αντιμετώπιση προβλημάτων της υπάρχουσας υλοποίησης, και της προετοιμασίας του συστήματος για περαιτέρω εξέλιξη από μεταγενέστερους μελετητές.

Πίνακας περιεχομένων

Εισαγωγή

Το Internet of Things (IoT) είναι μια έννοια που αφορά τα καθημερινά αντικείμενα μας, από τα βιομηχανικά μηχανήματα μέχρι τις φορητές συσκευές που χρησιμοποιούν ενσωματωμένους αισθητήρες για να συλλέγουν δεδομένα και να ενεργούν πάνω τους μέσα στο δίκτυο. Μία μεγάλη πρόκληση για τους κατασκευαστές ενσωματωμένων συστημάτων IoT είναι η διαχείριση και η ερμηνεία του τεράστιου όγκου πληροφοριών που παράγουν οι συσκευές λόγω της συνεχούς επικοινωνίας με το διαδίκτυο (streaming data).

Στην παρούσα εργασία ασχολούμαστε με μια νέα αρχιτεκτονική δρομολόγησης (KIOT) για το σχεδιασμό πληροφοριοκεντρικών δικτύων αισθητήρων με δυνατότητα IoT, βασισμένο σε σύστημα ονοματοθεσίας με λέξεις-κλειδιά που προσφέρει ένα πλουσιότερο μοντέλο επικοινωνίας για να περιγράψει δεδομένα IoT που είναι διαθέσιμα σε έναν τομέα IoT (π.χ. Τα δεδομένα εντός αυτού του τομέα ορίζονται από ένα μη εξουσιοδοτημένο σύνολο λέξεων-κλειδιών όπως {temperature, aueb, troias, floorl, mmlab} που επιτρέπουν σε κάθε εφαρμογή να χρησιμοποιεί ένα σύνολο λέξεων-κλειδιών που περιγράφουν καλύτερα τα κατάλληλα δεδομένα. Για να επιτευχθεί η δρομολόγηση, προτείνουμε τη χρήση ονομάτων που αποτελούνται από ένα ιεραρχικό πρόθεμα (λέξεις-κλειδιά τοποθεσίας) και ένα σύνολο λέξεων-κλειδιών ως επίθημα, υποδεικνύοντας συγκεκριμένα δεδομένα για αυτόν τον τομέα (λέξεις-κλειδιά τύπου αισθητήρα).

Έτσι, χρησιμοποιείται ένα ευέλικτο σχήμα ονοματοθεσίας που να συνδυάζει ονόματα NDN ως αναγνωριστικά τομέα και λέξεις-κλειδιά για να υποδείξει σύνολα δεδομένων. Στην ουσία, είναι μια υβριδική λύση NDN και δρομολόγησης βασισμένη σε λέξεις-κλειδιά για το IoT. Διατηρούμε τη λειτουργικότητα της ιεραρχικής ονομασίας εισάγοντας λέξεις-κλειδιά τοποθεσίας (περιγραφές πορείας), διευθύνσεις που καθορίζονται από το δίκτυο και βασίζονται στη λογική των συμβατικών δικτύων IP. Η ευελιξία των λέξεων-κλειδιών είναι χρήσιμη για τη συγκέντρωση πληροφοριών (π.χ. ανά τοποθεσία, ανά τύπο αισθητήρα) χωρίς να επιβαρύνει το δίκτυο. Στο NDN, αν θέλουμε να βρούμε πληροφορίες για τη θερμοκρασία, θα πρέπει να ρωτήσουμε όλους τους πιθανούς αισθητήρες αυτού του τύπου (ένα ερώτημα για κάθε αισθητήρα). Στο KIOT μπορούμε να βρούμε αυτό το είδος πληροφοριών με ένα μόνο ερώτημα, ως αποτέλεσμα του σχεδίου ονοματοδοσίας που βασίζεται σε λέξεις-κλειδιά.

Η λύση αυτή υλοποιήθηκε σε παλαιότερη εργασία [4], χρησιμοποιώντας Java για τον κώδικα των κόμβων και Python για την αρχικοποίηση και εκτέλεση πειραμάτων. Τα αποτελέσματα της εκτέλεσης του κώδικα όμως, υποδεικνύουν ότι υπήρχαν σφάλματα στην υλοποίησή του. Λόγω της περιορισμένης τεκμηρίωσης του κώδικα, κρίθηκε αναγκαίο πριν γίνουν αλλαγές στον κώδικα να αναλυθεί και να τεκμηριωθεί αναλυτικά, έτσι ώστε να είναι δυνατή η παρακολούθηση της λειτουργίας του. Το αντικείμενο της παρούσας εργασίας ήταν λοιπόν η τεκμηρίωση του κώδικα της υπάρχουσας υλοποίησης, η διόρθωση σφαλμάτων σε αυτόν και η προετοιμασία του για επέκταση και πειραματισμό από μεταγενέστερους ερευνητές.

Υπόβαθρο

ICN και NDN

Ο όρος "Πληροφοριοκεντρικό Δίκτυο" (ICN) εμφανίστηκε γύρω στο 2010, πιθανώς εμπνευσμένος από το 2006 Google Tech Talk του Van Jacobson "Ένας νέος τρόπος να δούμε τη δικτύωση". Αυτή η συζήτηση επισημαίνει μια νέα κατεύθυνση της μετάβασης του Διαδικτύου προς μια αρχιτεκτονική διανομής περιεχομένου. Σύμφωνα με αυτή την κοινή κατεύθυνση ICN, πολλά διαφορετικά σχέδια αρχιτεκτονικής προέκυψαν τα τελευταία χρόνια.

Αυτή η αρχιτεκτονική επιδιώκει τη δημιουργία δικτύων ικανών να διανέμουν περιεχόμενο στους τελικούς χρήστες, εστιάζοντας στις ίδιες τις πληροφορίες, και όχι στα τελικά σημεία. Αυτό στοχεύει στη μείωση της κυκλοφορίας, πιέζοντας και αποθηκεύοντας τις πληροφορίες πιο κοντά στους χρήστες. Η βασική ιδέα πίσω από αυτά τα δίκτυα είναι ότι όταν κάποιος χρήστης ζητήσει μια συγκεκριμένη πληροφορία, δεν θα ονομάσει το τελικό σημείο στο οποίο αναμένεται να διαμένουν αυτές οι πληροφορίες, αλλά θα απευθύνει το περιεχόμενο απευθείας καθορίζοντας μια περιγραφή αυτού του περιεχομένου. Το δίκτυο θα φροντίσει στη συνέχεια να δρομολογήσει το αίτημα με τέτοιο τρόπο ώστε ο χρήστης να λάβει ένα αντίγραφο του περιεχομένου.

Μια γνωστή προσέγγιση ICN είναι η NDN, μια αρχιτεκτονική που βασίζεται σε ιεραρχικά ονόματα, όπου κάθε πληροφορία χωρίζεται σε κομμάτια που αποτελούν τη βασική μονάδα διευθυνσιοδότησης. Μπορούμε να ορίσουμε συνοπτικά τις βασικές αρχές του NDN ως εξής: Πρώτον, αξιοποιούμε τις επιτυχημένες αρχές TCP / IP, συμπεριλαμβανομένης της “thin waist” στο επίπεδο του δικτύου, για την προώθηση της ανεξάρτητης καινοτομίας και του ελέγχου από άκρο σε άκρο, όπου ενδείκνυται. Δεύτερον, λαμβάνουμε υπόψη την εξέλιξη της χρήσης του διαδικτύου τα τελευταία τριάντα χρόνια, και συγκεκριμένα την αυξανόμενη έμφαση στο περιεχόμενο. Γενικεύουμε τη “thin waist” για να ικανοποιήσουμε αυτή την εξέλιξη, επιτρέποντας στα πακέτα να ονομάζουν (και να ζητούν) περιεχόμενο. Τέλος, ενσωματώνουμε θεμελιώδεις αρχιτεκτονικές βασισμένες σε αυτά που έχουμε μάθει από τις αδυναμίες του TCP / IP: κρυπτογράφηση της ταυτότητας κάθε πακέτου.

Η προσέγγιση KIOT

Το KIOT είναι η προσέγγισή μας για μια αρχιτεκτονική δρομολόγησης βασισμένη σε λέξεις-κλειδιά για το ICN. Το KIOT έχει εμπνευστεί από την TagNet [1], μια αρχιτεκτονική ICN που χρησιμοποιεί ετικέτες για να ονομάσει αντικείμενα περιεχομένου σε παγκόσμιο επίπεδο

Χρησιμοποιούμε ένα πιο ευέλικτο σύστημα ονομασίας (σε σχέση με το TagNet) για το KIOT, το οποίο είναι πιο κατάλληλο για ένα δίκτυο ICN που παράγει προσωρινές πληροφορίες σε μεγάλη κλίμακα [2]. Η διαφορά με το TagNet είναι ότι η σχέση αντιστοίχισης ορίζεται ως μια σχέση υπερσύνδεσης μεταξύ ενός συνόλου λέξεων-κλειδιών. Η λειτουργία δρομολόγησης στην πλατφόρμα μας είναι ένα υβρίδιο που βασίζεται σε μια τροποποιημένη έκδοση της δρομολόγησης με βάση την ετικέτα για τον τοπικό τομέα IoT και την τακτική προώθηση NDN εκτός αυτού του τομέα (επίπεδη και ιεραρχική). Οι τροποποιήσεις υποθέτουν ότι ο στόχος του συστήματος είναι η επεξεργασία ενός συνόλου δεδομένων IoT στον τοπικό τομέα, πριν από την αποστολή ενός αποτελέσματος στο

Internet, χρησιμοποιώντας ένα ευέλικτο και κλιμακωτό σχήμα ονομασίας με βάση λέξεις-κλειδιά.

Η δρομολόγηση πραγματοποιείται σε ένα μόνο δέντρο, με τη ρίζα του να είναι ο δρομολογητής συνόρων. Επιπλέον, υποθέτουμε ότι όλα τα αιτήματα περιεχομένου προέρχονται από τη ρίζα και επομένως όλες οι τελικές απαντήσεις πρέπει να επιστραφούν στη ρίζα. Αυτό σημαίνει ότι κάθε κόμβος χρειάζεται μόνο να γνωρίζει ποια από τις συνδέσεις του οδηγεί στη ρίζα για να επιστρέψει εκεί τα αποτελέσματα. Για να εντοπίσετε αντικείμενα περιεχομένου, κάθε στοιχείο περιεχομένου φέρει ετικέτα με ένα Bloom-filter [3] που υποδεικνύει τις λέξεις-κλειδιά στις οποίες συσχετίζεται και κάθε κόμβος στο δέντρο διατηρεί μια δήλωση για κάθε ένα downstream link, δηλαδή μια λίστα Bloom-filters, αναφέροντας ποιά στοιχεία περιεχομένου είναι διαθέσιμα μέσω αυτού του συνδέσμου. Σε αντίθεση με το TagNet, ένα αίτημα στο πρόγραμμά μας ταιριάζει με ένα στοιχείο περιεχομένου αν το Bloom-filter είναι ένα υποσύνολο του Bloom-filter του στοιχείου. Αυτό επιτρέπει τη συγχώνευση των Bloom-filters στην upstream κατεύθυνση, με την τιμή της εισαγωγής ψευδώς θετικών τιμών.

Υλοποίηση του KIOT

Τύποι κόμβων

Για λόγους αποδοτικότητας, η τοπολογία του δικτύου IoT υποτίθεται ότι έχει τη μορφή δέντρου. Επομένως, τα βασικά συστατικά του συστήματος είναι οι τρεις τύποι κόμβων, καθένας από τους οποίους αποτελεί αυτοτελές παράδειγμα της εφαρμογής:

- **Οι κόμβοι φύλλων** είναι τα κάτω άκρα του δένδρου, και συνδέονται με έναν γονέα. Κάθε περίπτωση ενός κόμβου φύλλων συνδέεται με ένα όνομα και ένα σύνολο λέξεων-κλειδιών που περιγράφουν τις πληροφορίες που προέρχονται από τους αισθητήρες. Κάθε αισθητήρας περιγράφεται σε ένα bloom-filter που ονομάζεται BF-Keyword. Η ένωση αυτών των λέξεων-κλειδιών BF οδηγεί στον κατάλογο BF, ο οποίος περιγράφει τα περιεχόμενα ενός κόμβου φύλλων. Για παράδειγμα, ένας κόμβος φύλλων μπορεί να έχει πληροφορίες σχετικά με τη θερμοκρασία (λέξη-κλειδί BF: 01001000) και την υγρασία (BF λέξη-κλειδί: 01000001), οπότε ο κατάλογός του BF θα είναι η ένωση αυτών των δύο με τιμή 01001001.
- **Οι μεσαίοι κόμβοι** είναι οι ενδιάμεσοι κόμβοι του δένδρου, ανάμεσα στη ρίζα και τα παιδιά (μπορούν να υπάρχουν πολλά επίπεδα). Αυτοί οι κόμβοι κατέχουν έναν πίνακα δρομολόγησης με δύο στήλες. Η πρώτη στήλη αποθηκεύει τους εισερχόμενους καταλόγους BF από τους θυγατρικούς κόμβους, ενώ η δεξιά αποθηκεύει τη διεύθυνση IP από τον κόμβο στον οποίο ανήκει ο κατάλογος BF (θα μπορούσε να είναι και διεύθυνση Ethernet).
- **Ο κόμβος ρίζας** είναι ο ίδιος με έναν μεσαίο κόμβο, χωρίς το ανώτερο τελικό σημείο, αφού είναι το ανώτερο επίπεδο του δικτύου. Τα ερωτήματα υποβάλλονται σε αυτό τον κόμβο, και εκεί επιστρέφουν οι απαντήσεις.

Το σύστημα είναι αυτο-οργανωμένο, απαιτώντας οι κόμβοι να γνωρίζουν μόνο το επίπεδό τους στο δέντρο κατά την αρχικοποίηση, με τις συνδέσεις των δέντρων να επιλέγονται δυναμικά. Οι κόμβοι φύλλων φυσικά πρέπει επίσης να γνωρίζουν τις λέξεις-κλειδιά BF για τους συνδεδεμένους αισθητήρες.

Τύποι πακέτων

Στην εφαρμογή μας έχουμε τέσσερα είδη πακέτων:

- **Parent packets**: Κατά την αρχικοποίηση του συστήματος, κάθε κόμβος πρέπει να βρει τον γονέα του για να μπορεί να επικοινωνεί μαζί του και να στέλνει τα κατάλληλα δεδομένα. Το μόνο που πρέπει να γνωρίζουμε είναι το επίπεδο στο οποίο θα διαμείνει ένας κόμβος. Επομένως, στην αρχή κάθε κόμβος στέλνει ένα μήνυμα εκπομπής στο δίκτυο (σε ένα ασύρματο δίκτυο, αυτοί θα είναι οι κόμβοι εντός του εύρους εκπομπής) και όλοι οι κόμβοι που λαμβάνουν αυτό το μήνυμα μπορούν να αποφασίσουν εάν είναι δυνητικοί γονείς του ζητούμενου κόμβου. Οι δυνητικοί γονείς είναι εκείνοι οι κόμβοι που βρίσκονται στο επόμενο (ανώτερο) επίπεδο του αιτούντος. Υπάρχουν δύο πολιτικές που χρησιμοποιούμε για να βρούμε αυτούς τους άνω συνδέσμους (γονείς):
    - **Load Balancing** Αυτή η πολιτική βασίζεται στον αριθμό των παιδιών δυνητικών γονέων. Όταν ένας κόμβος στέλνει την εκπομπή, οι δυνητικοί γονείς θα απαντήσουν. Αυτή η απάντηση θα περιέχει τον αριθμό των παιδιών τους. Έτσι, το ένα με τα λιγότερα παιδιά θα επιλεγεί ως γονέας του αιτούντος κόμβου. Εάν όλοι οι δυνητικοί γονείς έχουν τον ίδιο αριθμό παιδιών, τότε θα επιλέγεται τυχαία.
    - **Delay Minimization**: Αυτή η πολιτική βασίζεται στο Round Trip Time (RTT) μεταξύ κόμβων. Σε αυτήν την περίπτωση, όταν ο αιτών κόμβος λαμβάνει

την απάντηση από δυνητικούς γονείς, λαμβάνει επίσης δεδομένα σχετικά με το RTT. Ο κόμβος με χαμηλότερη τιμή RTT θα είναι ο γονέας του ζητούμενου κόμβου.

- **Advertisement packets**: Η ενέργεια αυτή λαμβάνει χώρα μόλις αρχίσει να τρέχει ένας κόμβος φύλλων. Κάθε κόμβος (bottom-to-top) στέλνει λέξεις-κλειδιά που διαφημίζουν τις πληροφορίες που προσφέρει, οι οποίες αποθηκεύονται στον πίνακα δρομολόγησης του γονέα του, ο οποίος αποθηκεύει αρχεία των πληροφοριών που προέρχονται από τους θυγατρικούς κόμβους (κατάλογοι BF όπως αναφέραμε παραπάνω) και τη διεύθυνση του θυγατρικού κόμβου. Ο κόμβος που λαμβάνει το διαφημιστικό μήνυμα θα εκτελέσει μια συνένωση μεταξύ του ληφθέντος Bloom filter και του τοπικού Bloom filter και θα το διαφημίσει στον δικό του γονέα, εάν υπάρχει. Ο γονέας θα κάνει το ίδιο με τη σειρά του.
- **Interest packets**: Κάθε φορά που γίνεται αίτημα ενδιαφέροντος, η διαδικασία είναι η ακόλουθη. Το αίτημα ερώτησης (π.χ. aueb, troias, υγρασία) αποστέλλεται στον κόμβο ρίζας. Ο ριζικός κόμβος ελέγχει αν το όνομά του περιλαμβάνεται στο ερώτημα ή όχι, ελέγχοντας τον κατάλογο BF. Στη συνέχεια, ελέγχει εάν το αίτημα ερώτησης είναι ένα υποσύνολο οποιουδήποτε από τους καταλόγους BF του FIB της ρίζας. Εάν αυτό είναι αληθινό, τότε το αίτημα ερωτήματος προωθείται στους θυγατρικούς κόμβους (ένα ή περισσότερα) των αντίστοιχων αποτελεσμάτων. Δεδομένου ότι οι διευθύνσεις IP αποθηκεύονται, ο γονικός κόμβος ξέρει πού να στείλει την αίτηση ερωτήματος, σύμφωνα με την αντιστοιχία του καταλόγου BF. Αυτό συμβαίνει μέχρι να φτάσει το αίτημα στους κόμβους των φύλλων όπου αποθηκεύονται οι πραγματικές πληροφορίες.
- **Data packets**: Αυτά τα πακέτα είναι τα επακόλουθα της αντιστοίχισης αιτήματος ερωτήματος. Όταν υπάρξει πρόσβαση στους κατάλληλους κόμβους και ανακτηθούν οι πληροφορίες, θα σταλθούν ξανά προς την κορυφή του δέντρου μέχρι να φτάσουν στον ριζικό κόμβο.

Μηνύματα

Για να κατανοήσουμε το πώς λειτουργεί το σύστημα, θα περιγράψουμε τη λειτουργία του από την αρχικοποίηση μέχρι την αποστολή ενός αιτήματος και τη λήψη μιας απάντησης.

1. Στη φάση αρχικοποίησης του συστήματός μας, κάθε κόμβος στέλνει ένα μήνυμα εκπομπής MSG_PARENT για να αφήσει τους άλλους κόμβους να γνωρίζουν ότι ψάχνει γονέα. Οι κόμβοι στο ανώτερο επίπεδο θα απαντήσουν στον αιτούντα με το MSG_PARENT_ACK. Επιλέγοντας την πολιτική εξισορρόπησης φορτίου, ο κόμβος θα επιλέξει έναν κόμβο με τα λιγότερα παιδιά, στέλνοντας ένα MSG_PARENT_CHILD στον επιλεγμένο γονικό κόμβο του, έτσι ώστε να μπορεί να αυξήσει τα παιδιά του. Σε περίπτωση ελαχιστοποίησης καθυστέρησης, υπολογίζουμε τους χρόνους RTT όλων των πιθανών γονέων και επιλέγουμε το κόμβο με τον συντομότερο χρόνο.
2. Το επόμενο βήμα είναι η διαφήμιση του περιεχομένου ότι οι κόμβοι φύλλων προέρχονται από τους αισθητήρες (θερμοκρασία, υγρασία, πίεση κ.λπ.). Όταν όλοι οι κόμβοι βρουν τους γονείς τους, περνάμε σε αυτή τη φάση της διαφήμισης. Οι κόμβοι φύλλων στέλνουν MSG_ADVERTISEMENT στους γονείς τους με τις πληροφορίες τους. Τώρα, αυτοί οι κόμβοι στέλνουν τις συγκεντρωτικές πληροφορίες στους γονείς τους και αυτή η διαδικασία επαναλαμβάνεται μέχρι να φτάσουμε στον κόμβο ρίζας. Κάθε μήνυμα MSG_ADVERTISEMENT επιβεβαιώνεται από ένα MSG_ADVERTISEMENT_ACK, για να γνωρίζει εάν το περιεχόμενο μετακινήθηκε σωστά.
3. Σε αυτό το σημείο το σύστημα είναι έτοιμο. Όλοι οι κόμβοι έχουν δημιουργήσει τους πίνακες FIB, γνωρίζουν τους γονείς τους και είναι έτοιμοι να επικοινωνούν μεταξύ τους. Ένας φιλοξενούμενος κόμβος μπορεί να ζητήσει πληροφορίες τώρα

στέλνοντας ένα MSG_INTEREST με το είδος των πληροφοριών (π.χ., (aueb, evelpidon, υγρασία)) που θέλει στη ρίζα. Η ρίζα ελέγχει το FIB για να στείλει αυτό το μήνυμα στο κατάλληλο παιδί (ή στα παιδιά). Τα παιδιά με τη σειρά τους ελέγχουν το FIB τους για να κατευθύνουν το πακέτο στα παιδιά τους μέχρι να φτάσουμε στο κάτω μέρος του δέντρου (κόμβοι φύλλων) για να πάρουμε το περιεχόμενο που ζητά ο φιλοξενούμενος.

4. Όταν φτάσουμε στους κόμβους των φύλλων, ένα μήνυμα MSG_DATA με το περιεχόμενο δρομολογείται από το κάτω προς τα πάνω στρώματα (κόμβος σε γονέα) μέχρι να φτάσει το περιεχόμενο στη ρίζα.

Τεκμηρίωση και βελτίωση της υλοποίησης

Τμήματα κώδικα Java

Σε κάθε κόμβο του δικτύου εκτελείται το πρόγραμμα KIOT που έχει ως αρμοδιότητες να συμβάλει στην δημιουργία του δέντρου δρομολόγησης, την δήλωση των τοπικών λέξεων κλειδιών, την μετάδοση τους ώστε να στηθούν οι πίνακες δρομολόγησης, την λήψη αιτημάτων και τέλος την αποστολή των δεδομένων στον αιτούντα. Σε έναν κόμβο εκτός του δένδρου εκτελείται το πρόγραμμα Guest. Ο ρόλος του Guest κόμβου είναι να δημιουργεί αιτήματα. Στο κόμβο αυτό δηλώνονται οι λέξεις κλειδιά οι οποίες απαρτίζουν το αίτημα μας. Οι λέξεις κλειδιά εισάγονται στη δομή του Bloom-Filter και αποστέλλονται στη ρίζα του δέντρου το οποίο θα επαληθεύσει αν του στάλθηκε γνήσιο υποσύνολο του.

Βήματα εκτέλεσης κώδικα Java

Παρακάτω φαίνεται η ροή ελέγχου του κώδικα σε Java.

1 Προετοιμασία κόμβου: **(Configuration.configure)**
   1.1 Ανάγνωση αρχείου ρυθμίσεων
   1.2 Ορισμός επιπέδου του κόμβου.
   1.3 Ορισμός του τύπου του κόμβου
   1.4 Δημιουργία της λίστας των λέξεων κλειδιών του κόμβου
2 Κατασκευή του δέντρου:
   2.1 Ο κόμβος στέλνει μήνυμα ότι ψάχνει για κόμβο-γονέα. **(Main:main:43)**
   2.2 Ένας άλλος κόμβος ελέγχει το αίτημα και αν βρίσκεται σε ανώτερο επίπεδο τότε προσφέρεται να γίνει αυτός γονέας, δηλώνοντας επίσης στο μήνυμα του, πόσα παιδιά έχει ήδη. **(Receiver.extractParent)**
   2.3 Όταν λαμβάνει ο κόμβος την απόκριση γονέα τότε την αποθηκεύει σε ένα μία δομή hashmap, όπου αντιστοιχίζει την IP με τον αριθμό των παιδιών. **(Receiver.extractPacket**)
   2.4 Από όλους τους προσφερόμενους κόμβους ελέγχει αυτόν με τα λιγότερα παιδιά και στέλνει μήνυμα αναγνώρισης. **(SendMessageTimer.chooseParentsByChildren)**
3 Διαφήμιση του κόμβου:
   3.1 Δημιουργία του τοπικού Bloom-Filter χρησιμοποιώντας τις λέξεις κλειδιά από το αρχείο ρυθμίσεων**. (BloomFilter.getBloomFilter)**
   3.2 Δημιουργία του μηνύματος δήλωσης του κόμβου που περιέχει το BloomFilter των λέξεων κλειδιών και αποστολή προς τον πατρικό κόμβο. **(Main:main:51)(Advertisement.Advertisement)**
   3.3 Ο πατρικός κόμβος αντιστοιχίζει το BloomFilter με το συγκεκριμένο θυγατρικό κόμβο και το συνενώνει με το δικό του. **(Reciever.extractAdvertisement)**
   3.4 Αν δεν είναι η ρίζα του δέντρου τότε το μεταβιβάζει στον αντίστοιχο πατέρα του**. (Receiver.extractAdvertisement)**
4 Αίτημα για λέξεις κλειδιά:
   4.1 Ο κόμβος Guest διαβάζει τις λέξεις-κλειδιά από το δικό του αρχείου ρυθμίσεων και τις εισάγει σε λίστα**. (Configuration.configure)**
   4.2 Οι λέξεις κλειδιά εισάγονται στο BloomFilter που ορίζουν το αίτημα. **(Main.main:29) (BloomFilter.getBF_catalog)**

4.3 Δημιουργείται το μήνυμα αιτήματος που περιέχει τις λέξεις κλειδιά και το στέλνει στη ρίζα του δέντρου. (new Interest) (new Sender) **(Main:main:29)**

4.4 Η ρίζα του δέντρου ελέγχει αν υπάρχει αντιστοιχία μεταξύ του των BloomFilter του κόμβου και του αιτήματος. **(Receiver.extractInterest)**

4.5 Αν υπάρχει αντιστοιχία με κάποιο πίνακα του BloomFilter θυγατρικού κόμβου, το μεταβιβάζει σε αυτό. **(Receiver.sendInterest)**

4.6 Αν έχει φτάσει στο συγκεκριμένο φύλο του δέντρου, στέλνει τα δεδομένα στο πατρικό κόμβο**. (Receiver.sendLeafTags)**

4.7 Αν ο παρών κόμβος είναι ενδιάμεσος τότε στέλνει στο πατρικό κόμβο. **(Receiver,extractData)**

4.8 Αν ο παρών κόμβος είναι η ρίζα τότε εμφανίζει τα δεδομένα. **(Receiver.extractData)**

Σχόλια και τεκμηρίωση του κώδικα Java

Η κλάση main του κώδικα αρχικά διαβάζει τις ιδιότητες του κόμβου, μέσω της κλάσης Configuration και στη μέθοδο configure **(general.Configuration.configure:114**). Οι ιδιότητες του κόμβου είναι:

- Το όνομα/ταυτότητα του κόμβου (String nodeID),
- το επίπεδο του κόμβου (int level),
- ο τύπος του κόμβου (int type) όπου: 1 για τη ρίζα , 2 για τον ενδιάμεσο κόμβο και 3 για τα φύλα.
- το όνομα του αρχείου αναφορών (String outputFile)
- το όνομα του αρχείου αναφορών σφαλμάτων (String warningFile)
- τα tags (Keywors) του κόμβου (ArrayList<String> tags).

Οι ιδιότητες διαβάζονται από ένα αρχείο αρχικοποίησης.

Μετά από την ταυτοποίηση του κόμβου ξεκινά το νήμα το οποίο λαμβάνει το κάθε μήνυμα και το επεξεργάζεται ανάλογα με το τύπο του (Receiver).

Το δίκτυο μας έχει τη τοπολογία δέντρου στο οποίο τα φύλα παρέχουν τα δεδομένα μαζί με τις λέξεις κλειδιά που τις ταυτοποιούν. Αν ο παρών κόμβος δεν έχει οριστεί ως ρίζα τότε στέλνει ένα **ParentMessage** για να συμβάλει στη κατασκευή του δέντρου (Main:main). Το συγκεκριμένο μήνυμα θέλει να πει σε όλους τους κόμβους ότι ψάχνει έναν γονέα και ποιο είναι το επίπεδο στο οποίο έχει τοποθετηθεί **(componements.messages.ParentMessage) (componements.Packet**). Για να αποσταλεί το μήνυμα αυτό πρέπει πρώτα να μετατραπεί σε ένα byte πίνακα ([]byte toByteArray). Η μετατροπή γίνεται ανάλογα με τον τύπο του μηνύματος.

Η αποστολή γίνεται μέσω της κλάσης-Νήματος **Sender**. Η κλάση αυτή έχει ως πεδία το UDP socket για να στείλει το πακέτο αλλά και το αντικείμενο προς μεταφορά. Εντός του νήματος ελέγχεται ο τύπος του αντικειμένου (connection.Sender) και καλεί την αντίστοιχη μέθοδο toByteArray του αντικειμένου με την οποία μετατρέπει το πακέτο σε πίνακα byte.

Το πακέτο το οποίο δημιουργήθηκε θα ληφθεί από τους άλλους κόμβους στη κλάση-Νήμα Receiver**.(Receiver.run)** Εκεί θα φιλτραριστεί κατά το messageID όπου στη συγκεκριμένη περίπτωση αν είναι MSG_PARENT **(Receiver:extractParent)** και αν είναι όντως από κόμβο χαμηλότερου επιπέδου τότε στέλνει μήνυμα MSG_PARENT_ACK όπου αναγνωρίζει το κόμβο που ψάχνει για πατέρα και τον ενημερώνει πόσα παιδιά έχει ήδη **(Receiver:sendParentACK**).

Από τη στιγμή που λαμβάνει ο θυγατρικός κόμβος την αναγνώριση από τον πατρικό κόμβο ή τους πατρικούς κόμβους, αποθηκεύει τις πατρικές IP μαζί με τον αντίστοιχο αριθμό παιδιών.

Όταν στέλνει ο γονικός κόμβος τον αριθμό των παιδιών, στέλνει επίσης ένα μήνυμα χρονοσφραγίδιας (MessageTimer) στο οποίο ενημερώνει τον άλλο κόμβο για το πόσα παιδιά έχει ο παρών κόμβος ή και προαιρετικά πόσο χρόνο χρειάζεται ένα μήνυμα parent για να φτάσει στο προορισμό του**.(ParentMessage)**

Από τη στιγμή που αναγνωρίσει το παιδί τον γονέα τότε ο θυγατρικός κόμβος στέλνει πακέτο αναγνώρισης προς τον πατέρα **(MSG_PARENT_CHILD)** και ολοκληρώνει τη διαδικασία.

Όταν ολοκληρωθεί το δέντρο τότε ξεκινά η διαδικασία της διαφήμισης του κόμβου προς τη ρίζα.

(Main:main)

Άν ο κόμβος είναι φύλο του δέντρου και ο πατρικός κόμβος έχει δηλωθεί τότε ξεκινά η δήλωση των λέξεων κλειδιών προς τη ρίζα του δέντρου. Για να γίνει αυτό δημιουργείται το μήνυμα Advertisement το οποίο εμπεριέχει τη ταυτότητα του μηνύματος αλλά και τον byte πίνακα του Bloom Filter.

Το Bloom Filter είναι μία δομή δεδομένων η οποία μας απαντά στο αν μία τιμή έχει πιθανότατα εισαχθεί στη δομή αυτή ή όχι που εξυπηρετεί τους σκοπούς μας. **(Main:main) (BloomFilter:getBF_catalog).** Άν δεν έχει οριστεί ήδη ο byte πίνακας τότε λαμβάνονται τα keywords από τη λίστα του κόμβου και περνούν μέσα από ένα σταθερό αριθμό συναρτήσεων κατακερματισμού για να εισαχθούν στο πίνακα byte και να είναι έτοιμα για μεταφορά.

Όταν ληφθεί το πακέτο Advertisement από τον πατρικό κόμβο ξεχωρίζουμε όπως και στις προηγούμενες περιπτώσεις τη ταυτότητα του μηνύματος. Όταν ο τύπος το μηνύματος είναι τύπου **MSG_ADVERTISEMENT** τότε συνοδεύεται με το πίνακα byte του BloomFIlter. Όταν ληφθεί το bloomFilter,αντιστοιχίζεται με την I.P. του πακέτου που έχει ληφθεί στη δομή HashMap που την ονομάσαμε **ForwardInformationBase(F.I.B.**) έτσι ώστε να αντιστοιχίζεται κάθε διακριτό BloomFilter σε μία μόνο IP. Στη συνέχεια γίνεται η πράξη ένωσης των BloomFilter των παιδιών και του παρόντος. Η διαδικασία αυτή συνεχίζεται μέχρι να φτάσει στο κόμβο της ρίζας.

Η διαδικασία της αίτησης δεδομένων με συγκεκριμένες λέξεις κλειδιά ξεκινάει με τη λήψη των λέξεων κλειδιών από ένα αντίστοιχο αρχείο Configuration στο οποίο δηλώνονται οι λέξεις κλειδιά του αιτήματος και εισάγονται στο BloomFilter και μεταφέρονται μέσω του μηνύματος **MSG_INTEREST** στη ρίζα του δέντρου.

Όταν φτάσει στη ρίζα το μήνυμα Interest (Receiver) ελέγχει πρώτα αν υπάρχει το αίτημα στο τοπικό BloomFilter. Άν αντιστοιχεί σε υποσύνολο στο τοπικό BloomFilter τότε ελέγχει και στις εγγραφές του FIB για να ελέγξει αν το αίτημα είναι υποσύνολο των BF των παιδιών. Με το που φτάσει το αίτημα στα φύλλα του δέντρου τότε ξεκινά η διαδικασία μεταφοράς των δεδομένων σε μορφή αλφαριθμητικού προς τη ρίζα**.(Receiver:sendData**). Όταν ληφθεί το μήνυμα των δεδομένων από το κόμβο τότε ελέγχεται αν ο παρών κόμβος είναι ενδιάμεσος ή ρίζα. Στη περίπτωση όπου ο κόμβος είναι ενδιάμεσος τότε μεταβιβάζει προς τη ρίζα ενώ αν είναι η ρίζα τότε εκτυπώνει το μήνυμα.

Κλάση **Configuration**: Η κλάση με τις ιδιότητες του κόμβου όπου χρησιμοποιούνται σε όλες τις λειτουργίες του κόμβου.

Μέλη:

- **RTT_Enabled:** Boolean μεταβλητή η οποία μας ορίζει αν θα χρησιμοποιήσουμε την πολιτική του Round Trip Time
- **Parent_Map:** Hash Map αντιστοίχησης IP γονέων με τον αριθμό των παιδιών τους.
- **tags:** λίστα των λέξεων κλειδιών του κόμβου.
- **Parent_IP:** Η IP του γονικού κόμβου.
- **Type:** Αριθμητική τιμή που θα ορίζει αν ο κόμβος είναι ρίζα, ενδιάμεσος ή φύλο.
- **Level:** Το επίπεδο του κόμβου που έχουμε ορίσει.
- **number_of_children:** Αριθμός των παιδιών του κόμβου.
- **Advertisement_ACK_List**: η λίστα των κόμβων που πρέπει να απαντήσει σε περίπτωση που λάβει μήνυμα δημοσίευσης.

Μέθοδοι:

- **configure()**: Διαβάζει το αρχείο ρυθμίσεων και δίνει τιμές στις ιδιότητες του κόμβου.

Κλάση **Receiver:** Η κλάση των νημάτων που παραλαμβάνουν πακέτα.

Μέλη:

ServerSocket: Ένα UDP Socket από το οποίο λαμβάνουμε δεδομένα.

Μέθοδοι:

- **Public Receiver(**): Δημιουργεί ένα UDP Socket στο οποίο θα λαμβάνουμε τα μηνύματα και θα τα επεξεργαζόμαστε.
- **Public Run()**: Εξάγει το μήνυμα από το πακέτο και το επεξεργάζεται.
- **Private extractPacket(String messageId,ByteBuffer buffer,DatagramPacket receivePacket,long startTime)**: Φιλτράρει το πακέτο ανάλογα με το messageID και κάνει την αντίστοιχη επεξεργασία.

- **Private extractInterest(ByteBuffer buffer)**: Αναλύει το Μήνυμα Interest,το οποίο έχει εξαχθεί από το πακέτο.
- **Private insertLeafTags**(): Για όλες τις λέξεις κλειδιά του κόμβου, να γίνει σύγκριση με τις λέξεις κλειδιά του μηνύματος Interest.
- **private insertTagtoBF():** Άν το BF το οποίο δημιουργήθηκε από το τα tag του αιτήματος αντιστοιχεί σε υποσύνολο του κόμβου τότε στέλνει τα δεδομένα προς τη ρίζα.
- **Private sendInterest():** Στέλνει το μήνυμα Interest στο παιδί όπου αντιστοιχεί το BloomFilter.
- **Private sendData():** Στέλνει τα δεδομένα του κόμβου προς τη ρίζα
- **Private extractParent():** Αν ο κόμβος όπου στέλνει το μήνυμα είναι στο αμέσως υψηλότερο επίπεδο τότε στέλνει μήνυμα αναγνώρισης προς αυτόν.
- **Private sendParentACK():** Στέλνει μήνυμα αναγνώρισης προς το πατρικό κόμβο.
- **Private extractAdvertisement(**): Λαμβάνει το μήνυμα Advertisement από τους θυγατρικούς κόμβους, δημιουργεί μία εγγραφή αντιστοιχίας του BloomFilter με την IP στο FIB και αν είναι ενδιάμεσος κόμβος στέλνει στο πατρικό.
- **Private extractData():** Λαμβάνει το μήνυμα δεδομένων και το μεταβιβάζει προς τη ρίζα.
- **private GetMessageID()**: Εξάγει τη ταυτότητα του μηνύματος για την επεξεργασία που θα ακολουθήσει.
- **private getBloomFilter():** Εξάγει το αντικείμενο BF χρησιμοποιώντας το πίνακα που έχει το μήνυμα που βρίσκεται στο μήνυμα Advertisement ή Interest.
- **Private getData():** Εξάγει σε αλφαριθμητική μορφή τα δεδομένα που έχουν εισαχθεί στο μήνυμα Data
- **private getLevel()**: Εξάγει τον αριθμό των παιδιών από το μήνυμα Parent.
- **Private getChildren():** Εξάγει τον αριθμό των παιδιών από το μήνυμα parent.

Κλάση **Sender**: Η κλάση των νημάτων που αποστέλλουν πακέτα.

Μέλη:

- **ctrlsocket**: το UDP Socket μέσα από το οποίο θα στείλουμε τα δεδομένα μας.
- **SendDataObject**: Το αντικείμενο προς αποστολή.
- **DestinationIP**: Η IP του προορισμού.
- **DestinationPort**: Η θύρα του προορισμού.

Μέθοδοι:

- **run():** Έχει ως στόχο να δημιουργεί το πακέτο του μηνύματος χρησιμοποιώντας την αντίστοιχη μετατροπή του μηνύματος σε byte πίνακα.
- **ProcessPacket()**: Ελέγχει το τύπο του μηνύματος και καλεί την αντίστοιχη μετατροπή του σε byte πίνακα.

Κλάση **FIB**: Έχει ως λειτουργία να αντιστοιχίζει τους πίνακες του BloomFilter με τις αντίστοιχες I.P.

Μέλη:

- **fib:** Δομή hashmap αντιστοίχησης Bloom-filter σε IP που χρησιμοποιείται ως δείκτες.

Μέθοδοι:

- **getInstance():** Επιστρέφει τη δομή έτσι ώστε να γίνεται αμοιβαία χρήση.
- **add():** Εισάγει νέα εγγραφή στη δομή, hashmap και άν υπάρχει ήδη ,την αντικαθιστά.

Κλάση **ParentMessage**: Εξάγει τα μηνύματα: Parent, ParentACK και ParentChildren.

Κλάση **MessageTimer**: Είναι η κλάση-Νήμα όπου ο κόμβος ορίζει ποίος θα είναι ο πατρικός του χρησιμοποιώντας την αντίστοιχη πολιτική.

Μέλη:

- **type**: Ο τύπος του μηνύματος
- **time**: Ο μέγιστος χρόνος αναμονής για την σύγκριση.

Μέθοδοι:

- **run**: Ελέγχει τον τύπο του μηνύματος. Στην περίπτωση Parent ξεκινά τις συγκρίσεις μεταξύ των γονέων ενώ στη περίπτωση Advertisement μεταβιβάζει τη δημοσίευση του κόμβου στο πατέρα.
- **ChooseParentByChildren():** Με κάθε νέο μήνυμα Parent που έρχεται στο κόμβο , γίνεται η εύρεση του κόμβου με τα λιγότερα παιδιά και στέλνει το μήνυμα αναγνώρισης.
- **ChooseParentByRTT():** Με κάθε νέο μήνυμα Parent που λαμβάνεται από τον κόμβο γίνεται εκ νέου εύρεση για τη μικρότερη σε διάρκεια μεταφορά του μηνύματος.

Η κλάση **Packet**: Ορίζει την δομή του πακέτου κρατώντας ιδιότητες όπως η ταυτότητα του. Εξειδικεύεται σε Advertisement, Data, Interest, ParentMesage. Ο στόχος ύπαρξης της είναι να προετοιμάσει τα δεδομένα για αποστολή με το να τα εξάγει σαν πίνακα byte.

Μέλη:

- **MessageID:** η ταυτότητα του μηνύματος. Χρησιμοποιείται σε όλα τα μηνύματα

- **BloomFilter:** το Bloom-Filter που μεταφέρεται. Χρησιμοποιείται στα μηνύματα Interest και Advertisement
- **Level:** Το επίπεδο του κόμβου, χρησιμοποιείται στο μήνυμα Parent
- **Data:** ο πίνακας των δεδομένων που στέλνουν τα φύλλα. Χρησιμοποιείται στα μηνύματα Data.

Κλάση **Advertisment**: Έχει σαν πεδία την ταυτότητα του μηνύματος και τον πίνακα του **BloomFilter** για την δήλωση του κόμβου στους πατρικούς.

Κλάση **Interest**: Έχει σαν πεδία την ταυτότητα του μηνύματος και τον πίνακα του BloomFilter για να μεταβιβάσει τα αίτημα για συγκεκριμένα δεδομένα προς τα φύλλα του δέντρου.

Κλάση **Main**: Ξεκινά τη λειτουργία του κόμβου με το να στέλνει μηνύματα προς τους άλλους κόμβους για την δημιουργία του δέντρου. Στη συνέχεια διαφημίζει τις λέξεις κλειδιά στους πατρικούς κόμβους. Στη περίπτωση που ο κόμβος είναι **Guest** στέλνει το αίτημα με τις λέξεις-κλειδια προς τη ρίζα του δέντρου

Κώδικας Python

Για να μπορέσουμε την δοκιμή των πειραμάτων τρέχουμε τους κόμβους μας σε ένα εξομοιωτή δικτύου, το mininet. Αυτός ο εξομοιωτής δικτύου έχει ένα API σε Python και τα εκτελούμε τοπικά. Ο εξομοιωτής λειτουργεί με το να δημιουργεί τη τοπολογία του δικτύου και σε καθένα από αυτά να αναθέτει τοπικές εντολές κελύφους.

Για να οριστεί το δίκτυο πρέπει να δημιουργηθούν τα αρχεία ιδιοτήτων (createConfigFiles.py). Σε αυτό το δίκτυο ορίζουμε τη τοπολογία αστέρα με όλους τους κόμβους να συνδέονται στον ίδιο δρομολογητή. Εκεί ορίζουμε το ύψος του δέντρου που θα θέλαμε να προσομοιώσουμε και να το χειριστούν οι κόμβοι KIOT (get_nodes) εκεί ορίζει πόσους κόμβους θα έχει το κάθε επίπεδο και στη συνέχεια ορίζει στο κάθε κόμβο το επίπεδο το οποίο βρίσκεται και αν είναι κόμβος-φύλλο να του προσθέσει τις ετικέτες.

Αρχείο πειραμάτων(topoex.py):

Κατά την εκκίνηση του προγράμματος δημιουργείται το αντικείμενο τοπολογίας. Το αντικείμενο της τοπολογίας έχει ως ιδιότητες: Τις ιδιότητες των κόμβων, τις καθυστερήσεις των κόμβων, τον αριθμό των κόμβων και προστίθενται οι κόμβοι στη λίστα των κόμβων και το δρομολογητή στον οποίο θα συνδεθούν οι κόμβοι. Τέλος ορίζουμε τις συνδέσεις μεταξύ κόμβων και δρομολογητών.

Αφού δημιουργηθεί η τοπολογία, την εκκινούμε και αναθέτουμε στους κόμβους να εκτελέσουν στο παρασκήνιο τα προγράμματα μας.

Για να εκτελεστούν τα προγράμματα μας, πρέπει να οριστεί ένα σύνθετο λεκτικό στο οποίο αποτελείται από: Το πρόγραμμα που θα εκτελεστεί, το αρχείο ρυθμίσεων για το συγκεκριμένο πρόγραμμα, το σύμβολο “>” για την έξοδο της κονσόλας σε αρχείο text και το σύμβολο “&” (ambersand) για την εκτέλεση στο παρασκήνιο.

Συμπέρασμα

Η ICN έχει τη φιλοδοξία να σχεδιάσει ένα νέο Διαδίκτυο που θα ταιριάζει καλύτερα στις σημερινές ανάγκες των χρηστών και θα λύσει όλα τα προβλήματα που προκύπτουν από τη μεγάλη ανάπτυξη του δικτύου. Στην παρούσα διατριβή προτείνουμε μια νέα αρχιτεκτονική ICN, KIOT. Για αυτή την προσέγγιση, χρησιμοποιήσαμε ένα σύστημα ονοματοδοσίας βασισμένο σε λέξεις-κλειδιά, διότι με αυτόν τον τρόπο είναι πιο ευέλικτο να ανακτώνται δεδομένα από αισθητήρες, επιπλέον του NDN. Ο στόχος μας με το KIOT ήταν να αντιμετωπίσουμε μερικά από τα θεμελιώδη προβλήματα του Διαδικτύου. Συζητήσαμε πώς η εισαγωγή ενός νέου πρωτότυπου επικοινωνίας και η δική μας δομή του δικτύου βοηθά στην αντιμετώπιση πολλών από τα ζητήματα που αντιμετωπίζουμε σήμερα με το παραδοσιακό TCP / IP.

Το KIOT χρησιμοποιεί μια μέθοδο για την δρομολόγηση πληροφοριών που πιστεύουμε ότι είναι πιο αναλυτικές και θα ενισχύσουν το σχεδιασμό εφαρμογών που λειτουργούν με έναν φιλικό τροπο προς τον άνθρωπο. Αυτές οι διευθύνσεις που ορίζονται από το χρήστη κωδικοποιούνται σε σύνολα λέξεων-κλειδιών. Ωστόσο, τα σύνολα αυτά μπορούν να χρησιμοποιηθούν για να μιμηθούν τη σημασιολογία των ιεραρχικών ονομάτων. Υποστηρίζουμε την επεκτασιμότητα του δικτύου παρέχοντας μηχανισμούς γρήγορης προώθησης και δρομολόγησης για τη μεγιστοποίηση της απόδοσης και την ελαχιστοποίηση των μηνυμάτων ελέγχου. Το πρόγραμμα δρομολόγησης που ακολουθήσαμε μπορεί να υποστηρίξει τις λειτουργίες push και pull με φυσικό τρόπο, χωρίς να εισάγει πρόσθετα έξοδα. Πρέπει ακόμα να εκτελέσουμε μια πιο εκτενή αξιολόγηση του KIOT και να το συγκρίνουμε με άλλες αρχιτεκτονικές του ICN, όπως το NDN και το CCN. Διερευνούμε επίσης καλύτερους τρόπους να διαμορφώσουμε την τοπολογία των δέντρων της αρχιτεκτονικής μας.

Στην εργασία αυτή ασχοληθήκαμε με την τεκμηρίωση και βελτίωση ενός συστήματος που υλοποιεί τη σχεδίαση του KIOT, το οποίο είναι γραμμένο σε Java, χρησιμοποιώντας σενάρια σε Python για να εκτελέσει τον προσομοιωτή mininet για πειραματισμό.

Βιβλιογραφία

1. M. Papalini, A. Carzaniga, K. Khazaei, and A. L. Wolf. Scalable routing for tag-based information-centric networking. In Proceedings of the ACM Conference on Information-Centric Networking, pages 17–26, September 2014.
2. O. Ascigil, S. Rene, G. Xylomenos, I. Psaras and G. Pavlou. A Keyword-based ICN-IoT Platform. In Proceedings of the ACM Conference on Information-Centric Networking, pages 22-28, September 2017.
3. S. Tarkoma, C. E. Rothenberg and E. Lagerspetz. Theory and Practice of Bloom Filters for Distributed Systems. IEEE Communications Surveys & Tutorials 14.1, pages 131–155, 2012.
4. Πτυχιακή εργασία “Keyword-based Routing in IoT (KIOT)” Evangelos Zafeiratos Supervisor: George Xylomenos, Athens, April 2018