ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΤΙΤΛΟΣ : «Υλοποίηση πολιτικών προσωρινής αποθήκευσης σε δίκτυα βασισμένα στο περιεχόμενο»

Επίβλεψη

- Τασιούλας Λέανδρος, Καθηγητής
- Κατσαρός Δημήτριος, Λέκτορας

Εκπόνηση

- Γιωργαλλίδης Αντρέας

Βόλος, Μάρτιος 2013

Περιεχόμενα

Ευχαριστίες

Εκφράζω θερμές ευχαριστίες στους καθηγητές μου κ. Λέανδρο Τασιούλα και κ. Δημήτριο Κατσαρό για την επίβλεψη της διπλωματικής μου εργασίας.

Ιδιαίτερα θέλω να ευχαριστήσω τον Πάρη Φλέγκα για την όλη καθοδήγηση και τις συμβουλές που μου παρείχε από την αρχή έως και το τέλος της διπλωματικής μου εργασίας.

Τέλος ευχαριστώ τους φίλους και την οικογένεια μου για την αμέριστη συμπαράσταση και στήριξη που μου έδειξαν καθ' όλη την διάρκεια των σπουδών μου.

1. Εισαγωγή

1.1. Εισαγωγή

Προκειμένου να καταστεί διαχειρίσιμη η αυξανόμενη ζήτηση και κίνηση δεδομένων στο διαδίκτυο, μια σειρά από τεχνολογίες και υπηρεσίες χρησιμοποιούνται για να αναπαράγουν, να αποθηκεύουν και να διανέμουν το περιεχόμενο με διαφορετικούς τρόπους. Τέτοιες τεχνολογίες όπως Content Distribute Networks και P2P, προάγουν ένα μοντέλο επικοινωνίας όπου η πρόσβαση στα δεδομένα γίνεται βάση ονόματος χωρίς την γνώση τοποθεσίας της πηγής. Η χρήση όμως διαφορετικών τεχνολογιών αποτρέπει την ανάθεση καθολικών μοναδικών ονομάτων σε αντικείμενα με αποτέλεσμα η συνολική προσπάθεια να είναι ανεπαρκής.

Τα Δίκτυα Βασισμένα στο Περιεχόμενο (Information Centric Networking) είναι μια προσέγγιση για την δημιουργία διαδικτύου όπου η ανάθεση καθολικών μοναδικών ονομάτων σε αντικείμενα αποτελεί βασική αρχή. Με τον τρόπο αυτό τα δεδομένα ανεξαρτητοποιούνται πλήρως από το είδος τεχνολογίας που χρησιμοποιείται και από την τοποθεσία προορισμού τους, επιτρέποντας έτσι την αναπαραγωγή και αποθήκευση τους μέσα στο διαδίκτυο.

1.2. Κίνητρο

Η πληροφορία που διαδίδεται μέσω διαδικτύου είναι τεράστια και το κόστος για ενοικίαση bandwidth σχετικά ακριβό. Ωστόσο, το χαμηλό κόστος αγοράς ή και ενοικίασης μεγάλου αποθηκευτικού χώρου στο διαδίκτυο επιτρέπει την αναπαραγωγή και αποθήκευση αντικειμένων σε διαφορετικές τοποθεσίες. Ένας χρήστης μπορεί να βρει και να λάβει τα δεδομένα που ζητά από πλησιέστερους κόμβους-αποθήκες παρά από την αρχική πηγή. Αυτό συντελεί στην ελάττωση χρόνου παράδοσης δεδομένων στον χρήστη, στην μείωση φόρτου εργασίας που έχει να επιτελέσει ο αρχικός κόμβος-πηγή και εντέλει στην γενική αποσυμφόρηση του διαδικτύου.

Ο τρόπος με τον οποίον θα επιλεχθούν οι κόμβοι-αποθήκες και το περιεχόμενο που θα ανατεθεί στον καθένα από αυτούς δεν είναι ντετερμινιστικός. Τα Δίκτυα Βασισμένα σε Περιεχόμενο αποτελούν μια νέα αρχιτεκτονική δικτύου για την οποία πρέπει να μελετηθούν αλγόριθμοι διανομής περιεχομένου σε κόμβους-αποθήκες ώστε να επιτυγχάνεται η καλύτερη δυνατή απόδοση δικτύου.

1.3. Ορισμός Προβλήματος

Τα τελευταία χρόνια γίνονται έρευνες για την ανάπτυξη νέων μοντέλων επικοινωνίας και αρχιτεκτονικών με σκοπό τον επανασχεδιασμό του διαδικτύου. Τέτοιες νεοσύστατες υπηρεσίες και αρχιτεκτονικές χρήζουν πειραματισμού ώστε να εξαχθούν τα κατάλληλα συμπεράσματα. Μελετώντας τα αποτελέσματα αυτά μπορεί κανείς να κρίνει εάν μια νέα αρχιτεκτονική είναι σε θέση να αντικαταστήσει πλήρως την προκάτοχό της.

Η δημιουργία ενός λογισμικού που θα επιτρέπει την αυτοματοποίηση πειραμάτων σε μια νέα αρχιτεκτονική κρίνεται απαραίτητη. Ο χρήστης της εφαρμογής θα είναι σε θέση να αναπτύξει το πειραματικό περιβάλλον και να κάνει λήψεις μετρήσεων με εύκολο και γρήγορο τρόπο. Στην παρούσα διπλωματική εργασία βασιζόμαστε στην αρχιτεκτονική PURSUIT-Blackadder, η οποία στηρίζεται στα Δίκτυα Βασισμένα σε Περιεχόμενο και χρησιμοποιεί το μοντέλο επικοινωνίας publish/subscribe.

2. Σχετικές Εργασίες

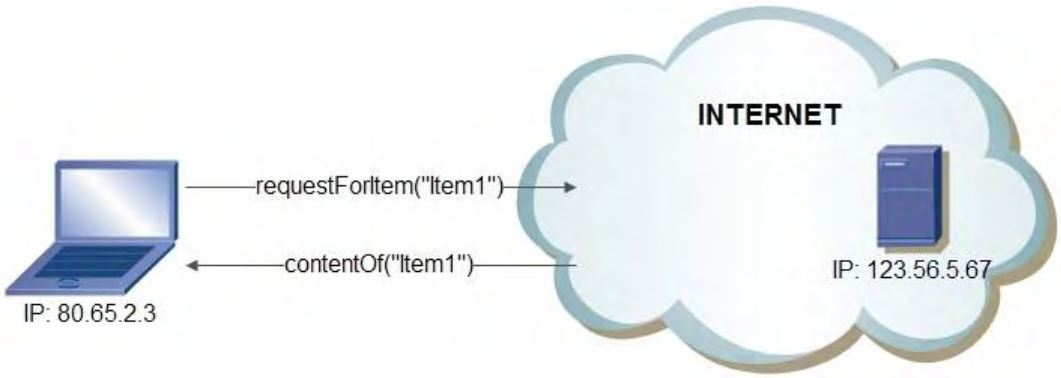
Σε αυτή την ενότητα γίνεται αναφορά σε ερευνητικά προγράμματα που σχετίζονται με την παρούσα διπλωματική εργασία. Αρχικά γίνεται μια εισαγωγή στα Δίκτυα Βασισμένα σε Περιεχόμενο – ICN [1] [2] και στην συνέχεια παρουσιάζονται αρχιτεκτονικές τέτοιων δικτύων όπως CNN και PURSUIT συνοδευόμενες από τις υλοποιήσεις τους.

Έπειτα, γίνεται λόγος για αλγορίθμους Αναπαραγωγής και Αποθήκευσης δεδομένων στο διαδίκτυο οι οποίοι αποτελούν κρίσιμο παράγοντα στην συνολική απόδοση των Δικτύων Βασισμένων στο Περιεχόμενο.

2.1. Δίκτυα Βασισμένα στο Περιεχόμενο (ICN)

Η κυρίαρχη αρχιτεκτονική, μέχρι σήμερα, δικτύου βασίζεται αποκλειστικά στην διευθυνσιοδότηση των κόμβων. Το IP πρωτόκολλο υλοποιεί μια τέτοια αρχιτεκτονική σε επίπεδο διαδικτύου και ως έργο του είναι η μεταφορά και παράδοση πακέτων από την πηγή στον προορισμό. Πιο συγκεκριμένα, όταν ένας κόμβος (π.χ υπολογιστής) ζητά κάποιο πόρο, πρέπει πρώτα να ανακαλύψει ποιος κόμβος(πχ. εξυπηρετητής) κατέχει τον συγκεκριμένο πόρο και στην συνέχεια να τον ζητήσει από αυτόν. Για να μπορεί να αναγνωριστεί οποιοσδήποτε κόμβος στο διαδίκτυο, χρειάζεται δυνητικά μοναδική IP διεύθυνση. Έτσι, κάθε πακέτο που ταξιδεύει στο διαδίκτυο εμπεριέχει τόσο την διεύθυνση του παραλήπτη, όσο και του αποστολέα. Συμπερασματικά, το μοντέλο διευθύνσεων στηρίζεται στο ‟”WHO”, δηλαδή ποιοι επικοινωνούν και στο “WHAT”, τι πληροφορία ανταλλάσουν.

Τα Δίκτυα Βασισμένα στο Περιεχόμενο ξεπερνούν το “ WHO ” και στοχεύουν κατευθείαν στο “ WHAT ”. Μία εφαρμογή ζητά κάποια πληροφορία και το δίκτυο είναι υπεύθυνο να την βρει και να την παραδώσει σε αυτήν. Στην Εικόνα 1 παρουσιάζεται η βασική ιδέα στην οποία στηρίζονται τα δίκτυα βασισμένα στο περιεχόμενο.

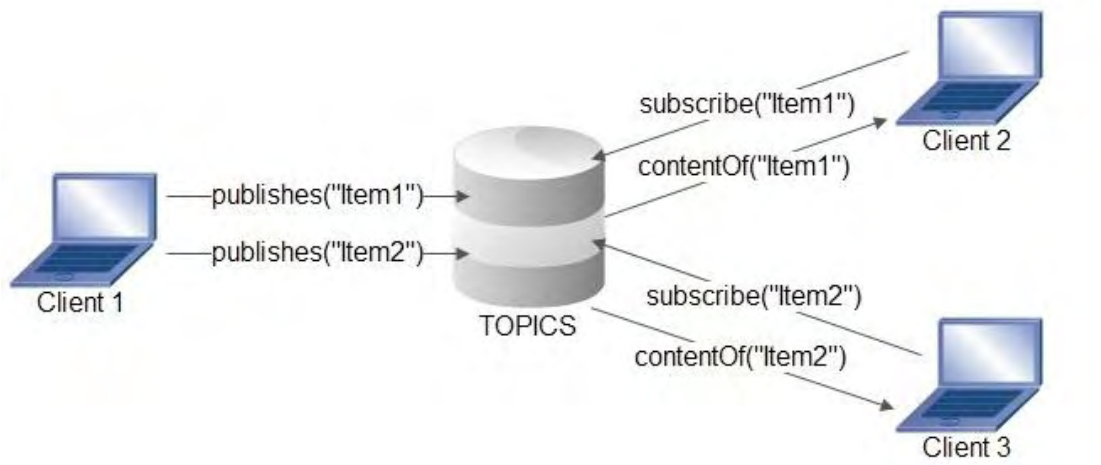


Εικόνα 1. Ο τρόπος λειτουργίας των δικτύων που βασίζονται στο περιεχόμενο. Μια εφαρμογή ζητά πληροφορία “Item1” από το διαδίκτυο και αυτό είναι υπεύθυνο για την εύρεση και παράδοση της στην εφαρμογή.

Αντίθετα με τα address-based networks (ABN) τα οποία διευθυνσιοδοτούν τους κόμβους του δικτύου, τα Information-Centric networks (ICN) διευθυνσιοδοτούν με

μοναδικό όνομα τις συνδέσεις μεταξύ αυτών. Έτσι, τα πακέτα δεν δρομολογούνται χρησιμοποιώντας τις διευθύνσεις κόμβων αλλά τις ονομασίες συνδέσεων.

Γενικά, οι αρχιτεκτονικές CBN (ή ICN ή CCN) είναι βασισμένες σε τεχνικές pull, request/response ή publish/subscribe. Η τελευταία μέθοδος είναι αυτή που χρησιμοποιείται συνήθως για την διάδοση δεδομένων τόσο στην CBN αρχιτεκτονική, όσο και σε πλήθος εφαρμογών στο διαδίκτυο. Αντίθετα με την Client-Server μέθοδο, στην Publish/Subscribe, συνδρομητής και εκδότης είναι χαλαρά συνδεδεμένοι. Καθένας από αυτούς μπορεί να συνεχίσει να λειτουργεί κανονικά, ανεξάρτητα από την κατάσταση στην οποία βρίσκεται ο άλλος. Η ενδιάμεση αποθήκευση δηλαδή προσφέρει την χωρίς συγχρονισμό, χρόνο και χώρο επικοινωνία μεταξύ αυτών.



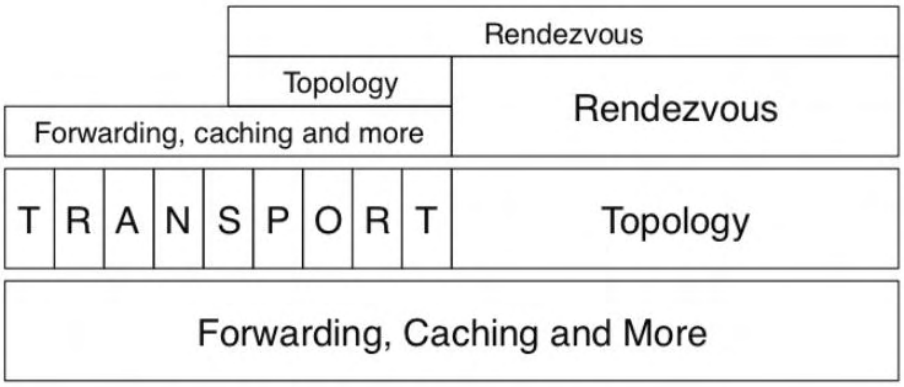
Εικόνα 2. Χαλαρή σύνδεση μεταξύ εκδότη και εγγραφέων. Η ενδιάμεση αποθήκη προσφέρει την χωρίς συγχρονισμό, χρόνο και χρόνο επικοινωνία μεταξύ αυτών.

Διάφορα ήταν τα ερευνητικά προγράμματα που στόχευαν στην ανάπτυξη μιας πρωτοποριακής και ανεξάρτητης αρχιτεκτονικής, κτισμένης από το μηδέν. PSIRP [3], CCNx [4], 4WARD [5], NDN [6] και PURSUIT [7] είναι μερικά από τα προγράμματα που εστιάζουν στην υλοποίηση αρχιτεκτονικής με επίκεντρο την ονομασία δεδομένων αντί τοποθεσιών. Μία από τις υλοποιήσεις τέτοιας αρχιτεκτονικής είναι η RTMF [8] και σε αυτήν θα βασιστούμε στην παρούσα διπλωματική εργασία.

2.1.1. Αρχιτεκτονική RTMF

Στα πλαίσια του PSIRP, το δίκτυο βασισμένο σε περιεχόμενο χρησιμοποιείται για την υλοποίηση μιας νέας αρχιτεκτονικής. Ένας συγκεκριμένος τρόπος μιας τέτοιας υλοποίησης που προέκυψε ονομάζεται RTMF και στηρίζεται αποκλειστικά στην publish/subscribe λειτουργία. Γενικά περιγράφει πώς ένας κόμβος εντάσσεται στο δίκτυο, πως δημοσιεύει δεδομένα και πως εγγράφεται σε αυτά.

Η συγκεκριμένη αρχιτεκτονική αποτελείται από 3 βασικά στοιχεία: Rendezvous, Topology, Forwarding (και Physical Media) από τα οποία πήρε και το όνομα της. Ο γενικός σχεδιασμός της RTMF απεικονίζεται πιο κάτω.



Εικόνα 3. Αρχιτεκτονική RTMF

2.1.2. Blackadder

Το λειτουργικό μοντέλο PURSUIT, βασίζεται άμεσα στο όραμα και στα αποτελέσματα του PSIRP, για να εξάγει την δική του αρχιτεκτονική. Ο Blackadder [9] είναι ένα πρότυπο που υλοποιεί την PURSUIT [10] αρχιτεκτονική και εξάγει ένα πλήρες εξυπηρετικό μοντέλο publish/subscribe για δίκτυα βασισμένα στο περιεχόμενο.

2.1.2.1. Σχεδιαστικές Ιδιότητες

Ο Blackadder βασίζεται σε πέντε βασικές σχεδιαστικές ιδιότητες οι οποίες ανήκουν στο σύνολο προδιαγραφών που μένουν αναλλοίωτες για όλες τις αρχιτεκτονικές δικτύων βασισμένων σε περιεχόμενο.

1. Identifying individual information items

Η πρώτη ιδιότητα είναι τα μέσο για την αναγνώριση ξεχωριστών αντικειμένων πληροφορίας (information items). Χρησιμοποιείται στατιστικά μοναδική, σταθερού μεγέθους ετικέτα για κάθε αντικείμενο ώστε αυτό να μπορεί να αναγνωριστεί. Επίσης, η ταυτοποίηση μπορεί να λάβει χώρα μέσω ιεραρχικών συστημάτων ονομασίας.

2. Scoping

Η δεύτερη ιδιότητα τοποθετεί τα αντικείμενα μέσα σε πλαίσια που καλούνται πεδία εφαρμογής, αλλιώς scopes. Κατά συνέπεια, ένα scope αντιπροσωπεύει ένα σύνολο αντικειμένων και επομένως θεωρείται και το ίδιο σαν ένα αντικείμενο πληροφορίας. Αφού και αυτό χαρακτηρίζεται από μοναδική ετικέτα, μπορεί να ενταχθεί κάτω από άλλα scopes, σχηματίζοντας έτσι σύνθετες δομές πληροφοριών ά-κυκλων κατευθυνόμενων γραφημάτων. Οι γράφοι αυτοί μπορούν να τύχουν επεξεργασίας από κατανεμημένες διεργασίες για οποιοδήποτε σκοπό.

3. Service Model

Η τρίτη ιδιότητα είναι το μοντέλο εξυπηρέτησης, service model, που λειτουργεί σε κάθε πληροφοριακή δομή. Διαμέσου της ροής πληροφοριών μεταξύ οντοτήτων πραγματοποιούνται κατανεμημένες υπολογιστικές διεργασίες που χρησιμοποιούν το εκτεθειμένο εξυπηρετικό μοντέλο. Στον Blackadder προτείνεται το publish/subscribe

service model ενώ μπορούν να υποστηριχτούν και εναλλακτικά μοντέλα, όπως το pull ή το request/response.

4. Main Functions of the Architecture

Η τέταρτη ιδιότητα είναι αυτή του διαχωρισμού των λειτουργιών που είναι υπεύθυνες για την διάδοση πληροφορίας που βρίσκεται μέσα σε ένα scope.

- **Rendezvous** (Ραντεβού)

Κάνει ταίριασμα συνδρομών και δημοσιεύσεων που αφορούν το ίδιο αντικείμενο πληροφορίας. Αυτό οδηγεί σε κάποια μορφή πληροφορίας τοποθεσίας που αφορά το σύνολο εκδοτών και συνδρομητών που έχουν ταιριάξει.

- **Topology Management and Formation** (Διαχείριση και Διαμόρφωση Τοπολογίας)

Χρησιμοποιεί τα αποτελέσματα της Rendezvous για να ορίσει κατάλληλη σχέση παράδοσης για την μεταφορά της πληροφορίας.

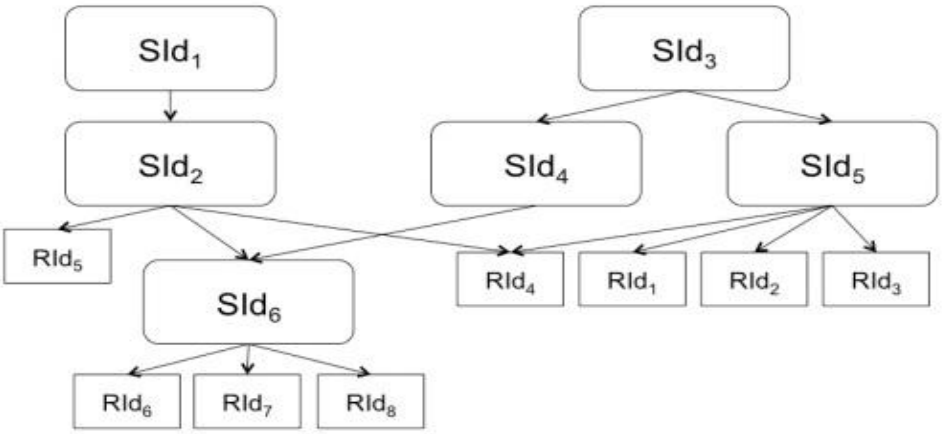
- **Forwarding** (Προώθηση)

Χρησιμοποιεί τα αποτελέσματα του Topology Formation και προωθεί τα δεδομένα.

5. Dissemination Strategy (Στρατηγική Διάδοσης)

Διευθύνει τις μεθόδους που χρησιμοποιούνται για την εφαρμογή των πιο πάνω κύριων λειτουργιών. Διευθύνει επίσης και θέματα που αφορούν την διαχείριση των πληροφοριών σε κάποια τμήματα της πληροφοριακής δομής. Για τον λόγο αυτό ορίζονται στρατηγικές διάδοσης της πληροφοριακής δομής ή μέρους αυτής. Ορίζοντας συγκεκριμένες στρατηγικές σε υποχώρους της δομής, μπορεί να βελτιστοποιηθεί η συνολική διαδικασία επικοινωνίας.

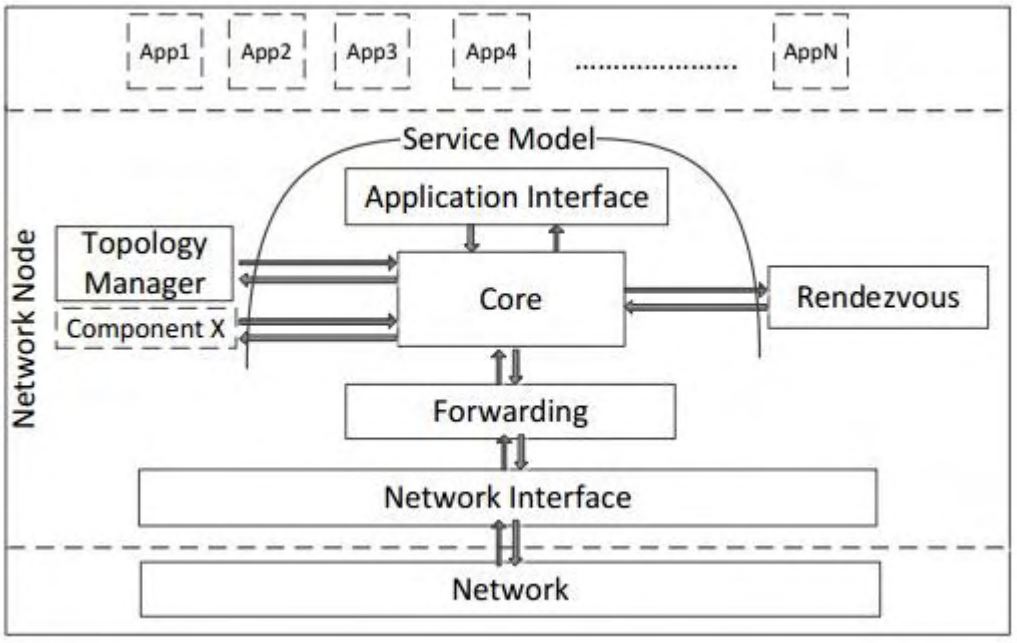
Παρακάτω, παρουσιάζεται ένα στιγμιότυπο μιας πληροφοριακής δομής. Με Sid χαρακτηρίζονται τα scopes ενώ με Rid τα information items. Ένα scope ή item μπορεί να προστεθεί στην δομή κάτω από ένα άλλο scope δημιουργώντας σύνθετες δομές πληροφορίας. Κάθε item ή scope αναγνωρίζεται με το πλήρες μονοπάτι που οδηγεί σε αυτό ξεκινώντας από την ρίζα. Για παράδειγμα, δεξιά το Rid3 αναγνωρίζεται ως Sid3/Sid5/Rid3 ενώ αριστερά το Rid7 μπορεί να αναγνωριστεί από διαφορετικά μονοπάτια, ξεκινώντας από το Sid1 και Sid3 αντίστοιχα. Με τον τρόπο αυτό, ένα συγκεκριμένο scope ή item μπορεί να αναγνωριστεί από διαφορετικά μονοπάτια.



Εικόνα 4. Πληροφοριακή Δομή σε μορφή γραφήματος. Τα scopes χαρακτηρίζονται ως Sid ενώ τα items ως Rid.

2.1.2.2. Αρχιτεκτονική κόμβου

Χρησιμοποιώντας και διασυνδέοντας elements από το εργαλείο Click Modular Router [11], ο Blackadder υλοποιεί το απαιτούμενο μοντέλο δρομολογητή το οποίο και τρέχει σε κάθε κόμβο του ICN δικτύου. Κάθε πακέτο που φθάνει στον κόμβο, ανάλογα με τη φύση του, ακολουθεί μια διαδρομή διαμέσου των κατάλληλων elements. Κάθε element παρέχει και μια λειτουργία, συμπεριλαμβανομένων και των πλέον βασικών που έχουμε προαναφέρει. Έτσι κάθε element με την σειρά του επεξεργάζεται ένα πακέτο και ανάλογα το προωθεί σε κάποιο άλλο element, εφαρμογή ή στο διαδίκτυο. Στην Εικόνα 5 φαίνεται η συνδεσμολογία των elements που χρησιμοποιεί ο Blackadder.



Εικόνα 5. Αρχιτεκτονική κόμβου Blackadder.

2.1.2.3. Βασικές λειτουργίες στοιχείων

- **Rendezvous**

Παραλαμβάνει και επεξεργάζεται όλες τις αιτήσεις που δημοσιεύονται από εφαρμογές που εκτελούνται τοπικά ή απομακρυσμένα. Τέτοιες αιτήσεις δημιουργούνται μέσω του εξαγόμενου μοντέλου εξυπηρέτησης. Χρησιμοποιώντας την τοπική πληροφοριακή δομή, ταιριάζει publishers και subscribers των οποίων οι αιτήσεις αφορούν στο ίδιο information item και εξάγει μια δομή πληροφορίας που αντιστοιχεί στην τοποθεσία αυτών. Οι ακριβείς πράξεις της λειτουργίας Rendezvous ορίζονται μέσω των στρατηγικών διάδοσης που βρίσκονται στην πληροφοριακή δομή και ανάλογα μπορεί να απαιτηθεί η επιβολή καθολικής μοναδικότητας αναγνωριστικών. Επιπλέον, ανάλογα με την στρατηγική διάδοσης μπορεί να ελαχιστοποιηθεί η λειτουργία της Rendezvous.

- **Topology Management and formation**

Υλοποιεί την διαχείριση της συνολικής τοπολογίας παράδοσης και την διαμόρφωση συγκεκριμένων γραφημάτων παράδοσης για τις σχέσεις μεταξύ Publishers και

Subscribers. Γι' αυτό, ο TM ενημερώνει την τοπολογία σύμφωνα με την σύνδεση ή αποσύνδεση ενός κόμβου από το δίκτυο και δημιουργεί την απαραίτητη πληροφορία προώθησης όταν της ζητηθεί. Για παράδειγμα, όταν ο Rendezvous ταιριάξει μια αίτηση publish/subscribe, ζητάει από τον TM να δημιουργήσει μονοπάτι παράδοσης από τον publisher στον subscriber. Αυτό όμως δεν συμβαίνει πάντα, αφού για συγκεκριμένες στρατηγικές διάδοσης, πχ link-local, η δημιουργία μονοπατιού δεν είναι απαραίτητη και ο TM δεν λαμβάνει χώρα.

- **Forwarding**

Λαμβάνει δημοσιεύσεις και τις προωθεί είτε στον τοπικό κόμβο, είτε στο διαδίκτυο. Για να συμβεί αυτό, κρατάει όλες τις απαραίτητες καταστάσεις σε ένα πίνακα προώθησης και για να επιτύχει την λειτουργία του μπορεί να χρειαστεί συντονισμός με τον Topology Manager.

2.1.2.4. **Μοντέλο Υπηρεσίας** (Service Model)

Όπως φαίνεται και στην Εικόνα 5 , οι εφαρμογές όπως επίσης και τα εσωτερικά στοιχεία που βρίσκονται στον κόμβο αλληλεπιδρούν μεταξύ τους και με το δίκτυο μέσω του εξυπηρετικού μοντέλου που προσφέρεται σαν εσωτερικό API. Το συγκεκριμένο μοντέλο επιτρέπει τον χειρισμό πληροφοριακών ροών με απλές σημασιολογικές ρουτίνες publish/subscribe.

Publishing & Un-publishing scope

Ένας publisher μπορεί να δημιουργήσει ένα scope στην πληροφοριακή δομή εκδίδοντας το αίτημα

- publishScope(string id, string prefix, strategy s)

Μετά την δημοσίευση ενός scope, η Rendezvous λειτουργία ενημερώνει όλους τους subscribers που έχουν κάνει subscribe στο scope το οποίο αποτελεί πατέρα του scope που μόλις δημοσιεύτηκε. Εάν το prefix scope δεν υπάρχει στην πληροφοριακή δομή, το αίτημα απορρίπτεται.

Όταν ένας publisher αποφασίσει να σταματήσει την δημοσίευση ενός scope, εκδίδει το αίτημα

- unpublishScope(string id, string prefix, strategy s)

Η Rendezvous λειτουργία θα σταματήσει για τον συγκεκριμένο publisher την δημοσίευση όλων των information items τα οποία βρίσκονται κάτω από το συγκεκριμένο scope. Αν αυτό πετύχει και δεν υπάρχει άλλο scope κάτω από αυτό, τότε διαγράφεται. Εάν το scope ήταν δημοσιευμένο κάτω από πολλαπλά scopes, τότε μόνο ο συγκεκριμένος κλάδος του γραφήματος διαγράφεται. Όλοι οι subscribers που είναι εγγεγραμμένοι σε scopes(s) που είχαν πατέρα αυτό το scope, θα ενημερωθούν.

Publishing & Un-publishing information items

Ένας publisher μπορεί να δημοσιεύσει στοιχεία πληροφορίας εκδίδοντας το αίτημα

- publishInfo(strind id, string prefix, strategy s)

Τα στοιχεία πληροφορίας μπορούν να τοποθετηθούν κάτω από περισσότερα του ενός scopes, δεδομένου πώς κάποιο από τα πατρικά scopes έχει περισσότερους από ένα πατέρα.

Για να σταματήσει η δημοσίευση του στοιχείου πληροφορίας εκδίδεται το αίτημα

- unpublishInfo(string id, string prefix, strategy s)

Σαν αποτέλεσμα, ο publisher που εκτέλεσε το αίτημα, διαγράφεται από τον ρόλο του publisher στο συγκεκριμένο στοιχείο πληροφορίας. Εάν δεν υπάρχουν άλλοι publishers ή subscribers στο στοιχείο αυτό, τότε αυτό διαγράφεται από τον γράφο πληροφορίας. Αλλιώς, η Rendezvous λειτουργία λαμβάνει χώρο ώστε να βρεθούν ένας ή περισσότεροι publishers για να δημοσιεύσουν δεδομένα για το συγκεκριμένο στοιχείο. Επιπλέον, εάν το στοιχείο πληροφορίας δημοσιεύεται κάτω από πολλαπλά scopes, τότε αυτό διαγράφεται μόνο από το scope που δηλώνεται στο prefix (θεωρώντας ότι δεν υπάρχουν άλλοι publishers ή subscribers για το αναγνωριστικό αυτό).

Subscribing and Un-subscribing scope

Ένας subscriber μπορεί να εγγραφεί σε ένα scope εκδίδοντας το αίτημα

- subscribeScope(string id, string prefix, strategy s)

Η Rendezvous λειτουργία λαμβάνει το αίτημα και εάν το scope δεν υπάρχει στον γράφο πληροφορίας τότε το δημιουργεί. Με αυτόν τον τρόπο ο αιτών δηλώνει το ενδιαφέρον του αυτόματα για όλα τα scopes και στοιχεία που βρίσκονται αμέσως κάτω από αυτό το scope. Έτσι, η λειτουργία Rendezvous ελέγχει τι υπάρχει κάτω από το συγκεκριμένο scope. Για κάθε scope στέλνει ενημέρωση για την ύπαρξη του στον αιτών. Για κάθε αντικείμενο πληροφορίας, οι publishers που προηγουμένως το δημοσίευαν θα ενημερωθούν.

Εκδίδοντας αίτημα

- unsubscribeScope(string id, string prefix, strategy s)

εάν δεν υπάρχει κάποιος publisher ή subscriber για κάποιο στοιχείο ή scope κάτω από το συγκεκριμένο scope, τότε αυτό διαγράφεται από τον γράφο πληροφορίας.

Subscribing and Un-subscribing from information items

Εκδίδοντας αίτημα

- subscribeInfo(string id, string prefix, strategy s)

εάν ένας publisher έχει διαφημίσει προηγουμένως το συγκεκριμένο αντικείμενο πληροφορίας, τότε η Rendezvous τον ταιριάζει με τον αιτών subscriber.

Αντίστοιχα, εκδίδοντας αίτημα

- unsubscribeInfo(string id, string prefix, strategy s)

ο Rendezvous διαγράφει τον συγκεκριμένο subscriber από το αντικείμενο. Εάν δεν υπάρχει άλλος subscriber ή publisher στο συγκεκριμένο αντικείμενο πληροφορίας, τότε διαγράφεται και αυτό από τον γράφο.

Publishing Data

Ένας publisher στέλνει δεδομένα για ένα συγκεκριμένο αντικείμενο πληροφορίας εκδίδοντας το αίτημα

- publishData(string id, strategy s, char *data)

Το id είναι το RID του στοιχείου πληροφορίας αρχίζοντας από την ρίζα του γράφου πληροφορίας. Τέτοια αιτήματα στέλνονται από τον publisher αφού λάβει πρώτα μήνυμα από την Rendezvous με στοιχεία προώθησης των δεδομένων.

Asynchronous upcalls

Στις περισσότερες περιπτώσεις, οι αιτήσεις που στέλνονται από μία εφαρμογή στο δίκτυο προκαλούν στο να παρθούν επιπλέον ενέργειες από την Rendezvous και τον Topology Manager. Οι ενέργειες αυτές μπορεί εντέλει να απαιτήσουν κοινοποιήσεις προς τους κόμβους που έκαναν μια αίτηση, και εν συνεχεία να ειδοποιήσουν μία ή περισσότερες εφαρμογές. Οι κοινοποιήσεις αυτές είναι ασύγχρονες.

- New Scope notification

Στέλνεται σε ένα subscriber όταν ένα νέο scope έχει δημιουργηθεί κάτω από το scope στο οποίο έχει γίνει subscribe ο αναφερόμενος. Η κοινοποίηση περιέχει το αναγνωριστικό του scope.

- Deleted Scope notification

Όταν ένα scope σταματά να δημοσιεύεται και διαγράφεται από τον γράφο πληροφορίας, οι subscribers που είναι εγγεγραμμένοι στο πατρικό αυτού του scope ενημερώνονται. Η κοινοποίηση περιέχει το αναγνωριστικό του διαγραφέντος scope.

- Start Publish notification

Στέλνεται σε μία εφαρμογή, που προηγουμένως είχε διαφημίσει ένα στοιχείο πληροφορίας ώστε να ξεκινήσει την αποστολή δεδομένων που αφορούν αυτό. Η κοινοποίηση περιέχει το αναγνωριστικό της πληροφορίας που πρόκειται να δημοσιευτεί.

- Stop Publish notification

Στέλνεται σε μια εφαρμογή όταν δεν υπάρχουν διαθέσιμοι subscribers για κάποιο στοιχείο πληροφορίας, για το οποίο έχει σταλεί προηγουμένως Start Publish κοινοποίηση.

- Published Data notification

Στέλνεται σε ένα subscriber όταν έχει φτάσει πληροφορία για μια συγκεκριμένη δημοσίευση. Η κοινοποίηση περιέχει το αναγνωριστικό της δημοσίευσης.

Dissemination strategies

Σε κάθε scope ανατίθεται μια στρατηγική διάδοσης. Τρία είδη στρατηγικών υπάρχουν.

- Node-local strategy

Επιτρέπει την δημιουργία γραφήματος πληροφορίας εντός κόμβου. Ο γράφος που διατηρείται από την Rendezvous είναι ορατός μόνο στις εφαρμογές και στα Click elements που τρέχουν στον συγκεκριμένο κόμβο. Η λειτουργία Topology Management and Formation είναι περιορισμένη αφού πρέπει να βρει τους ενδιαφερόμενους publishers και subscribers που βρίσκονται τοπικά.

- Link-local

Επιτρέπει σε ένα κόμβο του δικτύου να διαδίδει πληροφορίες μόνο στους άμεσα συνδεδεμένους μέσω φυσικής ζεύξης γείτονές του. Δεν χρειάζεται διατήρηση γράφου πληροφορίας οπότε η λειτουργία της Rendezvous είναι περιορισμένη. Οι subscribers εγγράφονται σιωπηρά σε συγκεκριμένα στοιχεία πληροφορίας. Ένας publisher μπορεί να δημοσιεύσει στοιχεία σε συγκεκριμένες ζεύξεις ή και σε όλες (multicast). Αν ένας subscriber υπάρχει στην ζεύξη, η πληροφορία προωθείται στην ενδιαφερόμενη εφαρμογή. Επειδή απαιτείται μόνο εύρεση κατάλληλης ζεύξης, η λειτουργία του Topology Manager είναι και πάλι περιορισμένη.

- Domain-local

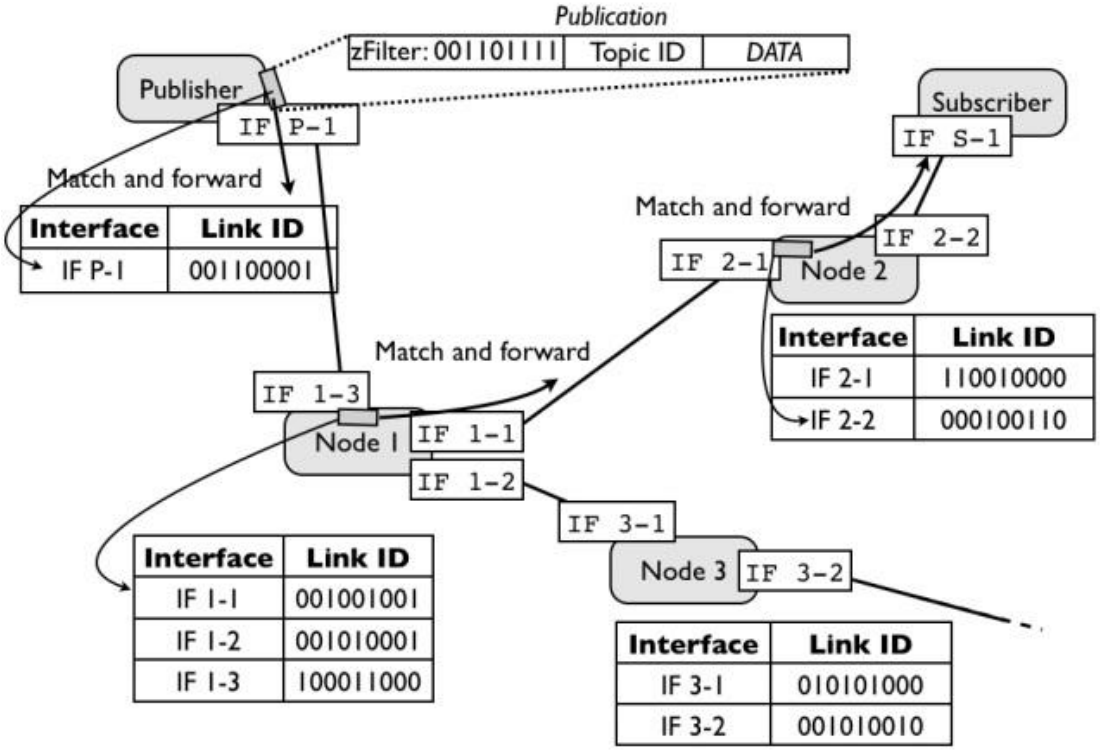
Σε αυτή την στρατηγική όλες οι λειτουργίες είναι σε πλήρη λειτουργικότητα. Η λειτουργία Rendezvous δρα σε έναν ή περισσότερους κόμβους. Η λειτουργία Topology Management and Formation δρα επίσης σε έναν η περισσότερους κόμβους και επικεντρώνεται στην παρακολούθηση του δικτύου δημιουργώντας δέντρα πολύ-εκπομπής για σύνολα publishers προς subscribers χρησιμοποιώντας αλγόριθμους εύρεσης συντομότερων μονοπατιών. Τέτοια δέντρα σχηματίζονται από υπολογισμούς που βασίζονται στα Bloom Filters [12], τα οποία χρησιμοποιούν μοναδικό αναγνωριστικό για κάθε ζεύξη. Πιο συγκεκριμένα, εκτελώντας την δυαδική πράξη OR σε όλες τις ζεύξεις του μονοπατιού από την πηγή στον προορισμό , ένα δέντρο πολύ-εκπομπής δημιουργείται από τον TM. Έτσι κάθε Forwarding element που βρίσκεται το μονοπάτι προωθεί αποδοτικά κάθε πακέτο εκτελώντας την δυαδική πράξη AND μεταξύ του δέντρου πολύ-εκπομπής που φέρει το πακέτο και του κάθε interface εξόδου.

Ο αλγόριθμος προώθησης πακέτων από έναν κόμβο, όπως επίσης και μια αναπαράσταση των δέντρων πολύ-εκπομπής και των interfaces των κόμβων φαίνονται στις Εικόνες 6 και 7 αντίστοιχα.

```
Input: Link IDs of the outgoing links; zFilters in the packet header

foreach  Link ID of outgoing interface do
      if zFitler & Link ID == Link ID then
            Forward packet on the link
      end
end
```

Εικόνα 6. Αλγόριθμος προώθησης που εκτελεί η λειτουργία Forwarding σε κάθε πακέτο που φθάνει στον κόμβο, ώστε να προωθηθεί στην κατάλληλη σύνδεση.

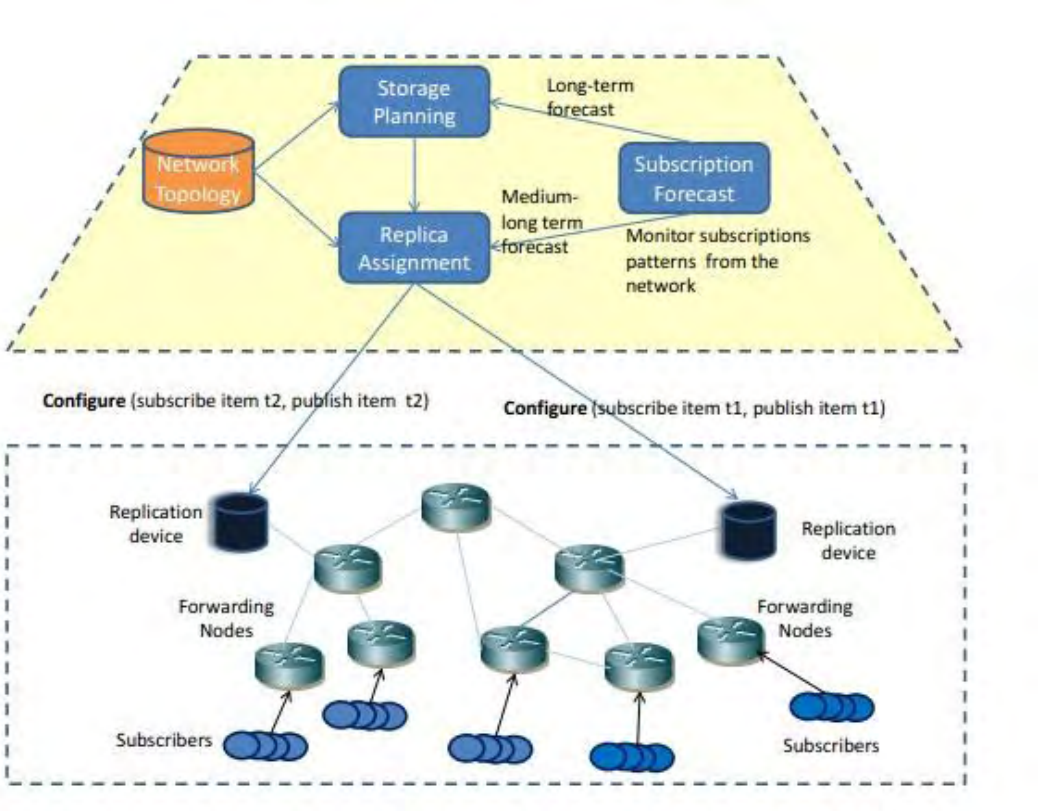


Εικόνα 7. Προώθηση ενός πακέτου με βάση το zFilter (BloomFilter) που διαθέτει.

3. CACHING

3.1. Σχεδιασμός Αρχιτεκτονικής Caching σε ICN

Η αποδοτική αναπαραγωγή και αποθήκευση πληροφορίας σε κόμβους του Information Centric δικτύου είναι το κλειδί για την βελτιστοποίηση της διάδοσης πληροφορίας μεταξύ άκρων και την ελαχιστοποίηση του συνολικού φόρτου κίνησης σε αυτό. Οι δύο αυτοί μηχανισμοί, Storage Planning και Replica Assignment [13], θα πρέπει να παρέχονται από σχετικά λειτουργικά στοιχεία. Τα εξαρτήματα αυτά συνήθως είναι εγκατεστημένα εκτός δικτύου και τρέχουν off-line αλγορίθμους σε δύο διαφορετικές αλλά μακρο-μεσοχρονικές κλίμακες. Τα δομικά στοιχεία και οι αλληλεπιδράσεις μεταξύ αυτών παρουσιάζονται στην Εικόνα 8.



Εικόνα 8. Αρχιτεκτονική των φάσεων Planning και Off-line Assignment.

- **Storage Planning Phase**

Παίρνει ως είσοδο τον αριθμό των διαθέσιμων κόμβων αποθήκευσης που ο διαχειριστής επιθυμεί να εγκαταστήσει, την τοπολογία του δικτύου και μία μακροπρόθεσμη πρόβλεψη των συνδρομών στο δίκτυο. Μπορεί να τρέξει περιοδικά και να αποφασίσει την βέλτιστη τοποθέτηση των αποθηκών σε ένα μακροπρόθεσμο χρονικό διάστημα (π.χ μια φορά τον χρόνο) ή πιο σύντομα όταν η τρέχουσα θέση των κόμβων αποθήκευσης οδηγεί σε αναποτελεσματική ανάπτυξη λόγω σημαντικών αλλαγών συνδρομών που δεν έχουν προβλεφθεί επιτυχώς από το Subscription Forecast στοιχείο. Η εκτέλεση και επιβολή της απόφασης του Storage Planning περιλαμβάνει συνήθως υψηλού επιπέδου επιχειρηματικές αποφάσεις, δεδομένου ότι η

μεταφορά μιας συσκευής αποθήκευσης σε μία διαφορετική φυσική τοποθεσία ή η επέκταση του αριθμού τέτοιων συσκευών έχει υψηλό κόστος.

- **Replica Assignment**

Παίρνει ως είσοδο την τοπολογία δικτύου, τα αποτελέσματα από την φάση Storage Planning σχετικά με τις θέσεις των συσκευών αποθήκευσης που είναι εγκατεστημένες σε αυτό, και την μεσοπρόθεσμη/μακροπρόθεσμη πρόβλεψη σχετικά με τις συνδρομές. Τρέχει περιοδικά σε μεσοπρόθεσμη (ή μακροπρόθεσμη) κλίμακα και είναι σε θέση να επιβάλει αυτόματα την απόφαση της στο δίκτυο. Η μετεγκατάσταση των αντιγράφων μπορεί να εκτελεστεί καθοδηγώντας τους κόμβους αποθήκευσης να εγγράφονται σε ένα διαφορετικό σύνολο αντικειμένων πληροφορίας. Η καθοδήγηση πραγματοποιείται σαν μία δημοσίευση ενός αντικειμένου πληροφορίας στο οποίο έχουν εγγραφεί οι κόμβοι αποθήκευσης.

- **Subscription Forecast**

[13] Παρακολουθεί τις συνδρομές που λαμβάνουν χώρα στο δίκτυο και δημιουργεί προβλέψεις για το Storage Planning και Replica Assignment σε μακροπρόθεσμη και μέσο - μακροπρόθεσμη περίοδο αντίστοιχα. Γενικά οι κόμβοι αποθήκευσης συμπεριφέρνονται ως εκδότες και ως συνδρομητές ταυτοχρόνως. Κάνουν εγγραφή ώστε να λαμβάνουν νέες εκδοχές αντικειμένων και στην συνέχεια τα δημοσιεύουν οι ίδιοι. Με τον τρόπο αυτό, όταν ένας συνδρομητής κάνει εγγραφή σε ένα συγκεκριμένο αντικείμενο, η λειτουργία Rendezvous σε συνεργασία με τον Topology Manager and Formation καθοδηγούν, σύμφωνα με την πολιτική του φορέα, έναν ή περισσότερους εκδότες να δημοσιοποιήσουν τις πληροφορίες που σχετίζονται με το συγκεκριμένο αντικείμενο.

Δεδομένου ότι οι περίοδοι που λαμβάνουν χώρα οι προβλέψεις συνδρομών μπορούν να είναι μεγάλες (πχ μια φορά την βδομάδα), οι συνδρομές ενδέχεται να τροποποιούνται σημαντικά στο διάστημα αυτό. Για τον λόγο αυτό, παράγονται διάφορες ρυθμίσεις οι οποίες καθορίζουν την περίοδο προβλέψεων ανάλογα με την απόδοση του δικτύου. Η εφαρμογή διαφορετικών ρυθμίσεων συνεπάγει κάποιο κόστος το οποίο πρέπει να λαμβάνεται υπόψη όταν αποφασίζεται μία τέτοια μετάβαση.

3.2. Υλοποίηση της Αρχιτεκτονικής Caching σε ICN

Στην ενότητα 3.2.1. γίνεται αναφορά στην στρατηγική τοποθέτησης και ανάθεσης αντιγράφων στο δίκτυο η οποία έχει υλοποιηθεί στα πλαίσια άλλης διπλωματικής εργασίας.

3.2.1. Στρατηγική Τοποθέτησης και Ανάθεσης Αντιγράφων

Αλγόριθμοι τοποθέτησης που παρουσιάστηκαν στο πλαίσιο των CDN [14] και [15] χρησιμοποιούν πληροφορίες για τον φόρτο εργασίας όπως η καθυστέρηση(συναρτήσει της απόστασης από τα σημεία της αποθήκευσης) και τα ποσοστά αιτήσεων για να πάρουν την απόφαση τοποθέτησης. Βασικό συμπέρασμα είναι πώς ο ‘‘άπληστος’’ αλγόριθμος βασιζόμενος σε μετρικό απόστασης και φόρτο αιτημάτων παρουσιάζει την καλύτερη και πολύ κοντά στην βέλτιστη λύση.

- **Άπληστος Αλγόριθμος**

Γενικά, υποθέτοντας πως πρέπει να επιλέξουμε M αντίγραφα ανάμεσα σε N πιθανές θέσεις, προχωράμε ως εξής. Σε κάθε επανάληψη αξιολογούμε κάθε μία από τις N-i θέσεις για να προσδιοριστεί η καταλληλότητα της για την φιλοξενία του αντιγράφου(i είναι ο αριθμός επανάληψης 0,…,M-1). Υπολογίζουμε το κόστος που σχετίζεται με την κάθε θέση και επιλέγουμε αυτή που έχει το χαμηλότερο. Σημείωση πως η συγκεκριμένη θέση θα πρέπει να είναι προσβάσιμη από όλους τους πελάτες. Σε κάθε επόμενη επανάληψη, σαν επόμενη θέση αντιγράφου επιλέγεται αυτή που σε συνδυασμό με όλες τις προεπιλεγμένες, αποδίδει το χαμηλότερο κόστος. Η διαδικασία συνεχίζεται μέχρι να επιλεχθούν και οι M θέσεις αντιγράφων. Γενικά, εδώ κάθε πελάτης θεωρείται πως απευθύνεται στον πλησιέστερο σε αυτόν κόμβο-αποθήκη για την απόκτηση κάποιου αντιγράφου.

Στον συγκεκριμένο αλγόριθμο θεωρούμε πως υπάρχει μόνο μια κατηγορία περιεχομένου στο σύστημα μας(ένα θέμα), ή ισοδύναμα δεν υπάρχει διάκριση όσο αφορά το περιεχόμενο. Ως ri θεωρούμε τη ζήτηση (αιτήματα/sec) από τους πελάτες που βρίσκονται στον κόμβο i. Επίσης ως pij θεωρούμε το ποσοστό ζήτησης αιτημάτων που ενώ προορίζονται για τον κόμβο j, βρίσκουν το περιεχόμενο που ζητούν στον ενδιάμεσο κόμβο i. Η καθυστέρηση διάδοσης σε άλματα από τον κόμβο i στον διακομιστή προορισμού j ορίζεται ως dij. Αφού το ποσοστό πληροφορίας pij δεν χρειάζεται να διασχίσει την απόσταση dij, η συνολική μείωση κίνησης στο δίκτυο gij = pij * dij

$$d_{ij} \cdot \sum_{l=1}^{N} R_l$$

$$R_l = \begin{cases} r_l & \text{if } i \text{ is on the path from } l \text{ to } j \\ 0 & \text{otherwise.} \end{cases}$$

Εικόνα 9. Κάθε πελάτης li που βρίσκει το περιεχόμενο που ζητά στον κόμβο i, αποφεύγει να διανύσει την απόσταση dij. Οπότε η συνολική απόσταση που δεν καλύπτεται από όλους τους πελάτες l είναι αυτή που φαίνεται σε αυτή την εικόνα.

- **Τροποποιημένος άπληστος αλγόριθμος**

Στην αρχιτεκτονική δικτύου publish/subscribe που υποθέτουμε σε αυτό το κείμενο, η έννοια του διακομιστή προέλευσης – ο οποίος είναι ζωτικής σημασίας για τον άπληστο αλγόριθμο – δεν υπάρχει. Οι publishers συνδέονται στο δίκτυο, δημοσιεύουν το περιεχόμενο τους και αποσυνδέονται. Συνεπώς ο αλγόριθμος πρέπει να τροποποιηθεί. Συγκεκριμένα, η παραπάνω διαδικασία επαναλαμβάνεται N φορές υποθέτοντας κάθε φορά σαν διακομιστή προορισμού και ένα διαφορετικό κόμβο του δικτύου. Με τον τρόπο αυτό παίρνουμε N διανύσματα k πιθανών αποθηκών. Κάθε διάνυσμα έχει N στοιχεία με k άσσους στους δείκτες των επιλεγμένων αποθηκών και N-k μηδενικά σε κάθε άλλη θέση. Για παράδειγμα το διάνυσμα [0 0 0 1 0 1] δείχνει ότι σε ένα δίκτυο N=6 κόμβων, οι k =2 κόμβοι, 4 και 6, έχουν επιλεχθεί σαν αποθήκες. Τέλος, επιλέγουμε ως αποθήκες τους k κόμβους που εμφανίστηκαν περισσότερες φορές στο ανά στοιχείο άθροισμα των N διανυσμάτων. Ο συγκεκριμένος αλγόριθμος στο παρόν κείμενο προϋποθέτει ομοιόμορφη κατανομή της πιθανότητας, μεταξύ των N κόμβων δικτύου, στο να μπορούν να συμβούν

δημοσιεύσεις. Φυσικά θα μπορούσαν να χρησιμοποιηθούν και άλλες μορφές κατανομών πιθανοτήτων, και κάθε διάνυσμα θα πρέπει πρώτα να σταθμίζεται με την πιθανότητα που του αντιστοιχεί πριν από την ανά στοιχείο άθροιση των N διανυσμάτων.

- **Αλγόριθμος Τοποθέτησης και Ανάθεσης Αντιγράφων για Δίκτυα Δημοσιεύσεων/Συνδρομών**

Χρησιμοποιούμε τον τροποποιημένο άπληστο αλγόριθμο για τοποθέτηση και ανάθεση αντιγράφων σε κόμβους όταν στο δίκτυο υπάρχουν T διαφορετικές κατηγορίες θεμάτων. Στην συνέχεια παρουσιάζονται τα βήματα του προτεινόμενου αλγορίθμου ενώ στην πιο κάτω εικόνα φαίνονται οι απαραίτητες παράμετροι.

r_i^t	: request rate for topic $t \in T$ in broker i
N	: number of nodes (brokers) in the network
M	: $(M < N)$ number of storages in the network
k_t	: $(k_t \leq M)$ replication of each topic $t \in T$ in the network
L	: storage capacity of each storage point in the network
T	: number of classes of content (topics)
w_t	: weight of topic $t \in T$ in the network
$\mathcal{S}$	: storage brokers vector
s_t	: possible stores vector for topic $t \in T$
ξ_t	: relative weight of topic $t \in T$

Εικόνα 10. Παράμετροι αλγορίθμου Τοποθέτησης και Ανάθεσης αντιγράφων για Δίκτυα δημοσιεύσεων/συνδρομών.

- **Βήμα 1**

Για κάθε θέμα t ϵ T εκτελούμε τον τροποποιημένο άπληστο αλγόριθμο και παίρνουμε T διανύσματα των πιθανών αποθηκών St. Για παράδειγμα :

sa = [0 3 5 0 2 2]
sb = [0 2 5 0 5 0]
sc = [0 2 5 0 5 0]

είναι τα τρία διανύσματα, ένα για κάθε θέμα t. Το [0 3 5 0 2 2] σημαίνει πως από τις N=6 εκτελέσεις του τροποποιημένου άπληστου αλγορίθμου, ο κόμβος 2 εμφανίστηκε 3 φορές, ο κόμβος 3 5 φορές, ο κόμβος 1 καμία φορά κ.ο.κ.

- **Βήμα 2**

Κάθε διάνυσμα st σταθμίζεται από το

$$w_t = \frac{\sum_{i=1}^{N} r_i^t}{\sum_{i=1}^{N} \sum_{t=1}^{T} r_i^t}.$$

Το wt δείχνει την δημοτικότητα ενός θέματος αναφορικά με τη ζήτηση κυκλοφορίας του στο δίκτυο. Οι συντελεστές στάθμισης για το συγκεκριμένο παράδειγμα είναι οι εξής:

wa= 17/50= 0.34 , wb= 27/50= 0.54, wc= 6/50= 0.12

Λαμβάνουμε τα ακόλουθα σταθμισμένα διανύσματα:
wa * sa = [0 1.02 1.7 0 0.68 0.68]
wb * sb = [0 1.08 2.7 0 2.7 0]
wc * sc = [0 0.24 0.6 0 0.6 0]

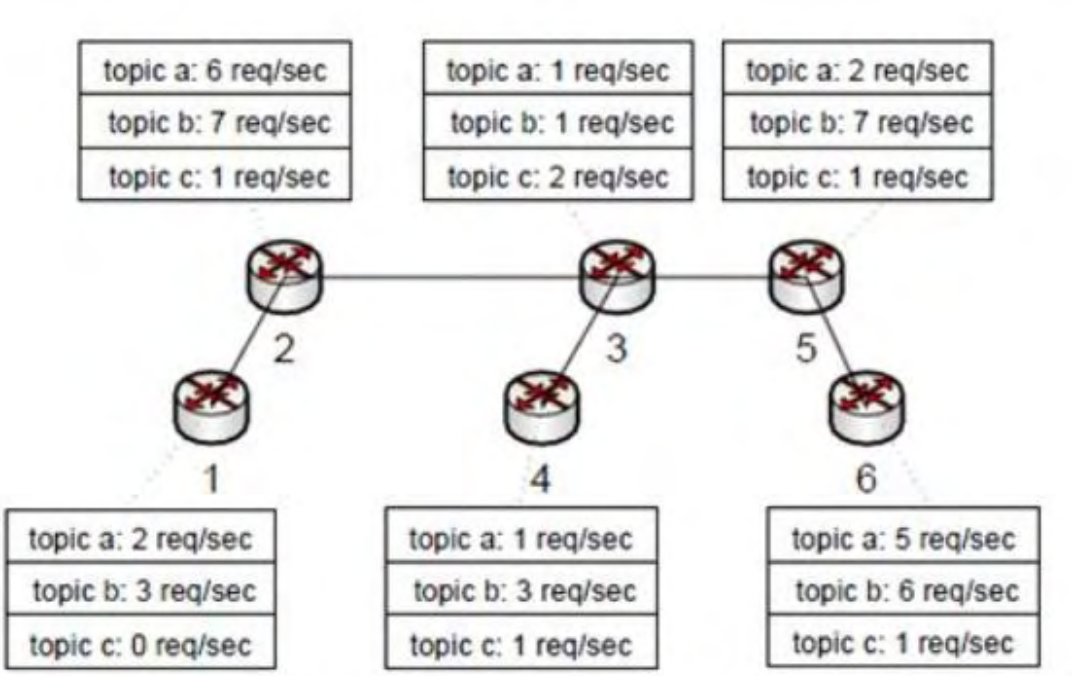
- **Βήμα 3**

Επιλέγουμε ως αποθήκες τους M κόμβους που εμφανίστηκαν τις περισσότερες φορές στην ανά στοιχείο σταθμισμένη άθροιση των T διανυσμάτων. Καλούμε αυτό το διάνυσμα ως brokers' storage vector S. Η άθροιση ανά στοιχείο των τριών, πιο πάνω, διανυσμάτων ισούτε με [0 2.34 5 3.98 0.68] υποδεικνύοντας τους κόμβους 3,5 και 2 σαν αποθήκες στο S, δεδομένου ότι M=3.

- **Βήμα 4**

Για κάθε θέμα t, ξεκινώντας από το δημοφιλέστερο, αναθέτουμε kt αποθήκες κυκλικά ακολουθώντας την παρακάτω διαδικασία:

- Για κάθε εγγραφή στο st του θέματος t που έχει υπολογιστεί στο Βήμα 1 ανάθεσε μια αποθήκη την φορά, εάν η καταχώρηση εμφανίζεται επίσης στο S, το οποίο υπολογίστηκε στο Βήμα 3, και μόνο αν στην εν λόγω αποθήκη έχουν ανατεθεί κάτω από L (χωρητικότητα αποθήκης) θέματα μέχρι να έχουμε kt αποθήκες (αναπαραγωγή του θέματος t).



Εικόνα 11. Τοπολογία δικτύου και πληροφορία φόρτου εργασίας (αιτήσεις/δευτερόλεπτο) για κάθε κλάση περιεχομένων. Είσοδοι για τον αλγόριθμο Τοποθέτησης για το publish/subscribe δίκτυο (T=3, k=2, SC=2, M=3).

Στο παράδειγμα ξεκινώντας από το θέμα b, έπειτα το θέμα a και τέλος το θέμα c (με βάση το βάρος τους) τα τοποθετούμε σε k = 2 αποθήκες. Το θέμα b ανατίθεται στους κόμβους 3 και 5 που ήταν οι κόμβοι στους οποίους το θέμα b εμφανίστηκε περισσότερες φορές στο Βήμα 1. Το θέμα a ανατίθεται επίσης στους κόμβους που παράχθηκαν από το Βήμα 1, 2 και 3, ενώ το θέμα c ανατίθεται στους κόμβους 2 και 5. Ο κόμβος 5 ήταν από τις πιο δημοφιλείς επιλογές που παράχθηκε στο Βήμα 1, ενώ ο κόμβος 2 ήταν η μόνη αποθήκη στο S με λιγότερο από το L=2 αναθέσεις.

Το Βήμα 4 του αλγορίθμου μας είναι επίσης γνωστό ως Γενικευμένο Πρόβλημα ανάθεσης το οποίο ακόμα και στην απλούστερη μορφή του μειώνεται σε NP-complete πρόβλημα πολλαπλών σακιδίων.

3.2.2. **Δημιουργία Εγγραφών στο Σύστημα** (Subscription Forecast)

Στην παρούσα εργασία, για την δημιουργία εγγραφών σε αντικείμενα έχουμε χρησιμοποιήσει τον νόμο του Zipf ο οποίος μας λέει πως η συχνότητα του i-στού πιο συχνά εμφανιζόμενου αντικειμένου στο δίκτυο είναι αντιστρόφως ανάλογη του i.
Μια απλή μορφή μορφή της κατανομής Zipf είναι η

Freq(i) = 1/ (i^a)

όπου a είναι ένας σταθμιστής που κυμαίνεται γύρω στο 0.6 – 0.75.

Ο αλγόριθμος που εξάγει τα διανύσματα Δημοτικότητας και Τοπικότητας των αντικειμένων είναι απλός και δίνεται παρακάτω.

```
for(int i=1; i<=numberOfObjects; i++){
    popularity.add((int)Math.round( (MAX_popularity*(1/Math.pow(i,a)))));
    locality.add((int)Math.round((MAX_locality*(1/Math.pow(i,a)))));
}
```

Εικόνα 12. Αλγόριθμος εξαγωγής διανυσμάτων δημοφηλίας και τοπικότητας αντικειμένων.

Σαν είσοδο παίρνει τον αριθμό είδους των αντικειμένων, τον μέγιστο αριθμό δημοφιλίας, τον μέγιστο αριθμό τοπικότητας και την μεταβλητή a. Για παράδειγμα όταν Objects_num = 4, Max_Popularity = 4, Max_Locality = 2 και a=0.65 το αποτέλεσμα έχει ως εξής:

Popularity = [4, 3, 2, 2]
Locality = [2, 1, 1, 1]

Αυτό σημαίνει πως το αντικείμενο 1 θα ζητηθεί από 4 εγγραφείς οι οποίοι θα βρίσκονται σε δύο γειτονικούς κόμβους του δικτύου. Αντίστοιχα, το αντικείμενο 2 θα ζητηθεί από 3 εγγραφείς οι οποίοι θα βρίσκονται όλοι στον ίδιο, μοναδικό κόμβο κ.ο.κ. Σημείωση πως η κατανομή μπορεί να δώσει και μηδενική τιμή, αλλά όποτε αυτή προκύψει αντικαθίσταται από την τιμή 1 ώστε να υπάρχει τουλάχιστο μία εγγραφή στο συγκεκριμένο αντικείμενο.

Για να κατανείμουμε τους εγγραφείς στο δίκτυο, ακολουθήσαμε την εξής διαδικασία. Για κάθε είδος αντικείμενου (αρχικά το 1, μετά το 2 κλπ) επέλεξε έναν τυχαίο κόμβο X μέσα στο δίκτυο και βρες όλους τους κόμβους που βρίσκονται το πολύ σε απόσταση Locality(X) από αυτόν. Οι κόμβοι αυτοί αποτελούν την γειτονία του X, Neigh(X). Ξεκινώντας από τον X και προχωρώντας κάθε βήμα προς το επόμενο πιο

κοντινό group γειτόνων, ενώσω υπάρχουν διαθέσιμα αντίγραφα του αντικειμένου 1 ανάθεσε τα με την σειρά σε αυτούς. Εάν σε όλους τους Neigh(X) έχει ανατεθεί από ένα αντικείμενο και εξακολουθούν να υπάρχουν διαθέσιμα αντίγραφα αντικειμένων, η ανάθεση ξεκινάει πάλι από τον X και ακολουθείται η ίδια διαδικασία. Είναι ουσιαστικά μια μορφή Round Robin ανάθεσης αντικειμένων σε όλη την γειτονία του X. Όταν ο αριθμός αντικειμένων ανάθεσης φτάσει να ισούται με Popularity(X), η ανάθεση του συγκεκριμένου είδους αντικειμένου σταματά.

Με αυτόν τον τρόπο υλοποιούμε το κομμάτι του Subscription Forecast, του οποίου η έξοδος χρησιμοποιείται για την Τοποθέτηση και Ανάθεση Αντιγράφων στο δίκτυο.

4. Αρχεία, Εφαρμογές και χρήση του Μοντέλου Υπηρεσίας

Σε αυτή την ενότητα θα γίνει μια αναφορά στο πώς και ποια αρχεία παράγονται για την ανάπτυξη δικτύου και δημιουργίας κίνησης σε αυτό. Αυτά τα αρχεία χρησιμοποιούνται από το MonitorGui Tool το οποίο αυτοματοποιεί την όλη διαδικασία. Επίσης παρουσιάζεται ο τρόπος με τον οποίο η κάθε εφαρμογή χρησιμοποιεί το εξαγόμενο από τον Blackadder Service Model. Επίδειξη του εργαλείου ακολουθεί στην επόμενη ενότητα.

4.1. Αρχεία

4.1.1. Ανάπτυξη Δικτύου (Deployment)

Το πρώτο πράγμα που χρειαζόμαστε είναι το αρχείο topology.cfg το οποίο αναπαριστά την τοπολογία δικτύου που θέλουμε να αναπτύξουμε. Δίνει πληροφορίες για κάθε κόμβο, όπως η IP διεύθυνση του, τα μοναδικό Label_ID που τον αντιπροσωπεύει, τον Ρόλο του στο δίκτυο και με ποιους κόμβους είναι απευθείας συνδεδεμένος. Για παράδειγμα, στην Εικόνα 13, ο κόμβος με IP= 192.168.100.101 έχει LabelID= 00000001, συμπεριφέρεται σαν Rendezvous και Topology Manager, και συνδέεται απευθείας με τον κόμβο που έχει LabelID= 00000002.

Εικόνα 13. Αρχείο αναπαράστασης τοπολογίας δικτύου

Ο Blackadder παρέχει μια εφαρμογή, Deployment, η οποία παίρνει σαν όρισμα ένα τέτοιο αρχείο και εξάγει τα απαραίτητα αρχεία. Τέτοια αρχεία είναι τα Label_ID.conf τα οποία στέλνονται σε κάθε κόμβο, όπως επίσης και το topology.graphml το οποίο

στέλνεται μόνο στους κόμβους που τους έχει ανατεθεί ο ρόλος του "TM". Τα Label_ID.conf αρχεία δίνονται σαν όρισμα στον Click Router και περιέχουν την συνδεσμολογία των elements που θα τον απαρτίζουν. Τα στοιχεία INcounter, OUTcounter και controlsocket έχουν προστεθεί μετέπειτα από εμάς για να γίνει εφικτή η λήψη μετρήσεων από τον Click.

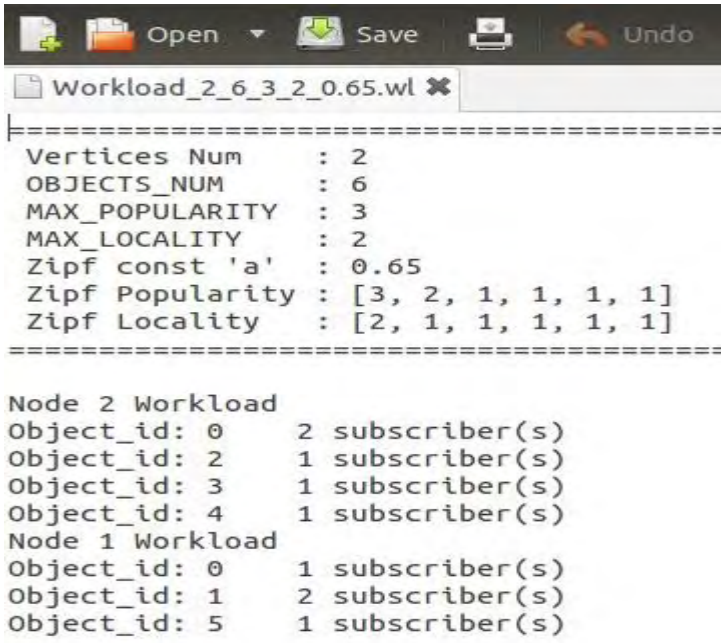
```
 8 localRV::LocalRV(globalconf);
 9 netlink::Netlink();
10 tonetlink::ToNetlink(netlink);
11 fromnetlink::FromNetlink(netlink);
12
13 proxy::LocalProxy(globalconf);
14
15 fw::Forwarder(globalconf,1,
16 1,147.102.224.228,155.207.48.52,0000000000000000000000000000000000
17
18 tsf::ThreadSafeQueue(1000);
19 rawsocket::RawSocket(UDP, 55555)
20 INcounter::Counter();
21 OUTcounter::Counter();
22 controlsocket::ControlSocket(unix, /tmp/clickSocket);
23
24
25 proxy[0]->tonetlink;
26 fromnetlink->[0]proxy;
27 localRV[0]->[1]proxy[1]->[0]localRV;
28 proxy[2]-> [0]fw[0] -> [2]proxy;
29 fw[1] ->  tsf -> OUTcounter -> rawsocket -> IPClassifier(dst udp p
```

Εικόνα 14. Αρχείο .conf το οποίο δίνεται σαν όρισμα στον Blackadder και του υποδεικνύει ποια elements θα χρησιμοποιηθούν και ποια η συνδεσμολογία αυτών.

Αντίστοιχα, το topology.graphml αρχείο δίνεται σαν όρισμα στον Topology Manager που θα τρέξει σε έναν ή περισσότερους κόμβους.

4.1.2. **Πρόγνωση Εγγραφών** (Subscription Forecast)

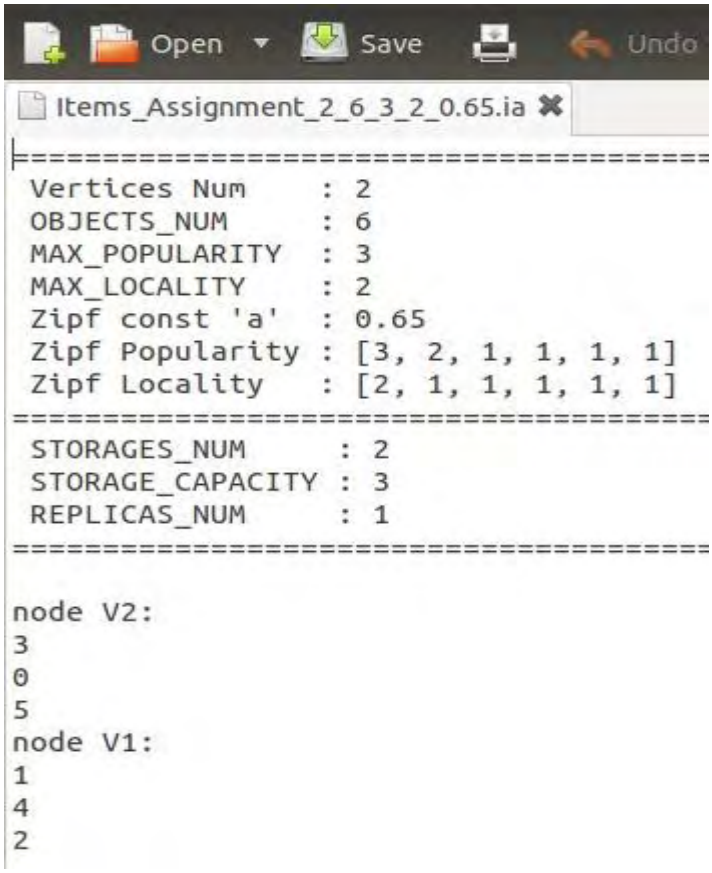
Εδώ παρουσιάζεται το αρχείο workload.wl το οποίο δημιουργείται βάσει του αλγορίθμου που περιγράψαμε στην ενότητα 3.2.3. Στην κορυφή του αρχείου φαίνονται τα στοιχεία εισόδου που έχει θέσει ο χρήστης όπως επίσης και τα διανύσματα Popularity και Locality που έχουν εξαχθεί. Στην συνέχεια ακολουθεί αναλυτική κατανομή των Subscribers σε αντικείμενα και κόμβους. Για παράδειγμα στον κόμβο 2 θα υπάρξουν δύο (2) subscribers που θα ζητήσουν το αντικείμενο 0, ενώ από ένας (1) subscriber θα ζητήσει τα αντικείμενα 2,3 και 4 αντίστοιχα. Ανάλογα μπορείτε να συμπεράνετε την ζήτηση στον κόμβο 1.



Εικόνα 15. Αρχείο που περιέχει την πρόβλεψη εγγραφέων στον δίκτυο.

4.1.3. **Τοποθέτηση και Ανάθεση Αντιγράφων** (Storage and Replica Assignment)

Παίρνει σαν όρισμα τον αριθμό Αποθηκών στο δίκτυο, την Χωρητικότητα κάθε αποθήκης, τον αριθμό Αντικειμένων και πόσες Replicas ανά Αντικείμενο θα υπάρχουν. Επίσης, διαβάζει το Workload.wl που μόλις αναφέραμε για να λάβει υπόψη την πρόβλεψη εγγραφέων. Βάσει του αλγορίθμου που περιγράψαμε εξάγεται το ItemsAssignment.ia αρχείο. Στην κορυφή αυτού, φαίνονται τα ορίσματα που έδωσε ο χρήστης, όπως επίσης και βασικά δεδομένα από το αρχείο Workload.wl στο οποίο και στηρίχθηκε. Στην συνέχεια φαίνονται οι κόμβοι που έχουν επιλεχθεί σαν Αποθήκες και τα Αντικείμενα τα οποία θα φυλάσσουν και θα δημοσιεύουν. Για παράδειγμα η πρώτη Αποθήκη είναι ο κόμβος 2 και θα δημοσιεύει τα Αντικείμενα 3,0,5. Αντίστοιχα ο κόμβος 1 σαν Αποθήκη θα δημοσιεύει τα Αντικείμενα 1,4,2.



Εικόνα 16. Αρχείο που περιέχει την επιλογή αποθηκών στο δίκτυο και την ανάθεση αντικειμένων σε αυτές.

4.2. Εφαρμογές και χρήση του Μοντέλου Υπηρεσίας

4.2.1. Topology Manager

Στα πλαίσια αυτής της διπλωματικής εργασίας, έπρεπε να τροποποιηθεί ο εξορισμού Topology Manager και να υλοποιηθεί ένας εναλλακτικός ώστε να γίνει σύγκριση μεταξύ αυτών των δύο.

- **Default Topology Manager (Shortest Path)**

```
void handleRequest(char *request, int request_len) {
        if (request_type == MATCH_PUB_SUBS) {
                ...
                tm_igraph->calculateFID(publishers, subscribers, result);
                ...
        }
    ..
}


void calculateFID(&pubs, &subs, map<string, Bitvector *> &result) {
        for (subs_it = subs.begin(); subs_it != subss.end(); subs_it++) {
                for (pubs_it = pubs.begin(); pubs_it != pubs.end(); pubs_it++) {
```

```
                    /* Για τον τρέχον subscriber, βρες τον publisher ο οποίος
                                βρίσκεται πιο κοντά σε αυτόν */
                }
        }
}
```

Όταν φθάσει στον Topology Manager ένα request “MATCH_PUB_SUBS” για ένα αντικείμενο, υπάρχει ένα group από subscribers που ζητούν το συγκεκριμένο αντικείμενο και ένα group από publishers(Caches) που το παρέχουν. Ο Topology Manager για κάθε subscriber ελέγχει την απόσταση μεταξύ αυτού και όλων των publishers. Αντιστοιχεί στον subscriber τον publisher που απέχει την μικρότερη απόσταση από αυτόν. Έτσι, εντέλει ενημερώνει τους κατάλληλους publishers με τα ανάλογα FIDs ώστε να προωθήσουν τα περιεχόμενα του αντικειμένου στον/στους subscribers.

- **Round Robin Topology Manager**

```
void handleRequest(char *request, int request_len) {
        if (request_type == MATCH_PUB_SUBS) {
                ...
                my_calculateFID(pubs, subs, result, reqString);
                ...
        }
    ..
}

void my_calculateFID(&pubs, &subs, map<string, Bitvector *> &my_result, string&
                                                                reqString) {
        ...
        for (subs_it = subs.begin(); subs_it != subs.end(); subs_it++) {
                ...
                findPublisher (list, subs_onList, str1, pubs, f_pub, reqString);
                 //str1 = (*subs_it)
                //f_pub holds the best publisher found using RoundRobbin algorithm
                ...
        }
}

void findPublisher (vector<Element>& list, vector<string>& subs_onList, string sub,
                        vector<string> pubs, string& f_pub, string& reqString ){
        int index;

        index=indexOfElem(subs_onList, sub); // check if subscriber ID is on subs list
        if(index==-1){//add new element
                //cout << "create newElement" << endl;
                if( reqString.rfind("GGGGGGGG")!= std::string::npos ){
                        //update structure only for GGGGGGGG scope
                        Element newElement(sub,pubs[0]);
                        subs_onList.push_back(sub);
                        // new subscriber ID is added on list
                        list.push_back(newElement);
```

```
                        // new element for new subscriber is added on list
                }
                f_pub = pubs[0]; // by default return publisher[0]
        }
        else{ // update element on index
                //cout << "update element" << endl;
                list[index].findBestPublisher(pubs, reqString);
                // find publisher who have sent the least items to current subscriber
                f_pub = list[index].getBestPublisher();
        }
}
```

Αυτό που συμβαίνει στην παραλλαγή αυτή είναι πως ο Topology Manager κρατάει μια λίστα από Objects της κλάσης Element. Κάθε Element Object που υπάρχει στην λίστα αναφέρεται και σε έναν κόμβο, στον οποίον και βρίσκεται ο Subscriber. Η κλάση Elements περιέχει πεδία όπως:

- το ID του κόμβου (Subscriber) στον οποίο ανήκει το Element
- μια λίστα με τα IDs των κόμβων (Publishers) που έχουν στείλει μέχρι στιγμής στον συγκεκριμένο κόμβο (Subscriber)
- μια λίστα με τον αριθμό των φορών που έχει στείλει ο κάθε κόμβος (Publisher) στον συγκεκριμένο κόμβο (Subscriber)

Όταν φθάσει στον Topology Manager ένα request “MATCH_PUB_SUBS” για ένα αντικείμενο, υπάρχει ένα group από subscribers που ζητούν το συγκεκριμένο αντικείμενο και ένα group από publishers (Caches) που το παρέχουν. Για κάθε subscriber, πρέπει να βρει τον καλύτερο publisher βάση του RoundRobin αλγορίθμου. Ψάχνει στην λίστα από Elements και βρίσκει αυτό που αναφέρεται στον συγκεκριμένο κόμβο-subscriber. Εάν δεν βρει τον συγκεκριμένο κόμβο-subscriber στην λίστα, σημαίνει πώς αυτός ζητάει αντικείμενο για πρώτη φορά, οπότε εξ' ορισμού επιστρέφεται ο πρώτος κόμβος-publisher που βρίσκεται στην λίστα. Αλλιώς, βρίσκει ποιος απ' όλους τους κόμβους-publishers έχει στείλει τις λιγότερες φορές στον συγκεκριμένο κόμβο-subscriber και αυτός επιλέγεται σαν βέλτιστος publisher.

Πιο κάτω φαίνεται ένα στιγμιότυπο όπου επιλέγεται ο καλύτερος Publisher χρησιμοποιώντας την λίστα από Elements. Αφού επιλεγεί ο κατάλληλος
Publisher, η δομή ανανεώνεται.

```
// previous state of structure
Element 1
sub_id: 00000001
pub_id(s)  /  pub_times:
00000004   2
00000003   1
00000001   1


Element 2
sub_id: 00000002
pub_id(s)  /  pub_times:
00000004   1
00000002   1
```

```
// new request just arrived
==== *request is: ====
i__00000004000000200000001_00000002__GGGGGGGG00000002_
Publishers: 00000004 00000002 00000001
Subscribers: 00000002

// current state of structure
Element 1
sub_id: 00000001
pub_id(s)  /  pub_times:
00000004   2
00000003   1
00000001   1


Element 2
sub_id: 00000002
pub_id(s)  /  pub_times:
00000004   1
00000002   1
00000001   1       ← UPDATED!


Publisher 00000001, FID:
00000000000000000000000000000000000000000000000000000000010000
00000000000000000000000000000000000000000000000000000000000000
00000000000000001000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000
00000000    ← INFORMS PUBLISHER 00000001 WITH THE PROPER FID
Publisher 00000002, FID: NULL
Publisher 00000004, FID: NULL
```

4.2.2. Cache

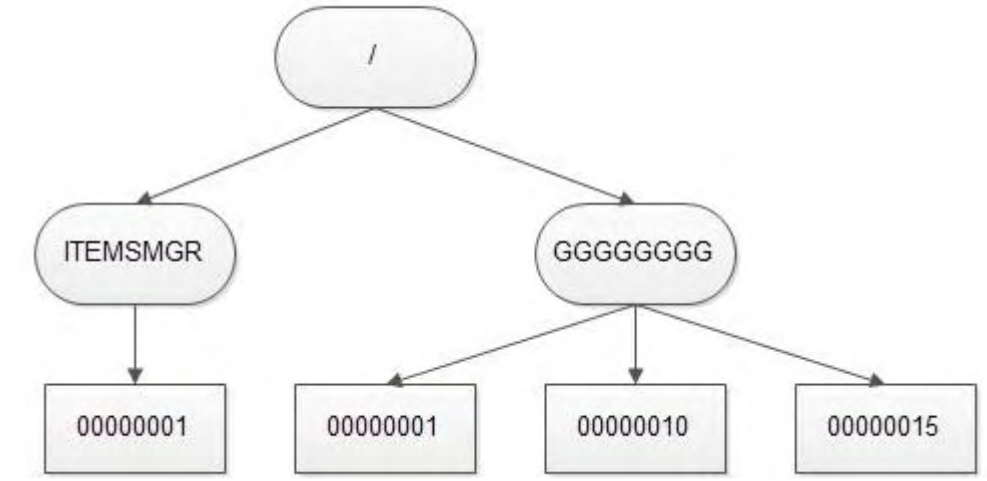
Τρέχει σε κάθε κόμβο και είναι υπεύθυνη για την αποθήκευση αντικειμένων. Ενημερώνεται από τον Items Manager για το ποια αντικείμενα θα αποθηκεύσει και στην συνέχεια θα δημοσιεύσει.

Ψευδοκώδικας

- publish_scope ("ITEMSMGR", new byte[0], DOMAIN_LOCAL, null)
- subscribe_info (myNodeLabel, "ITEMSMGR", DOMAIN_LOCAL , null)
Μόλις πάρει event απο τον Items Manager
- unpublish_info (itemID[i], "GGGGGGGG", DOMAIN_LOCAL, null)
// Σταματάει δηλαδή να δημοσιεύει αντικείμενα τα οποία δημοσίευε μέχρι
// στιγμής.
- publish_scope("GGGGGGGG", new byte[0], BA.DOMAIN_LOCAL, null)
- publish_info (itemsToPublish[i], "GGGGGGGG", DOMAIN_LOCAL, null)
//Δημοσιεύει τα νέα αντικείμενα που υπέδειξε ο Items Manager
Μόλις λάβει START_PUBLISH event
- item_payload = find(event.getId())
- publish_data (event.getId(), DOMAIN_LOCAL, null, item_payload[i])
// Η publish_data εκτελείτε item_payload.size() φορές

Υποδέντρο Πληροφορίας

Αρχικά η Cache εγγράφεται με το LabelID της κάτω από το Scope ITEMSMGR ώστε να ενημερωθεί για το ποια Αντικείμενα θα φυλάσσει και θα δημοσιεύει. Αφού ενημερωθεί από τον Items Manager δημοσιεύει τα Αντικείμενα κάτω από το Scope GGGGGGGG. Εδώ υποθέτουμε πως η συγκεκριμένη Cache βρίσκεται στον κόμβο με LabelID=00000001 και δημοσιεύει τα Αντικείμενα 00000001, 00000010, 00000015.



Εικόνα 17. Υποδέντρο πληροφορίας που σχετίζεται με τους κόμβους-Αποθήκες.

4.2.3. Items Manager

Διαβάζει το αρχείο "ItemsAssignment.ia" και ενημερώνει όλες τις Caches σχετικά με τα αντικείμενα που πρέπει να αποθηκεύσει και να δημοσιεύσει η κάθε μια. Το συγκεκριμένο αρχείο εξάγεται βάση του αλγορίθμου "Storage and Replica Planning" που έχουμε αναφέρει.

Ψευδοκώδικας

- διάβασε αρχείο
- publish_scope ("ITEMSMGR", new byte[0], DOMAIN_LOCAL, null)

Για κάθε κόμβο
    - publish_info (nodeLabel[i], "ITEMSMGR", DOMAIN_LOCAL, null)

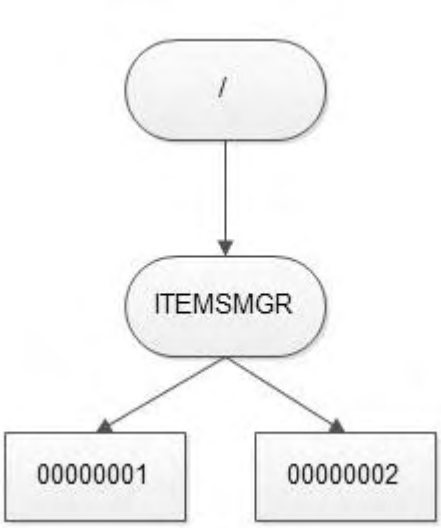
Για κάθε κόμβο περίμενε START_PUBLISH event
    - publish_data (event.getId(), DOMAIN_LOCAL, null, items_payload)

Για κάθε κόμβο (αφού έχει δημοσιεύσει σε όλες τις Caches)
    - unpublish_info (nodeLabel[i], "ITEMSMGR", DOMAIN_LOCAL, null)

- unpublish_scope ("ITEMSMGR", new byte[0], DOMAIN_LOCAL, null)
- έξοδος

Υποδέντρο Πληροφορίας

Ο Items Manager για κάθε Cache που έχει εγγραφεί στο Scope ITEMSMGR δημοσιεύει το κατάλληλο μήνυμα. Εδώ υποθέτουμε πως υπάρχουν δύο κόμβοι-Caches στο δίκτυο με LabelID 00000001 και 00000002 αντίστοιχα.



Εικόνα 18. Υποδέντρο πληροφορίας που σχετίζεται με τον Items Manager

4.2.4. Subscriber

Ζητά συγκεκριμένο αντικείμενο από το δίκτυο και περιμένει να το λάβει από "κάποια" Cache. Αναλόγως ποια έκδοση του Topology Manager χρησιμοποιείται, προσδιορίζεται και η λέξη "κάποια". Μόλις λάβει τα περιεχόμενα του αντικειμένου, ενημερώνει το MonitorGUI για τον χρόνο παραλαβής τους.

Ψευδοκώδικας

- startTime = System.currentTimeMillis()
- subscribe_info(item, "GGGGGGGG", DOMAIN_LOCAL, null)

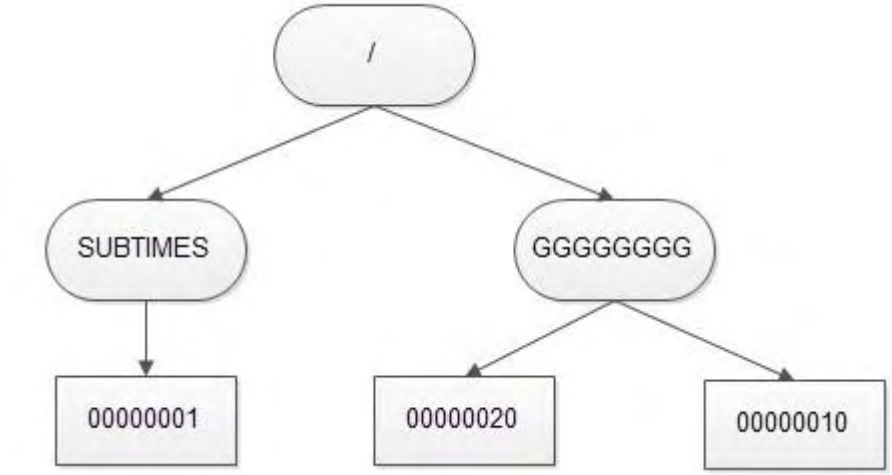
Για να μπορέσει ο Subscriber να καταλάβει πως η Cache δεν πρόκειται να στείλει άλλα περιεχόμενα για το συγκεκριμένο item, ελέγχουμε εάν στο πακέτο περιλαμβάνεται το substring "_EOF".

Ενώσω λαμβάνει event PUBLISHED_DATA
    - Εάν το event.getData() περιέχει "_EOF", η Cache δεν πρόκειται να στείλει κάτι άλλο, οπότε βγες από τον βρόγχο
    - Αλλιώς περίμενε και επόμενο event
- endTime = System.currentTimeMillis();
- unsubscribe_info (item, "GGGGGGGG", DOMAIN_LOCAL, null)
- finalTime = endTime-startTime

- publish_info(myNodeLabel, "SUBTIMES", DOMAIN_LOCAL, null)
Όταν λάβει event START_PUBLISH
- publish_data (event.getId(), DOMAIN_LOCAL, null, finalTime)
- unpublish_info(myNodeLabel, "SUBTIMES", DOMAIN_LOCAL, null)
- έξοδος

Υποδέντρο Πληροφορίας

Ο Subscriber εγγράφεται στα Αντικείμενα 00000020 και 00000010 κάτω από το Scope GGGGGGGG. Μόλις τα παραλάβει δημοσιεύει τον χρόνο παραλαβής αυτών μέσω του Αντικειμένου 00000001 (που χαρακτηρίζει το LabelID του κόμβου) κάτω από το Scope SUBTIMES.



Εικόνα 19. Υποδέντρο πληροφορίας που σχετίζεται με κάθε Subscriber.

4.2.5. Agent

Τρέχει σε κάθε κόμβο του δικτύου και είναι υπεύθυνος για λήψεις μετρήσεων. Επικοινωνεί με τον Click-Blackadder τοπικά μέσω unix socket και παίρνει μετρήσεις όπως rate, bit rate, byte rate, packet count και byte count τόσο για είσοδο όσο και για έξοδο. Η περίοδος λήψης μετρήσεων παρέχεται μέσω του MonitorGUI, ενώ οι μετρήσεις στέλνονται πίσω στο MonitorGUI για επεξεργασία. Όταν ο χρήστης θελήσει να σταματήσει τις λήψεις μετρήσεων, ενημερώνει με "STOP" μήνυμα τους Agents.

Ψευδοκώδικας

- publish_scope("MONITORS", new byte[0], DOMAIN_LOCAL, null)
- subscribe_info("interval", "MONITORS", DOMAIN_LOCAL, null)
- publish_info(myNodeLabel, "MONITORS", DOMAIN_LOCAL, null)

Όταν λάβει event PUBLISHED_DATA
- εάν το πακέτο περιέχει "START:", διάβασε το interval_time που περιέχεται σε αυτό
- άνοιξε νέο thread "stop_thread" και περίμενε για "STOP" μήνυμα

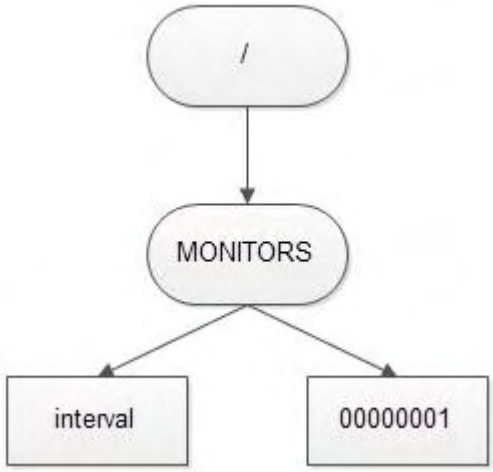
Κάθε interval_time
- metrics = getMetrics();
- publish_data(event.getId(), DOMAIN_LOCAL, null, metrics)
- Εάν stop_thread.getState().equals("TERMINATED") βγες από τον βρόγχο και περίμενε ξανά για μήνυμα "START"

stop_thread:
Όταν λάβει event PUBLISHED_DATA
-εάν το πακέτο περιέχει "STOP" exit thread

Υποδέντρο Πληροφορίας

Κάθε κόμβος-Agent εγγράφεται στο Αντικείμενο interval που βρίσκεται κάτω από το Scope MONITORS ώστε να παραλάβει την περίοδο δειγματοληψίας μετρήσεων. Επίσης, υποθέτοντας πως αναφερόμαστε στον κόμβο-Agent με LabelID = 00000001,

δημοσιεύει κάτω από το ίδιο Scope στο Αντικείμενο 00000001 τους χρόνους που έλαβε από την δειγματοληψία.



Εικόνα 20. Υποδέντρο πληροφορίας που σχετίζεται με κάθε Agent.

4.2.6. Monitor GUI Tool

Είναι υπεύθυνο για να στείλει το START:interval_time ή STOP μήνυμα στους Agents. Επίσης λαμβάνει μηνύματα από τους Agents και τους Subscribers που περιέχουν τις ανάλογες μετρήσεις.

Ψευδοκώδικας

- publish_info("interval", "MONITORS", DOMAIN_LOCAL, null)
Όταν λάβει event START_PUBLISH
- publish_data(event.getId(), BA.DOMAIN_LOCAL, null, msg.getBytes())
//Στέλνει δηλαδή σε όλους τους Agents το START:interval_time μήνυμα

- publish_scope("SUBTIMES", new byte[0], DOMAIN_LOCAL, null)
Για κάθε κόμβο
    - subscribe_info(nodeLabel[i], "MONITORS", BA.DOMAIN_LOCAL, null);
    - subscribe_info(nodeLabel[i], "SUBTIMES", BA.DOMAIN_LOCAL, null);
    // ή εναλλακτικά subscribe_scope("MONITORS"),
    // subscribe_scope("SUBTIMES")

While(true){
  - Εάν ο χρήστης πατήσει Stop
  - unsubscribe_scope ("MONITORS", new byte[0], DOMAIN_LOCAL, null)
  - unsubscribe_scope ("SUBTIMES", new byte[0], DOMAIN_LOCAL, null)
  - unpublish_scope ("SUBTIMES", new byte[0], DOMAIN_LOCAL, null)
  - unpublish_info("interval", "MONITORS", DOMAIN_LOCAL, null)

  /* στείλε "STOP" σε όλους τους Agents */
  - publish_info ("interval", "MONITORS", DOMAIN_LOCAL, null)
  Όταν λάβει event START_PUBLISH
    - publish_data (event.getId(), DOMAIN_LOCAL, null, "STOP")
    - unpublish_info ("interval", "MONITORS", DOMAIN_LOCAL, null)

- break και περίμενε τον χρήστη να ξαναπατήσει το κουμπί Start

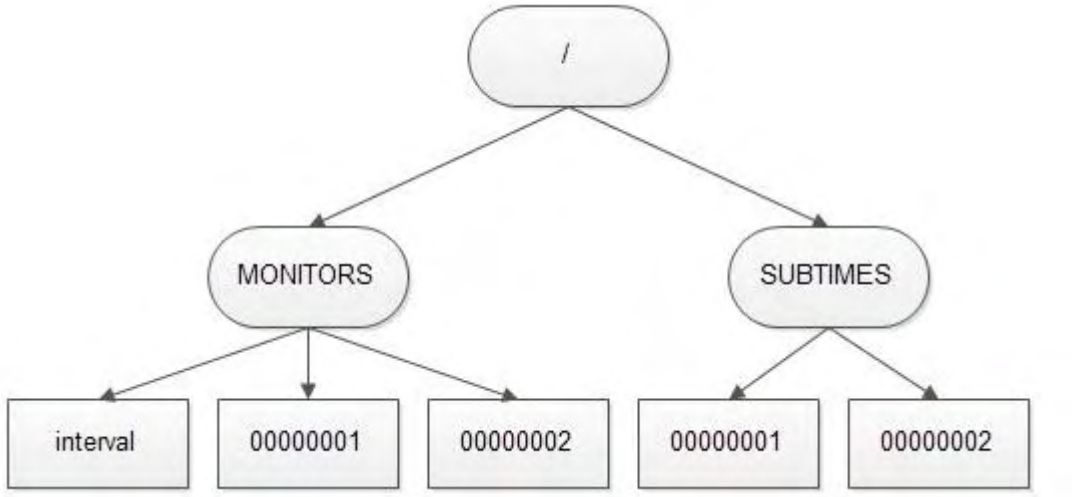
Αλλιώς όταν λάβει event PUBLISHED_DATA

- nodeLabel = findNode (event.getId())
- metric_type = findMetricType (event.getId())
- Εάν metricType.equals("MONITOR"), σώσε το μήνυμα στο κατάλληλο αρχείο.
- Εάν metricType.equals("SUBTIMES"), σώσε το μήνυμα στο κατάλληλο αρχείο.

}

Υποδέντρο Πληροφορίας

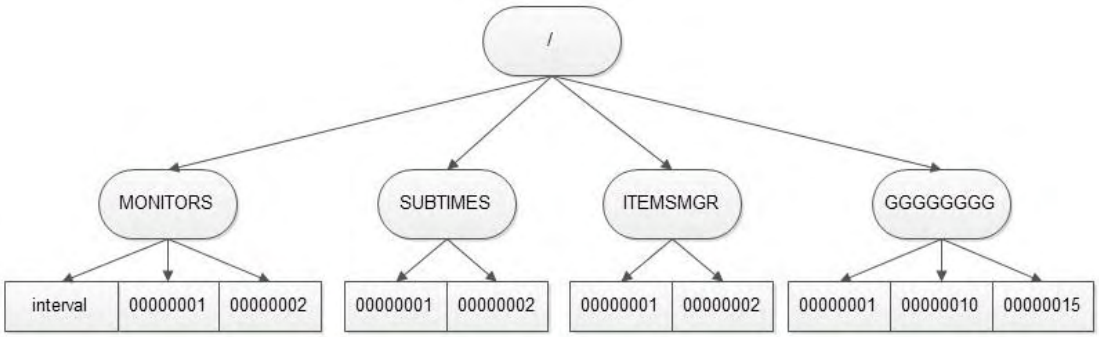
Το MonitorGUI δημοσιεύει το Αντικείμενο interval κάτω από το Scope MONITORS ώστε να στείλει το interval_period ή το STOP μήνυμα στους Agents. Αντίστοιχα εγγράφεται σε όλα τα αντικείμενα LabelIDs κάτω από τα Scopes MONITORS και SUBTIMES ώστε να λαμβάνει τις μετρήσεις τόσο από τους Agents, όσο και από τους Subscribers (υποθέτουμε μόνο δύο κόμβους στο δίκτυο).



Εικόνα 21. Υποδέντρο πληροφορίας που σχετίζεται με το εργαλείο Monitor GUI.

4.2.7 Δέντρο Πληροφορίας

Το συνολικό Δέντρο Πληροφορίας που σχετίζεται με όλες τις εφαρμογές που έχουμε προαναφέρει παρουσιάζεται πιο κάτω. Πιο συγκεκριμένα αναφέρεται σε δίκτυο δύο κόμβων και τριών αντικειμένων.



Εικόνα 22. Δέντρο Πληροφορίας που σχετίζεται με όλες τις εφαρμογές. Αναφέρεται σε δίκτυο δύο κόμβων και τριών αντικειμένων.

5. Υλοποίηση της Εφαρμογής Παρακολούθησης

Σε αυτή την ενότητα αναφέρονται τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη των αλγορίθμων και γενικότερα του εργαλείου Monitor GUI. Αρχικά κάνουμε μια αναφορά στην πλατφόρμα NetBeans όπως επίσης και σε βιβλιοθήκες που φάνηκαν εξαιρετικά χρήσιμες για την διευκόλυνση υλοποίησης του όλου εγχειρήματος. Απαραίτητη επίσης υπήρξε η βιβλιοθήκης Blackadder_Java για την χρήση του εξαγόμενου από τον Blackadder publish/subscribe API. Τέλος ακολουθεί μία λεπτομερής παρουσίαση της λειτουργικότητας του εργαλείου Monitor GUI.

5.1. Εργαλείο NetBeans

Το NetBeans IDE είναι ένα open-source περιβαλλοντικό ανάπτυγμα που επιτρέπει στους προγραμματιστές να γράψουν, να κάνουν compile, debug και να αναπτύξουν προγράμματα σε διάφορες γλώσσες προγραμματισμού όπως Java, C/C++, HTML5, PHP. Το εργαλείο αυτό είναι γραμμένο σε Java και μπορεί να τρέξει σε Windows, OS X, Linux, Solaris και σε άλλες πλατφόρμες που υποστηρίζουν συμβατό Java Virtual Machine.

5.2. Βασικές Βιβλιοθήκες

5.2.1. JUNG – the Java Universal Network/Graph Framework

[16] Είναι μια βιβλιοθήκη γραμμένη σε Java που παρέχει μια κοινή και επεκτάσιμη γλώσσα για την μοντελοποίηση, ανάλυση και απεικόνιση δεδομένων τα οποία μπορούν να αναπαρασταθούν ως ένα γράφημα ή δίκτυο. Έχει σχεδιαστεί για να υποστηρίζει μία ποικιλία από αναπαραστάσεις των οντοτήτων και των σχέσεων τους όπως κατευθυνόμενοι και μη γράφοι, πολυτροπικοί γράφοι, γράφοι με παράλληλες ακμές και υπερ-γράφοι.

Περιέχει υλοποιήσεις αλγορίθμων από την θεωρεία γράφων, την εξόρυξη δεδομένων καθώς επίσης και την ανάλυση κοινωνικών δικτύων όπως την τυχαία παραγωγή γράφων, τον υπολογισμό αποστάσεων σε δίκτυο, στατική ανάλυση κ.α. Παρέχει δε ένα πλαίσιο οπτικοποίησης που καθιστά εύκολη την υλοποίηση εργαλείων για την διαδραστική εξερεύνηση δεδομένων του δικτύου. Πέρα από τους αλγόριθμους οπτικοποίησης που παρέχονται, ο χρήστης μπορεί να δημιουργήσει τις δικές του προσαρμοσμένες εμφανίσεις.

Γενικά είναι μια open-source βιβλιοθήκη η οποία διευκολύνει σημαντικά όσους ασχολούνται με την ανάλυση και οπτικοποίηση γράφων και δικτύων.

5.2.2. JSch – Java Secure channel

[17] Είναι μια βιβλιοθήκη που υλοποιεί το SSH2 πρωτόκολλο αποκλειστικά σε Java. Επιτρέπει την σύνδεση σε sshd server και χρήση του port forwarding, X11 forwarding, μεταφορά αρχείων κ.λ.π. Μερικά επιπλέον χαρακτηριστικά που παρέχονται από την βιβλιοθήκη αυτή είναι user authentication, stream forwarding, signal sending, environment variable passing, generating DSA and RSA key pairs και

remote exec. Ο χρήστης μπορεί να ενσωματώσει τις όποιες λειτουργίες επιθυμεί στο πρόγραμμα του με σκοπό την ανάπτυξη πιο πολύπλοκων λειτουργικοτήτων.

5.2.3. Blackadder Java

Στα πλαίσια του Blackadder έχουν αναπτυχθεί βιβλιοθήκες που επιτρέπουν την χρήση του εξαγόμενου publish/subscribe API για διάφορες γλώσσες όπως C, Java, Python και Ruby. Στην παρούσα διπλωματική χρησιμοποιήσαμε το Java API.

5.3 Εφαρμογές εντός και εκτός επιπέδου Δικτύου

5.3.1 Γενική εικόνα

Η υλοποίηση του περιβάλλοντος αποτελείται από εφαρμογές που τρέχουν σε επίπεδο δικτύου σε κάθε κόμβο όπως επίσης και από εφαρμογές που τρέχουν εκτός δικτύου. Με τον τρόπο αυτό θα καταστεί δυνατή η ανάπτυξη δικτύου και η παρακολούθηση κίνησης σε αυτό.

Ξεκινώντας από το επίπεδο δικτύου, οι εφαρμογές που τρέχουν είναι οι εξής

- Ο Blackadder που είναι υπεύθυνος για την δρομολόγηση των πακέτων που κινούνται στο δίκτυο. Η λειτουργία Rendezvous (RV) λαμβάνει χώρα μόνο εάν υποδειχθεί από την Deployment εφαρμογή που τρέχει εκτός δικτύου.

- Ο Topology Manager είναι υπεύθυνος για την δημιουργία μονοπατιών μεταξύ publishers και subscribers όταν του ζητηθεί από την Rendezvous (RV) λειτουργία που αποτελεί μέρος του Click-Blackadder. Η Deployment εφαρμογή υποδεικνύει και ενεργοποιεί τον Topology Manager σύμφωνα με το αρχείο τοπολογίας δικτύου.

- Η εφαρμογή Cache ενημερώνεται από τον Items Manager για το ποια αντικείμενα θα αποθηκεύει και θα δημοσιεύει.

- Η εφαρμογή Agent επικοινωνεί με τον Click-Blackadder ανά χρονικά διαστήματα για να λάβει τις μετρήσεις κίνησης που αφορούν τον κόμβο. Ο χρόνος δειγματοληψίας παρέχεται από την εφαρμογή Monitor GUI και οι μετρήσεις στέλνονται πίσω σε αυτή.

- Η εφαρμογή Subscriber ζητά ένα συγκεκριμένο Αντικείμενο και καταμετρά τον χρόνο που χρειάστηκε για την παραλαβή του. Ενημερώνει την εφαρμογή Monitor GUI για την συγκεκριμένη μέτρηση.

Συνεχίζοντας, οι εφαρμογές που τρέχουν εκτός δικτύου είναι οι εξής

- Η εφαρμογή Deployment είναι υπεύθυνη για την διαμόρφωση του δικτύου. Αρχικά γίνεται ο σχεδιασμός της τοπολογίας και στην συνέχεια τα κατάλληλα αρχεία στέλνονται σε κάθε κόμβο. Τέλος ενεργοποιούνται οι Click-Blackadder Routers με ή χωρίς την λειτουργία Rendezvous και ο Topology Manager.

- Η εφορμογή Workload χρησιμοποιώντας την Zipf κατανομή εξάγει το workload που θα υπάρχει στο δίκτυο. Με άλλα λόγια, εξάγει ένα αρχείο που περιλαμβάνει τους subscribers που θα τρέξουν σε κάθε κόμβο.

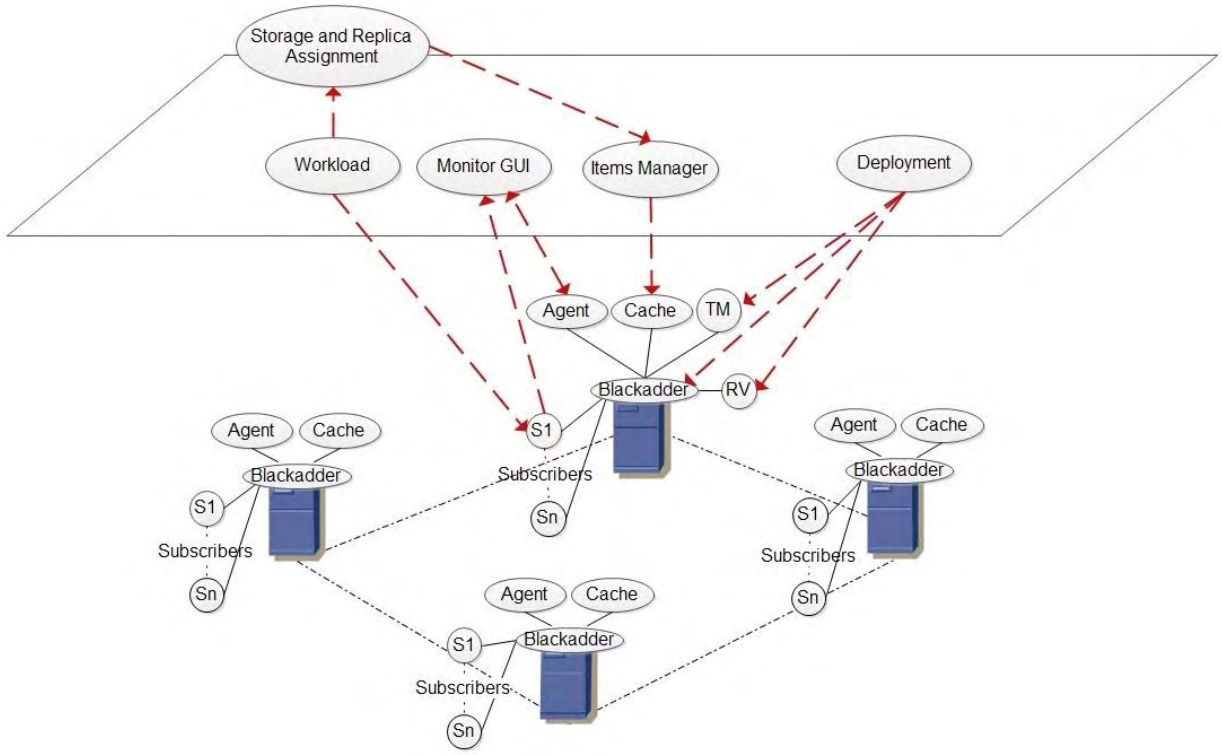
- Η εφαρμογή Storage and Replica Assignment παίρνει σαν είσοδο το εξαγόμενο από την Workload εφαρμογή αρχείο και εξάγει ένα άλλο αρχείο που περιέχει τις επιλεγμένες αποθήκες-κόμβους και τα αντικείμενα που θα αποθηκεύουν.

- Η εφαρμογή Items Manager διαβάζει το αρχείο που έχει εξαχθεί από το Storage and Replica Assignment και ενημερώνει τις Caches για το ποια αντικείμενα θα αποθηκεύουν.

- Η εφαρμογή Monitor GUI είναι υπεύθυνη για τον συντονισμό όλων των εφαρμογών που τρέχουν εκτός δικτύου. Αρχικά εκτελεί με την σειρά τις εφαρμογές Deployment, Workload, Storage and Replica Assignment ώστε να ενεργοποιηθούν κατάλληλα όλες οι εφαρμογές σε επίπεδο δικτύου. Στην συνέχεια ενημερώνει τους Agents με χρόνο δειγματοληψίας, τρέχει τους Subscribers σε κάθε κόμβο σύμφωνα με το Workload αρχείο και αναμένει για την παραλαβή μετρήσεων τόσο από τους Agents όσο και από τους Subscribers.

Στην παρούσα διπλωματική εργασία όλες οι εφαρμογές που βρίσκονται εκτός δικτύου λαμβάνουν χώρα στον τοπικό υπολογιστή που αποτελεί μέρος του δικτύου.

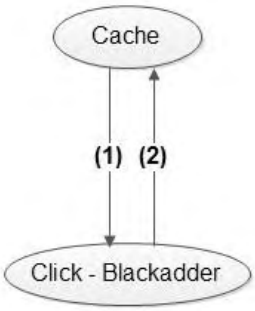
Η αλληλεπίδραση των εφαρμογών που βρίσκονται σε επίπεδο δικτύου και μη απεικονίζεται πιο κάτω.



Εικόνα 23. Εφαρμογές που τρέχουν σε επίπεδο δικτύου και εκτός αυτού.

5.3.2 Εφαρμογές και χρήση του publish/subscribe API

Πιο κάτω παρουσιάζεται αναλυτικά η αλληλεπίδραση της κάθε εφαρμογής με τον Click-Blackadder μέσω του publish/subscribe API.



(1) publish_scope("ITEMSMGR")

(1) subscribe_info(myLabelID, "ITEMSMGR")

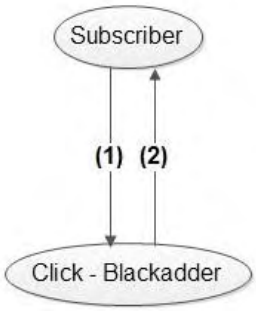
(2) Published Data notification

(1) publish_scope("GGGGGGGG")

(1) publish_info(item[i], "GGGGGGGG")

(2) Start Publish notification

(1) publish_data(ev.getID(), *item_payload)



(1) subscribe_info(item, "GGGGGGGG")

(2) Published Data notification

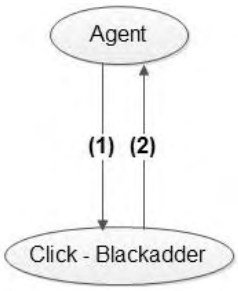
(1) unsubsribe_info(item, "GGGGGGGG")

(1) publish_info(myLabelID, "SUBTIMES")

(2) Start Publish notification

(1) publish_data(ev.getID(), *item_time)

(1) unpublish_info(myLabelID, "SUBTIMES")



(1) publish_scope("MONITORS")

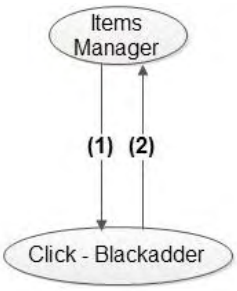
(1) subscribe_info(interval, "MONITORS")

(1) publish_info(myLabelID, "MONITORS")

(2) Published Data notification

```
if ( msg == "START:interval" ) {
    for every interval period
        open unix socket
        metrics = getMetrics from Click
        publish_data (ev.getID(), *metrics)
}
esle if ( msg == "STOP") {
     exit for every interval period loop
}

/* Separate Thread is waiting for "STOP" msg*/
```



(1) publish_scope(item, "ITEMSMGR")

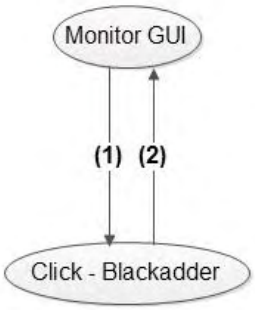
(1) publish_info(nodeLabelID[i], "ITEMSMGR")

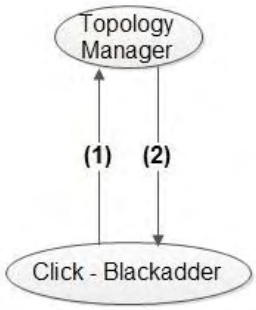
(2) Start Publish notification

(1) publish_data(ev.getID(), *items_for_node)

(1) unpublish_info(nodeLabelID[i], "ITEMSMGR")

(1) unpublish_scope("ITEMSMGR")





(1) publish_info("interval", "MONITORS")

(2) Start Publish notification

(1) publish_data(ev.getID(), *start_interval)

(1) publish_scope("SUBTIMES")

(1) subscribe_info(nodeLabelID[i], "SUBTIMES")

(1) subscribe_info(nodeLabelID[i], "MONITORS")

(2) Published Data notification

```
if (ev.getID().contains("MONITORS") ){
   save_to_mon_file()
}
else if ( ev.getID().contains("SUBTIMES")){
   save_to_sub_file()
}
```

(1) handle_request ("MATCH_PUB_SUBS")

(2) return (Pub_Sub_Path)

5.4. Επίδειξη Monitor GUI

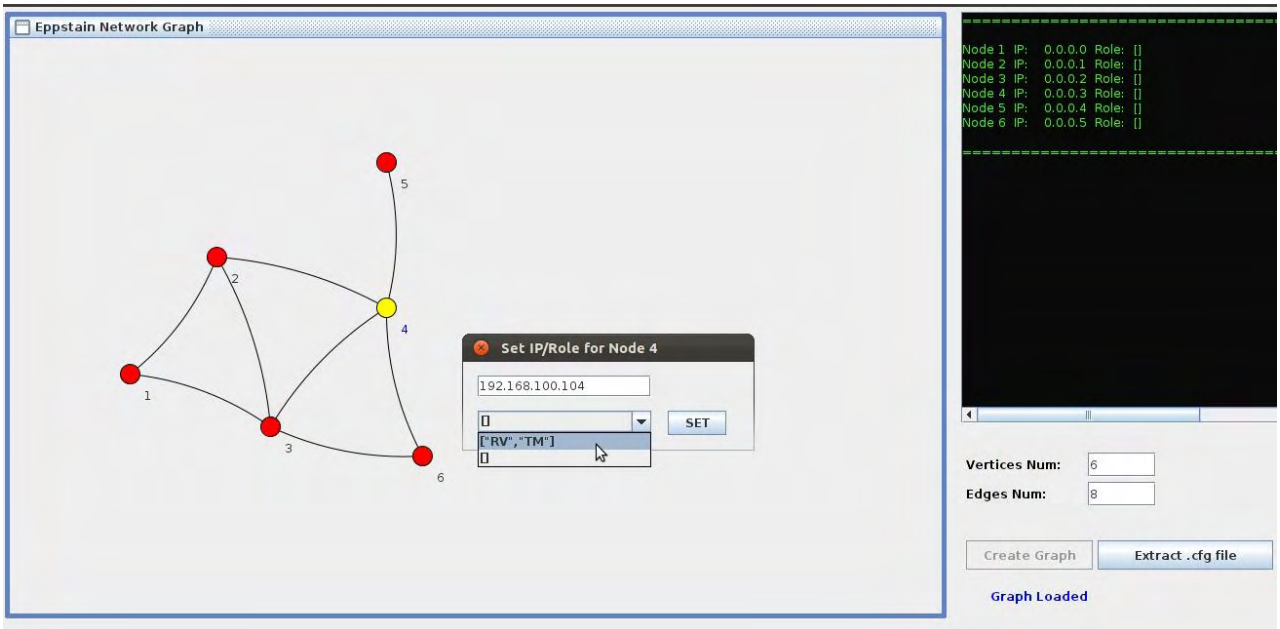
Προχωρούμε σε μία συνολική εκτέλεση του εργαλείου με το οποίο θα σχεδιάσουμε και θα αναπτύξουμε το δίκτυο. Κατά την διάρκεια του πειράματος ο χρήστης μπορεί να παρακολουθήσει την κίνηση που λαμβάνει χώρα στο δίκτυο. Αφού τελειώσει το πείραμα (όλοι οι εγγραφείς έχουν λάβει τα αντικείμενα στα οποία εγγράφηκαν) ο χρήστης μπορεί να δει τα συνολικά αποτελέσματα.

5.4.1. Βήμα Πρώτο – Προσδιορισμός Τοπολογίας Δικτύου

Εικόνα 23. Επιλογή τρόπου δημιουργίας δικτύου.

- **Τρόπος Πρώτος** : Δημιούργησε Τυχαίο Δίκτυο (Eppstain Algorithm)

Ο χρήστης δίνει σαν όρισμα τον αριθμό κορυφών και ακμών, πατά "Create Graph" και το δίκτυο, βάση αλγορίθμου, φορτώνεται. Ο χρήστης μπορεί να εισάγει IP και Role σε κάθε κόμβο πατώντας πάνω σε αυτούς. Όταν τελειώσει με όλους τους κόμβους, πατά "Extract .cfg" για να δημιουργηθεί το αρχείο topology.cfg.

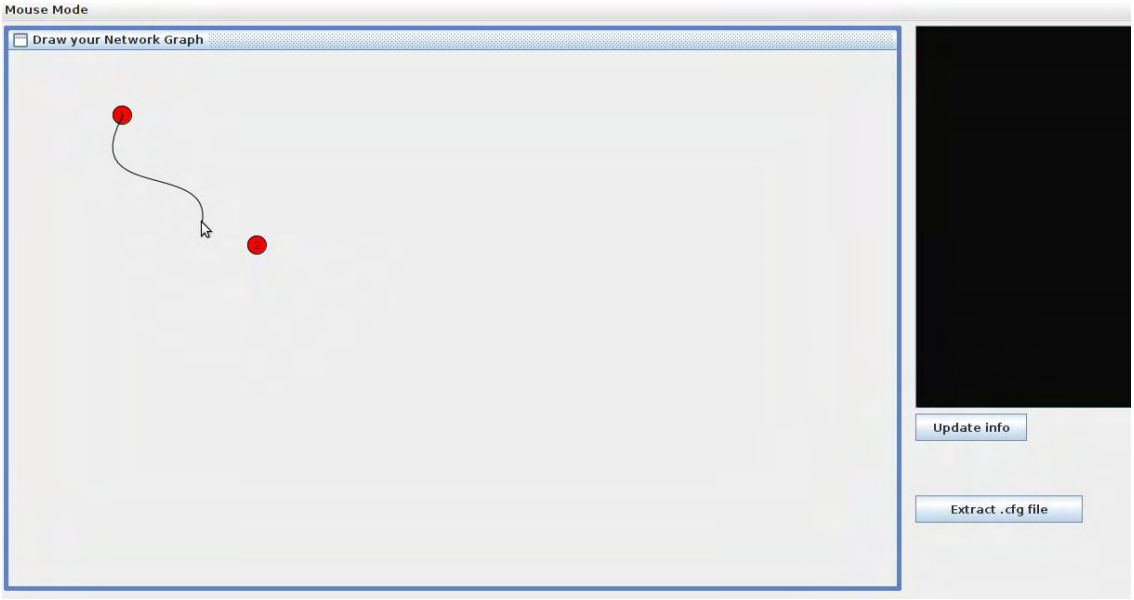


Εικόνα 24. Δημιουργία γράφου και εξαγωγή του αρχείου τοπολογίας.

Ο αλγόριθμος Eppstain παρέχεται από την βιβλιοθήκη JUNG που έχουμε προαναφέρει.

- **Τρόπος Δεύτερος** : Σχεδίασε το δικό σου Δίκτυο (Draw)

Κάνοντας click στην επιφάνεια, δημιουργείται νέος κόμβος. Σύροντας το ποντίκι από ένα κόμβο σε ένα άλλον δημιουργείται ακμή μεταξύ αυτών. Αντίστοιχα με τον τρόπο 1, ο χρήστης μπορεί να θέσει IP και Role σε κάθε κόμβο. Όταν τελειώσει με όλους τους κόμβους μπορεί να πατήσει "Extract .cfg".

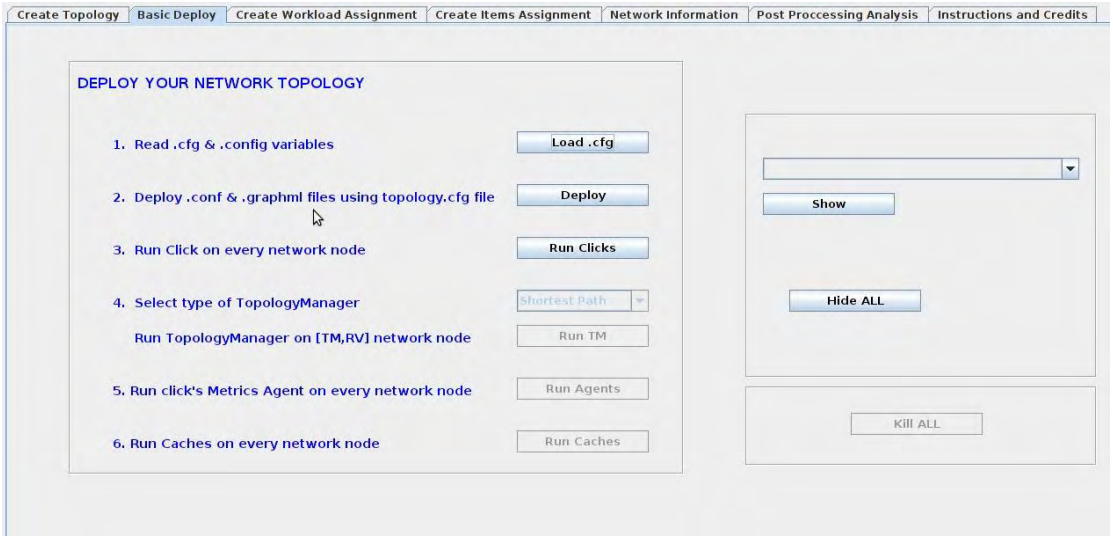


Εικόνα 25. Δημιουργία συγκεκριμένου γράφου και εξαγωγή του αρχείου τοπολογίας.

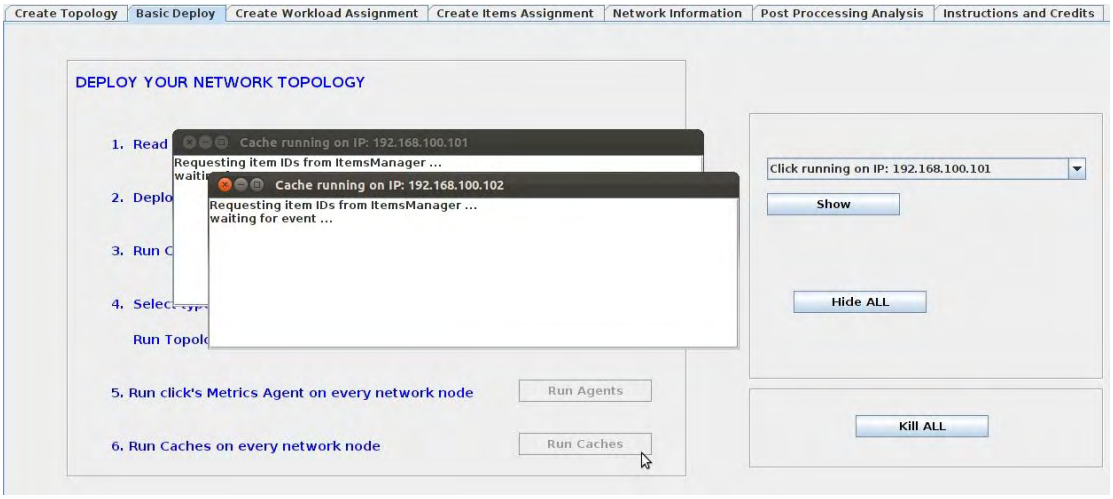
5.4.2. Βήμα Δεύτερο – Ανάπτυξη Δικτύου

Στην συνέχεια ο χρήστης είναι έτοιμος να αναπτύξει το δίκτυο και να τρέξει σε κάθε κόμβο τα απαραίτητα αρχεία, τα οποία αναφέραμε στην προηγούμενη ενότητα. Αρχικά με το "Load .cfg" φορτώνεται το αρχείο τοπολογίας, έστω 2_nodes.cfg. Με το "Deploy" τα αρχεία .conf και .graphml που έχουν δημιουργηθεί αρχικά από το Deployment εργαλείο, τροποποιούνται και στέλνονται στους κόμβους. Η τροποποίηση περιλαμβάνει την εισαγωγή νέων Elements στην δομή του Click Router που θα μας επιτρέψουν να λάβουμε τις κατάλληλες μετρήσεις. Πατώντας "Run Clicks" οι Blackadders σηκώνονται σε κάθε κόμβο διαβάζοντας το .conf αρχείο σαν όρισμα. Στην περίπτωση του "Run TM" δίνεται η δυνατότητα επιλογής μεταξύ του Shortest Path Topology Manager και του RoundRobbin Topology Manager. Τέλος μπορούμε να τρέξουμε τους Agents και τις Caches.

Ακολουθούν δύο εικόνες όπου στην πρώτη φαίνεται το αρχικό παράθυρο Ανάπτυξης, ενώ στην δεύτερη παρουσιάζεται ένα στιγμιότυπο όπου έχουμε τρέξει όλα τα προγράμματα στους κόμβους.



Εικόνα 26. Παράθυρο Ελέγχου για Ανάπτυξη Δικτύου.

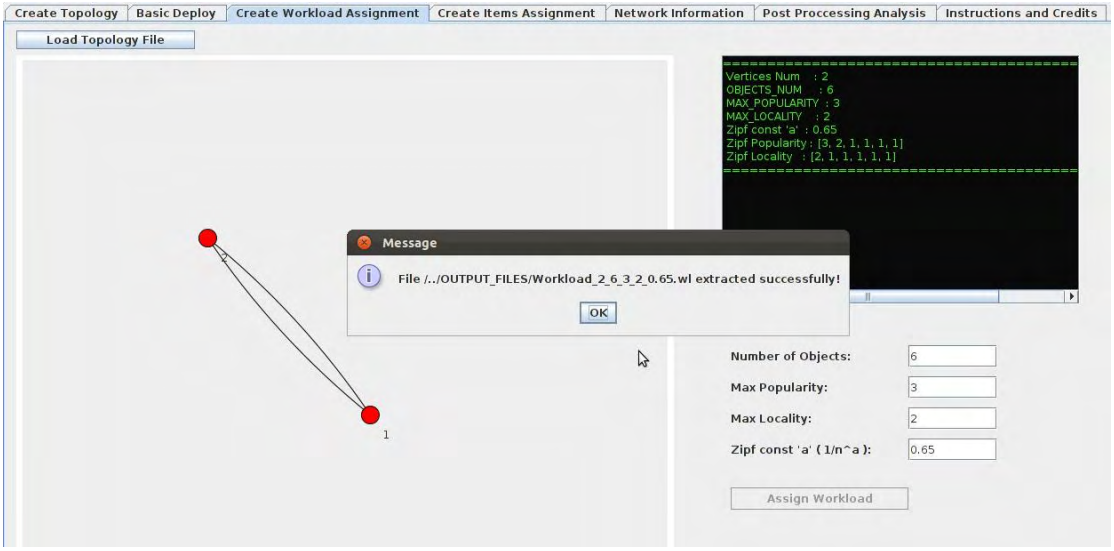


Εικόνα 27. Ανάπτυξη Δικτύου και έλεγχος όλων των εφαρμογών που τρέχουν.

Για ευκολία χρήσης, μέσω του Hide/Show, ο χρήστης μπορεί να κρύψει ή να εμφανίσει τα παράθυρα που θέλει να παρακολουθήσει. Εάν επιθυμεί να "ρίξει" όλο το δίκτυο και να το επανεκκινήσει, μπορεί να το κάνει μέσω του "Kill All".

5.4.3. Βήμα Τρίτο – Δημιουργία Αρχείου Εγγραφέων

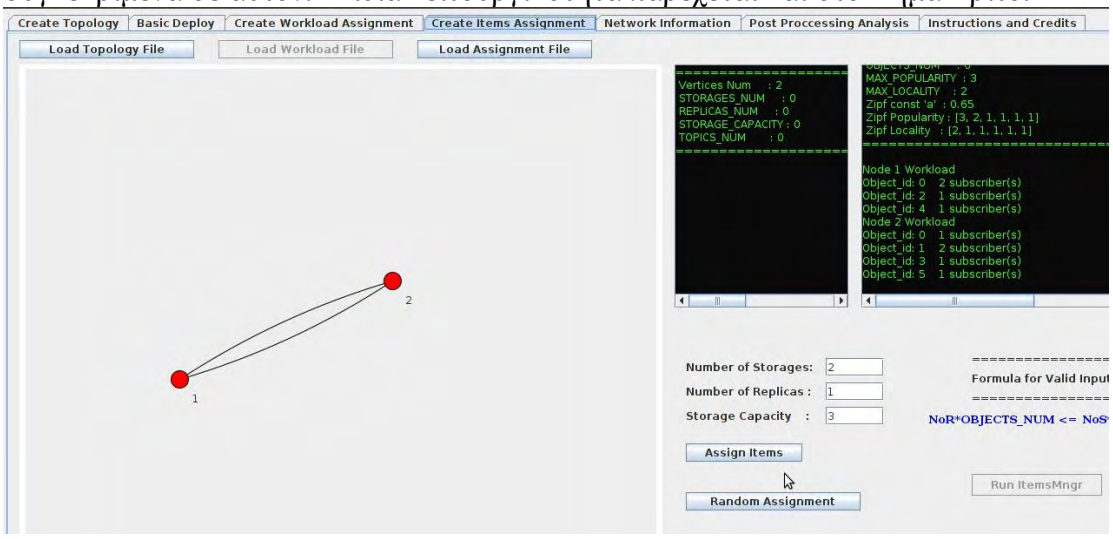
Ο χρήστης φορτώνει το .cfg αρχείο και δίνει σαν όρισμα τον αριθμό αντικειμένων, την μέγιστη δημοσιότητα, την μέγιστη τοπικότητα και την σταθερά a. Πατώντας "Assign Workload" το αρχείο εγγραφέων .wl εξάγεται. Βασικά αποτελέσματα παρουσιάζονται πάνω δεξιά με πράσινα γράμματα.



Εικόνα 28. Δημιουργία αρχείου εγγραφέων(subscribers).

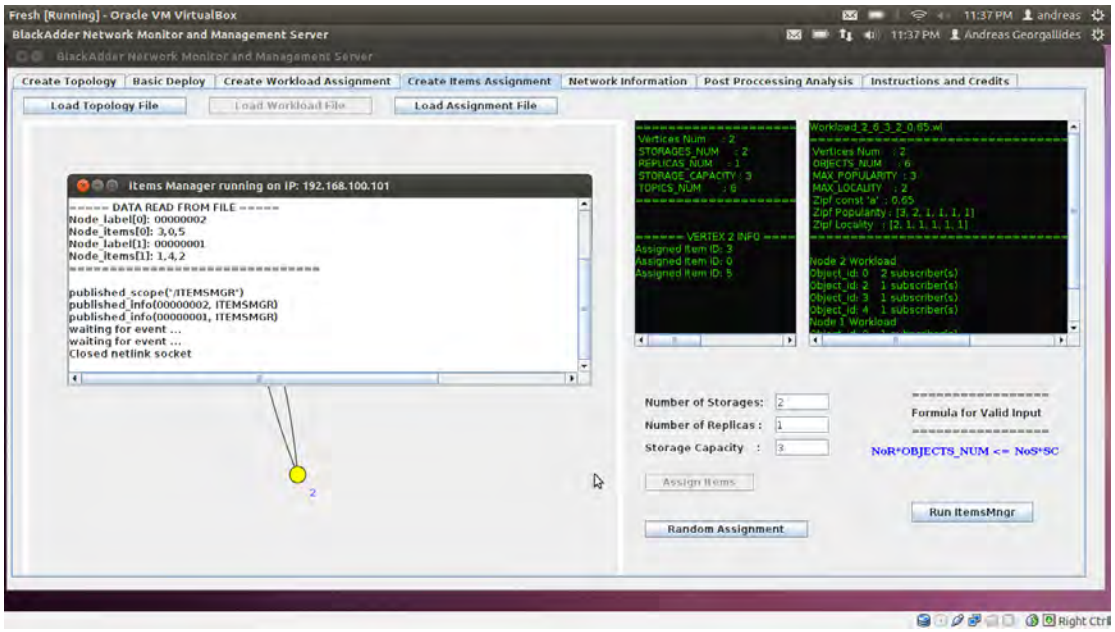
5.4.4. Βήμα Τέταρτο – Εξαγωγή Αρχείου ανάθεσης Αντικειμένων και εκτέλεση του Items' Manager

Ο χρήστης αρχικά φορτώνει το αρχείο .cfg και .wl(που έχει δημιουργηθεί στο Βήμα Τρίτο). Έπειτα δίνει σαν όρισμα τον αριθμό αποθηκών, τον αριθμό αντιγράφων κάθε αντικειμένου και την χωρητικότητα μνήμης κάθε κόμβου. Πατώντας "Assign Items" παράγεται το αρχείο .ia που εμπεριέχει πληροφορία για τα αντικείμενα που πρέπει να αποθηκεύσει τοπικά κάθε κόμβος βάση του αλγορίθμου ανάθεσης. Επίσης πατώντας "Random Assignment" παράγεται αντίστοιχο αρχείο με έναν τυχαίο αλγόριθμο που έχουμε υλοποιήσει ώστε να γίνουν συγκρίσεις στα πειράματα που θα ακολουθήσουν. Επιλέγοντας ένα κόμβο μπορούμε να δούμε ποια αντικείμενα έχουν ανατεθεί συγκεκριμένα σε αυτόν. Η ίδια λειτουργικότητα παρέχεται και στο Βήμα Τρίτο.



Εικόνα 29. Εξαγωγή αρχείου ανάθεσης αντικειμένων.

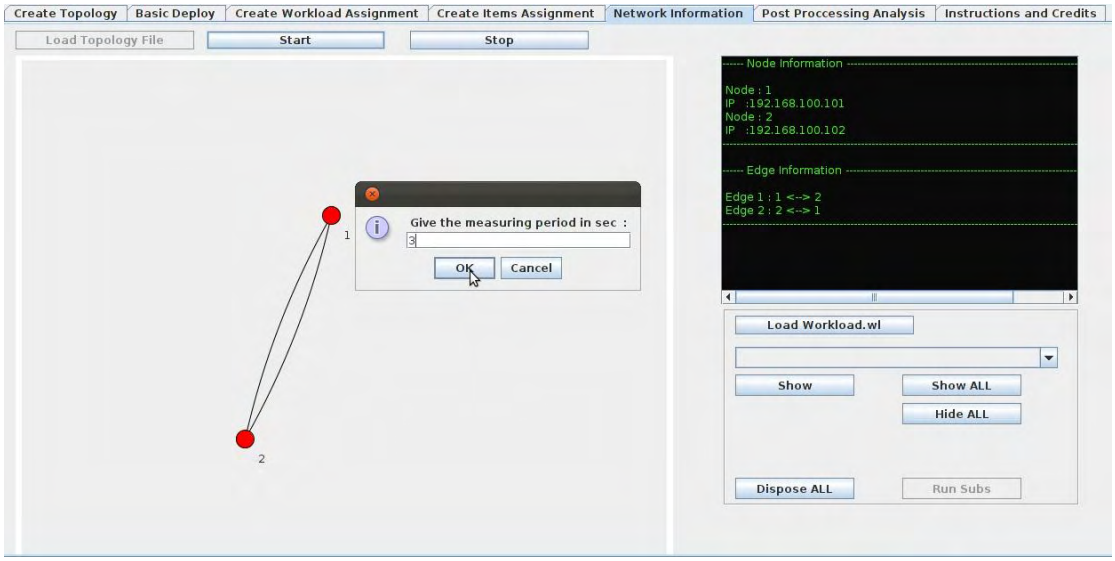
Φορτώνοντας το .ia αρχείο με το κουμπί "Load Assignment File", ο χρήστης τρέχει τον Item Manager πατώντας "Run ItemsManager". Η εφαρμογή Items Manager διαβάζει το .ia αρχείο και ενημερώνει της αποθήκες (Caches) για το ποια αντικείμενα πρέπει να αποθηκεύσουν.



Εικόνα 30. Τρέξιμο του Items Manager και ενημέρωση των Αποθηκών.

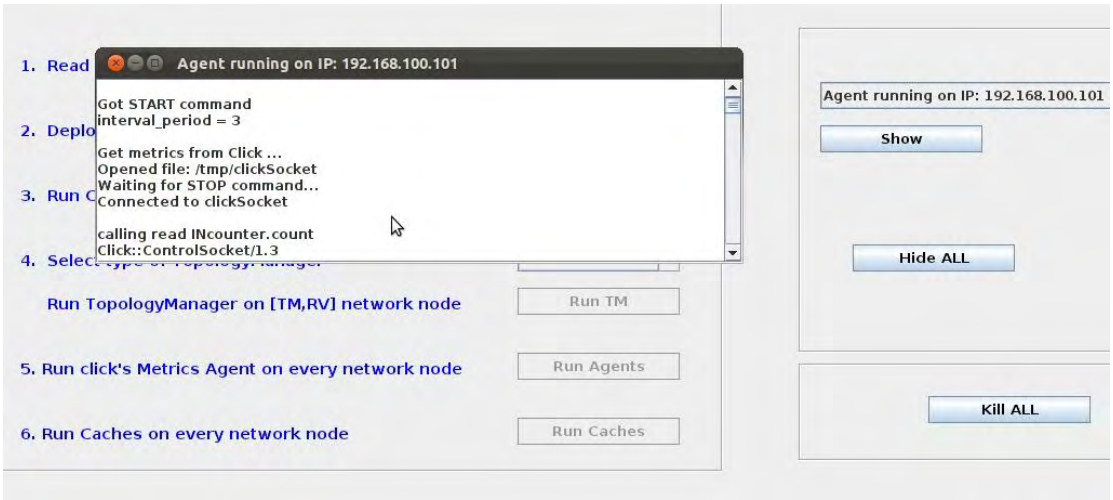
5.4.5 Βήμα Πέμπτο – Ενημέρωσε τους Agents και τρέξε τους Subscribers

Αφού φορτώσει το .cfg αρχείο, ο χρήστης πατά “Start” για να δώσει measuring period στους Agents. Στο συγκεκριμένο παράδειγμα δίνουμε σαν χρόνο δειγματοληψίας 3 seconds.



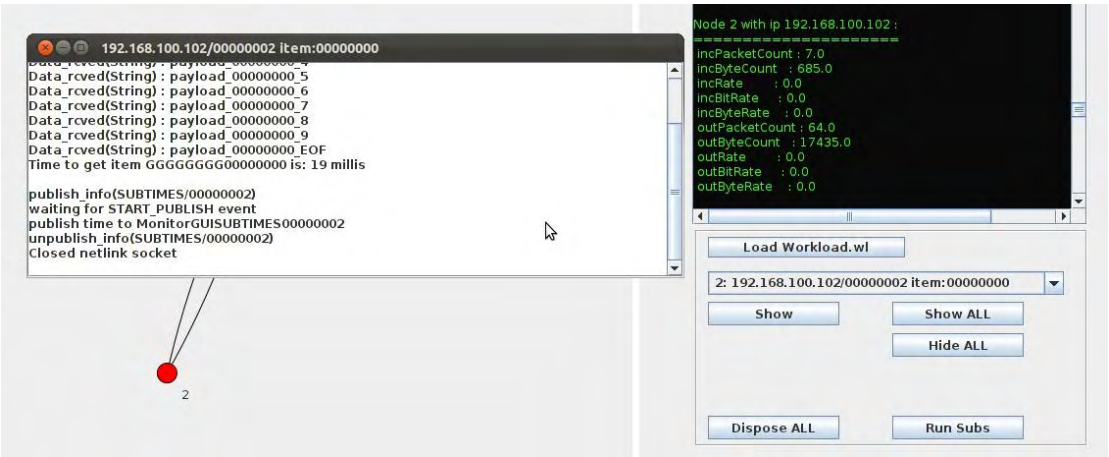
Εικόνα 31. Ενημέρωση Agents για τον χρόνο δειγματοληψίας.

Οι Agents λαμβάνουν την τιμή και ξεκινούν την δειγματοληψία.



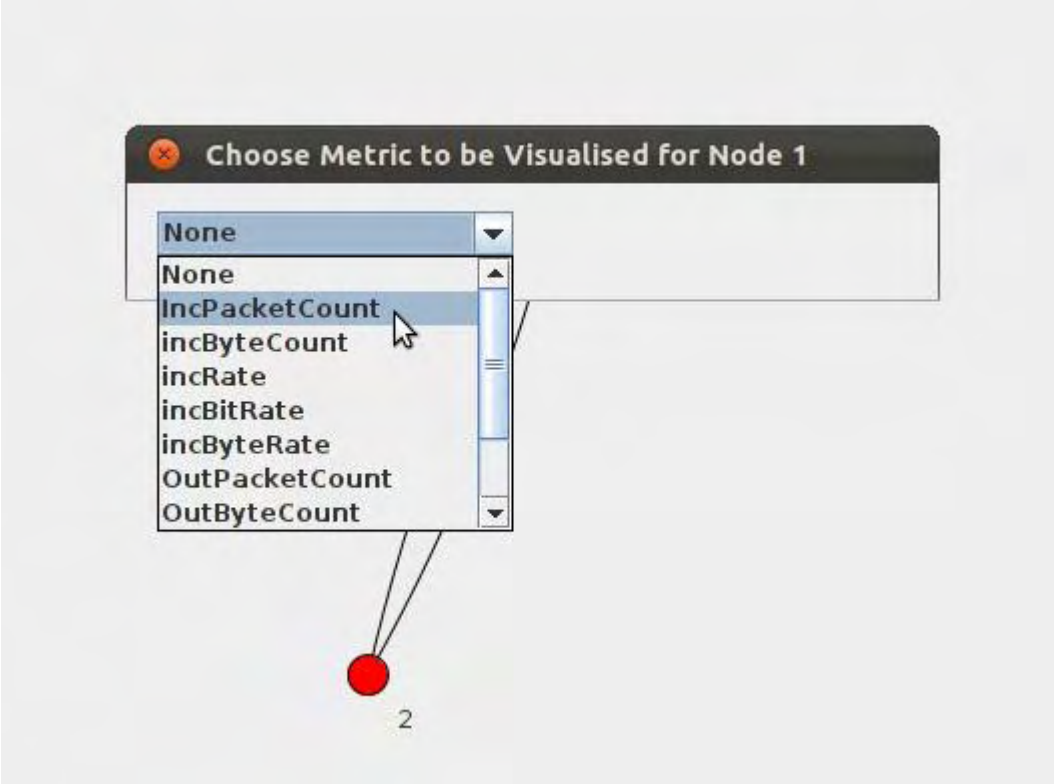
Εικόνα 32. Έναρξη δειγματοληψίας από τους Agents σε κάθε κόμβο.

Τώρα ο χρήστης είναι σε θέση να τρέξει τους Subscribers. Αρχικά φορτώνει το .wl αρχείο με το "Load Workload.wl" και στην συνέχεια πατάει "Run Subs". Ένα στιγμιότυπο παρουσιάζεται στην πιο κάτω εικόνα.

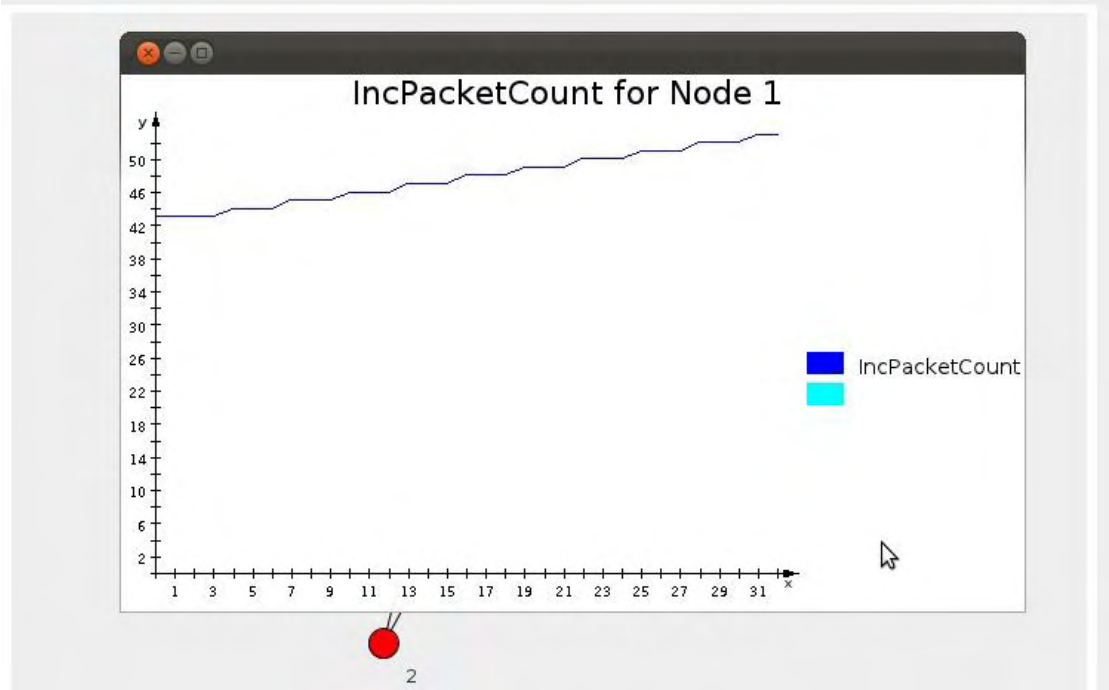


Εικόνα 33. Τρέξιμο των Subscribers για δημιουργία φόρτου στο δίκτυο.

Ο χρήστης μπορεί να παρακολουθήσει την κίνηση στο δίκτυο που σχετίζεται με τον κάθε κόμβο σε πραγματικό χρόνο. Πατώντας πάνω σε ένα κόμβο, επιλέγεται ο τύπος μέτρησης και η γραφική παράσταση εμφανίζεται. Στο συγκεκριμένο παράδειγμα, παρακολουθούμε το Incoming Packet Count στον κόμβο 1.



Εικόνα 34. Επιλογή παρακολούθησης Incoming Packet Count στον κόμβο 1.

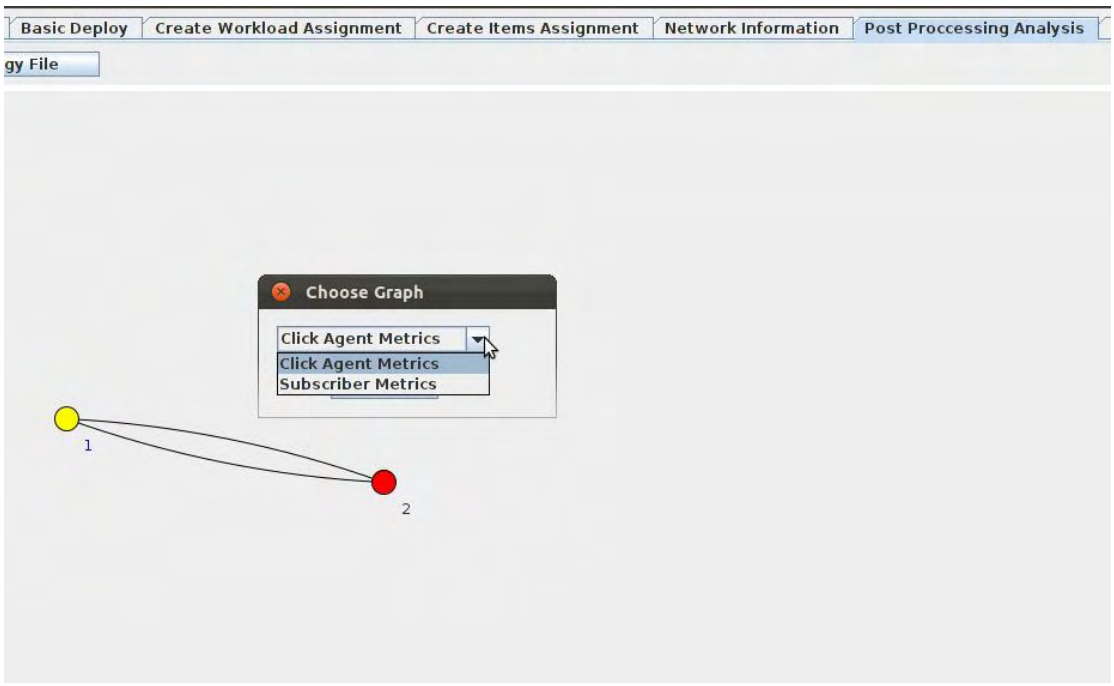


Εικόνα 35. Παρακολούθηση επιλεγμένης μέτρησης σε πραγματικό χρόνο.

5.4.6 Βήμα Έκτο – Παρακολούθηση συνολικών μετρήσεων

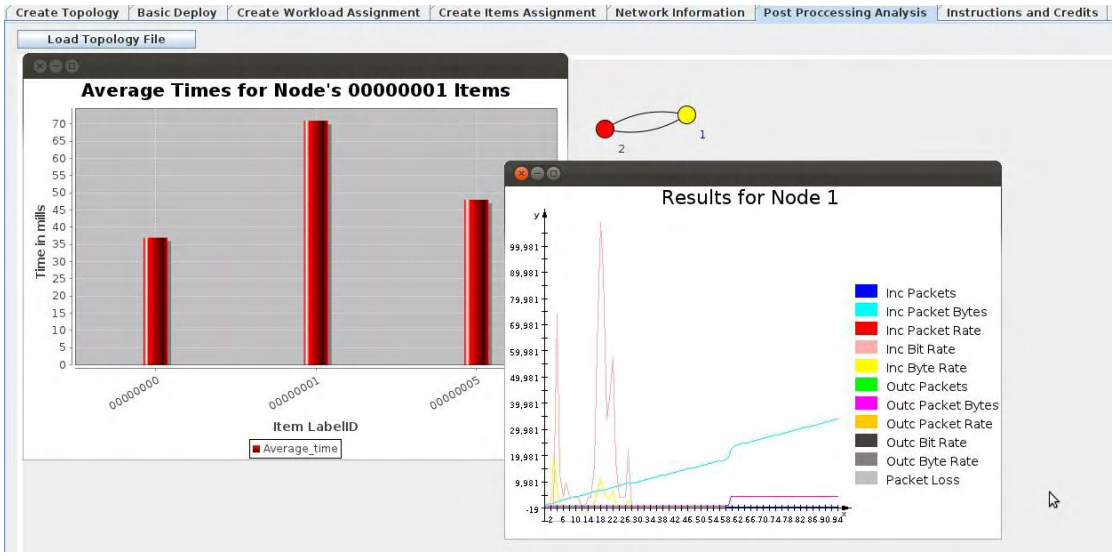
Φορτώνουμε το .cfg αρχείο και στην συνέχεια πατώντας πάνω σε ένα κόμβο μπορούμε να επιλέξουμε την γραφική αναπαράσταση των συνολικών μετρήσεων που

έχουν λάβει χώρα στο δίκτυο. Υπάρχουν δύο τύπων μετρήσεις, αυτές που στέλνουν οι Agents και αυτές που στέλνουν οι Subscribers.



Εικόνα 36. Επιλογή προβολής μετρήσεων από Agents ή Subscribers που αφορούν συγκεκριμένο κόμβο.

Η γραφική παράσταση στα αριστερά μας δείχνει πως ο κόμβος 1 έχει ζητήσει 3 ειδών items. Κάθε είδος item μπορεί να έχει περισσότερες από μία εγγραφές. Τις μετρήσεις αυτές τις στέλνουν οι Subscribers που τρέχουν στον κόμβο 1 και το MonitorGUI είναι υπεύθυνο για τον υπολογισμό των μέσων χρόνων σε Milliseconds. Στα δεξιά φαίνονται όλες οι μετρήσεις που έχει στείλει, σε βάθος χρόνου, ο Agent που τρέχει στον κόμβο 1.



Εικόνα 37. Προβολή μετρήσεων από Agent και Subscribers που έχουν τρέξει στον κόμβο 1.

Τα αρχεία μετρήσεων μπορούν να αποθηκευτούν και να μελετηθούν σε επόμενο χρόνο. Ο χρήστης μπορεί να ακολουθήσει ξανά όλη την διαδικασία για την εκτέλεση διαφορετικών πειραμάτων.

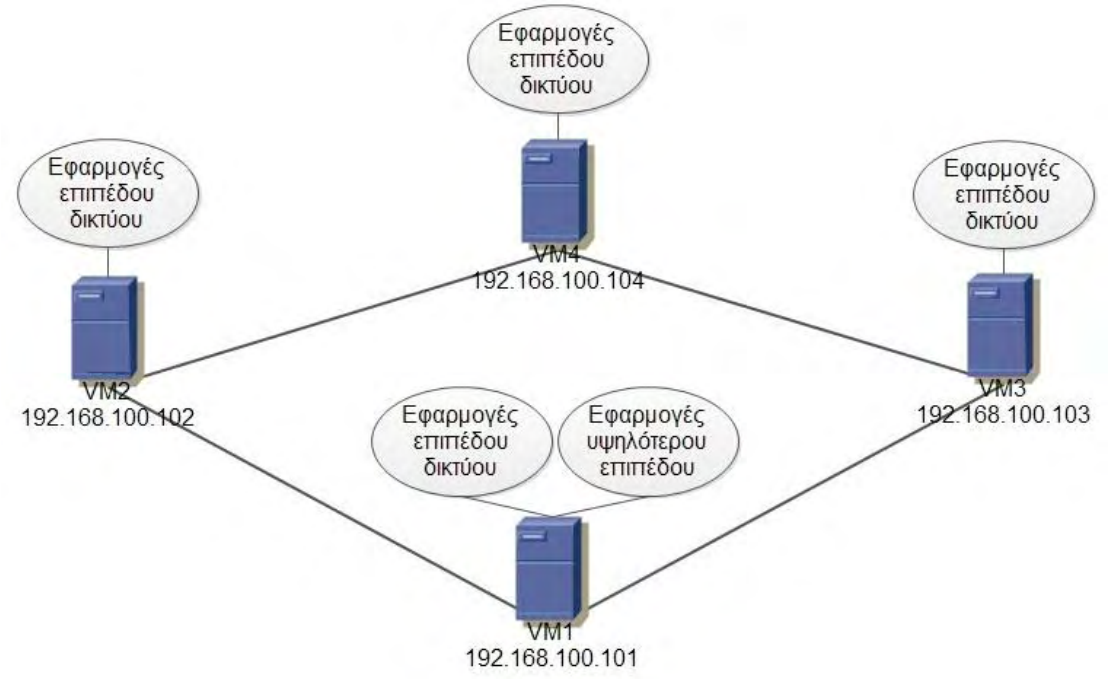
6. Πειράματα

Το Monitor GUI εργαλείο αναπτύχθηκε χρησιμοποιώντας μία συνδεσμολογία τεσσάρων κόμβων από Virtual Machines σε τοπικό υπολογιστή. Αφού ολοκληρώθηκε η υλοποίηση του επιχειρήθηκε να γίνουν πειράματα σε πραγματικούς κόμβους στο παγκόσμιο δίκτυο Planet Lab [18]. Ή λήψη όμως αντιπροσωπευτικών μετρήσεων δεν κατέστη δυνατή. Ο Blackadder δεν χρησιμοποιεί πρωτόκολλο που να εγγυάται παράδοση όλων των πακέτων οπότε οι απαραίτητες για την παρακολούθηση μετρήσεις αλλοιώνονταν σημαντικά.

Αποφασίσαμε έτσι να πάρουμε ενδεικτικές μετρήσεις στο τοπικό εικονικό δίκτυο των τεσσάρων κόμβων οι οποίες και παρουσιάζονται πιο κάτω.

6.1. Τοπολογία τοπικού εικονικού δικτύου

Η τοπολογία δικτύου που χρησιμοποιήθηκε για την λήψη πειραμάτων φαίνεται πιο κάτω. Οι εφαρμογές που τρέχουν σε κάθε κόμβο σε επίπεδο δικτύου όπως επίσης και οι εφαρμογές που τρέχουν σε υψηλότερο επίπεδο έχουν αναφερθεί στο 5.3.1.



Εικόνα 38. Τοπολογία των εικονικών κόμβων και οι εφαρμογές που τρέχουν σε αυτούς.

6.2. Μετρήσεις

Οι γραφικές παραστάσεις δείχνουν τα αποτελέσματα για τέσσερις διαφορετικούς συνδυασμούς χρησιμοποιώντας δυο διαφορετικούς τρόπους ανάθεσης αντικειμένων στις αποθήκες και δύο διαφορετικές υλοποιήσεις του Topology Manager.

- **Χρώμα Κόκκινο** – Shortest Path Topology Manager και Αλγόριθμος Τοποθέτησης και Ανάθεσης Αντιγράφων για Δίκτυα Δημοσιεύσεων/Συνδρομών.

- **Χρώμα Μπλε** – Round Robin Topology Manager και Αλγόριθμος Τοποθέτησης και Ανάθεσης Αντιγράφων για Δίκτυα Δημοσιεύσεων/Συνδρομών.

- **Χρώμα Πράσινο** - Shortest Path Topology Manager και Τυχαίος Αλγόριθμος Τοποθέτησης και Ανάθεσης Αντιγράφων.

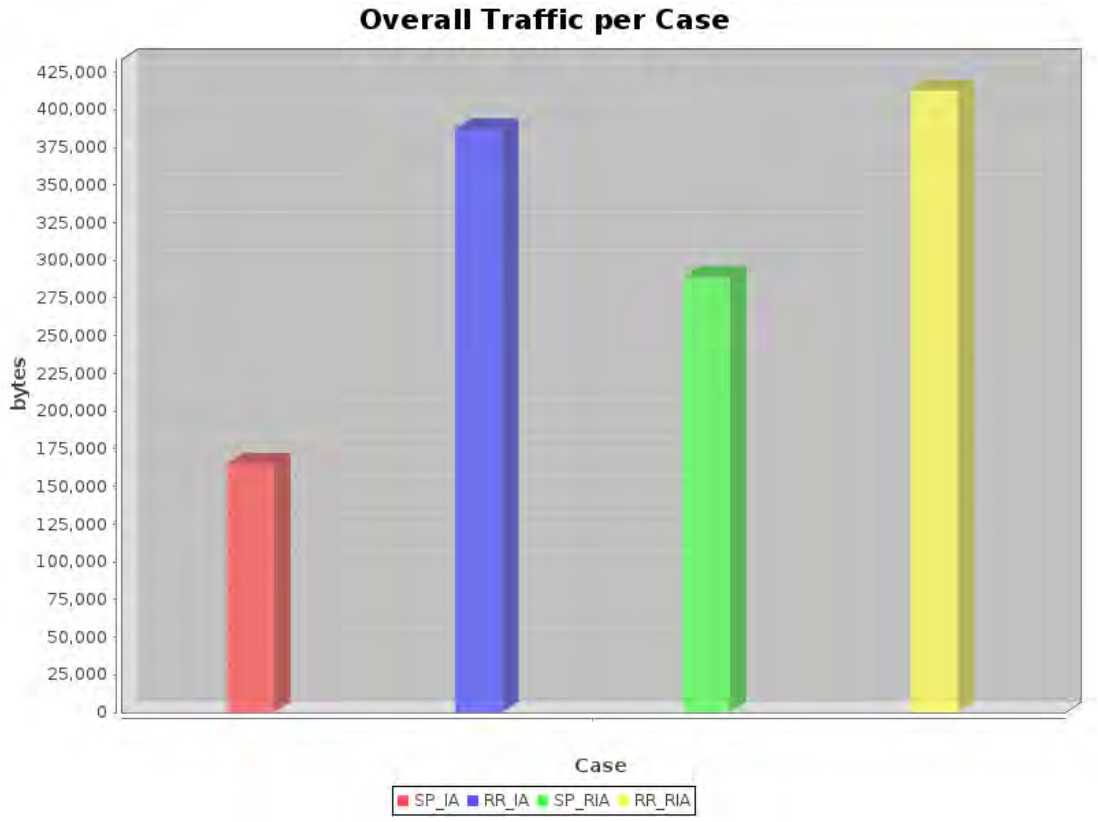
- **Χρώμα Κίτρινο** - Round Robin Topology Manager και Τυχαίος Αλγόριθμος Τοποθέτησης και Ανάθεσης Αντιγράφων.

Πείραμα 1

Αριθμός Κόμβων: 4
Αριθμός Αποθηκών: 4
Αριθμός Αντικειμένων: 4
Αριθμός Αντιγράφων ανά Αντικείμενο: 2
Χωρητικότητα Αποθήκης: 3

Αριθμός Subscribers: 40

```
Zipf Popularity : [15, 10, 8, 7]
Zipf Locality   : [3, 2, 2, 1]
```



Εικόνα 38. Συνολική κίνηση στο δίκτυο σε bytes.

Συγκρίνοντας αρχικά τον συνδυασμό Shortest Path και Round Robin Topology Manager βλέπουμε πως όποιο αλγόριθμο τοποθέτησης και να χρησιμοποιήσουμε, η επιλογή του Shortest Path Topology Manager δίνει χαμηλότερες μετρήσεις κίνησης στο δίκτυο όπως και αναμενόταν. Επιλέγοντας τον Shortest Path TM βλέπουμε πως η

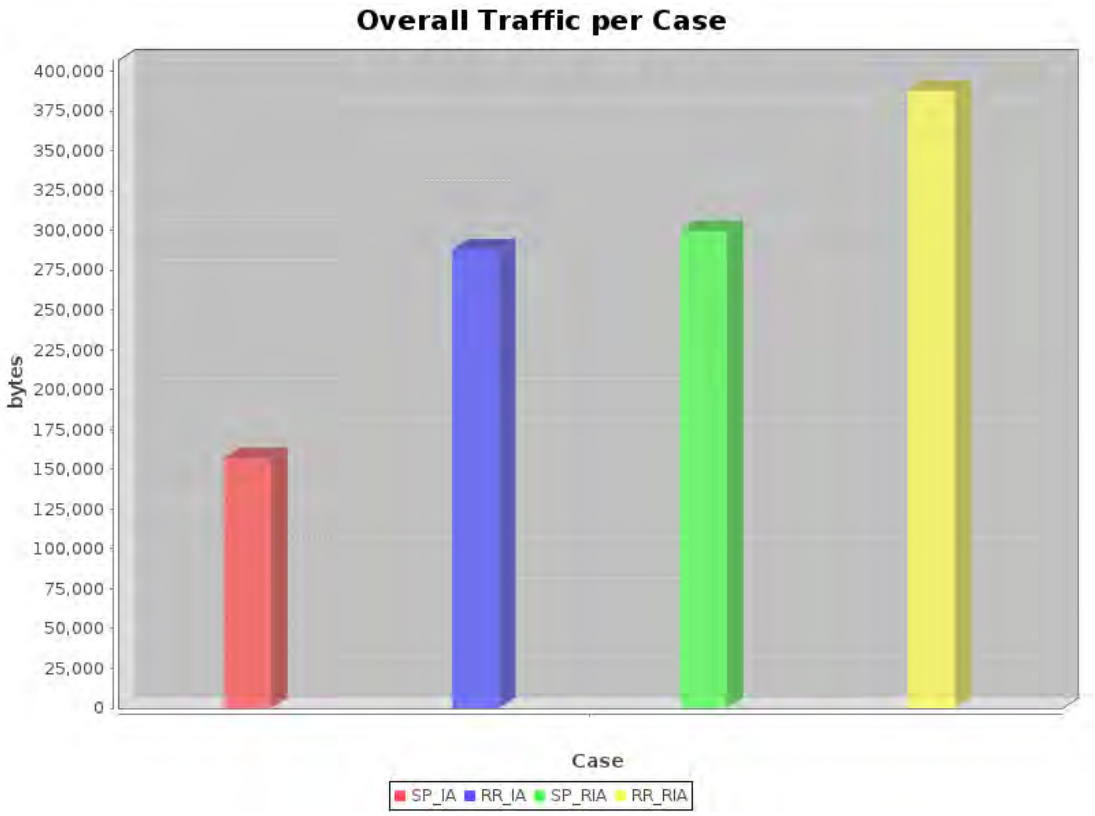
Τυχαία Ανάθεση Αποθηκών και Αντιγράφων στο δίκτυο δημιουργεί σχεδόν διπλάσιο φόρτο κίνησης στο δίκτυο συγκριτικά με τον Αλγόριθμο Τοποθέτησης και Ανάθεσης Αντιγράφων.

Πείραμα 2

Αριθμός Κόμβων: 4
Αριθμός Αποθηκών: 4
Αριθμός Αντικειμένων: 7
Αριθμός Αντιγράφων ανά Αντικείμενο: 2
Χωρητικότητα Αποθήκης: 4

Αριθμός Subscribers: 28

```
Zipf Popularity : [8, 5, 4, 3, 3, 3, 2]
Zipf Locality   : [1, 1, 1, 1, 1, 1, 1]
```



Εικόνα 39. Συνολική κίνηση στο δίκτυο σε bytes.

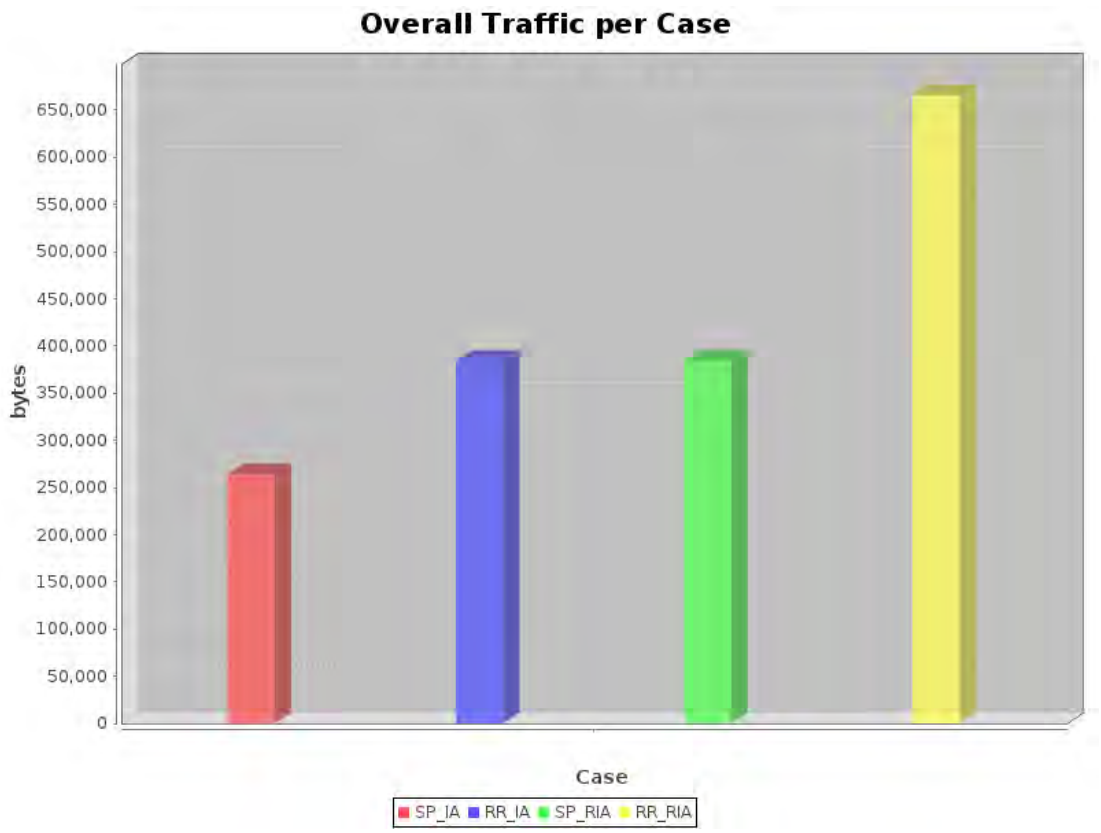
Παρόμοια με το πείραμα 1, ο συνδυασμός Shortest Path TM και Αλγορίθμου Τοποθέτησης και Ανάθεσης Αντιγράφων δίνει εμφανή μείωση στο φόρτο δικτύου. Εδώ παρατηρούμε πως ακόμα και με την χρήση του Shortest Path TM, όταν γίνεται Τυχαία Ανάθεση Αποθηκών και Αντιγράφων, ο φόρτος στο δίκτυο μπορεί να προσεγγίσει αυτόν που παράγεται με την χρήση Round Robin TM και Αλγορίθμου Τοποθέτησης και Ανάθεσης Αντιγράφων.

Πείραμα 3

Αριθμός Κόμβων: 4
Αριθμός Αποθηκών: 3
Αριθμός Αντικειμένων: 12
Αριθμός Αντιγράφων ανά Αντικείμενο: 2
Χωρητικότητα Αποθήκης: 9

Αριθμός Subscribers: 65

```
Zipf Popularity : [15, 9, 7, 6, 5, 4, 4, 3, 3, 3, 3, 3]
Zipf Locality   : [3, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```



Εικόνα 39. Συνολική κίνηση στο δίκτυο σε bytes.

Στο συγκεκριμένο πείραμα αυτό που παρατηρείται είναι πως η μείωση φόρτου δικτύου χρησιμοποιώντας τον Shortest Path TM και εναλλάσσοντας τους δύο αλγόριθμους ανάθεσης είναι μικρότερη. Αυτό ίσως να έγκειται στο γεγονός ότι ο αλγόριθμος τυχαίας ανάθεσης να έχει προσεγγίσει περισσότερο την ανάθεση που έχει κάνει ο μη τυχαίος αλγόριθμος. Εξακολουθεί όμως να υπάρχει αρκετή διαφορά μεταξύ τους.

Και στα τρία πειράματα βλέπουμε πώς ο φόρτος που δημιουργείται από την χρήση του Round Robin TM και Τυχαίου Αλγόριθμου Ανάθεσης παραμένει σταθερά μεγαλύτερος. Αντίστοιχα ο φόρτος που δημιουργείται από την χρήση Shortest Path TM και Αλγόριθμου Τοποθέτησης και Ανάθεσης Αντιγράφων παραμένει σταθερά μικρότερος.

Για την λήψη περαιτέρω πειραμάτων απαιτείται χρήση πραγματικού δικτύου με αρκετούς κόμβους ώστε να υπάρχει μεγάλη ευελιξία στην επιλογή παραμέτρων όπως ο αριθμός αποθηκών, αριθμός αντικειμένων κ.λ.π. Με αυτόν τον τρόπο θα μπορεί να ξεχωρίσει η ιδιότητα του αλγορίθμου Τοποθέτησης και Ανάθεσης Αντιγράφων έναντι του Τυχαίου αλγόριθμου. Τέλος, θα καταστούν αξιοποιήσιμες οι μετρήσεις χρόνων που λαμβάνονται από του Subscribers και να συγκριθούν χρησιμοποιώντας τους διάφορους συνδυασμούς Topology Manager – Αλγόριθμου Ανάθεσης.

7. Μελλοντικές Ιδέες

Στην παρούσα διπλωματική εργασία βασικό αντικείμενο μελέτης υπήρξε η αρχιτεκτονική των Δικτύων Βασισμένων στο Περιεχόμενο. Αποτελούν μια προσέγγιση για την δημιουργία διαδικτύου όπου η ανάθεση καθολικών ονομάτων σε αντικείμενα αποτελεί βασική αρχή. Λόγω του αυξημένου φόρτου που παράγεται σήμερα στο δίκτυο, κρίνεται απαραίτητη η μελέτη στρατηγικών και υλοποιήσεων που θα βοηθήσουν γενικά στην βελτίωση απόδοσης του δικτύου. Όπως έχουμε δει, αυτό που επιχειρείται να γίνει είναι η αποθήκευση πληροφορίας σε ενδιάμεσους κόμβους ώστε να επιτυγχάνεται η αποσυμφόρηση τόσο του αρχικού αποστολέα αλλά και συνολικά του δικτύου. Για να γίνει αυτό εφικτό, χρησιμοποιούνται εφαρμογές έξω από το επίπεδο δικτύου που τρέχουν σε μεσο-μακροπρόθεσμες περιόδους ώστε να επιλέγουν τους κατάλληλους κόμβους-αποθήκες και τα αντικείμενα που πρέπει να φυλάσσει και να δημοσιεύει η κάθε μια. Η αρχιτεκτονική κόμβου που χρησιμοποιήσαμε είναι η PURSUIT - Blackadder η οποία και εξάγει το μοντέλο υπηρεσίας publish/subscribe για επικοινωνία μεταξύ των εφαρμογών. Σε επιλεγμένους κόμβους του δικτύου τρέχει η εφαρμογή Rendezvous η οποία διαχειρίζεται τον γράφο πληροφορίας και κάνει το ταίριασμα εκδοτών και εγγραφεών που αφορούν στο ίδιο αντικείμενο πληροφορίας. Το γεγονός οτι η εφαρμογή του Topology Manager βρίσκεται εκτός του Blackadder σαν ξεχωριστή εφαρμογή δίνει την δυνατότητα στον χρήστη-προγραμματιστή να κάνει τροποποιήσεις και να ενσωματώσει δικές του πολιτικές που αφορούν στην δημιουργία μονοπατιών μεταξύ publishers και subscribers.

Πιο συγκεκριμένα στα πλαίσια αυτής της εργασίας έχουν χρησιμοποιηθεί και υλοποιηθεί αλγόριθμοι για διαφορετικά τμήματα της αρχιτεκτονικής των Information Centric Networks. Αρχικά έχει χρησιμοποιηθεί ο αλγόριθμος για το Storage and Planning Assignment που έχει υλοποιηθεί από τον συμφοιτητή Αρχάγγελο Ευθυμιάδη για την Επιλογή Αποθηκών και Ανάθεση Αντικειμένων σε αυτές. Το τμήμα του Subscription Forecast, ο Αλγόριθμος Τυχαίας Ανάθεσης Αντικειμένων στις Αποθήκες όπως επίσης και η μετατροπή του Topology Manager για Round Robin επιλογή των publishers έχουν υλοποιηθεί στα πλαίσια της εργασίας αυτής.

Το εργαλείο παρακολούθησης Monitor GUI θα μπορούσε να επεκταθεί και να ενσωματώσει διαφορετικούς αλγόριθμους που αφορούν το Subscription Forecast, Storage and Planning Assignment και τον Topology Manager. Με τον τρόπο αυτό θα μπορούσαν να γίνουν περεταίρω συγκρίσεις μεταξύ των συνδυασμών αυτών των τριών τμημάτων και να επιλεχθεί αυτός με τα καλύτερα αποτελέσματα.

Πειράματα επιχειρήθηκαν να γίνουν στο δίκτυο Planet Lab το οποίο αποτελείται από πολλούς κόμβους ανά το παγκόσμιο. Λόγω όμως του ότι ο Blackadder δεν στηρίζεται σε πρωτόκολλο που να εγγυάται την παράδοση όλων των πακέτων (π.χ TCP) οι μετρήσεις δεν θα μπορούσαν να είναι επαρκής. Ένα προβληματικό παράδειγμα είναι όταν ένα μήνυμα που εντάσσεται στην κατηγορία Asynchronous Upcalls, Start Publish Notification, δεν φθάσει ποτέ στον εκδότη. Τότε αυτός ποτέ δεν αποστέλλει τα δεδομένα στον subscriber με αποτέλεσμα να υπάρχει ανακρίβεια τόσο στις μετρήσεις του Overall Traffic στο δίκτυο όπως επίσης και στον χρόνο παραλαβής αντικειμένου που υπολογίζει ο subscriber. Για τον λόγο αυτό ήταν αναγκαίο τα πειράματα να λάβουν χώρα σε έναν υπολογιστή με ένα δίκτυο από Virtual Machines όπου η απώλεια πακέτων ήταν αμελητέα.

Μία εναλλακτική και προτεινόμενη ιδέα θα ήταν τα πειράματα να πραγματοποιηθούν σε πραγματικό αλλά VPN δίκτυο όπου η απώλεια πακέτων είναι επίσης περιορισμένη. Έτσι θα μπορούσαν να ληφθούν ακριβέστερες μετρήσεις όπως το Overall Traffic και ο Μέσος Χρόνος παραλαβής αντικειμένων από τους subscribers.

Παραπομπές

[1] Dirk Trossen and George Parisis, "Designing and Realizing An Information-Centric Internet", University of Cambridge UK

[2] George Parisis and Dirk Trossen, "Towards an implementation of an information-centric network". Euro-NF International Workshop on Traffic and Congestion Control for the Future Internet, Volos, Greece, March 2011

[3] "PSIRP Publish Subscribe Internet Routing Paradigm" at http://www.psirp.org/

[4] "CCNx Project" at https://www.ccnx.org/

[5] "4WARD Project" at http://www.4ward-project.eu/

[6] "NDN Project" at http://www.named-data.net/

[7] "PURSUIT Blackadder Project" at http://www.fp7-pursuit.eu/PursuitWeb/

[8] Somaya Arianfar, Pekka Nikander, "Packet-level Caching for Information-centric Networking"

[9] SIGCOMM Blackadder 2012, "Node Design for an Information-Centric Network Architecture"

[10] Dirk Trossen (ed.) (UCAM), George Parisis (UCAM), Kari Visala (AALTO - HIIT), Borislava Gajic (RWTH), Janne Riihijarvi (RWTH), Paris Flegkas (CERTH), Pasi Sarolahti (AALTO - HIIT), Petri Jokela (LMF), Xenofon Vasilakos (AUEB), Christos Tsilopoulos (AUEB), Somaya Arianfar (AALTO – HIIT), "PURSUIT,Publish Subscribe Internet Technology",FP7 - INFSO- ICT - 257217

[11] "Click Modular Router" at http://www.read.cs.ucla.edu/click/click

[12] Petri Jokela, András Zahemszky, Christian Esteve Rothenberg, Somaya Arianfar, and Pekka Nikander, "LIPSIN: Line Speed Publish/Subscribe Inter-Networking"

[13] Vasilis Sourlas, Paris Flegkas and Leandros Tassioulas, “Replication and Opportunistic Caching in Information-Centric Networking”, Department of Computer & Communication Engineering. University of Thessaly, Greece. Centre for Research and Technology Hellas - ITI

[14] J. Kangasharju, J. Roberts, K. Ross, “Object replication strategies in content distribution networks”, Comput. Commun. Elsevier, vol. 25, pp. 376–383, March 2002.

[15] L. Qiu, V.N. Padmanabhan and G. Voelker, “On the placement of web server replicas”, In Proc. of IEEE INFOCOM, Anchorage, USA, Apr. 2001.

[16] JUNG - Java Universal Network/Graph Framework at http://jung.sourceforge.net/

[17] JCraft – JSch – Java Secure Channel at http://www.jcraft.com/jsch/

[18] Planet Lab at http://www.planet-lab.org/