ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ, ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

Υλοποίηση μηχανισμού αποθήκευσης (caching) σε επίπεδο πακέτων για δίκτυα βασισμένα στο περιεχόμενο

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΔΗΜΗΤΡΗ ΓΕΩΡΓΙΟΥ

Επιβλέποντες : Λέανδρος Τασιούλας
Καθηγητής Τ.Μ.Η.Υ.Τ.Δ
Δημήτριος Κατσαρός
Λέκτορας Τ.Μ.Η.Υ.Τ.Δ

Βόλος, Φλεβάρης 2012

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ, ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

Υλοποίηση μηχανισμού αποθήκευσης (caching) σε επίπεδο πακέτων για δίκτυα βασισμένα στο περιεχόμενο

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΔΗΜΗΤΡΗ ΓΕΩΡΓΙΟΥ

Επιβλέποντες : Λέανδρος Τασιούλας
Καθηγητής Τ.Μ.Η.Υ.Τ.Δ
Δημήτριος Κατσαρός
Λέκτορας Τ.Μ.Η.Υ.Τ.Δ

Βόλος, Φλεβάρης 2012

Ευχαριστίες

Θερμές ευχαριστίες εκφράζω στους καθηγητές μου κύριο Λέανδρο Τασιούλα και κύριο Δημήτριο Κατσαρό στην εμπιστοσύνη που έδειξαν στην επίβλεψη της διπλωματικής μου εργασίας.

Ιδιαίτερα θέλω να ευχαριστήσω τον Πάρη Φλέγκα για την όλη καθοδήγηση του, για τις συμβουλές και την βοήθεια που μου παρείχε στην εκπόνησης της εργασίας αυτής.

Επίσης θέλω να ευχαριστήσω τον Δημήτρη Συρίβελη για την σημαντική υποστήριξη και βοήθεια που μου παρείχε σε θέματα σχετικά με την εργασία αυτή.

Θέλω να ευχαριστήσω τους συγκατοίκους μου Αδάμου Γιώργο, Ανδρέου Σταύρο και Βαγιανού Πόλυ καθώς και τους φίλους μου Χριστοφή Μάριο και Γιωργαλλίδη Ανδρέα για το ενδιαφέρον που έδειχναν με την κάθε εξέλιξη της διπλωματικής μου εργασίας μέχρι και την ολοκλήρωση της.

Τέλος θέλω να ευχαριστήσω και να αφιερώσω την διπλωματική μου εργασία στην οικογένεια μου για την όλη συμπαράσταση και βοήθεια που μου παρείχε στα χρόνια της φοιτητικής μου ζωής.

Δημήτρης Γεωργίου

Βόλος, 2012

Περίληψη

Ο κύριος στόχος της διπλωματικής εργασίας είναι η δημιουργία ενός μηχανισμού αποθήκευσης (caching) σε επίπεδο πακέτων που θα ενσωματώνεται σε δίκτυα βασισμένα στο περιεχόμενο. Η βασική λειτουργία του μηχανισμού αποθήκευσης είναι να τρέχει σε κάθε κόμβο του δικτύου και να παρέχει μια ενδιάμεση αποθήκευση για τα πακέτα δεδομένων τα οποία περνούν από τον κόμβο. Έτσι τα δεδομένα που είναι αποθηκευμένα θα μπορούν ανά πάσα στιγμή να ανακτηθούν από αιτήσεις που πιθανόν να περνούν από τους κόμβους αυτούς. Με τη δημιουργία αυτή του μηχανισμού μας δίνεται η δυνατότητα να δούμε την συνεισφορά που θα έχει η χρήση των ενδιάμεσων μνημών cache πάνω στα δίκτυα τα οποία βασίζονται στο περιεχόμενο ως προς τον χρόνο που ανακτάται ένα πακέτο από έναν ενδιάμεσο κόμβο παρά από τον αρχικό κόμβο που είναι η πηγή του. Επιπλέον μας δίνεται η δυνατότητα να δούμε την συνολική μείωση στο φόρτο του δικτύου. Για τον σκοπό αυτό πραγματοποιήθηκαν πειράματα στο διεθνές δίκτυο planetlab για διάφορες τοπολογίες δικτύων.

Η υλοποίηση του μηχανισμού εφαρμόστηκε και ενσωματώθηκε στο υπό ανέγερση project Blackadder το οποίο αναπτύσσεται στα πλαίσια ερευνητικών υλοποιήσεων και μέσω διάφορων demo εφαρμογών εκτελεί πλήρη λειτουργικότητα των δικτύων βασισμένων στο περιεχόμενο. Συγκεκριμένα για την υλοποίηση του μηχανισμού αποθήκευσης παρουσιάσαμε κάποιες νέες ιδέες. Μερικές στηριχτήκαν στην προηγούμενη μορφή λειτουρίας του Blackadder, αλλά κάποιες από αυτές μας ανάγκασαν να κάνουμε επέκταση της λειτουργίας του Blackadder. Υλοποιήθηκαν τρεις αλγόριθμοι για την απόρριψη των πακέτων, FIFO, LRU και LFU.

Η αρχική αυτή υλοποίηση είναι δυνατόν να γίνει πρόκληση για την πραγματοποίηση ολοκληρωμένων εφαρμογών που θα μπορούν να κρατούν αρχεία και δεδομένα μεγάλου όγκου καθώς επίσης να αποτελέσει κίνητρο για μελλοντική επέκταση του μηχανισμού.

Περιεχόμενα

1.Εισαγωγη

Η ραγδαία ανάπτυξη που παρουσιάζεται στο διαδίκτυο, δημιούργησε την ανάγκη για δημιουργία νέων αρχιτεκτονικών δικτύων. Οι αρχιτεκτονικές θα πρέπει να έχουν την ικανότητα να διαχειριστούν το μεγάλο φόρτο που παρουσιάζεται σήμερα στο διαδίκτυο παρέχοντας παράλληλα αξιοπιστία και ασφάλεια στα δεδομένα των χρηστών. Μια σύγκλιση που προέκυψε μέσα από τις έρευνες, είναι η δημιουργία δικτύων που θα έχουν ως βασικό συστατικό το περιεχόμενο της πληροφορίας που είναι κατανεμημένη στο διαδίκτυο. Τα δίκτυα βασισμένα στο περιεχόμενο (Content-Based Networks - CBN ή αλλιώς Information-Centric Networks - ICN) είναι μια νεοεμφανιζόμενη αρχιτεκτονική δικτύου η οποία καταλαμβάνει ολοένα και μεγαλύτερο έδαφος στον χώρο της δικτύωσης. Κάποιες από τις αρχιτεκτονικές που υποστηρίζουν δίκτυα βασισμένα στο περιεχόμενο βασίζονται στην τεχνική του publish/subscribe.

Το λογισμικό Blackadder εφαρμόζει μια υλοποίηση των δικτύων βασισμένων στο περιεχόμενο και στηρίζεται στην τεχνική του publish/subscribe. Ο Blackadder είναι ένα υπό ανάπτυξη λογισμικό και μέσω διάφορων βοηθητικών εφαρμογών(application) εκτελεί παράδοση και παραλαβή δεδομένων μέσω νέων χαρακτηριστικών ιδιοτήτων. Πολλές από αυτές τις νέες χαρακτηριστικές ιδιότητες παραμένουν αναλλοίωτες για όλες τις αρχιτεκτονικές που υποστηρίζουν δίκτυα βασισμένα στο περιεχόμενο. Επίσης οι ιδιότητες αυτές, όσο άφορα τον τρόπο ανταλλαγής των δεδομένων και πιο γενικά οποιασδήποτε επικοινωνίας απέχουν κατά πολύ από τα μέχρι στιγμής γνωστά δίκτυα IP/TCP.

Επομένως τα δίκτυα βασισμένα στο περιεχόμενο γενικά σαν μια νέα αρχιτεκτονική καθώς και συγκεκριμένα το νέο λογισμικό του Blackadder είναι ευπρόσδεκτα σε οποιεσδήποτε βελτιστοποίησης παρέχονται από τους ερευνητές.

Εμείς την παρούσα διπλωματική εργασία καλούμαστε να συνεφέρουμε στο νέο λογισμικό του Blacakdder. Η συνεισφορά μας θα είναι κάποιες προσθήκες που ευελπιστούμε ότι θα παρέχουν βελτίωση σε συγκριμένα θέματα πάνω στο λογισμικό Blackadder.

1.1. Κίνητρο

Το σημαντικότερο κίνητρο που μας ώθησε να ασχοληθούμε με το συγκεκριμένο θέμα είναι ότι με την αποθήκευση διαδικτυακού περιεχομένου μέσα στο ίδιο δίκτυο θα πετύχουμε την μείωση του χρόνου παράδοσης των δεδομένων που ζητούν οι χρηστές. Επίσης πετυχαίνουμε μείωση στο συνολικό φόρτο στο δίκτυο. Με την αποθήκευση λοιπόν του διαδικτυακού περιεχομένου σε κρυφές μνήμες, ο χρήστης έχει τη δυνατότητα να βρει τα δεδομένα σε ενδιάμεσους κόμβους οι οποίοι είναι πιο κοντά σε αυτούς παρά από τον αρχικό που είναι η πηγή τους. Με τη χρήση κρυφών μνημών μειώνεται παράλληλα και ο φόρτος του αρχικού κόμβου ο οποίος είναι η πηγή των δεδομένων, αφού δεν δέχεται πλέον όλες τις αιτήσεις. Η τεχνική των κρυφών μνημών από πλευράς

στρατηγικών αντικατάστασης, μπορεί να διαφοροποιηθεί σε πολλά μοντέλα είτε βάσει αλγορίθμου είτε βάσει της πληροφορίας των δεδομένων.

Για το λόγο ότι τα δίκτυα που βασίζονται στο περιεχόμενο αποτελούν μια νεοεμφανιζόμενη αρχιτεκτονική δικτύου, δεν περιέχουν μέχρι στιγμής κάποιο είδος μηχανισμού αποθήκευσης (caching). Επομένως στην παρούσα διπλωματική εργασία, έχουμε σαν στόχο την δημιουργία κάποιου μηχανισμού αποθήκευσης. Τελειώνοντας λοιπόν αυτήν την διπλωματική εργασία θα είμαστε σε θέση να στηρίξουμε την άποψη, ότι η χρήση μηχανισμών αποθήκευσης στα δίκτυα που βασίζονται στο περιεχόμενο θα επιφέρει μεγάλα ωφελήματα στο χώρο της δικτύωσης.

1.2. Ορισμός του Προβλήματος

Το πρόβλημα το οποίο μας απασχολεί στην συγκεκριμένη διπλωματική εργασία είναι η δημιουργία ενός μηχανισμού αποθήκευσης που να μπορεί να αποθηκεύσει πακέτα δεδομένων σε δίκτυα που βασίζονται στο περιεχόμενο. Επιπλέον ο συγκεκριμένος μηχανισμός αποθήκευσης μας ενδιαφέρει να έχει την δυνατότητα να ενσωματώνεται στο λογισμικό Blackadder, το οποίο τρέχει σε κάθε δρομολογητή του δικτύου. Επίσης να είναι σε θέση να παραλαμβάνει και να επεξεργάζεται πακέτα δεδομένων και πακέτα αιτήσεων δεδομένων. Τα πακέτα που πρέπει να επεξεργάζεται προέρχονται από το δίκτυο ή και από εφαρμογές που τρέχουν στον ίδιο κόμβο. Ακόμη να είναι σε θέση να στέλνει δεδομένα στο δίκτυο και σε εφαρμογές που τρέχουν στον ίδιο κόμβο. Επίσης να αποθηκεύει μόνο πακέτα δεδομένων και όχι οποιαδήποτε άλλα πακέτα του πρωτοκόλλου επικοινωνίας. Εκτός απ' αυτό θέλουμε να είναι σε θέση να επεξεργάζεται αιτήσεις για δεδομένα. Με αυτό εννοούμε ότι κάθε αίτηση που λαμβάνει ο δρομολογητής για δεδομένα, ο μηχανισμός αποθήκευσης να μπορεί να ελέγχει αν η αίτηση απευθύνεται σε κάποια δεδομένα που είναι αποθηκευμένα στην μνήμη του. Έτσι είναι δυνατόν να εξυπηρετήσει τον χρήστη που έστειλε την αίτηση, στέλνοντας τα δεδομένα από τον δρομολογητή σε αυτόν. Αλλιώς να την προωθεί στον επόμενο κόμβο.

1.3. Δομή της Εργασίας

Η διπλωματική εργασία ακολουθεί μια δομή που αποτελείται από 5 κεφάλαια. Το πρώτο κεφάλαιο αποτελεί την «Εισαγωγή» και περιλαμβάνει τα κίνητρα και τους στόχους που μας ώθησαν στην ανάπτυξη του μηχανισμού αποθήκευσης. Επίσης περιγράφει τον ορισμό του προβλήματος που καλείται να λύσει. Επιπλέον σε αυτό το κεφάλαιο αναφέρονται οι λόγοι που οδήγησαν στη συγγραφή αυτής της διπλωματικής και αναλυτικά τους στόχους που έχει να επιτύχει κατά την ολοκλήρωσή της. Στο δεύτερο κεφάλαιο γίνεται λόγος για σχετικές εργασίες που αποτέλεσαν τις βασικές ιδέες και τα κίνητρα ώστε να δημιουργηθεί η

ανάγκη της ύπαρξης αυτού του μηχανισμού αποθήκευσης. Στο κεφαλαίο 3 αναπτύσσουμε την σχεδίαση εξολοκλήρου του μηχανισμού αποθήκευσης από τη σχεδίαση και το πώς λειτούργει μέχρι και την ενσωμάτωση του στο προηγούμενο λογισμικό. Στο κεφάλαιο 4 παρουσιάζουμε πειράματα που κάναμε στο διαδίκτυο, έτσι βλέπουμε στην πράξη τα πλεονεκτήματα που έχει ο μηχανισμός αποθήκευσης στους χρήστες αλλά και γενικά στο δίκτυο. Τέλος στο κεφάλαιο 5 όπου ολοκληρώνεται η διπλωματική εργασία γράφουμε τα γενικά μας συμπεράσματα στα οποία καταλήξαμε. Επιπλέον δίνουμε μελλοντικές ιδέες που είναι δυνατόν να αποτελέσουν νεότερα θέματα για άλλες εργασίες που θα βασίζονται στην ύπαρξης αυτής της διπλωματικής.

2. Σχετικές Εργασίες

Σε αυτό το κεφάλαιο συγκεντρώνονται πληροφορίες από διάφορες εργασίες που σχετίζονται με την παρούσα διπλωματική εργασία. Αρχικά κάνουμε μια γενική εισαγωγή στα δίκτυα βασισμένα στο περιεχόμενο. Στην συνέχεια περιγράφουμε την αρχιτεκτονική RTFM που είναι σχεδιασμένη να λειτούργει αποκλειστικά για δίκτυα που επικεντρώνονται στο περιεχόμενο και βασίζεται στην τεχνική του publish/subscribe. Στη συνέχεια παρουσιάζουμε αναλυτικά την γενική λειτουργία του Blackadder, που είναι το λογισμικό όπου θα ενσωματωθεί ο μηχανισμός αποθήκευσης. Βλέπουμε ιδέες όπως τα zFilters που βασίζονται στα Silicon Bloom-Filters και χρησιμοποιούνται στην δρομολόγηση των πακέτων διαμέσω των συνδέσεων του δικτύου. Κάνουμε μια εισαγωγή για τις στρατηγικές απόρριψης πακέτων από μνήμες cache. Στο τέλος περιγράφουμε συνοπτικά τον Click Modular Router που είναι το εργαλείο που τρέχει το λογισμικό του Blackadder.

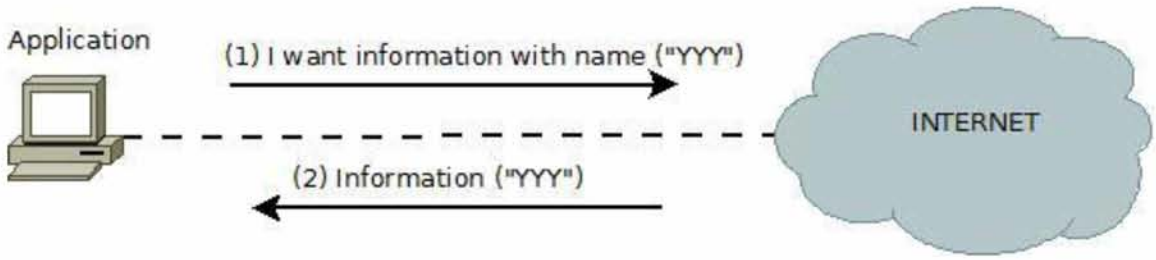
Γενικά το συγκεκριμένο κεφάλαιο περιλαμβάνει τις βασικές ιδέες, που αποτελούν τα κλειδιά για την υλοποίηση του μηχανισμού αποθήκευσης. Επίσης μας βοηθούν να συνθέσουμε νέες δικές μας ιδέες για το μοντέλο μας. Η ακρίβεια των στοιχείων που συλλέξαμε κρίθηκε απαραίτητη ώστε να οδηγηθούμε στην σωστή υλοποίηση του μηχανισμού αποθήκευσης.

2.1. Δίκτυα Βασισμένα στο Περιεχόμενο

Η παρούσα αρχιτεκτονική δικτύου (TCP/IP) επιτρέπει την επικοινωνία μεταξύ ακριβώς δυο κόμβων ένας κόμβος θεωρείται ένα μηχάνημα. Π.χ.: ένας υπολογιστής, ένας εξυπηρετητής, ένας δρομολογητής, κ.ο.κ. Όταν ένας κόμβος αιτείται έναν πόρο που βρίσκεται σε κάποιον άλλο κόμβο, τότε ο δεύτερος που δέχεται την αίτηση παρέχει στον πρώτο πρόσβαση ώστε να παραλάβει τον πόρο. Οποιαδήποτε επικοινωνία μεταξύ δυο κόμβων απαιτεί πακέτα της μορφής IP, που περιέχουν τις IP διευθύνσεις και των δυο μηχανημάτων, πηγής και προορισμού, ώστε να επιτευχθεί η επικοινωνία. Έτσι οι χρήστες αναγκάζονται πρώτα να προσδιορίσουν την τοποθεσία κάποιου κόμβου που έχει την πληροφορία που ζητούν και στην συνέχεια να την παραλάβουν.

Είναι προφανές ότι οι χρήστες χρησιμοποιούν το διαδίκτυο τόσο για το τι περιέχει, αλλά στην πραγματικότητα με την παρούσα υλοποίηση του IP/TCP δικτύου γίνεται πρώτα η επικοινωνία με τον κόμβο που έχει την πληροφορία που ζητούν. Αν δούμε το μοντέλο διευθύνσεων σε βήματα. Το πρώτο βήμα είναι να βρούμε ποιός "WHO" έχει την πληροφορία και το δεύτερο βήμα είναι να παραλάβουμε τι "WHAT" πληροφορία θέλουμε. Τα δίκτυα βασισμένα στο περιεχόμενο έρχονται να προσπεράσουν το πρώτο στάδιο που είναι το ποιός έχει την πληροφορία και στοχεύουν απευθείας στο τι πληροφορία θέλουμε. Στο Σχήμα 1 βλέπουμε την βασική ιδέα της λειτουργίας των δικτύων που βασίζονται στο περιεχόμενο, όπου μια εφαρμογή ζητά από το δίκτυο μια πληροφορία και το

δίκτυο είναι υπεύθυνο να βρει την πληροφορία αυτή και να την παράδοση στην εφαρμογή.



Σχήμα 1: *Ο τρόπος λειτουργίας των δικτύων που βασίζονται στο περιεχόμενο. Μια εφαρμογή ζητά από το δίκτυο τι πληροφορία θέλει να παραλάβει και το δίκτυο είναι υπεύθυνο να βρει και να της παραδώσει την πληροφορία αυτή.*

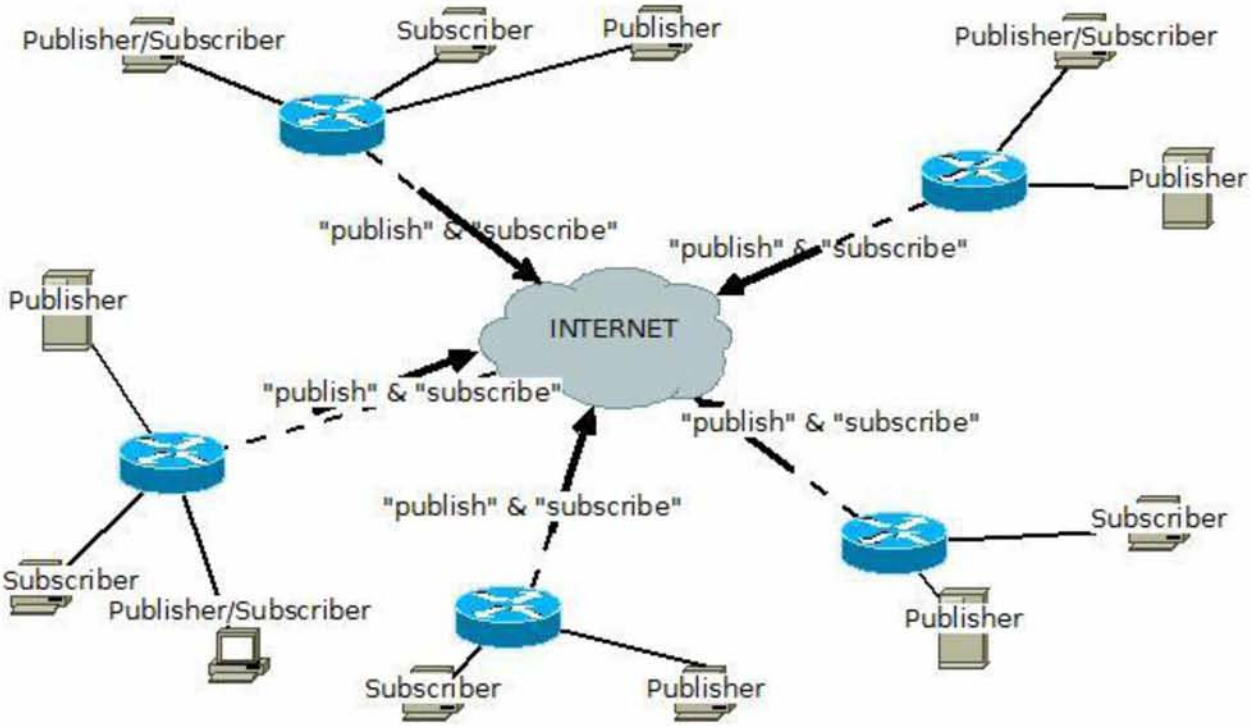
Παρόμοια στα παραδοσιακά μοντέλα δικτύων που επικεντρώνονται στις διευθύνσεις Address-Based Network (ABN), τα Δίκτυα Βασισμένα στο περιεχόμενο (CBN) περιλαμβάνουν ένα σύνολο από κόμβους και δρομολογητές που ενώνουν τους κόμβους αυτούς μέσω συνδέσεων. Η κύρια διαφορά μεταξύ των δύο τύπων δικτύων είναι το ότι στα CBN δεν αναγνωρίζονται οι κόμβοι με μοναδικό όνομα δικτύου, αλλά οι συνδέσεις μεταξύ των κόμβων. Για αυτό τον λόγο στα CBN τα μηνύματα δεν δρομολογούνται σε συγκεκριμένους κόμβους αλλά στις συνδέσεις που συνδέουν τους κόμβους. Τα μηνύματα που στέλνονται περιλαμβάνουν τις ονομασίες των πληροφοριών που ενδιαφέρεται ο χρήστης. Η κύρια αυτής προσέγγισης έχει ως στόχο να γίνεται η εύρεση των δεδομένων ανεξαρτήτως του που βρίσκονται αποθηκευμένα.

Το μοντέλο δικτύου που βασίζεται στο περιεχόμενο μεταφέρει τον έλεγχο αποστολής δεδομένων από τον αποστολέα στον παραλήπτη. Αυτό επιτυγχάνεται με την αλλαγή της αρχιτεκτονικής δικτύου από το μοντέλο push στο μοντέλο pull. Δηλαδή το δίκτυο να μπορεί να παραδώσει τα δεδομένα σε κάποιον που ενδιαφέρεται να τα λάβει, παρά να ξέρει ποιος θα είναι ο αποστολέας των δεδομένων εκ των προτέρων.

Οι πιο πολλές αρχιτεκτονικές δικτύων που βασίζονται στο περιεχόμενο βασίζονται στην τεχνική του publish/subscribe. Ένα μεγάλο μέρος του πλήθους των εφαρμογών που παρέχονται στο διαδίκτυο σήμερα βασίζονται στην τεχνική του publish/subscribe. Σε γενικές γραμμές η τεχνική publish/subscribe είναι μια μέθοδος διάδοσης των δεδομένων που παρέχει αποσύνδεση σε χρόνο, χώρο και συγχρονισμό στην επικοινωνία μεταξύ των παραγωγών και των καταναλωτών. Βασικά χαρακτηριστικά που εκφράζουν την τεχνική αυτή είναι ο χειρισμός των δεδομένων με βάση την ονομασία τους (data-oriented naming), προσωρινή αποθήκευση για να διακόπτεται η σύνδεση μεταξύ παραγωγών και καταναλωτών (network caching), αποδοτική πολύ-εκπομπή (efficient multicast) για την διάδοση των δεδομένων.

Στο Σχήμα 2 βλέπουμε διάφορες εφαρμογές που δρουν ως publisher είτε ως

subscriber είτε ως publisher και subscriber μαζί, όπου στέλνουν τα αιτήματα τους "publish" ή/και "subscribe" στο δίκτυο.



Σχήμα 2: *Αιτήσεις "publish" και "subscribe" των χρηστών σε Δίκτυο Βασισμένο στο Περιεχόμενο.*

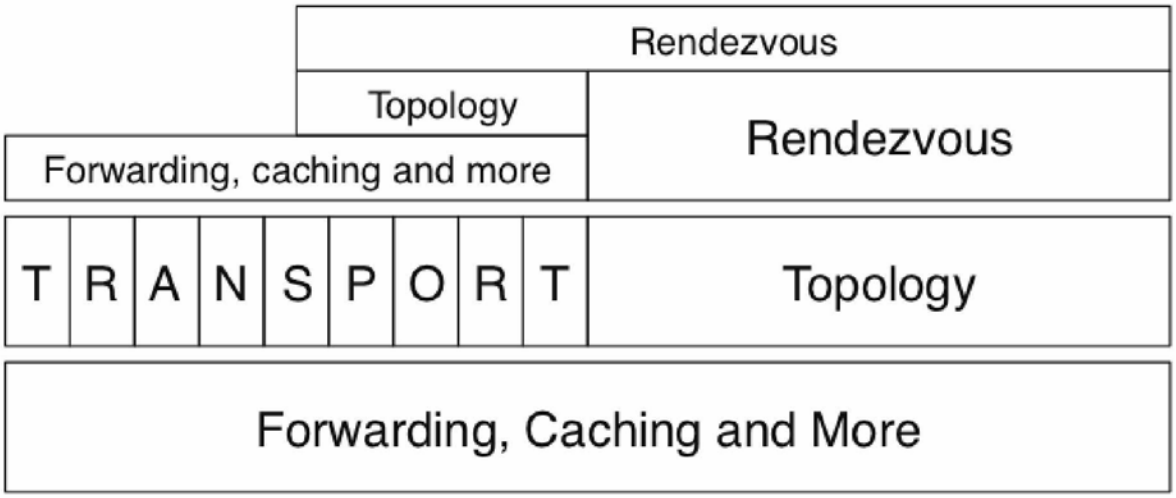
Διάφορα ερευνητικά προγράμματα στόχευαν στην ανάπτυξη μιας εξολοκλήρου αρχιτεκτονικής που να βασίζεται στην τεχνική publish/subscribe. Περασμένα ερευνητικά προγράμματα, όπως PSIRP, CCNx και 4WARD, εστιάζονται σε προσεγγίσεις που βασίζονται στο περιεχόμενο. Πρόσφατα άρχισαν τα προγράμματα, όπως το NDN και το PURSUIT που έχουν ως στόχο την ίδια κατεύθυνση. Μια από αυτές τις νέες αρχιτεκτονικές που προέκυψαν είναι η αρχιτεκτονική RTFM, που εξολοκλήρου βασίζεται στην τεχνική publish/subscribe και πάνω σε αυτήν θα βασιστούμε στην παρούσα διπλωματική.

2.1.1. Αρχιτεκτονική RTFM

Η αρχιτεκτονική RTFM περιγράφει τα βασικά στοιχεία που είναι απαραίτητα για τη λειτουργία της τεχνικής publish/subscribe στα δίκτυα που βασίζονται στο περιεχόμενο. Η τεχνική του publish/subscribe δίνει την δυνατότητα σε κάθε κόμβο του δικτύου να έχει μόνο δύο πρότυπες λειτουργίες: Τη δημοσίευση (publish) και εγγραφή σε μια δημοσίευση (subscribe). Η Αρχιτεκτονική RTFM

γενικά περιγράφει το πως ένας κόμβος εντάσσεται στο δίκτυο, πως δημοσιεύει δεδομένα και πως εγγράφεται σε κάποιες δημοσιεύσεις. Επιπλέον περιγράφει και τις απαιτήσεις των publish/subscribe δρομολογητών (sprouters), στο πως θα βρεθεί ένα μονοπάτι προς μια δημοσίευση από μια εγγραφή και τι συμβαίνει όταν ένας κόμβος αποφασίζει να δημοσιεύσει δεδομένα.

Η Αρχιτεκτονική αποτελείται από τρία λειτουργικά μέρη: Rendezvous, Topology και Forwarding (και τα physical Media). Από τα αρχικά αυτών των λειτουργικών μερών πήρε και το όνομα της, RTFM . Μια γενική εικόνα της αρχιτεκτονικής RTFM απεικονίζεται στο Σχήμα 3.



Σχήμα 3: *Η αρχιτεκτονική RTFM.*

2.1.1.1. Λειτουργικό μέρος Forwarding

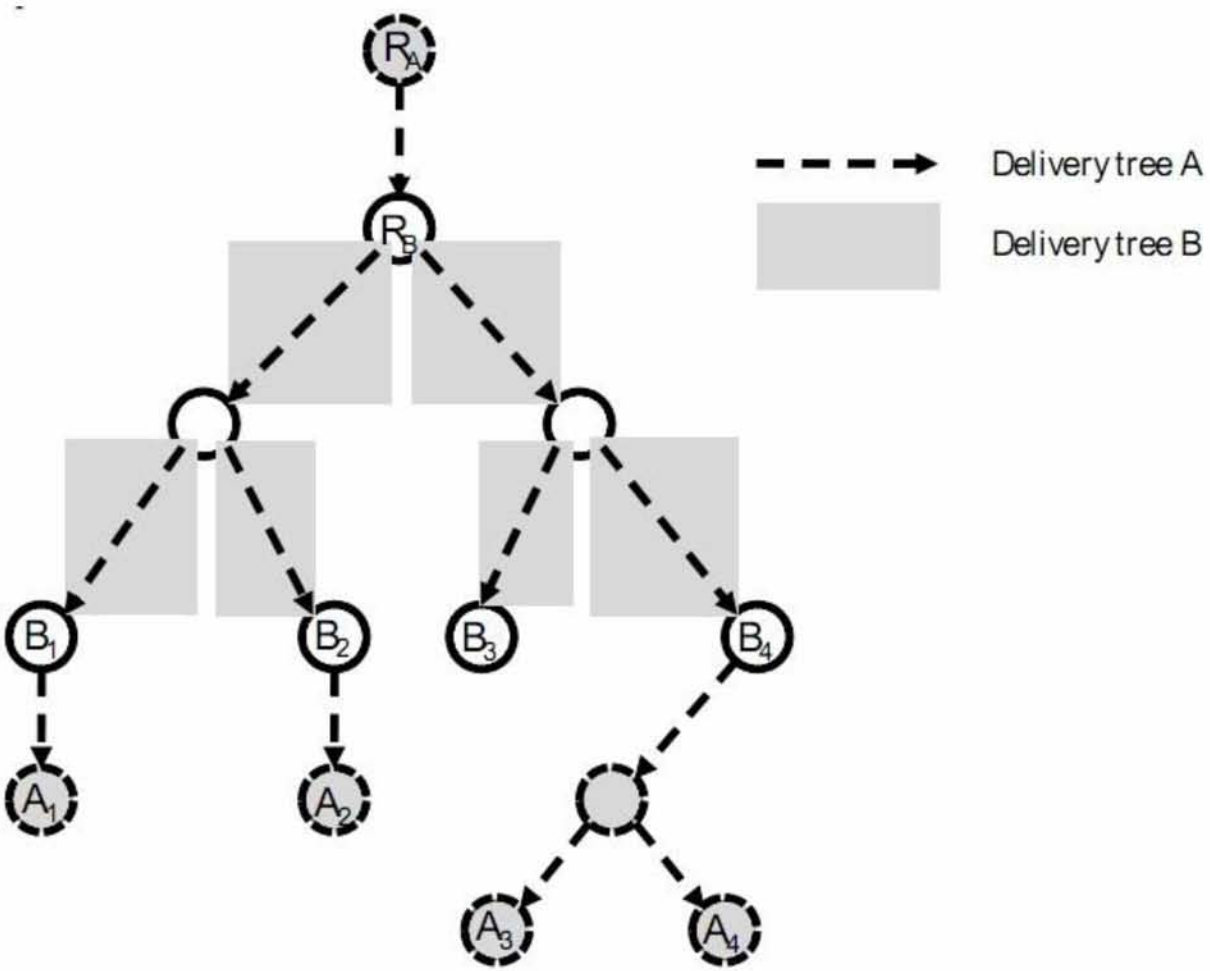
Χρησιμοποιείται για να παραδώσει δεδομένα από μια τοποθεσία σε μια άλλη. Συγκεκριμένα βασίζεται σε διευθυντικές ετικέτες που φέρουν τα πακέτα. Μια ετικέτα είναι ένας δυαδικός αριθμός τον οποίο χρησιμοποιούν οι δρομολογητές ώστε να αποφασίσουν προς τα που να προωθήσουν το πακέτο. Στον πίνακα φαίνεται ένας πίνακας προώθησης κάποιου δρομολογητή. Εξερχόμενες θύρες, είναι η αρίθμηση των διάφορων συνδέσεων προς κόμβους που είναι ενωμένες με τον δρομολογητή και είναι αυτές οι συνδέσεις στις οποίες μπορεί το πακέτο να προωθηθεί. (Για παράδειγμα από την πρώτη γραμμή του πίνακα, το πακέτο με ετικέτα "X" θα προωθηθεί προς την θύρα 1 και θα προστεθεί στο πεδίο που είχε την ετικέτα "X", η ετικέτα "AX". Ενώ αν έρθει κάποιο πακέτο που έχει ως πρόθεμα την ετικέτα "Z", θα προωθηθεί προς την θύρα 3 και θα του προστεθεί η ετικέτα "*", που σημαίνει ότι στην συνέχεια θα μπορεί να προωθηθεί προς όλες τις θύρες του επόμενου κόμβου.)

Εισερχόμενη Ετικέτα	Εξερχόμενες Θύρες	Εξερχόμενες Ετικέτες
X	1	AX
Y	1 2	Y Y
Z*	3	*

Πίνακας 1: *Πίνακας προώθησης που χρησιμοποιεί το λειτουργικό μέρος Forwarding*

2.1.1.2. Λειτουργικό μέρος Topology

Το λειτουργικό μέρος Topology έχει ως στόχο την δημιουργία και την διατήρηση δέντρων παράδοσης. Τα δέντρα παράδοσης χρειάζονται για την δρομολόγηση της κίνησης των πακέτων στο δίκτυο για κάποια συγκεκριμένη πληροφορία. Τα δέντρα που δημιουργούνται τα χρησιμοποιεί το λειτουργικό μέρος Forwarding για να προωθεί τα πακέτα προς τον προορισμό τους. Το Topology δρα με δύο τρόπους: Είτε άμεσα είτε αντιδραστικά. Η άμεση ενέργεια του δημιουργεί τα δέντρα παράδοσης που χρειάζονται για την παράδοση της πληροφορίας. Η αντιδραστική ενέργεια του δημιουργεί νέα δέντρα παράδοσης, όταν τα υπάρχοντα δέντρα παράδοσης δεν παρέχουν την κατάλληλη συνδεσιμότητα ή όταν ένας σύνδεσμος χαλάσει, ή όταν το παρόν δέντρο παράδοσης είναι μη βέλτιστο. Και αυτό γιατί μπορεί να έγιναν αλλαγές στους παραλήπτες των δεδομένων. Ένα παράδειγμα φαίνεται στο Σχήμα 4, όπου το δέντρο παράδοσης Α χρησιμοποιεί το δέντρο παράδοσης Β, ώστε να επιτευχθεί η παράδοση των δεδομένων από τον κόμβο ρίζα RA προς στους κόμβους φύλλα, A1, ..., A4. Από την παράδοση φαίνεται ότι το δέντρο Α χρησιμοποιεί το δέντρο Β καθώς επίσης και μια μη βέλτιστη λύση αφού ο κόμβος B3 παραλαμβάνει δεδομένα που δεν ενδιαφέρεται να πάρει. Αυτό μπορεί να βελτιωθεί από το λειτουργικό μέρος Topology και έτσι να δημιουργηθεί ένα βέλτιστο δέντρο παράδοσης που δεν θα χρησιμοποιεί τον κόμβο B3.



Σχήμα 4: Δέντρο παράδοσης πληροφορίας που φτιάχνει το λειτουργικό μέρος Topology.

2.1.1.3. Λειτουργικό μέρος Rendezvous

Όταν ένας κόμβος γίνεται συνδρομητής σε μια δημοσίευση, το δίκτυο πρέπει να βρει ένα αντίγραφο και να του το παραδώσει. Το λειτουργικό μέρος Rendezvous είναι μια κατανεμημένη δομή που παρέχει αυτήν την υπηρεσία. Μπορεί να είναι ένας κατανεμημένος πίνακας κατακερματισμού ή μια διαφορετική κατανεμημένη δομή. Χρησιμοποιεί την κατανεμημένη δομή συγκεντρώνοντας αρκετές πληροφορίες για τους εγγραφείς σε μια δημοσίευση (subscribers) και για αυτούς που την δημοσιεύουν (publishers). Τις πληροφορίες τις παραδίδει στο λειτουργικό μέρος Topology.

2.1.2. Blackadder

Ο Blackadder είναι μια υλοποίηση της αρχιτεκτονικής υποδομής δικτύου που βασίζεται στις αρχιτεκτονικές σταθερές που προτείνονται στο πλαίσιο του λειτουργικού μοντέλου PURSUIT. Ο Blackadder εξάγει ένα απλό και πλήρη λειτουργικό εξυπηρετικό μοντέλο των δικτύων βασισμένων στο περιεχόμενο και βασίζεται στην χωρίς συγχρονισμό επικοινωνία τεχνική publish/subscribe.

2.1.2.1. Σχεδιαστικές ιδιότητες

Ο Blackadder βασίζεται σε πέντε σχεδιαστικές ιδιότητες. Οι σχεδιαστικές ιδιότητες αυτές ανήκουν στο σύνολο των αναλλοίωτων προδιαγραφών οι οποίες παραμένουν αμετάβλητες για όλες τις υπό ανάπτυξη αρχιτεκτονικές που έχουν ως στόχο τα δίκτυα που βασίζονται στο περιεχόμενο.

1. **(Identifying individual information items)** Η πρώτη ιδιότητα είναι η έννοια του μοναδικού χαρακτηρισμού των *στοιχείων πληροφορίας (information items)* που αποτελεί το μέσο για τον εντοπισμό τους. Κάθε στοιχείο πληροφορίας αναγνωρίζεται χρησιμοποιώντας μια μοναδική αναγνωριστική ετικέτα. Η αναγνώριση μπορεί να λάβει χώρο σε ένα σχήμα ιεραρχικής ονομασίας.

2. **(Scoping)** Η δεύτερη ιδιότητα τοποθετεί τα στοιχειά πληροφορίας μέσα σε πλαίσια που καλούνται *πεδία εφαρμογής (scopes)*. Έτσι, ένα scope ορίζει ένα σύνολο από πληροφορίες και είναι και το ίδιο από μόνο του ένα στοιχείο πληροφορίας και αναγνωρίζεται με ένα μοναδικό αναγνωριστικό. Σαν στοιχειά πληροφορίας, τα scopes μπορούν να τοποθετηθούν κάτω από άλλα scope, επιτρέποντας να δομούνται σύνθετα κατευθυνόμενα α-κύκλα γραφήματα από πληροφορίες που καλούνται *πληροφοριακές δομές (data structure)* ή αλλιώς γράφοι πληροφορίας. Οι γράφοι πληροφορίας μπορούν να χρησιμοποιηθούν από κατανεμημένες υπολογιστικές διεργασίες για κάθε σκοπό.

3. **(Service model)** Η Τρίτη ιδιότητα είναι το *μοντέλο εξυπηρέτησης (service model)* που λειτούργει σε κάθε πληροφοριακή δομή. Κατανεμημένες υπολογιστικές διεργασίες πραγματοποιούνται διαμέσω της ροής πληροφοριών μεταξύ οντοτήτων που χρησιμοποιούν το εκτεθειμένο εξυπηρετικό μοντέλο. Στον Blackadder προτείνεται το εξυπηρετικό μοντέλο publish/subscribe, που επίσης μπορεί να υποστηρίξει άλλα μοντέλα, όπως το pull ή το request/response.

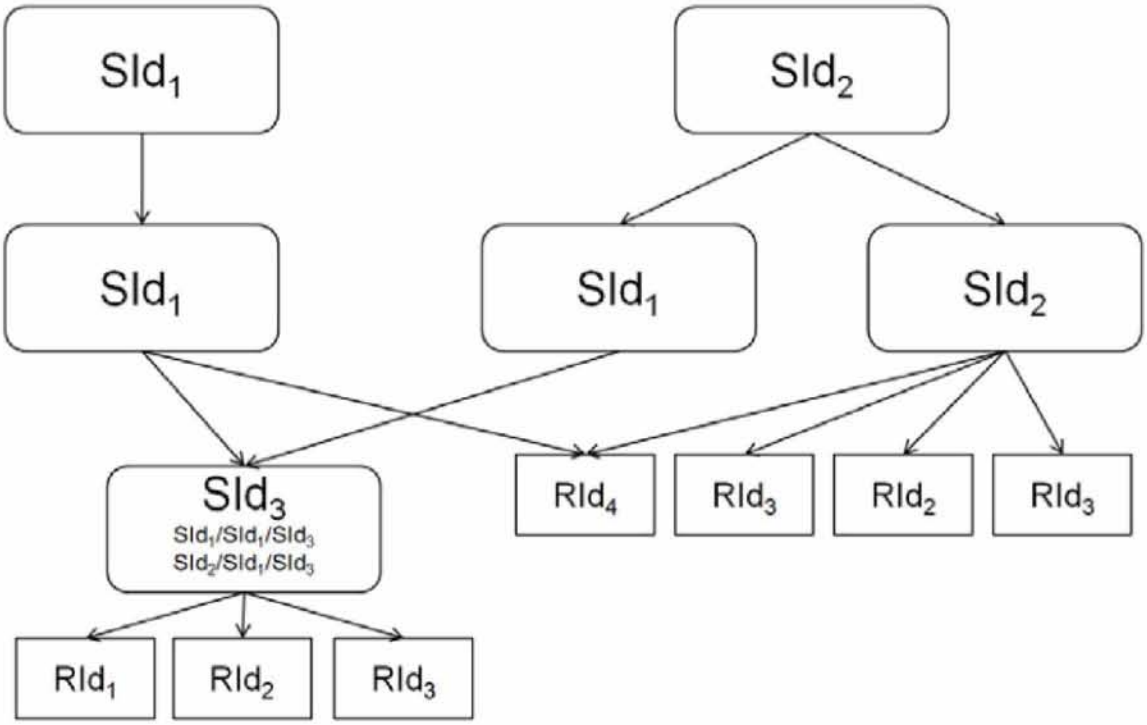
4. **(Rendezvous, topology management and formation, forwarding)** Η τέταρτη ιδιότητα είναι οι κύριες λειτουργίες της αρχιτεκτονικής. Αυτές οι

λειτουργίες αναλαμβάνουν την διάδοση της πληροφορίας διαμέσω των scopes μέσα στην πληροφοριακή δομή. Η πρώτη λειτουργία είναι η *rendezvous*, που ο ρόλος της είναι να ταιριάζει την διαθέσιμη πληροφορία σε αυτούς που ενδιαφέρονται για αυτήν. Τα αποτελέσματα αυτής της διαδικασίας είναι σαν μια μορφή τοποθεσιών που χρησιμοποιείται για την δέσμευση των θέσεων των πελατών που σχετίζονται με την συγκεκριμένη πληροφορία στο δίκτυο. Τα αποτελέσματα που παραδίδει η rendezvous λειτουργία χρησιμοποιούνται από την δεύτερη κύρια λειτουργία, *topology management and formation*, για να ορίσει την κατάλληλη σχέση παράδοσης για την μεταφορά της πληροφορίας. Η μεταφορά εκτελείται από την τρίτη βασική λειτουργία, *forwarding*. Η ακριβής φύση και η εφαρμογή αυτών των λειτουργιών καθορίζεται από τη στρατηγική διάδοσης του scopes για την οποία πραγματοποιούνται αυτές οι λειτουργίες.

5. **(dissemination strategy)** Η Πέμπτη ιδιότητα διευθύνει τις μεθόδους που χρησιμοποιούνται για την εφαρμογή των πιο πάνω κύριων λειτουργιών. Επίσης διευθύνει και ιδιαίτερα θέματα που αφορούν την διαχείριση των πληροφοριών σε κάποια τμήματα της πληροφοριακής δομής. Για τον λόγο αυτό ορίζονται στρατηγικές διάδοσης της πληροφοριακής δομής ή σε τμήματα της πληροφοριακής δομής. Μαζί με τα scoping σε κάποια υπό πεδία πληροφορίας, οι στρατηγικές αυτές ορίζουν υπό-χώρούς που εφαρμόζουν *functional scoping (λειτουργικά πεδία εφαρμογής).* Διαμέσω αυτών των διακριτών λειτουργιών μπορεί να βελτιστοποιηθεί η συγκεκριμένη υπολογιστική διαδικασία που χρησιμοποίει αυτό το είδος επικοινωνίας.

Στο Σχήμα 5 παρουσιάζεται μια πληροφοριακή δομή (data structure) και φαίνεται πως ένα scope τοποθετείται κάτω από ένα άλλο scope, καθώς επίσης και πως ένα στοιχειό πληροφορίας τοποθετείται κάτω από ένα scope. Στην υλοποίηση αυτή κάθε κόμβος του γραφήματος αναγνωρίζεται με το πλήρες μονοπάτι που αρχίζει από το scope που βρίσκεται στην ρίζα του γραφήματος. (Για παράδειγμα το δεξιότερο στοιχείο του γραφήματος αναγνωρίζεται ως /Sid2/SId2/RId3). Επίσης, όπως φαίνεται και στο σχήμα, ένα scope ή ένα στοιχείο πληροφορίας μπορεί να αναγνωριστεί από διάφορα μονοπάτια όταν δημοσιεύεται κάτω από περισσότερα από ένα scope ή όταν ένα από γονικά του scopes δημοσιεύεται κάτω από περισσότερα του ενός scope.

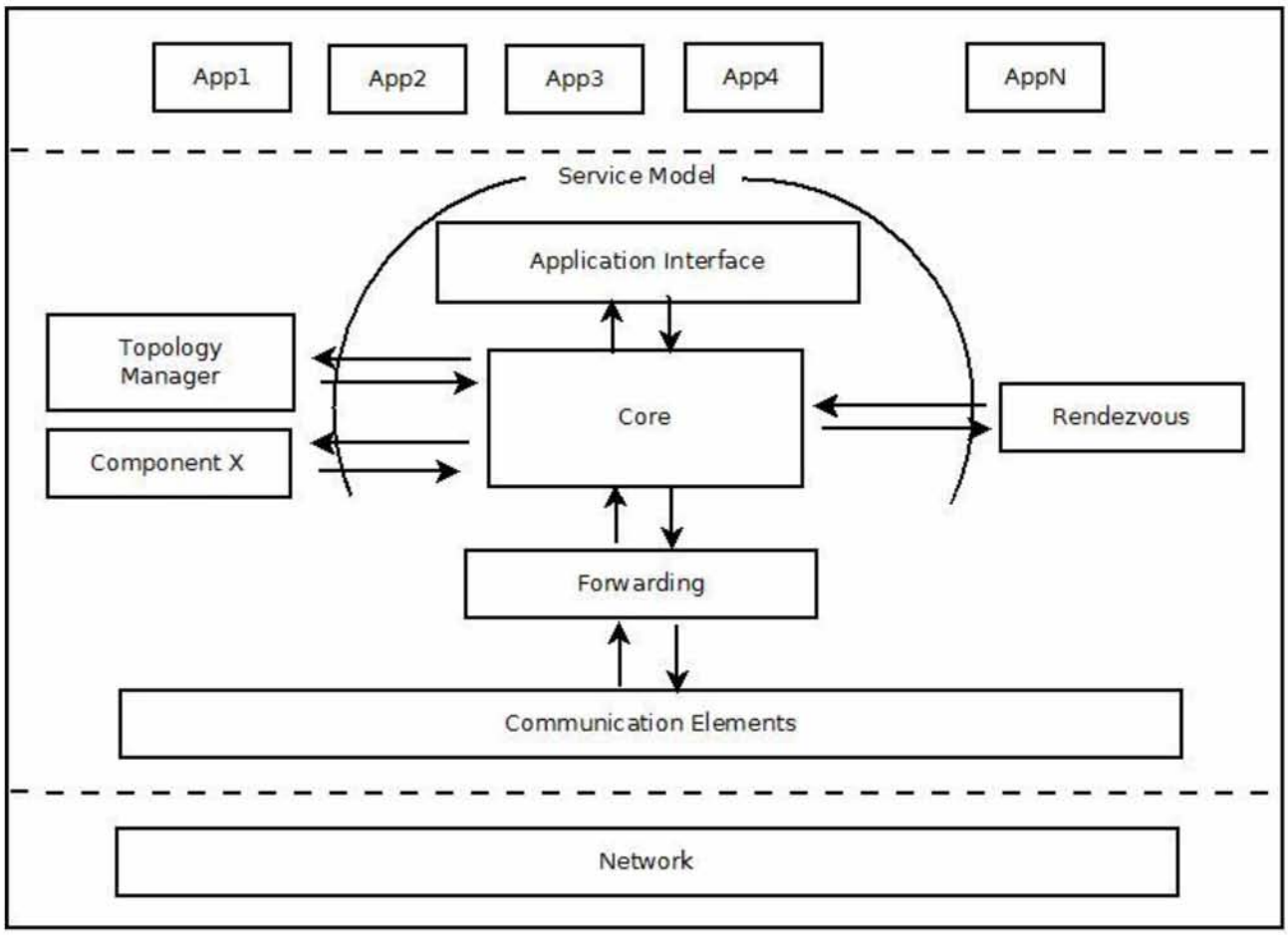
Ένας publisher μπορεί να δημιουργήσει μια πληροφοριακή δομή (information structure) για παράδειγμα σαν αυτή που φαίνεται στο Σχήμα 5, ή να προσθέσει και άλλα scopes ή να διαφημίσει στοιχεία πληροφορίας σε μια άλλη δομή πληροφοριών που δημιουργήθηκε από κάποιον άλλο publisher.



Σχήμα 5: *Μια απλή πληροφοριακή δομή σε μορφή γραφήματος.*

2.1.2.2. Σχεδίαση του κόμβου

Ο Blackadder βασίζεται στην αρχιτεκτονική του εργαλείου Click Modular Router και από άλλες συμπληρωματικές εφαρμογές. Το κύριο μέρος του λειτουργεί σαν λογισμικό ενός δρομολογητή κι τρέχει σε κάθε κόμβο του δικτύου. Κάθε Click element παρέχει και μια λειτουργία, που πιο συγκεκριμένα είναι εκτελεί μια επεξεργασία πακέτου. Επίσης οι βασικές λειτουργίες που αναφέραμε πιο πάνω, επίσης υλοποιούνται μέσα από elements του εργαλείου Click Router. Κάθε πακέτο που φθάνει στον δρομολογητή, ανάλογα με την φύση του, ακολουθεί μια διαδρομή διαμέσου των κατάλληλων elements. Στο Σχήμα 6 βλέπουμε πως είναι συνδεδεμένα τα elements μεταξύ τους και μπορούμε να διακρίνουμε πως μπορεί ένας δρομολογητής να επεξεργαστεί ένα πακέτο περνώντας διαμέσω των elements. Πιο κάτω αναλύουμε την λειτουργία αυτών των Elements που υλοποιούν τις βασικές λειτουργίες του Blackadder.



Σχήμα 6: *Η αρχιτεκτονική κόμβου του Blackadder.*

2.1.2.3. Βασικές λειτουργίες

Για να επιτευχθεί η όλη λειτουργία του Blackadder, φυσικά είναι αναγκαίο να συμβάλουν όλα τα Click elements. Όπως προαναφέραμε βασικές λειτουργίες της αρχιτεκτονικής είναι τρεις (3) και εκτελούνται μέσα από μεμονωμένα Click elements. Από την ονομασία που έχει κάθε element, καταλαβαίνουμε και ποια βασική λειτουργία εκτελεί. Αυτά είναι Elements Rendezvous, Element Forwarder και Element TopologyManager. Πιο κάτω αναλύουμε τις βασικές λειτουργιές.

1. **(Rendezvous)** Η λειτουργία Rendezvous λαμβάνει και επεξεργάζεται όλες τις αιτήσεις που δημοσιεύονται από τις εφαρμογές που εκτελούνται τοπικά ή από άλλους κόμβους. Τέτοιου είδους δημοσιεύσεις δημιουργήθηκαν διαμέσου της σημασιολογίας του εξαγόμενου εξυπηρετικού μοντέλου. Μετά την λήψη των αιτημάτων η λειτουργία Rendezvous ταιριάζει τους δυνητικούς στους publishers και subscribers των στοιχείων πληροφορίας, προκειμένου να διευκολύνει την ανταλλαγή πληροφοριών μεταξύ τους. Οι πράξεις της λειτουργίας Rendezvous ορίζονται μέσω της στρατηγικής διάδοσης που κρύβεται πίσω από την πληροφοριακή δομή. Οι πράξεις της λειτουργίας Rendezvous μπορεί να απαιτούν την επιβολή καθολικής (παγκόσμιας)

μοναδικότητας αναγνωριστικών. Επιπλέον ανάλογα με την στρατηγική διάδοσης μπορεί να ελαχιστοποιηθεί η λειτουργία της Rendezvous.

2. **(Topology management and formation - TM)** Η λειτουργία TM πραγματοποιεί την διαχείριση της συνολικής τοπολογίας παράδοσης και την διαμόρφωση συγκεκριμένων γραφημάτων παράδοσης για τις σχέσεις μεταξύ Publishers και Subscribers. Για αυτό, η TM ενημερώνει την τοπολογία συμφώνα με το κατά πόσο ένας κόμβος μπαίνει ή φεύγει από το δίκτυο και δημιουργεί την απαραίτητη πληροφορία προώθησης όταν της ζητηθεί.

3. **(Forwarding)** Η λειτουργία forwarding λαμβάνει δημοσιεύσεις και τις προωθεί στο δίκτυο ή στον τοπικό κόμβο. Για να το κάνει αυτό, κρατάει όλες τις απαραίτητες καταστάσεις σε ένα πινάκας προώθησης και για να επιτύχει την λειτουργία της μπορεί να συντονιστεί με τη λειτουργία Topology management and formation.

2.1.2.4 Εξυπηρετικό μοντέλο (Service model)

Όπως φαίνεται στο Σχήμα 6, οι εφαρμογές που ανήκουν στον ίδιο κόμβο αλληλεπιδρούν μεταξύ τους σαν εσωτερικά στοιχειά της όλης εφαρμογής ενώ στο δίκτυο αλληλεπιδρούν διάμεσο του εξυπηρετικού μοντέλου, που δείχνει σαν εσωτερική εφαρμογή. Το εξυπηρετικό μοντέλο επιτρέπει τον χειρισμό μια πληροφοριακής ροής με άπλες publish/subscribe σημασιολογικές ρουτίνες. Οι περισσότερες από ρουτίνες αυτές απευθύνονται στην λειτουργία rendezvous ώστε να συνθέτει τον πληροφοριακό γράφο και κατ' επέκταση να διευθύνει την επικοινωνία. Κάποιες απευθύνονται από κόμβους σε άλλους κόμβους, αλλά και πάλι διευθύνονται από την λειτουργία rendezvous. Στη συνεχεία παρουσιάζουμε αυτές τις πράξεις και περιγραφούμε τι ακριβώς λειτουργία εκτελούν.

Για την κατανόηση των ρουτινών, εξηγούμε τις παραμέτρους που λαμβάνουν ώστε να γίνεται κατανοητή η παρουσίαση τους. Το *id* είναι το τελευταίο αναγνωριστικό κομμάτι ενός μονοπατιού που απευθύνεται σε ένα στοιχείο πληροφορίας, το *prefix* είναι το υπάρχον μονοπάτι στον γράφο πληροφορίας και αποτελείται από τα scopes και το *s* είναι η στρατηγική διάδοσης που χρησιμοποιείται.

(Publishing & Unpublishing Scopes)

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

publishScope(string id, string prefix, strategy s)

Μέτα την δημοσίευση ενός scope, η rendezvous λειτουργία, ενημερώνει όλους τους subscribers που είναι εγγεγραμμένοι (subscribed) στο scope που είναι πατέρας του νέου scope που δημοσιεύτηκε.

Για να σταματήσει την δημοσίευση σε ένα scope μπορεί να το κάνει εκτελώντας το αίτημα:

unpublishScope(string id, string prefix, strategy s)

Μετά από αυτό το αίτημα, η rendezvous λειτουργία θα σταματήσει για τον publisher την δημοσίευση όλων των στοιχείων πληροφορίας που είναι παιδιά αυτού του scope. Αν το επιτύχει αυτό και δεν υπάρχει κάποιο άλλο scope κάτω από το scope που είναι στο αίτημα, τότε διαγράφεται. Εάν το scope ήταν δημοσιευμένο κάτω από πολλαπλά scopes, τότε διαγράφεται μόνο ένας συγκεκριμένος ο κλάδος του γραφήματος. Όλοι οι subscribers που είναι εγγεγραμμένοι σε scope(s) που είχαν πατέρα αυτό το scope θα ενημερωθούν.

(Publishing & Unpublishing Information Items)

Ένας publisher μπορεί να διαφημίζει στοιχειά πληροφορίας διάμεσου του αιτήματος:

publishInfo(string id, string prefix, strategy s)

Τα στοιχειά πληροφορίας τοποθετούνται κάτω από ένα ή περισσότερα scopes.

Επίσης μπορεί να σταματήσει την διαφήμιση ενός στοιχείου πληροφορίας εκτελώντας το αίτημα:

unpublishInfo(string id, string prefix, strategy s)

Σαν αποτέλεσμα αυτής της πράξης, ο publisher που εκτέλεσε το αίτημα αυτό, διαγράφεται από τον ρόλο του publisher για το συγκεκριμένο στοιχείο πληροφορίας στον γράφο πληροφορίας. Αν δεν υπάρχουν άλλοι publishers ή subscribers τότε το συγκεκριμένο στοιχειό πληροφορίας διαγράφεται εντελώς από τον γράφο πληροφορίας. Εάν παραμένουν publisher και subscribers για το συγκεκριμένο στοιχείο πληροφορίας, η λειτουργία rendezvous θα λάβει χώρο ώστε να βρεθούν ένας ή περισσότεροι publishers για να δημοσιεύουν δεδομένα για το συγκεκριμένο στοιχείο πληροφορίας. Εάν το στοιχείο πληροφορίας διαφημίζεται κάτω από πολλαπλά scope τότε το στοιχείο, διαγράφεται από το scope που δηλώνεται στο prefix του αιτήματος (θεωρώντας ότι δεν υπάρχουν publisher ή subscribers για το αναγνωριστικό αυτό).

(Subscribing to & Unsubscribing from Scopes)

Ένα subscription σε ένα scope γίνεται διάμεσο του αιτήματος:

subscribeScope(string id, string prefix, strategy s)

Κατ την παραλάβει του αιτήματος αυτού από την λειτουργία rendezvous, αν δεν υπάρχει το scope που υπάρχει στην αίτηση τότε δημιουργεί ένα νέο scope. Κάνοντας subscribe σε ένα scope, ένας subscriber δηλώνει τον ενδιαφέρον του αυτόματα και για όλα τα scope και όλα τα στοιχεία πληροφορίας που διαμένουν από κάτω του scope αυτού. Για αυτό τον λόγο, η λειτουργία rendezvous ελέγχει όλα τα scope και τα στοιχεία πληροφορίας που είναι δημοσιευμένα κάτω από το scope αυτό και δρα ως ακολούθως: για κάθε scope, θα στείλει μια ενημερώσει

για την υπάρξει του. Για κάθε στοιχείο πληροφορίας, ένας ή περισσότεροι publisher που προηγμένως διαφήμιζαν θα ενημερωθούν. Ένα δεν υπάρχει κοπής publisher ή subscriber για ένα στοιχείο πληροφορίας ή ένα scope κάτω από αυτό το scope, τότε το scope διαγράφεται από τον γράφο πληροφορίας.

(Subscribing to & Unsubscribing from Information Items)

Ένα subscription σε ένα στοιχείο πληροφορίας γίνεται με το αίτημα:

subscribeInfo(string id, string prefix, strategy s)

Εάν ένας publisher διαφήμισε προηγουμένως το στοιχείο πληροφορίας αυτό, τότε η λειτουργία rendezvous ταιριάζει τον publisher με αυτόν με τον subscriber που έστειλε το αίτημα.

(Publishing Data)

Ένας publisher στέλνει δεδομένα για ένα συγκεκριμένο στοιχείο πληροφορίας εκτελώντας το αίτημα:

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

Το id, είναι το RID του στοιχείου πληροφορίας αρχίζοντας από την ρίζα του γράφου πληροφορίας. Ένας publisher στέλνει τέτοια αιτήματα εάν πρώτα λάβει χώρο η λειτουργία rendezvous και ξέρει που θα προωθηθούν τα δεδομένα.

(Asynchronous Upcalls)

Στις περισσότερες περιπτώσεις τα αιτήματα που στέλνονται από τις εφαρμογές στο δίκτυο έχουν σαν αποτέλεσμα να καταλήξουν στην rendezvous ή στην TM λειτουργία. Αυτά τα αιτήματα μπορεί να έχουν σαν τελικό σκοπό την ενημερώσει άλλων εφαρμογών του δικτύου. Αυτά τα αιτήματα παρέχουν την ασύγχρονη επικοινωνία. Αυτά είναι:

- *New scope notification.* Στέλνεται σε ένα subscriber όταν ένα νέο scope έχει δημιουργηθεί. Στην ενημέρωση περιλαμβάνεται το αναγνωριστικό του scope.
- *Deleted scope notification.* Όταν ένα scope σταματήσει να δημοσιεύεται (unpublished) και διαγράφεται από τον γράφο πληροφορίας, οι subscribers στα scopes που είναι γονικά αυτού, ενημερώνονται για την διαγραφή του. Στην ενημέρωση περιλαμβάνεται το αναγνωριστικό του scope.
- *Starts publish notification.* Στέλνεται σε μια εφαρμογή, που προηγουμένως είχε διαφημίσει ένα στοιχείο πληροφορίας. Ώστε να αρχίσει την αποστολή δεδομένων για το συγκεκριμένο στοιχείο πληροφορίας. Περιλαμβάνει το αναγνωριστικό της πληροφορίας που πρόκειται να δημοσιευτεί.
- *Stops publish notification.* Στέλνεται σε μια εφαρμογή όταν δεν υπάρχουν διαθέσιμοι subscribers για κάποιο συγκεκριμένο στοιχείο πληροφορίας, όπου για αυτό το στοιχείο πληροφορίας προηγουμένως έχει σταλεί ένα μήνυμα

start publish.

- *Published data notification.* Στέλνεται σε μια εφαρμογή (subscriber) όταν έχουν φθάσει δεδομένα για μια συγκεκριμένη δημοσίευση. Η ενημέρωση περιλαμβάνει το ID του στοιχείου πληροφορίας για τα δεδομένα αυτά.

2.1.2.5. Στρατηγικές διάδοσης (Dissemination strategies)

Κάθε scope ανατίθεται με μια στρατηγική διάδοσης. Υπάρχουν τρεις στρατηγικές διάδοσης: στρατηγική domain-local, στρατηγική node-local και η στρατηγική link-local.

1. **(domain-local)** Στην στρατηγική domain-local όλες οι λειτουργίες είναι σε πλήρη λειτουργικότητα. Η λειτουργία rendezvous δρα σε μόνο ένα ή περισσότερους κόμβους του δικτύου. Επίσης η λειτουργία Topology management and formation δρα μόνο σε ένα ή περισσότερους κόμβους, επικεντρώνεται στην παρακολούθηση του δικτύου, δημιουργώντας δέντρα πολύ-εκπομπής για σύνολα από publishers σε σύνολα προς subscribers, χρησιμοποιώντας αλγορίθμους εύρεσης συντομότερου μονοπατιού. Τέτοια δέντρα πολύ-εκπομπής σχηματίζονται από υπολογισμούς που βασίζονται στα Bloom Filters, τα οποία χρησιμοποιούν τα μοναδικά αναγνωριστικά κάθε ζεύξης. Βασιζόμενο σε αυτά τα μοναδικά αναγνωριστικά των ζεύξεων, η λειτουργία Forwarder προωθεί αποτελεσματικά κάθε πακέτο στον προορισμό του εκτελώντας την δυαδική πράξη AND στο Bloom Filter που φέρει το πακέτο, το αποτέλεσμα είναι ένα δέντρο πολύ-εκπομπής.

2. **(node-local)** Η στρατηγική node-local επιτρέπει την δημιουργία του γραφήματος πληροφορίας εντός του κόμβου. Η λειτουργία rendezvous διατηρεί τον γράφο πληροφορίας έτσι ώστε να είναι ορατό στις εφαρμογές και στα Click Elements που τρέχουν μόνο στον κόμβο. Η λειτουργία Topology management and formation είναι απλή και περιορισμένη, γιατί πρέπει να βρει τους ενδιαφερόμενους publishers και subscribers τοπικά. Τέλος η λειτουργία Forwarding προωθεί επίσης μόνο τοπικά σε εφαρμογές και Click elements.

3. **(link-local)** Η στρατηγική link-local επιτρέπει σε ένα κόμβο του δικτύου να διαδίδει πληροφορίες μόνο στους άμεσα συνδεδεμένους μέσω φυσικής ζεύξης γείτονες του. Η λειτουργία της rendezvous είναι περιορισμένη γιατί δεν έχει να διατηρήσει γράφο πληροφορίας. Αντί αυτού οι subscribers εγγράφονται έμμεσα σε συγκεκριμένα στοιχεία πληροφορίας. Ένας publisher μπορεί να δημοσιεύσει στοιχεία πληροφορίας σε συγκεκριμένες ζεύξεις ή σε όλες. Αν ένας subscriber υπάρχει στην ζεύξη, η πληροφορία προωθείται στην ενδιαφερομένη εφαρμογή. Η λειτουργία του Topology management and formation είναι και πάλι περιορισμένη, επειδή το μόνο που έχει να κάνει είναι να βρει την κατάλληλη ζεύξη.

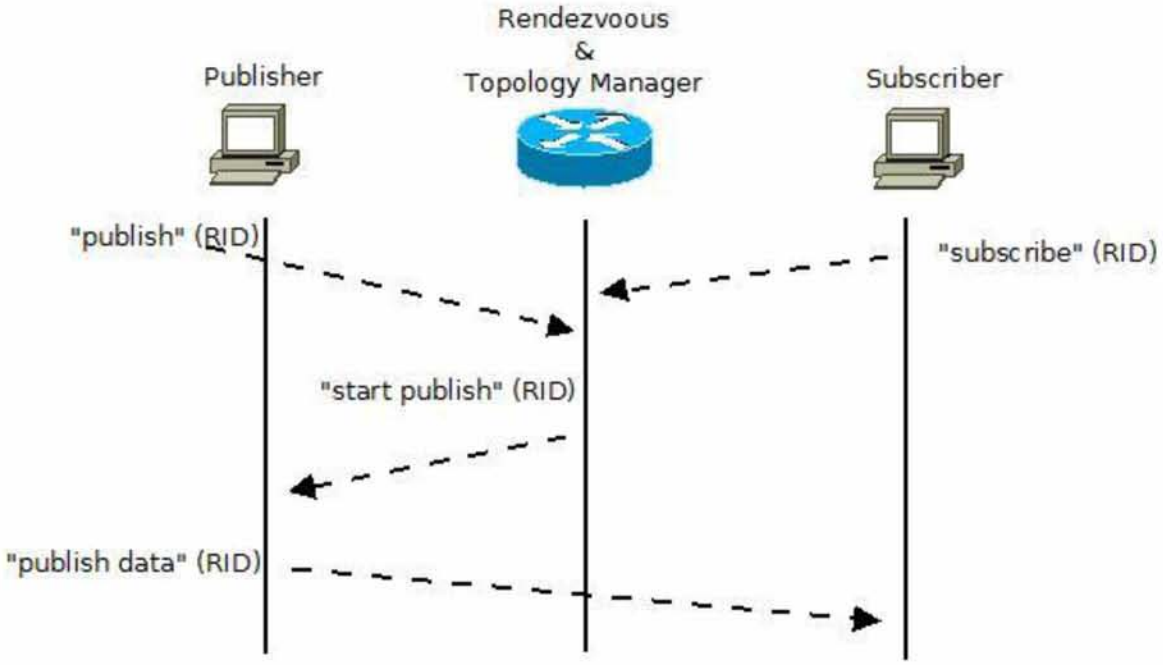
2.1.2.6. Συνοπτική περιγραφή λειτουργίας publish/subscribe

Όταν μια εφαρμογή έχει έτοιμα δεδομένα προς δημοσίευση, στέλνει ενημέρωση στην λειτουργία rendezvous του δικτύου του ότι θα είναι publisher στα δεδομένα αυτά, τα οποία σηματοδοτεί με ένα συγκεκριμένο μοναδικό αναγνωριστικό RID (Rendezvous ID). Η ενημέρωση γίνεται μέσω των ρουτινών publishScope() και publishInfo(), ανάλογα ώστε να ενημερώσει το γράφημα πληροφορίας ποιες πληροφορίας θέλει να δημοσιεύει. Συμπληρωματικά, όταν μια εφαρμογή θέλει να λαμβάνει κάποια συγκεκριμένα δεδομένα, επίσης ενημερώνει την λειτουργία Rendezvous του δικτύου του ότι θα είναι subscriber (συνδρομητής) στα δεδομένα αυτά, τα οποία σηματοδοτεί με ένα συγκεκριμένο μοναδικό αναγνωριστικό RID (Rendezvous ID). Η ενημερώσει γίνεται μέσω τον ρουτινών subscribeScope() ή subscribeInfo(). Η εφαρμογή μπορεί να καλέσει subscribeScope() αν θέλει όλες τις πληροφορίες που διαμένουν κάτω από κάποιο scope. Με την κλήση της ρουτίνας subscribeInfo() η εφαρμογή δηλώνει την επιθυμία να παραλάβει μια συγκεκριμένη πληροφορία, δηλαδή επιθυμεί ένα στοιχείο πληροφορίας το οποίο αντιστοιχεί σε συγκεκριμένα δεδομένα.

Όπως προαναφέραμε η λειτουργία Rendezvous είναι υπεύθυνη για να λαμβάνει τις δημοσιεύσεις (publication) και της συνδρομές σε δημοσιεύσεις (subscription) από τους διάφορες εφαρμογές του δικτύου της. Στην συνέχεια κάνει την σύγκριση των RIDs που έφτασαν από publishers μεταξύ των RIDs που έφτασαν από subscribers. Στην περίπτωση που βρεθεί κάποια αντιστοιχία των RIDs τότε η λειτουργία Rendezvous ενημερώνει την λειτουργία Topology management and formation ώστε να φτιάξει ένα μονοπάτι από ένα[1] publisher προς τον/τους subscriber/s που βρέθηκαν να είναι εγγεγραμμένοι στην δημοσίευση με το ίδιο RID. Στην συνέχεια ο publisher ενημερώνεται από την Rendezvous ώστε να αρχίσει να στέλνει δεδομένα μέσω του αιτήματος "start publish". Ο publisher όταν παραλάβει την ενημέρωση "start publish" αρχίζει να αποστέλλει τα δεδομένα που αντιστοιχούν στο RID που έκανε publish. Στην περίπτωση που μια εφαρμογή είναι publisher και κάνει δημοσιεύσει κάποια πληροφορία που δεν βρεθεί άμεσα κάποιος subscriber, τότε στην εφαρμογή publisher αποστέλλεται από την λειτουργία Rendezvous το μήνυμα "stop publish". Έτσι ενημερώνεται ότι δεν έχει άμεσα κάποιο subscriber εγγραμμένο στην δημοσίευση με το συγκεκριμένο RID και παραμένει σε αναμονή μέχρι να βρεθεί κάποιος subscriber. Καθώς επίσης όταν βρεθεί κάποιος subscriber στο δίκτυο, ο publisher θα παραλάβει εκείνη την στιγμή το μήνυμα start_publish και θα εκτελέσει την διαδικασία αποστολής δεδομένων που περιγράψαμε πιο πριν.

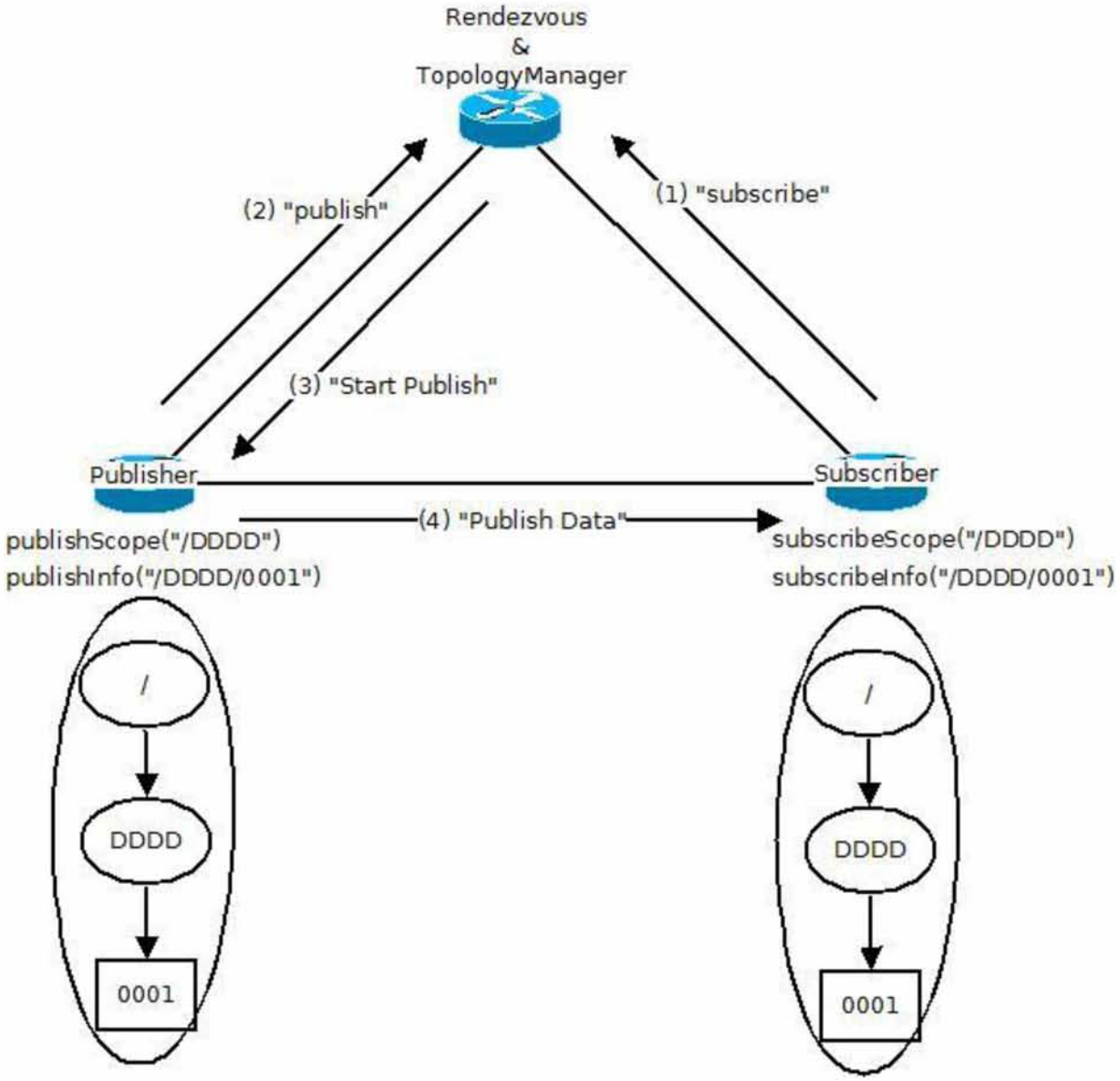
Στο Σχήμα 7 παρουσιάζονται οι βασικές ενημερώσεις ώστε να γίνει η παράδοση δεδομένων από ένα publisher προς ένα subscriber.

[1] Θα επιλεχτεί ένας, πιθανόν να υπάρχουν περισσότεροι που δημοσιεύουν τα ίδια δεδομένα.



Σχήμα 7: *Ανταλλαγή ενημερώσεων που έχει ως αποτέλεσμα την παράδοση δεδομένων από τον publisher προς τον subscriber.*

Στο Σχήμα 8 μέσω ενός παραδείγματος περιγράφουμε την όλη λειτουργία. Ένας publisher κάνει publish ένα πληροφοριακό στοιχειό με όνομα "/DDDD/0001" κάτω από ένα scope με όνομα "/DDDD". Παράλληλα ένας subscriber κάνει subscribe ένα πληροφοριακό στοιχειό με όνομα "/DDDD/0001" κάτω από το scope "/DDDD". Ο Rendezvous επιστρέφει στον publisher το μονοπάτι προς τον subscriber και έτσι ο publisher αρχίζει την αποστολή των δεδομένων.



Σχήμα 8: *Σχηματικό παράδειγμα λειτουργίας του Blackadder.*

2.1.3. Silicon Bloom Filters και zfilters

Για να υποστηριχτεί αποτελεσματικά η τεχνική του publish/subscribe σε δίκτυα μεγάλης κλίμακας, απαιτείται αποδοτική επικοινωνία πολύ-εκπομπής (multicast), ώστε να προωθείται η πληροφορία με το λιγότερο υπολογιστικό φόρτο στους διάφορους πόρους του δικτύου. Με μια αποδοτική προωθητική μέθοδο μπορούμε σε κάθε κόμβο να έχουμε μικρούς πίνακες δρομολόγησης και απλές πράξεις στις αποφάσεις δρομολόγησης.

Ένας τέτοιου είδους μηχανισμός βασίζεται στην μοναδική αναγνώριση των συνδέσεων μεταξύ των κόμβων και χρησιμοποιώντας τα Silicon Bloom Filters ώστε να κωδικοποιεί και να ενσωματώνει την πληροφορία δρομολόγησης μέσα στην κεφαλίδα κάθε πακέτου. Οι αποφάσεις δρομολόγησης είναι απλές και οι πίνακες δρομολόγησης είναι μικροί, οπότε μπορούμε να έχουμε πιο γρήγορους, πιο μικρούς και πιο αποδοτικούς δρομολογητές.

Η πιο πάνω μέθοδος λαμβάνει το πλεονέκτημα της, αναστρέφοντας το σκεπτικό των Bloom Filters. Αντί να κρατάει δηλαδή τα Bloom Filters μέσα στους κόμβους του δικτύου και να ελέγχει εάν κάποιο εισερχόμενο πακέτο περιλαμβάνεται σε κάποιο σύνολο εξερχόμενων συνδέσεων που ορίζονται από Bloom Filters, τοποθετεί τα Bloom Filters μέσα στο πακέτο και επιτρέπει στους κόμβους διαμέσου του μονοπατιού να αποφασίσουν σε ποιες εξωτερικές συνδέσεις θα πρέπει να προωθηθεί το πακέτο.

Ο προωθητικός μηχανισμός δεν χρησιμοποιεί την μέθοδο διευθυνσιοδότησης άκρου προς άκρο (end to end addresses) και αντί να ονομάζονται οι κόμβοι, ονομάζονται όλες οι συνδέσεις μεταξύ των κόμβων. Για να προωθούνται τα πακέτα διάμεσω του δικτύου χρησιμοποιεί την προσέγγιση των υβριδικών Bloom-filter-based. Η λειτουργία του Topology δημιουργεί αναγνωριστικά προώθησης (forwarding identifier) κωδικοποιώντας τα αναγνωριστικά των συνδέσμων κατά κάποιο τρόπο δρομολόγησης.

Για κάθε σύνδεσμο που ενώνει δύο κόμβους, εφαρμόζονται δυο αναγνωριστικά. Τα αναγνωριστικά αυτά λέγονται Link IDs, ένα για κάθε κατεύθυνση. Για παράδειγμα ένα από τον κόμβο Α προς τον κόμβο Β και ένα από τον κόμβο Β προς τον κόμβο Α.

Βασικά, ένα Link ID αποτελείται από ένα δυαδικό αριθμό αποτελούμενο από m-bits με k-bits από αυτά τα είναι ίσα με ένα '1'. Επιλέγοντας k<<m και m σχετικά μεγάλο μπορούμε να κάνουμε τα Link ID μοναδικά γιατί μπορούν να γίνουν πάρα πολλοί συνδυασμοί τέτοιων Link IDs. (για παράδειγμα, με m = 248, k = 5, αριθμός των Link IDs ≈ m!/(m-k)! ≈ 9*10^11).

Η λειτουργία Topology δημιουργεί ένα γράφημα του δικτύου και χρησιμοποιεί τα Link IDs για την συνδεσιμότητα. Όταν πάρει ένα αίτημα για να καθορίσει το δέντρο για μια δημοσίευση, πρώτα δημιουργεί ένα εννοιολογικό δέντρο παράδοσης χρησιμοποιώντας το γράφημα του δικτύου και τις τοποθεσίες των publishers και των subscribers. Από τη στιγμή που έχει μια τέτοια εσωτερική αναπαράσταση του δέντρου, ξέρει από ποια Links θα περάσει το πακέτο. Βασικά κωδικοποιούν όλα τα Link IDs του δέντρου μέσα σε Bloom Filters και τοποθετούνται στην κεφαλίδα του πακέτου. Έτσι όταν όλα τα Link IDs τοποθετηθούν στο φίλτρο, γίνεται μια χαρτογράφηση σε Bloom Filters του

αναγνωριστικού των δεδομένων, από τον κόμβο που δρα ως η πηγή στο δέντρο παράδοσης. Στην αναπαράσταση των δέντρων παράδοσης σε κεφαλίδες των πακέτων έχει σημαντικό ρόλο η πηγή των δεδομένων, αφού από ίδιες πήγες πολύ πιθανόν να χρησιμοποιηθούν διαφορετικά Bloom Filters για το ίδιο σύνολο των subscribers. Για την διάκριση των Bloom Filters της κεφαλίδας του πακέτου από τα άλλα Bloom Filters, αναφέρονται τα Bloom Filters πακέτου σαν zFilters.

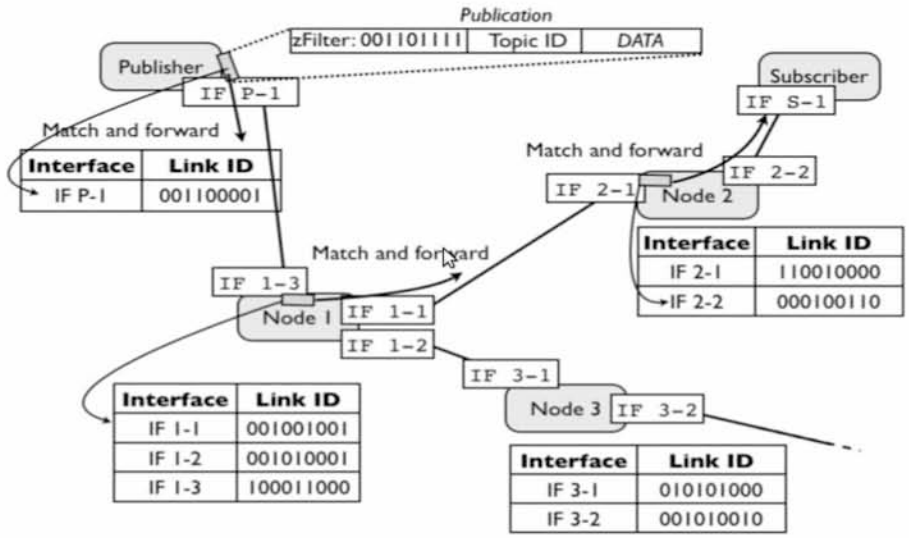
Ο πυρήνας του μηχανισμού προώθησης, που είναι η απόφαση προώθησης, βασίζεται στο δυαδικό AND και σε πράξεις σύγκρισης. Και οι δυο αυτές πράξεις είναι πολύ απλές να υλοποιηθούν σε hardware. Σε κάθε κόμβο ο μηχανισμός δρα ως ακολούθως: για κάθε σύνδεσμο, το εξερχόμενο Link ID γίνεται AND με το zFilter του πακέτου. Εάν το αποτέλεσμα είναι ίσο με το Link ID, τότε θεωρείται ότι το Link ID τοποθετήθηκε στο zFilter και έτσι το πακέτο θα πρέπει να προωθηθεί προς αυτόν τον σύνδεσμο. Ο βασικός αλγόριθμος που χρησιμοποιείται παρουσιάζεται στην πιο κάτω εικόνα.

```
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
```

***Αλγόριθμος 1**: Αλγόριθμος προώθησης που εκτελεί η λειτουργία Forwarding σε κάθε πακέτο που φθάνει στον κόμβο, ώστε να προωθηθεί στην κατάλληλη σύνδεση.*

Ένα στιγμιότυπο δρομολόγησης ενός πακέτου που βασίζει την δρομολόγηση του σε zFilter που διαθέτει, φαίνεται στο Σχήμα 9. Το πακέτο ξεκινώντας από τον publisher, με βάση το zFilter που φαίνεται και τον αλγόριθμο που παρουσιάζεται πιο πάνω μπορούμε να διακρίνουμε πιο μονοπάτι θα ακολουθήσει.



Σχήμα 9: *Προώθηση ενός πακέτου με βάση το zFilter που διαθέτει.*

2.2. Κρυφή Αποθήκευση (Caching)

Στο Διαδίκτυο, ένα μεγάλο μέρος του περιεχομένου μεταφέρεται επανειλημμένα. Τις περισσότερες φορές το περιεχόμενο αναμεταδίδεται από την πηγή για να εξυπηρετήσει διάφορες αιτήσεις που έρχονται από το δίκτυο. Το Caching είναι μια από τις πολλές τεχνικές που στοχεύει στην μείωση του φόρτου γενικά στο διαδίκτυο και του φόρτου στους κόμβους που διαθέτουν την πληροφορία. Τρία χαρακτηριστικά του Web caching το καθιστούν ελκυστικό για όλους τους συμμετέχοντες στο διαδίκτυο. Σε αυτούς συμπεριλαμβάνονται οι χρηστές, οι διαχειριστές του δικτύου και οι δημιουργοί του περιεχομένου.

Αυτά είναι:

1. Μειώνει το συνολικό bandwidth του δικτύου.
2. Μειώνει την καθυστέρηση που αντιλαμβάνεται ο χρήστης.
3. Μειώνει το φόρτο στους διακομιστές προέλευσης της πληροφορίας.

2.2.1. Στρατηγικές αντικαταστάσεις

Το κεντρικό θέμα στο caching είναι η στρατηγική αντικατάστασης. Στρατηγική αντικατάσταση της cache είναι η διαδικασία που λαμβάνει χώρα όταν η μνήμη γεμίσει και κάποιο παλιό αντικείμενο πρέπει να αφαιρεθεί ώστε να κάνει χώρο για εισαγωγή νέου αντικειμένου. Πιο κάτω αναφέρουμε τις βασικές στρατηγικές αντικατάστασης. Βλέπουμε τα ωφελήματα που προσφέρει η κάθε μια.

Έχουμε αναλύσει τις ακόλουθες κατηγορίες στρατηγικών αντικατάστασης, στρατηγικές που βασίζονται στο πιο πρόσφατο, στρατηγικές που βασίζονται στην συχνότητα και στρατηγική πρώτο μπαίνει - πρώτο βγαίνει (FIFO).

2.2.1.1. Στρατηγικές που βασίζονται στο πιο πρόσφατο (Recent-Based Strategies)

Οι στρατηγικές αυτές χρησιμοποιούν την πιο πρόσφατη αναφορά ως κύριο παράγοντα. Οι περισσότερες από αυτές είναι περισσότερο ή λιγότερο επεκτάσιμες της γνωστής στρατηγικής LRU (Last Recently Used). Η στρατηγική LRU εφαρμόζεται με επιτυχία σε πολλούς διαφορετικούς τομείς των υπολογιστών. Η LRU βασίζεται στην τοπικότητα αναφοράς που φαίνεται σε αιτήματα. Η τοπικότητα της αναφοράς χαρακτηρίζει την ικανότητα να προβλέπει τις μελλοντικές προσβάσεις σε κάποια αντικείμενα, βασιζόμενη σε προσβάσεις από το παρελθόν.

Υπάρχουν δύο κύριοι τύποι της τοπικότητας: χρονικές και χωρικές. Η χρονική τοπικότητα αναφέρεται σε επαναλαμβανόμενες προσβάσεις στο ίδιο αντικείμενο σε σύντομα χρονικά διαστήματα. Αυτό σημαίνει ότι εφόσον έχει τώρα πρόσφατές προσβάσεις το ίδιο αντικείμενο, είναι πιθανό να προσπελαστεί ξανά στο μέλλον. Η χωρική τοπικότητα αναφέρεται σε πρότυπα πρόσβασης. Όπου προσβάσεις σε κάποιο αντικείμενο συνεπάγεται προσβάσεις σε κάποια άλλα αντικείμενα. Συνεπάγεται επίσης ότι οι αναφορές σε κάποια αντικείμενα μπορεί να γίνουν μέσω πρόβλεψης των μελλοντικών αναφορών σε άλλα αντικείμενα.

2.2.1.2 Στρατηγικές που βασίζονται στην συχνότητα (Frequent-Based Strategies)

Οι στρατηγικές αυτές χρησιμοποιούν την συχνότητα ως κύριο παράγοντα. Οι στρατηγικές που βασίζονται στην συχνότητα είναι περισσότερο ή λιγότερο επεκτάσιμες της γνωστής στρατηγικής LFU (Last Frequency Used). Συγκεκριμένα βασίζονται στο γεγονός ότι διαφορετικά αντικείμενα έχουν διαφορετικές τιμές δημοσιότητας και αυτές οι τιμές δημοσιότητας αντιστοιχούν σε διαφορετικές τιμές συχνότητας. Οι στρατηγικές που βασίζονται στην συχνότητα, παρακολουθούν αυτές τις τιμές και τις χρησιμοποιούν μελλοντικές σε αποφάσεις.

Η LFU (και οι επεκτάσεις της) μπορεί να εφαρμοστούν σε δύο διαφορετικές μορφές: Τέλεια LFU (Perfect LFU) και LFU σε μνήμες (In-Cache LFU). Τέλεια LFU μετράει όλα τα αιτήματα για ένα αντικείμενο. Οι αιτήσεις εμμένουν και μετά τις αντικαταστάσεις των αντικειμένων. Από τη μία πλευρά, αυτό εξασφαλίζει ότι οι μετρήσεις αιτημάτων αντιπροσωπεύουν όλες τις αιτήσεις από το παρελθόν. Από την άλλη πλευρά, οι στατιστικές αυτές πρέπει να τηρούνται για όλα τα αντικείμενα που είδαν στο παρελθόν (αυτό επιφέρει επιβάρυνση χώρου). Στις LFU σε μνήμες, οι μετρήσεις καθορίζονται μόνο για τα αντικείμενα που είναι αποθηκευμένα. Αν και αυτό δεν αντιπροσωπεύει το σύνολο των αιτήσεων του παρελθόντος, εξασφαλίζει μια απλούστερη διαχείριση (επιφέρει λιγότερη επιβάρυνση χώρου).

2.2.1.3 Στρατηγική πρώτο μπαίνει - πρώτο βγαίνει (First In - First Out)

Πρόκειται για την πιο απλή στρατηγική αντικατάστασης, που βρίσκει εφαρμογή σε πολλές περιπτώσεις στον τομέα των υπολογιστών και ειδικά σε ουρές αναμονής. Η στρατηγική FIFO λέει απλά ότι σε περίπτωση που γεμίσει η μνήμη και έρθει ένα νέο αντικείμενο όπου πρέπει να αποθηκευτεί, τότε θα διαγράφει από την μνήμη αυτό που αποθηκεύτηκε αρχικά πρώτο.

2.3. Click Modular Router

Ο Click Modular Router είναι μια αρχιτεκτονική για την δημιουργία ευέλικτων και ρυθμιζόμενων δρομολογητών. Ο Click Router αποτελείται από οντότητες επεξεργασίας πακέτων που λέγονται Elements (στοιχεία). Μεμονωμένα elements υλοποιούν μια απλή λειτουργία δρομολογητή όπως για παράδειγμα διαχωρισμό πακέτων, queueing, scheduling, αλληλεπίδραση με συσκευές δικτύου και διάφορα άλλα.

Η διαμόρφωση του δρομολογητή είναι ένας κατευθυνόμενος γράφος με elements στις κορυφές του. Τα πακέτα περνούν από τις ακμές του γράφου. Οι ρυθμίσεις γράφονται σε δηλωτική (declarative) γλώσσα που καθορίζει ο χρήστης. Η γλώσσα αυτή είναι αναγνώσιμη τόσο από τον άνθρωπο όσο και από το εργαλείο του Click Router.

Βάσει της αρχιτεκτονικής και της γλώσσας του, οι Click Router ρυθμίσεις είναι αρθρωτές (Modular) και εύκολα επεκτάσιμες. Κάθε element είναι ένα λογισμικό που αναπαριστά μια οντότητα επεξεργασίας του δρομολογητή. Γράφοντας το δικό μας λογισμικό μπορούμε να φτιάξουμε δικά μας elements ή να τροποποιήσουμε τα ήδη υπάρχοντα. Για την δημιουργία νέου element πρέπει να γραφτεί μια κλάση σε κώδικα C++ και να ενσωματωθεί στο εργαλείο Click Router.

Τα elements έχουν πέντε σημαντικές ιδιότητες: element class, ports, configuration strings, method interfaces, and handlers.

- **(Element class)** Καθορίζει την διάταξη και την συμπεριφορά των δεδομένων στο Element.

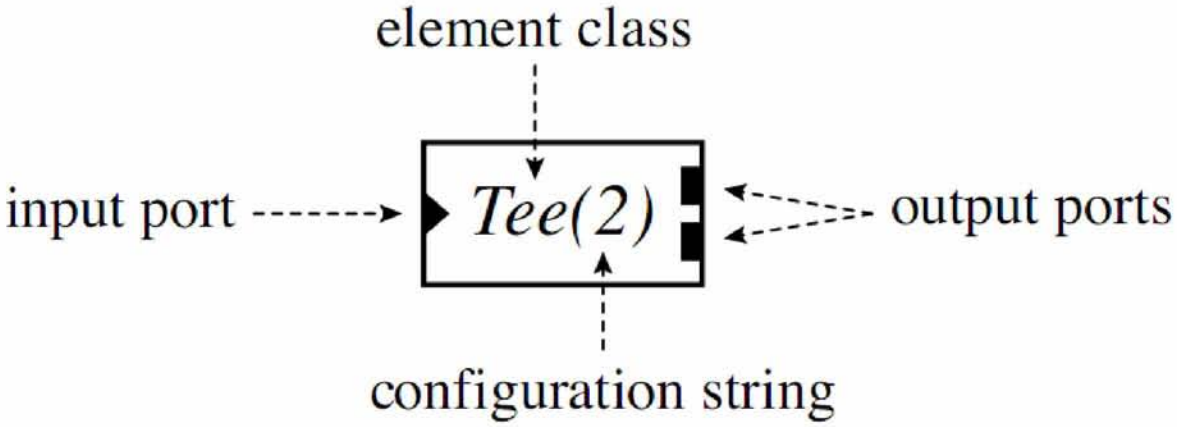
- **(Ports)** Κάθε Element έχει κάποιο αριθμό εισερχόμενων και εξερχόμενων θυρών.

- **(Configurations Strings)** Προαιρετικά αλφαριθμητικά διαμόρφωσης που παρέχουν επιπλέον ορίσματα στο Element κατά την αρχικοποίηση του δρομολογητή.

- **(Method Interface)** Κάθε Element έχει εξάγει μεθόδους που μπορεί να έχει πρόσβαση σε άλλα Elements.

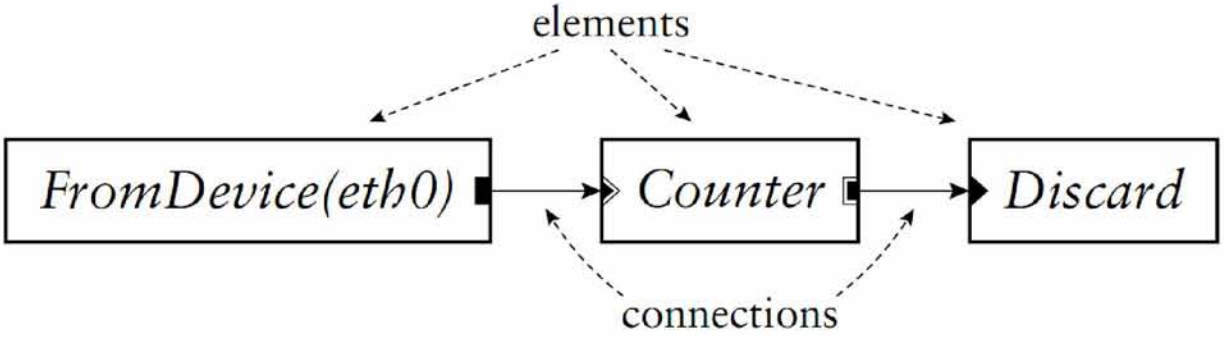
- **(Handlers)** Είναι μέθοδοι που μπορούν να εξάγουν δεδομένα στον χρήστη και όχι αλλά Elements.

Στο Σχήμα 10 διακρίνονται οι σημαντικές ιδιότητες πάνω σε ένα Click element.



***Σχήμα 10**: Σχηματικά, απεικονίζονται οι ιδιότητες των Click elements.*

Στο Σχήμα 11 φαίνεται μια συνδεσμολογία Click elements εκτελεί μια λειτουργία.



***Σχήμα 11:** Μια απλή συνδεσμολογία από elements του Click Router.*

3. Caching στον Blackadder

Σε αυτό το κεφάλαιο παρουσιάζουμε αναλυτικά την σχεδίαση του μηχανισμού αποθήκευσης πακέτων καθώς και την ενσωμάτωση του στον Blackadder. Αρχικά παρουσιάζουμε κάποιες νέες συμβάσεις και επεκτάσεις που χρειάστηκαν να γίνουν στον Blackadder ώστε να επιτευχθεί η ορθή του λειτουργία. Για παράδειγμα συμβάσεις κατά τις οποίες μια εφαρμογή publisher γίνεται και subscriber σε κάποια δεδομένα καθώς επίσης και το αντίστοιχο. Δηλαδή όπου μια εφαρμογή subscriber γίνεται και publisher σε κάποια δεδομένα. Σκοπός είναι να δημιουργηθεί μονοπάτι από τον subscriber στον publisher, όπου θα αποστέλλεται δια μέσω αυτού αίτημα δεδομένων. Στην συνέχεια παρουσιάζουμε ένα αλγόριθμο δημιουργίας ανάποδου μονοπατιού από τον κόμβο που βρεθούν τα δεδομένα μέχρι τον subscriber, που είναι ο αιτών. Καθώς και την ενσωμάτωση του μηχανισμού αποθήκευσης μέσα στα Click element του Blackadder. Επίσης παρουσιάζουμε την εσωτερική βασική σχεδίαση της λειτουργίας του μηχανισμού και τις απαραίτητες λειτουργίες, όπως εισαγωγής, αναζήτησης, ταξινόμησης και διαγραφής πακέτου. Στο τέλος του κεφαλαίου δείχνουμε συγκεκριμένα σε ποια elements υλοποιήσαμε τις λειτουργιές που προαναφέρουμε στην σχεδίαση του μηχανισμού. Καθώς επίσης παρουσιάζουμε και τον τρόπο που τροποποιήσαμε εφαρμογές (clients) για να είναι συμβατές σύμφωνα με το μοντέλο μας.

3.1. Σχεδίαση

3.1.1. Γενικά

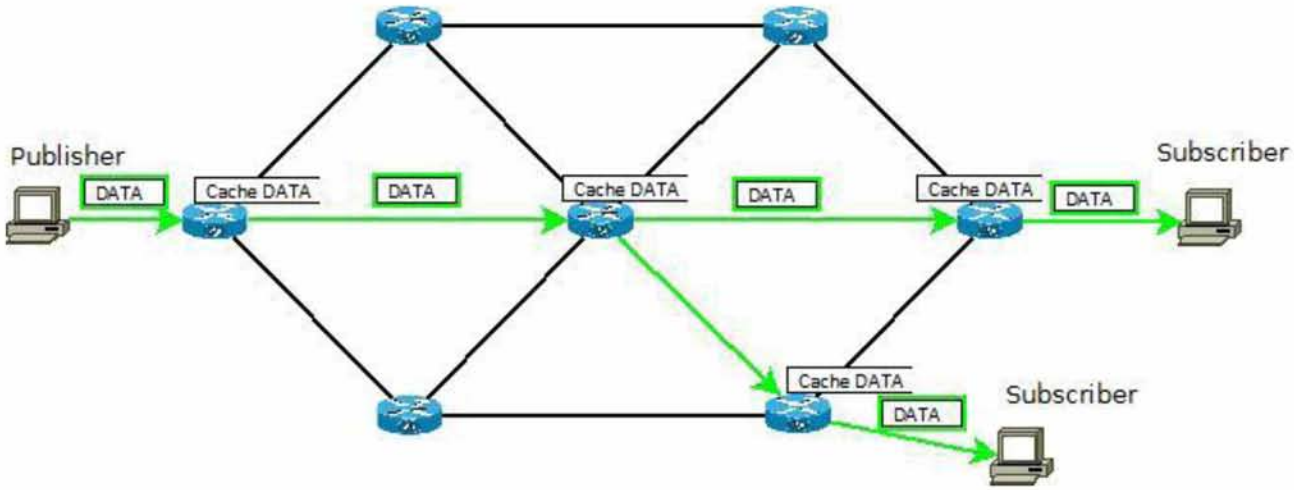
Η λειτουργία του μηχανισμού αποθήκευσης που παρουσιάζουμε ενσωματώνεται στο λογισμικό του Blackadder που τρέχει σε δρομολογητές. Έτσι επεκτείνουμε την λειτουργία του Blackadder ώστε να έχει την δυνατότητα να κάνει αποθήκευση πακέτα δεδομένων. Πιο κάτω αναλύουμε αυτή την διαδικασία.

Μέχρι στιγμής στην υλοποίηση του Blackadder ξέρουμε ότι ο κάθε publisher κάνει publish σε κάποια scopes και σε κάποια στοιχεία πληροφορίας (information items). Ενώ ένας subscriber κάνει subscribe σε κάποια scopes και σε κάποια στοιχεία πληροφορίας. Έτσι δεν υπάρχει κάποιου είδους μήνυμα που να ζητά δεδομένα από κόμβους του δικτύου, παρά μόνο μηνύματα που στέλνονται στον Rendezvous. Δηλαδή, σε μια παράδοση δεδομένων υπάρχει μια έμμεση επικοινωνία μεταξύ των publishers και των subscribers δια μέσω της λειτουργίας Rendezvous.

Συγκεκριμένα, οι subscribers κάνουν subscribe κάποιο RID στην λειτουργία Rendezvous. Αν υπάρχει κάποιος publisher που έκανε publish στο ίδιο RID, τότε ο publisher ενημερώνεται από την Rendezvous ώστε να αρχίσει να στέλνει δεδομένα στον subscriber. Μαζί με την ενημέρωση που έρχεται στον publisher

για να αρχίσει να στέλνει τα δεδομένα περιέχεται και το μονοπάτι από τον publisher προς τον/τους subscriber/s, μονοπάτι το οποίο το έφτιαξε η λειτουργία Topology management and formation (TM).

Ο Μηχανισμός αποθήκευσης λειτούργει σε κάθε κόμβο του δικτύου και επιλεγεί να αποθηκεύει μόνο πακέτα που περιέχουν δεδομένα και όχι άλλου είδους πακέτα (π.χ. του πρωτοκόλλου). Κατά την αποστολή δεδομένων, οι ενδιάμεσοι κόμβοι που περιέχονται στο μονοπάτι/α, από τον publisher προς τον/τους subscriber/s, έχουν την δυνατότητα να αποθηκεύσουν τα πακέτα δεδομένων που περνάνε από αυτούς. Στο Σχήμα 12 απεικονίζεται ένα τέτοιο στιγμιότυπο όπου στέλνονται δεδομένα από τον publisher προς τους subscribers. Βλέπουμε ότι στους κόμβους όπου περνούν τα δεδομένα, αυτά αποθηκεύονται από στον μηχανισμό αποθήκευσης.



Σχήμα 12: *Αποστολή δεδομένων από τον Publisher προς τους Subscribers και παράλληλα η αποθήκευση των δεδομένων στους ενδιάμεσους κόμβους.*

3.1.2. Συμβάσεις στο προηγούμενο μοντέλο

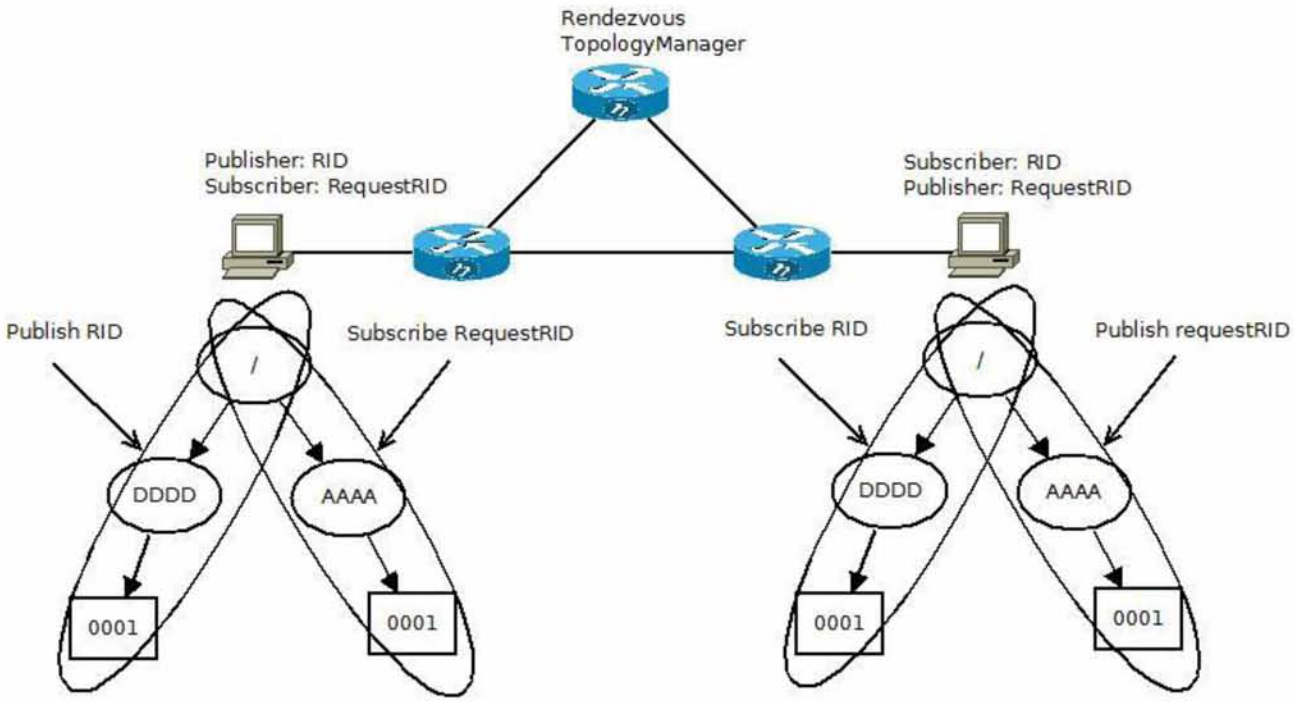
Ξέρουμε από την μέχρι στιγμής λειτουργία του Blackadder ότι οι εφαρμογές αλληλεπιδρούν με βάσει κάποιων ρουτινών που συνιστούν το εξυπηρετικό του μοντέλο. Ρουτίνες οποίες απευθύνονται στην λειτουργία Rendezvous, δηλώνουν ποιο ρόλο θα έχουν οι εφαρμογές που τις εκτελούν, είτε ως publishers είτε ως subscribers. Συγκεκριμένα κάνουμε τις εξής συμβάσεις που υποστηρίζονται από το προηγούμενο μοντέλο του Blackadder:

1. Για μια εφαρμογή που έχει δεδομένα προς δημοσίευση στέλνει ενημέρωση μέσω των ρουτινών publishScope() και publishInfo(), έτσι ενημερώνει την Rendezvous ότι θα είναι publisher για κάποια συγκεκριμένα RIDs πληροφορίας πάνω στον γράφο πληροφορίας. Αυτά τα RIDs αντιστοιχούν στα δεδομένα που έχει προς δημοσίευση. Παράλληλα στέλνει ενημέρωση μέσω των ρουτινών subscribeScope() και subscribeInfo() που με αυτά θα

ενημερώσει την λειτουργία Rendezvous ότι θα είναι subscriber για κάποια συγκεκριμένα RIDs, αυτά τα RIDs τα ονομάζουμε RequestRIDs και αντιστοιχούν σε πληροφορίες που περιέχουν αιτήσεις για τα δεδομένα που έγινε προηγουμένως publisher. Κατ' αντιστοιχία ένα RequestRID περιέχει αίτηση δεδομένων για ένα RID δεδομένων.

2. Για μια εφαρμογή που θέλει να εγγράφει ώστε να παραλαμβάνει δεδομένα στέλνει ενημέρωση στην λειτουργία Rendezvous μέσω της ρουτίνας subscribeInfo() για κάποια RIDs που αντιστοιχούν σε δεδομένα τα οποία ζητά. Παράλληλα στέλνει ενημέρωση μέσω της ρουτίνας publishInfo() και δηλώνει ότι θέλει να είναι publisher για κάποια RIDs που αντιστοιχούν σε RequestRIDs τα οποία αντιπροσωπεύουν πληροφορίες που περιέχουν αιτήσεις για τα συγκεκριμένα δεδομένα που έγινε προηγουμένως subscriber. Κατ' αντιστοιχία και πάλι ένα RequestRID περιέχει αίτηση δεδομένων για ένα RID δεδομένων.

Στο Σχήμα 13 βλέπουμε την αναπαράσταση των πιο πάνω συμβάσεων με την μορφή ενός παραδείγματος. Υπάρχει ο publisher που κάνει publish μια πληροφορία με RID="DDDD/0001" και subscribe στο RequestRID="AAAA/0001", ενώ ο subscriber κάνει subscribe μια πληροφορία με RID="DDDD/0001" και publish το RequestRID="AAAA/0001".

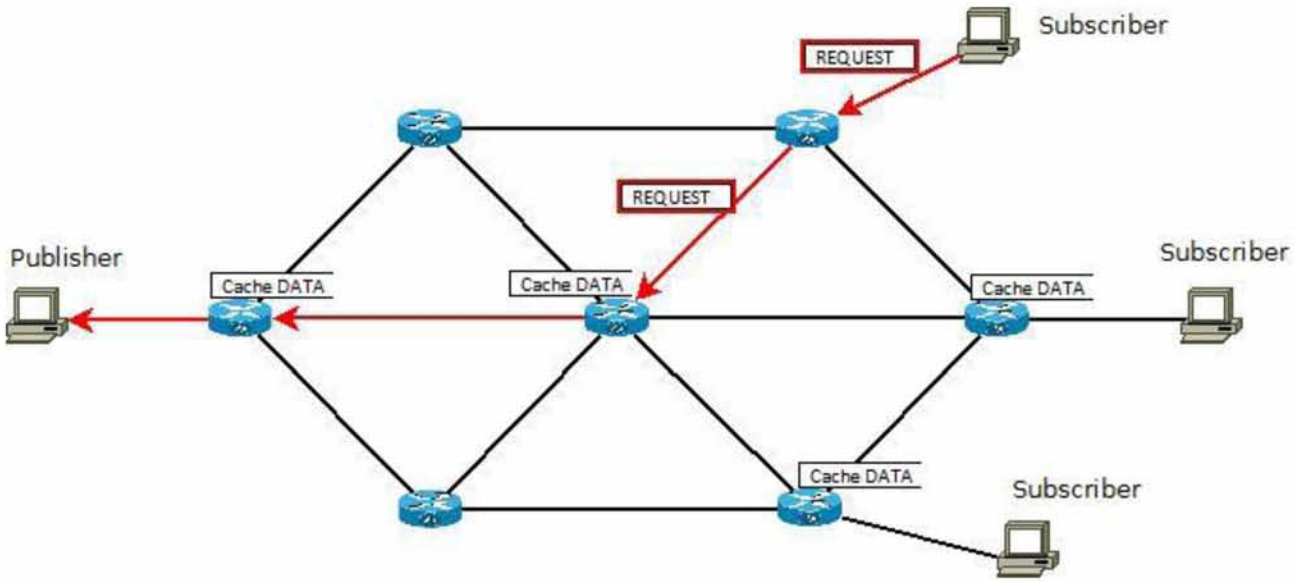


Σχήμα 13: *Σχηματικό παράδειγμα Publish και Subscribe και από τους δύο κόμβους (πελάτες).*

3.1.2.1. Μοντέλο από push σε pull

Ο πρώτος σκοπός αυτής της σύμβασης που επινοήσαμε είναι να δώσουμε την δυνατότητα στον subscriber να λάβει και αυτός από την λειτουργία ΤΜ του δικτύου, ένα μονοπάτι από τον ίδιον προς έναν publisher που περιέχει τα δεδομένα. Εδώ υφίσταται και η έννοια της αίτησης δεδομένων που δημιουργήσαμε και κατ' επέκταση του μηχανισμού αποθήκευσης.

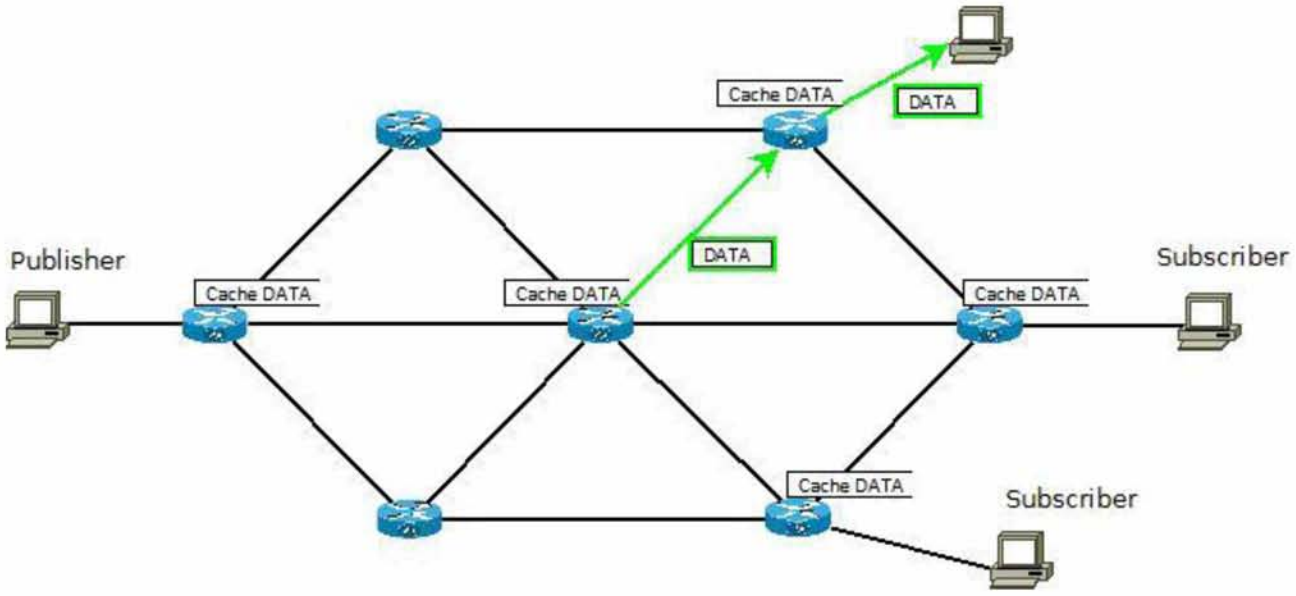
Όταν ο subscriber θα έχει το μονοπάτι προς τον publisher θα μπορεί να υπάρξει μια άμεση επικοινωνία μεταξύ αυτών των δύο κόμβων. Δηλαδή θα μπορεί να του στείλει ένα αίτημα για δεδομένα που ζήτα από αυτόν. Στην αποστολή του αιτήματος, ο publisher είναι και ο τελικός προορισμός και παράλληλα αυτός που έχει σίγουρα τα δεδομένα. Σε συνέχεια με το προηγούμενο σχήμα βλέπουμε στο Σχήμα 14 ένα subscriber που έχει το ανάποδο μονοπάτι προς τον publisher και το πιθανό δρομολόγιο της αίτησης.



Σχήμα 14: *Αποστολή του αιτήματος για δεδομένα από τον Subscriber με προορισμό τον Publisher και παράλληλα ο τερματισμός προώθησης του αιτήματος από κόμβο που περιέχει τα δεδομένα.*

Κατά την διάρκεια της προώθησης της αίτησης δεδομένων προς τον publisher υπάρχουν και οι ενδιάμεσοι κόμβοι στους οποίους είναι πιθανόν από προηγούμενες ανταλλαγές δεδομένων, να υπάρχουν αποθηκευμένα τα ίδια δεδομένα που αναζητά ο subscriber. Έτσι, αν βρεθούν σε κάποιο ενδιάμεσο κόμβο, να είναι δυνατόν να του επιστραφούν πριν φθάσει το αίτημα του στον publisher (τον τελικό προορισμό). Κάθε ενδιάμεσος κόμβος όταν λαμβάνει το αίτημα δεδομένων θα ελέγχει το δικό του μηχανισμό αποθήκευσης για τα δεδομένα που ζητά το αίτημα. Σε περίπτωση που βρεθούν τα δεδομένα σε έναν ενδιάμεσο κόμβο, τότε αυτός τα στέλνει πίσω στον subscriber. Στην περίπτωση που δεν βρεθεί ενδιάμεσος κόμβος που να περιέχει τα δεδομένα, το αίτημα δεδομένων θα φθάσει στον publisher που έχει σίγουρα τα δεδομένα και έτσι τα στέλνει στον subscriber με τον κλασσικό τρόπο που θα τα έστελνε χωρίς την

λειτουργία του μηχανισμού αποθήκευσης. Στο Σχήμα 15 μας δείχνει ένα στιγμιότυπο όπου βρέθηκαν τα δεδομένα στον δεύτερο κόμβο. Οπότε αποστέλλονται στον subscriber, επίσης αποθηκευτήκαν σε κόμβο που δεν τα είχε προηγουμένως.

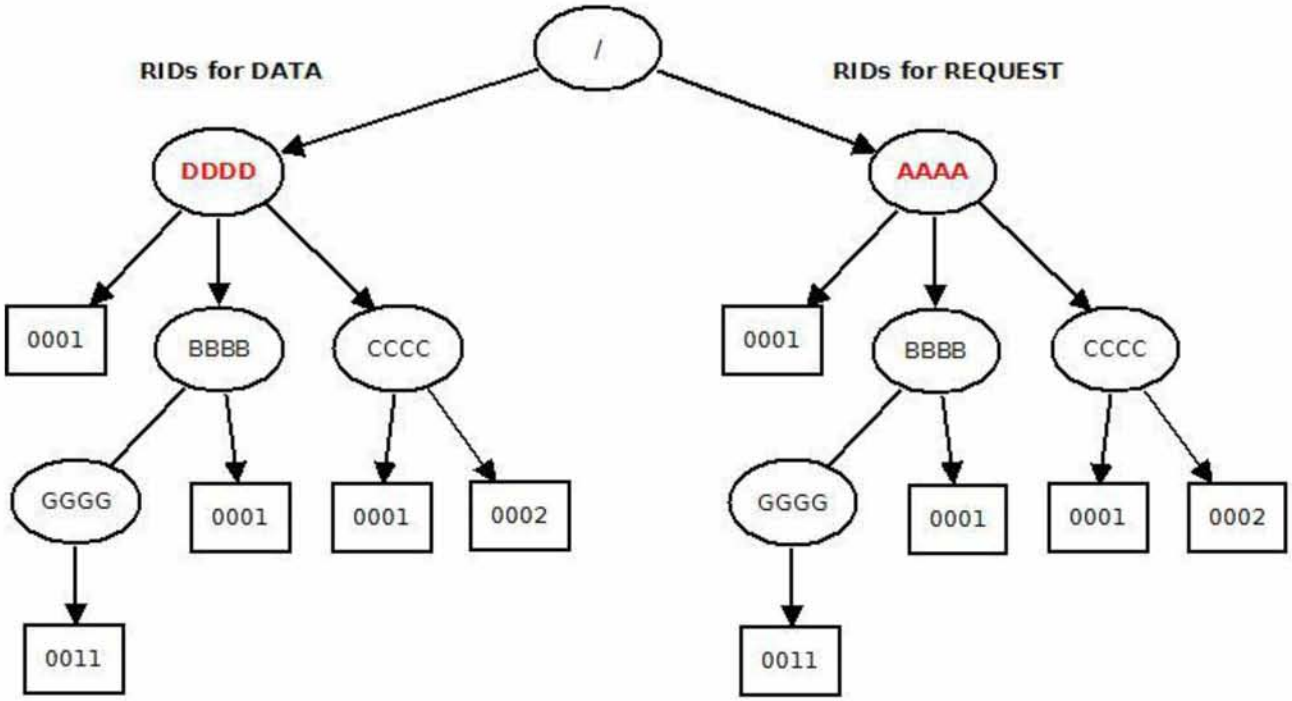


Σχήμα 15: *Αποστολή των δεδομένων που αιτήθηκε ο Subscriber από ενδιάμεσο κόμβο που περιείχε τα δεδομένα και παράλληλα η αποθήκευση των δεδομένων αυτών στον ενδιάμεσο κόμβο που βρίσκεται στο μονοπάτι που ακολουθούν προς τον Subscriber.*

3.1.2.2. Διαχωρισμός RIDs δεδομένων από RIDs αιτημάτων

Ο δεύτερος σκοπός της πιο πάνω σύμβασης ήταν αναγνώρισης των RIDs των δεδομένων από τα RIDs των αιτημάτων. Οπότε για να ξεχωρίσουμε τα RIDs των δεδομένων από τα RIDs των αιτημάτων, ορίσαμε δυο συγκεκριμένα scopes τα οποία θα μπαίνουν ως πρόθεμα στην κάθε περίπτωση, δεδομένων ή αιτημάτων. Το υπόλοιπο RID εκτός από το πρόθεμα παραμένει το ίδιο.

Στο Σχήμα 16 βλέπουμε ένα γράφημα πληροφορίας που αναπαριστά τη πιο πάνω σύμβαση, ώστε να ξεχωρίζει τα RIDs. Βλέπουμε ότι τα RIDs που αντιστοιχούν σε δεδομένα αρχίζουν από το scope "DDDD", ενώ τα RIDs που αντιστοιχούν σε αιτήματα για τα αντίστοιχα δεδομένα αρχίζουν με το πρόθεμα "AAAA". Για παράδειγμα το RID δεδομένων που αναγνωρίζεται από το αναγνωριστικό "/DDDD/BBBB/0001", θα αναζητείται από μια αίτηση που έχει ως RID το αναγνωριστικό "/AAAA/BBBB/0001".



Σχήμα 16: *Ένα γράφημα πληροφορίας που ξεχωρίζει τα RIDs των δεδομένων από τα RIDs των αιτημάτων για τα αντίστοιχα δεδομένα.*

3.1.3. Αλλαγές στους clients του Blackadder

Για τους σκοπούς της υλοποίηση μας τροποποιούμε κατάλληλα τους clients του Blackadder. Ο publisher πρέπει να αρχίσει την αποστολή δεδομένων προς τον subscriber μόνο και μόνο όταν λάβει την αίτηση από αυτόν. Δηλαδή, αγνοεί την ενημέρωση "start publish" που λαμβάνει από την Rendezvous που του λέει να αρχίσει να στέλνει δεδομένα. Ο subscriber που ζητεί δεδομένα αφού είναι και publisher στα αιτήματα των δεδομένων που γίνεται subscriber τότε θα λάβει από την Rendezvous, το μήνυμα "start publish" (που περιέχει και το μονοπάτι προς τον publisher) και του λέει να αρχίσει να στέλνει δεδομένα, τότε ο subscriber στέλνει το αίτημα για τα δεδομένα "*send request*".

3.1.4. Δομή πακέτου αίτησης

Για τις ανάγκες φτιάξαμε το πακέτο αίτησης και επιλέξαμε να τοποθετήσουμε τόσα πεδία όσα είναι το λιγότερο δυνατό ώστε να μην επιφέρουμε φόρτο στο δίκτυο.

Το πακέτο αίτησης περιέχει αναγκαία τα εξής: 1^{ο} ένα FID ώστε να βρει το μονοπάτι προς τον προορισμό του, 2^{ο} ένα RID που είναι το αναγνωριστικό του Rendezvous, 3^{ο} ένα RID το οποίο είναι το αναγνωριστικό του Rendezvous για τα

δεδομένα που αναζητά και 4° ένα πεδίο στο οποίο θα δομείτε το ανάποδο FID της διαδρομής (Reverse FID). Στο Σχήμα 17 φαίνεται η δομή του πακέτου αίτησης.

HEADER (IP or ETHERNET)	FID	RID	Data RID	Reverse FID

Σχήμα 17: *Η Δομή του πακέτου αίτησης (Request Packet).*

3.1.5. Δημιουργία ανάποδου μονοπατιού (Reverse FID)

Όταν μια αίτηση δεδομένων που φθάνει σε ένα κόμβο και βρει επιτυχώς τα δεδομένα στον μηχανισμό αποθήκευσης, τα δεδομένα αυτά θα πρέπει να επιστραφούν στο αποστολέα της αίτησης. Εδώ γεννάται το ερώτημα, «πως θα βρουν τον μονοπάτι να επιστραφούν πίσω στον αποστολέα της αίτησης;».

Στην σχεδίαση μας σκεφτήκαμε ότι καθώς μια αίτηση προωθείται κόμβο προς κόμβο μέσα στο δίκτυο από τον αποστολέα προς τον παραλήπτη, θα μπορεί να κωδικοποιεί το ανάποδο μονοπάτι, από τον μέχρι στιγμής κόμβο που θα βρίσκεται προς στον αποστολέα της. Το ανάποδο μονοπάτι είναι ένα FID (Forward ID) και το καλούμε ReverseFID. Αυτό το ReverseFID θα μπαίνει στην θέση του FID για το νέο πακέτο δεδομένων που θα δημιουργεί ο μηχανισμός αποθήκευσης στην περίπτωση που βρεθούν τα δεδομένα. Έτσι το πακέτο δεδομένων που θα δημιουργηθεί θα χρησιμοποιείται κανονικά από την λειτουργία Forwarder κάθε κόμβου για την δρομολόγησή του. Έτσι στο πακέτο αίτησης προσθέτουμε ένα πεδίο μεγέθους FID (σε bytes), επιλέγουμε να το τοποθετήσουμε στο τέλος του πακέτου. Αυτό το πεδίο θα ενημερώνεται κατάλληλα σε κάθε κόμβο που θα φθάνει. Η ενημέρωση του πακέτου γίνεται από τη λειτουργία Forwarder και βασίζεται στην δυαδική πράξη OR.

Σε κάθε ενδιάμεσο κόμβο που φθάνει το πακέτο αίτησης, θα πρέπει να κωδικοποιείται στο πεδίο ReverseFID το αναγνωριστικό της σύνδεσης (Link ID) που συνδέει τον κόμβο που είναι παρόν με τον κόμβο που βρισκόταν πριν. Αναγκαία πεδία για την κωδικοποίηση του ανάποδου μονοπατιού είναι ο πίνακας προωθήσεως του κόμβου και το πακέτο αίτησης. Σε κάθε κόμβο που φθάνει το πακέτο αίτησης το ReverseFID ισούται με το OR αυτού μαζί με τον Link ID που ενώνει τον κόμβο που είναι παρόν και τον κόμβο που βρισκόταν πριν. Στην περίπτωση της λειτουργίας του Blackadder, όπου οι εφαρμογές και το λογισμικό του δρομολογητή τρέχουν στον ίδιο κόμβο, γίνεται η αρχική πράξη OR με τον εσωτερικό σύνδεσμο του κόμβου(Internal Link ID). Αυτή την προσέγγιση που ακολουθήσαμε για την εύρεση και κωδικοποίηση ανάποδου μονοπατιού την αναπαριστούμε αλγοριθμικά στον αλγόριθμό που φαίνεται πιο κάτω.

Input of node: *Source Address (srcLink), Destination Address (dstLink) and Link IDs (linkID) of the outgoing links; internal Link ID (internalLink) if needed;*

Input of packet: *Source Address (srcPkt), Destination Address (dstPkt) and Reverse FID (reverseFID);*

If *(Subscriber is in this node)* **then**

reverseFID = internalLink

end

For each: *Link ID of outgoing interface* **do**

If *(src_link == dst_ pkt) && (dst_link == src_ pkt)* **then**

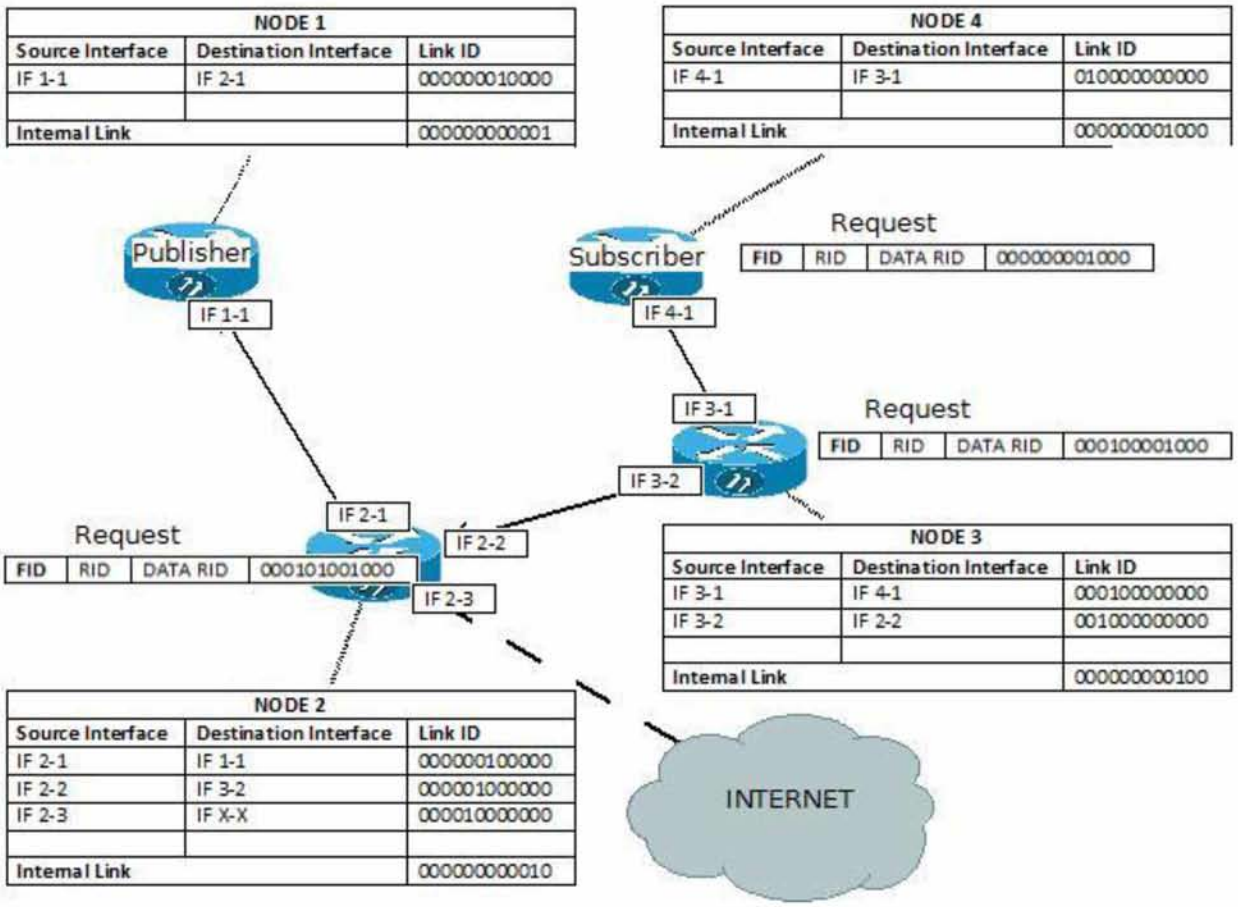
reverseFID = reverseFID | link

End

End

Αλγόριθμος 2: *Ο αλγόριθμος για την κωδικοποίηση του ανάποδου μονοπατιού στο πεδίο ReverseFID του πακέτου αίτησης.*

Στο Σχήμα 18 που ακολουθεί δίνεται ένα παράδειγμα κωδικοποίησης ενός ανάποδου μονοπατιού στο πεδίο ReverseFID του πακέτου, κατά την διάρκεια της αποστολής μιας αίτησης από τον subscriber προς τον publisher. Το πεδίο ReverseFID φαίνεται στο πακέτο αίτησης (request packet) σε δυαδική αναπαράσταση. Για κάθε άφιξη του πακέτου σε κάθε κόμβο γίνεται αναβάθμιση του πεδίου ReverseFID. Συγκεκριμένα η αναβάθμιση γίνεται στην λειτουργία Forwarder που κρατά στην μνήμη της τον πινάκα προωθήσεως του κόμβου. Έτσι με βάση τον αλγόριθμο που παρουσιάζεται πιο πάνω που έχει ως κύριο συστατικό στοιχείο την δυαδική πράξη OR, κωδικοποιείται το ανάποδο μονοπάτι.



Σχήμα 18: *Σχηματικό παράδειγμα κωδικοποίησης ενός ανάποδου μονοπατιού στο πεδίο (Reverse FID) του πακέτου αίτησης, καθώς προωθείται από τον Subscriber προς τον Publisher.*

3.1.6. Ενσωμάτωση του μηχανισμού μέσα στο Blackadder

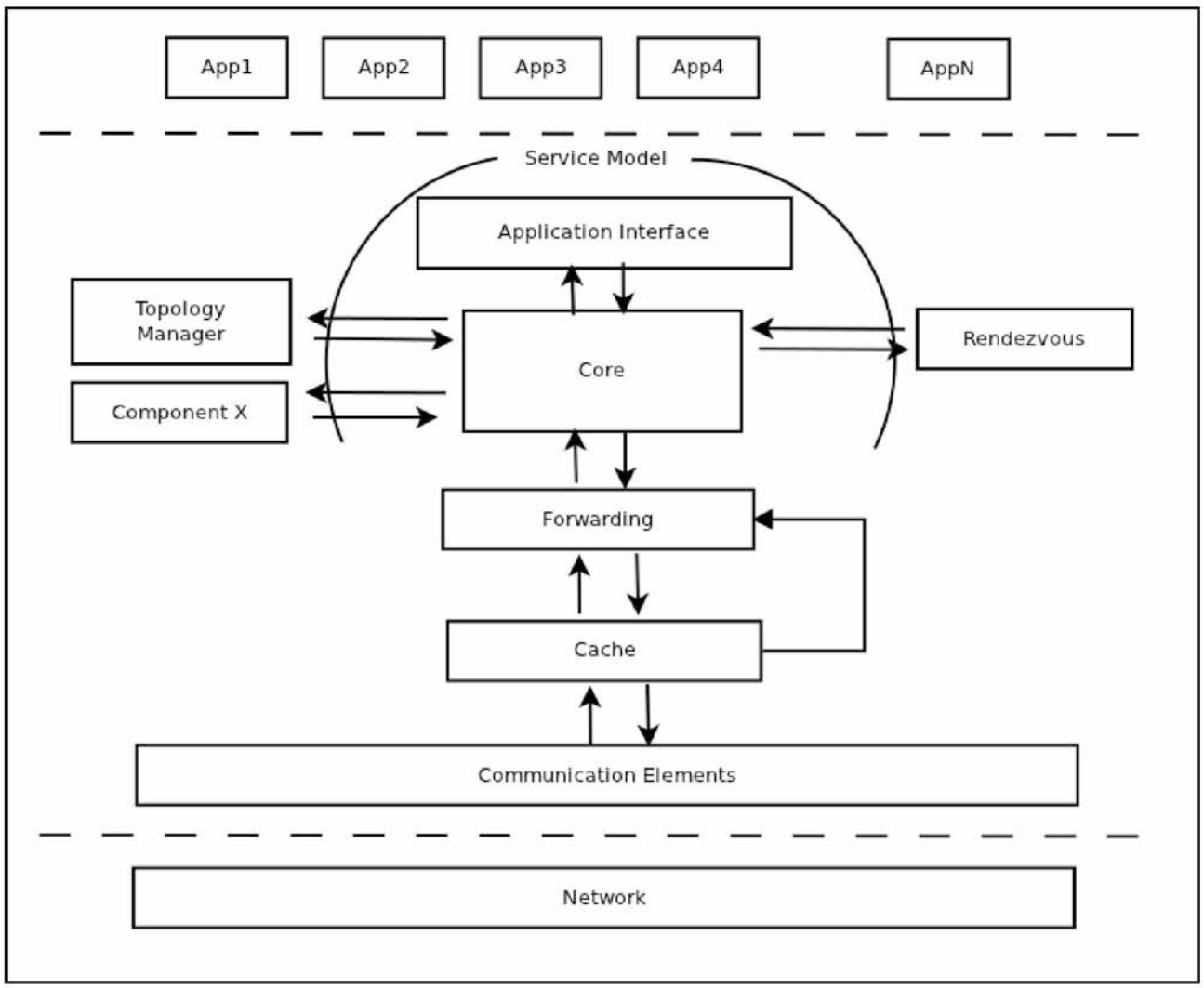
Τα πακέτα που εισέρχονται του κόμβο από το δίκτυο δέχονται επεξεργασία από το λογισμικό του Blackadder, το οποίο τρέχει μέσα από το εργαλείο Click Router.

Έτσι κάθε πακέτο περνάει από διάφορα elements του Click και ανάλογα με το τι περιέχει, με το τι πακέτο είναι, για ποιο/ποιούς προορίζεται και διάφορα άλλα, το λογισμικό του Blackadder κάνει τις απαραίτητες λειτουργιές. Έτσι για τους σκοπούς της υλοποίησης δημιουργήσαμε το δικό μας Click element. Του δώσαμε το όνομα Cache και εκτελεί την βασική λειτουργία του μηχανισμού αποθήκευσης.

Συλλογισμός: *Ο μηχανισμός αποθήκευσης που τρέχει σε κάθε κόμβο, πρέπει να είναι σε θέση να μπορεί να παραλαμβάνει πακέτα δεδομένων και πακέτα αιτήσεων που προέρχονται από το δίκτυο ή/και τις εφαρμογές που τρέχουν στον ίδιο του κόμβο. Επίσης, να είναι σε θέση να στέλνει δεδομένα στο δίκτυο ή σε εφαρμογές που τρέχουν στον ίδιο του κόμβο. Άρα θα πρέπει να μπει στην κατάλληλη θέση ώστε να αλληλεπιδρά με τις υπόλοιπες λειτουργίες και να παρέχει το επιθυμητό αποτέλεσμα.*

Με βάση τον πιο πάνω συλλογισμό, συμπεραίνουμε ότι πρέπει μέσα στα Click elements του Blackadder να επιλέγει η κατάλληλη τοποθεσία όπου θα ενσωματωθεί το δικό μας element Cache. Η τοποθεσία είναι οι κατάλληλες συνδέσεις που συνδέουν το δικό μας element με τα ήδη υπάρχοντα elements του Blackadder.

Στο Σχήμα 19 φαίνεται η ενσωμάτωση και η συνδεσμολογία του μηχανισμού αποθήκευσης (element Cache), μέσα στα Click elements του Blackadder.



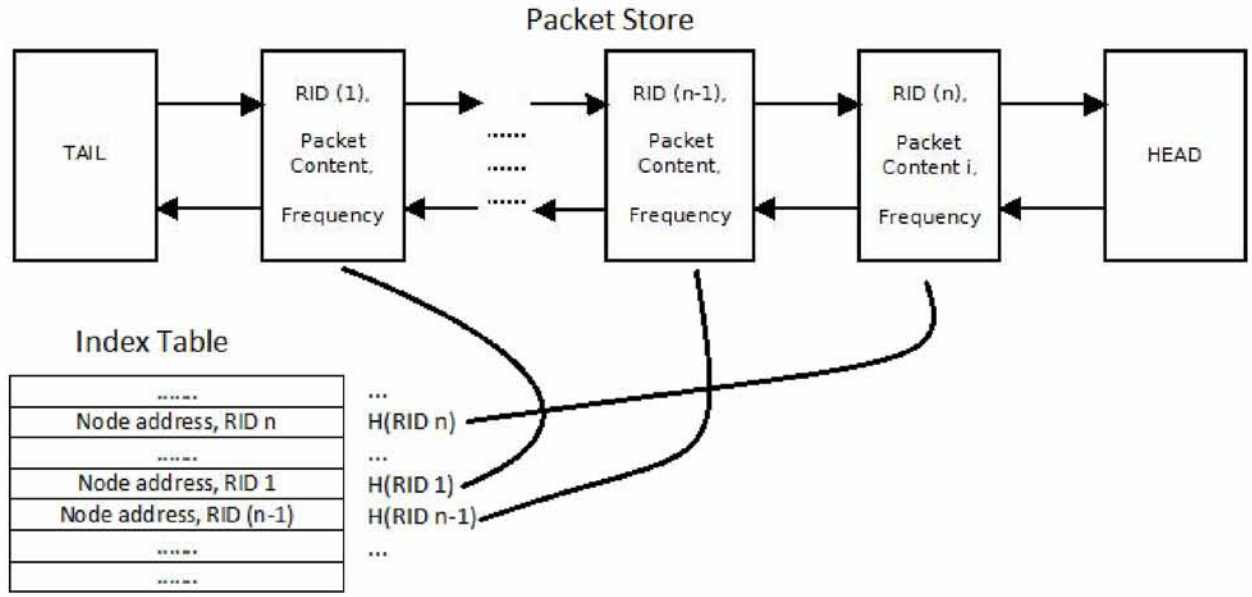
Σχήμα 19: *Ενσωμάτωση του μηχανισμού αποθήκευσης (Cache) μέσα στην αρχιτεκτονική κόμβου του Blackadder.*

Το element Cache που υλοποιεί τον μηχανισμό αποθήκευσης λειτούργει κατά την είσοδο και την έξοδο κάποιου πακέτου από τον κόμβο. Συγκεκριμένα το element Cache έχει αμφίδρομη επικοινωνία με τα Elements Forwarder και Communication Element. Επίσης έχει και μια δεύτερη εξερχομένη θύρα που επικοινωνεί πάλι με το element Forwarder. Ελέγχει τα πακέτα που εισέρχονται από το Communication element και/που εισέρχονται από το Element Forwarder. Σε περίπτωση που για κάποιο εισερχόμενο πακέτο αίτησης βρει δεδομένα να στείλει, τότε ετοιμάζει το πακέτο δεδομένων. Στη συνέχεια του βάζει το κατάλληλο FID (Reverse FID) και τα δεδομένα και το προωθεί πίσω στο Element Forwarder διαμέσου της θύρας 2. Στο Forwarder με βάση το FID θα γίνει η απόφαση για το που θα σταλεί, και έτσι θα τοποθετηθούν τα κατάλληλα header στο πακέτο και κατ' επέκταση να οδηγηθεί στον κόμβο που έστειλε την αίτηση.

3.2. Δομή του Μηχανισμού Αποθήκευσης

3.2.1. Γενική περιγραφή

Η σχεδίαση της μνήμης cache όπου λειτουργεί μέσα στον δρομολογητή περιλαμβάνει δυο δομές δεδομένων που είναι και τα βασικά αποθηκευτικά στοιχεία του μηχανισμού: μια διπλά συνδεμένη λίστα αποθήκευσης πακέτου (Packet Store) και ένα πίνακα κατακερματισμού (Index Table). Οι πληροφορίες και τα δεδομένα των πακέτων αποθηκεύονται σε διαφορετικούς κόμβους της λίστας Packet Store. Ο Index Table περιλαμβάνει πληροφορίες για να βρίσκει ένα πακέτο στην λίστα Packet Store. Δηλαδή στον πίνακα Index Table αποθηκεύονται οι διευθύνσεις των κόμβων της λίστας. Το μέγεθος της λίστας Packet Store είναι δυναμικό και μπορεί να μεγαλώσει ανάλογα όσο θέλουμε. Μια γενική εικόνα της δομής του μηχανισμού αποθήκευσης παρουσιάζεται στο Σχήμα 20.



***Σχήμα 20**: Δομές δεδομένων που αποτελούν την βασική δομή του μηχανισμού αποθήκευσης. Ένας πίνακας κατακερματισμού (Index Table) και μια διπλά συνδεδεμένη λίστα (Packet Store).*

3.2.2. Λειτουργίες

Οι βασικές λειτουργίες που είναι απαραίτητες σε ένα μηχανισμό αποθήκευσης (cache) είναι απαραίτητα οι πράξεις της εισαγωγής, διαγραφής και της αναζήτησης ενός στοιχείου. Περεταίρω για την υλοποίηση αλγορίθμων απόρριψης στοιχείων γίνεται αναγκαία και η πράξη της ταξινόμησης, ώστε να

έχουμε ταξινομημένα τα στοιχεία ως προς μια παράμετρο. Έτσι ξέρουμε ποιο να διαγράψουμε ανάλογα με τον αλγόριθμο που τρέχει κάθε φόρα στον μηχανισμό.

3.2.2.1. Εισαγωγή

Στο μοντέλο αποθηκεύουμε μόνο πακέτα δεδομένων που εισέρχονται στον δρομολογητή. Για κάθε νέα εισαγωγή πακέτου δεδομένων αποθηκεύονται και στις δυο δομές δεδομένων οι κατάλληλες πληροφορίες. Συγκεκριμένα ο πίνακας κατακερματισμού Index Table αποθηκεύει το RID του πακέτου και μια διεύθυνση κόμβου της λίστας Packet Store. Κάθε κόμβος της λίστας Packet Store αποθηκεύει το RID του πακέτου, τα δεδομένα του, το μέγεθος σε bytes των δεδομένων και την συχνότητα που περνάει το πακέτο δεδομένων από τον δρομολογητή. Στο Σχήμα 20 φαίνεται πως συνδέονται οι δύο βασικές δομές δεδομένων. Εφόσον ο πίνακας Index Table είναι ένας πίνακας κατακερματισμού, ο κατακερματισμός του RID, H(RID), καθορίζει την θέση εγγραφής στον πίνακα Index Table. Κάθε νεοεισερχόμενο πακέτο αποθηκεύεται στην ουρά της λίστας ανεξαρτήτως αλγόριθμου.

3.2.2.2. Ταξινόμηση

Στην υλοποίηση λαμβάνουν χώρο τρεις(3) διαφορετικοί αλγόριθμοι ώστε να κάνουν την κατάλληλη ταξινόμηση της λίστας (packet store). Οι αλγόριθμοι είναι οι εξής: FIFO, LRU και LFU. Για κάθε πακέτο που εισέρχεται στον δρομολογητή, γίνεται με βάση αλγορίθμου και η κατάλληλη ταξινόμηση του στην λίστα.

Για τον απλό αλγόριθμο FIFO δεν γίνεται ταξινόμηση.

Για τον αλγόριθμο LRU γίνεται ταξινόμηση με βάση την τελευταία φορά που χρησιμοποιήθηκε το πακέτο. Έτσι κάθε πακέτο που έρχεται στον δρομολογητή μπαίνει πρώτο, δηλαδή στην κεφαλή της λίστας.

Για τον αλγόριθμο LFU γίνεται ταξινόμηση με βάση την συχνότητα του πακέτου. Κάθε φορά που περνάει ένα πακέτο από τον δρομολογητή το οποίο είναι ήδη αποθηκευμένο στον μηχανισμό, τότε αυξάνεται κατά μια ακέραια μονάδα το πεδίο συχνότητας που βρίσκεται στον κόμβο του αντιστοίχου πακέτου. Επομένως ένα πακέτο μπαίνει στην κατάλληλη θέση προτεραιότητας στην λίστα ανάλογα με το πόσες φορές πέρασε από τον δρομολογητή.

3.2.2.3. Διαγραφή

Κάθε πακέτο που αποθηκεύεται στον δρομολογητή θεωρούμε ότι παραμένει αποθηκευμένο στον μηχανισμό, ανεξαρτήτως του χρόνου παραμονής του. Στην

περίπτωση που γεμίσει η μνήμη και θέλουμε να κάνουμε νέα αποθήκευση, διαγράφουμε το τελευταίο πακέτο, δηλαδή αυτό που βρίσκεται στην ουρά της λίστας.

3.2.2.3. Αναζήτηση

Καθώς φθάνει ένα πακέτο αίτησης (request packet) δεδομένων στον δρομολογητή, ο μηχανισμός κάνει κατακερματισμό του Data RID που μεταφέρει η αίτηση, H(RID), έτσι ελέγχει αν υπάρχει ίδιο RID στον πίνακα Index Table. Αν βρεθεί ότι το πακέτο είναι αποθηκευμένο στον δρομολογητή, τότε από τον Index Table βρίσκουμε την διεύθυνση του κόμβου όπου είναι αποθηκευμένα τα δεδομένα του πακέτου στην λίστα Packet Store. Αφού βρεθούν τα δεδομένα τότε δημιουργείται ένα νέο πακέτο δεδομένων το οποίο περιέχει τα δεδομένα που αιτήθηκαν και προορίζεται προς τον subscriber. Στην περίπτωση που βρεθεί ότι δεν υπάρχουν τα δεδομένα στον μηχανισμό, τότε δρομολογητής προωθεί την αίτηση στον επόμενο κόμβο.

3.2.3. Εντοπισμός και προώθηση δεδομένων

Όπως είπαμε και πιο πριν, ο subscriber αφού έχει την διαδρομή από αυτόν προς τον publisher, στέλνει αυτός την αίτηση δεδομένων. Μαζί με την αίτηση εκτός από το Data RID των δεδομένων που ψάχνει, υπάρχει και το πεδίο ReverseFID όπου είναι κωδικοποιημένο το ανάποδο μονοπάτι από τον μέχρι στιγμής κόμβο που φθάνει η αίτηση προς τα πίσω στον subscriber.

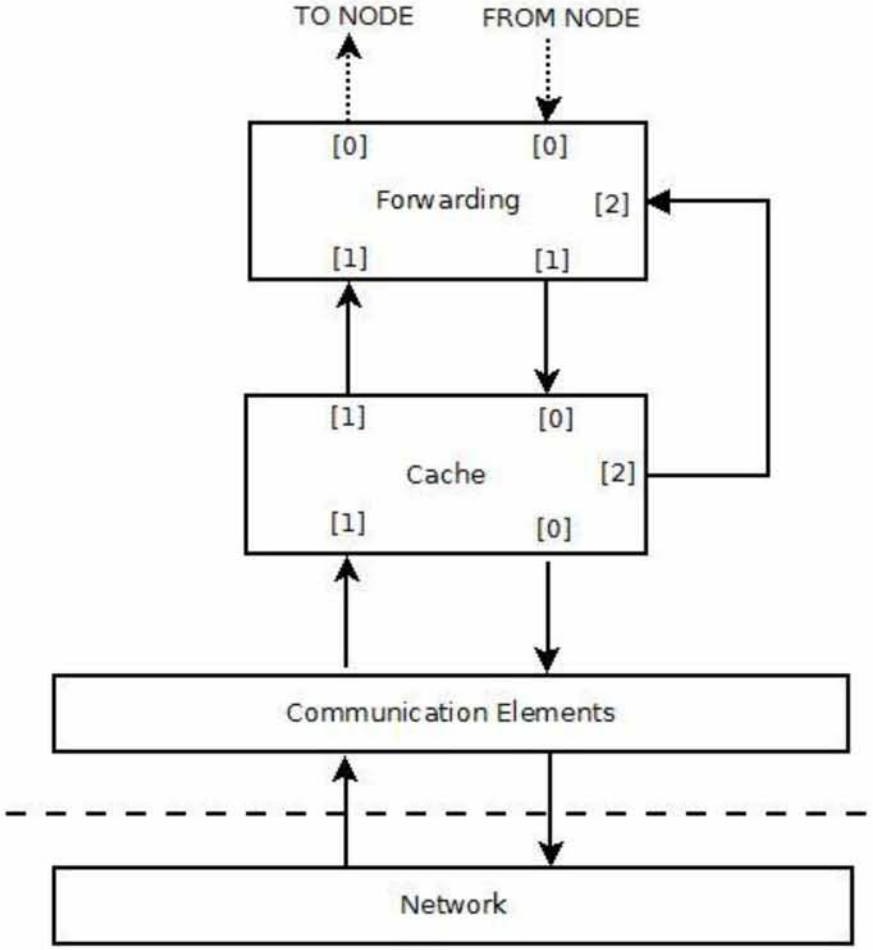
Έτσι σε κάθε κόμβο εκτελείται μια διαδικασία που χτίζεται κάθε φορά το νέο μονοπάτι. Στην περίπτωση που βρεθούν τα δεδομένα σε έναν ενδιάμεσο κόμβο, τότε δημιουργείται ένα νέο πακέτο δεδομένων. Στην θέση του FID τοποθετείται το ReverseFID. Στην θέση του RID τοποθετείται το Data RID που μεταφέρεται μαζί με την αίτηση. Στην θέση των δεδομένων τοποθετούνται τα δεδομένα που ανακτούνται από τον κόμβο της λίστας packet store. Έτσι δημιουργείται ένα έτοιμο πακέτο δεδομένων το ποιο προωθείται προς τον subscriber.

3.3. Υλοποίηση

Συγκεκριμένα για την υλοποίηση του μηχανισμού αποθήκευσης, δημιουργήσαμε ένα Click Element και τροποποιήσαμε κάποια αλλά υπάρχοντα Elements. Το Element που δημιουργήσαμε είναι το Element Cache και τροποποιήσαμε τα Elements Forwarder και GlobalConf. Επίσης δημιουργήσαμε και τους κατάλληλους Clients, που συμπεριφέρονται ταυτόχρονα ως Publisher και Subscriber.

3.3.1. Element Cache

Το element Cache είναι ο πυρήνας της παρούσας διπλωματικής εργασίας. Για την υλοποίηση του element Cache γράψαμε μια κλάση σε κώδικα C++ στα δύο αρχεία που δημιουργήσαμε, cache.cc και cache.hh. Είναι το στοιχειό που εκτελεί την λειτουργία αποθήκευσης που περιγράψαμε στο προηγούμενο υποκεφάλαιο. Συγκεκριμένα, αποθηκεύει πακέτα δεδομένων και επεξεργάζεται και αιτήσεις που ζητούν δεδομένα. Στην περίπτωση που βρεθούν δεδομένα, τότε ανακτούνται και στέλνονται σε αυτόν που τα αιτήθηκε. Όπως φαίνεται στο Σχήμα 21 το element Cache έχει (2) θύρες εισόδου και (3) θύρες εξόδου. Είναι αμφίπλευρα συνδεδεμένο με το element Forwarder για να στέλνει και να παραλαμβάνει πακέτα από τον κόμβο. Επίσης είναι αμφίπλευρα συνδεδεμένο με το element Communications για να παραλαμβάνει πακέτα που εισέρχονται και εξέρχονται από τον κόμβου. Επιπλέον έχει και μια τρίτη θύρα που επικοινωνεί και πάλι με το element Forwarder, όπου είναι η θύρα η οποία σε περίπτωση που βρεθούν δεδομένα για μια αίτηση στέλνονται μέσω αυτής στον Forwarder.



Σχήμα 21: *Οι συνδέσεις του Element Cache με τα άλλα Elements.*

Κάθε πακέτο που εισέρχεται στον κόμβο, και πιο συγκεκριμένα έρχεται από τα Communication elements και προωθείται προς τον κόμβο, θα περάσει από το element Cache, συγκεκριμένα θα εισέλθει από την θύρα εισόδου (1). Διακρίνονται τρεις (3) διαφορετικές περιπτώσεις πακέτων και κατά συνέπεια τρεις (3) διαφορετικές αντιδράσεις του μηχανισμού:

1. Αν είναι πακέτο δεδομένων αποθηκεύεται το περιεχόμενο του πακέτου μέσα στον μηχανισμό ή αν ήδη υπάρχει το πακέτο δεδομένων αποθηκευμένο, τότε εκτελούνται οι κατάλληλες λειτουργίες των αλγορίθμων ταξινόμησης. Στην συνέχεια προωθείται από την θύρα εξόδου (1) προς το element Forwarder.
2. Αν είναι πακέτο πρωτοκόλλου δεν γίνεται καμία επεξεργασία στον μηχανισμό και προωθείται από την θύρα εξόδου (1) προς το element Forwarder.
3. Αν είναι πακέτο αίτησης δεν γίνεται καμία επεξεργασία στον μηχανισμό και προωθείται από την θύρα εξόδου (1) προς το element Forwarder.

Κάθε πακέτο που εξέρχεται από τον κόμβο και πιο συγκεκριμένα έρχεται από το Element Forwarder και προωθείται προς το δίκτυο, θα περάσει από το element Cache, συγκεκριμένα θα εισέλθει από την θύρα εισόδου (0). Διακρίνονται και πάλι τρεις (3) διαφορετικές περιπτώσεις πακέτων και κατά συνέπεια τρεις (3) διαφορετικές αντιδράσεις του μηχανισμού:

1. Αν είναι πακέτο δεδομένων αποθηκεύεται το περιεχόμενο του πακέτου μέσα στον μηχανισμό ή αν ειδή υπάρχει το πακέτο δεδομένων αποθηκευμένο, τότε εκτελούνται οι κατάλληλες λειτουργίες των αλγορίθμων ταξινόμησης. Στην συνέχεια προωθείται από την θύρα εξόδου (0) προς τα Communication elements.
2. Αν είναι πακέτο πρωτοκόλλου δεν γίνεται καμία επεξεργασία στον μηχανισμό και προωθείται από την θύρα εξόδου (0) προς τα Communication elements.
3. Αν είναι πακέτο αίτησης τότε ο μηχανισμός ελέγχει το DATA RID (RID των δεδομένων) του πακέτου. Αν βρεθούν τα δεδομένα τότε δημιουργείται ένα νέο πακέτο δεδομένων. Στο νέο πακέτο τοποθετούνται τα δεδομένα που ζητά η αίτηση, στην θέση του FID τοποθετείται το ReverseFID που επίσης φέρει το πακέτο αίτησης. Η αίτηση τερματίζεται να προωθείται άλλο στο δίκτυο. Στη συνέχεια στέλνεται από την θύρα εξόδου (2) στην θύρα εισόδου (2) του Element Forwarder[2]. Στην περίπτωση που δεν εξυπηρετηθεί η αίτηση τότε το πακέτο αίτησης προωθείται από την θύρα εξόδου (0) στο δίκτυο.

Πιο κάτω δίνουμε ένα ψευδοκώδικα του Element Cache. Μέσα στον ψευδοκώδικα αναλύουμε την περιγραφή της πιο πάνω λειτουργίας του element Cache .

[2] Εκεί βάσει του νέου FID που έχει το πακέτο θα τοποθετηθούν και τα κατάλληλα headers είτε IP είτε ETHERNET.

```
struct index_table{
      char *RID;
      struct pakcet_store address;
};

struct packet_store{
      char *RID;
      char *data;
      int data_len;
};

void Cache::push(int port, Packet *p){

      if(p is protocol_packet){
            output(port).push(p);
      }
      else if((port == 1) &&(p is a request packet)){
            output(port).push(p);
      }
      /*    Έλεγχος μόνο κατά την έξοδο του πακέτου από τον κόμβο (port = 0)  */
      else if((port == 0) &&(p is a request packet))
      {
            if (hash_table_index.find(p->RID))/*      Αν βρεθεί το πακέτο στον
πίνακα index_table      */
            {
            /*    Εύρεση του κόμβου που περιέχει τα δεδομένα για το συγκεκριμένο
RID στον πίνακα packet_store  */
                  node = fnd_address_packet_store(p->RID);
                  node->frequency++;

            /*    Λειτουργία αλγορίθμων   */
                  if(LRU){    list_move_node_front(node);   }
                  else if(LFU){ list_sort_node(node);       }
                  else{       }     //FIFO

            /*    Δημιουργία ένος νέου πακέτου στο όποιο τοποθετούνται τα
δεδομένα που βρέθηκαν   */
                  WritablePacket *newpacket = Packet::make(FID_LEN + RID_LEN +
node->data_len);
                  memcpy(newpacket, p->ReverseFID, FID_LEN);
                  memcpy(newpacket   +   FID_LEN   p->RID.length(),   p->RID,   p-
>RID.length());
```

```
                memcpy(newpacket->data()  +  FID_LEN  p->RID.length(),  node-
>data, node->data_len);


                p->kill();


          /*    Προώθηση του πακέτου δεδομένων που μόλις δημιουργήθηκε προς το
ELement Forwarder μέσω της θύρας 2  */
                output(2).push(newpacket);
          }
          else  /*    Αν δεν βρει το πακέτο στον index_table    */
          {
          /*    Προώθησης της αίτησης στο επόμενο Element. Ανάλογα με την θύρα
που προέρχεται          */
                output(port).push(p);
          }
     }
     else (p is packet data)
     {
          /*    Αν υπάρχει το πακέτο στον πίνακα index_table    */
          if (hash_table_index.find(p->RID))
          {
          /*    Εύρεση του κόμβου που περιέχει τα δεδομένα για το συγκεκριμένο
RID στον πίνακα packet_store  */
                node = fnd_address_packet_store(p->RID);
                node->frequency++;


          /*    Λειτουργία αλγορίθμων   */
                if(LRU){    list_move_node_front(node);   }
                else if(LFU){ list_sort_node(node);       }
                else{       }     //FIFO
          }
          else  /*    Αν το πακέτο δεν είναι αποθηκευμένο       */
          {
                if(current_cache_size == cache_size)/*    Αν  είναι  γεμάτη  η
Cache */
                {
                /*    Ευρεση του κόμβου μέτα την TAIL     */
                      node = store_tail->nxt;
                      hash_table_index.erase(node->RID);
                      node->nxt->prv = store_tail;
                      store_tail->nxt = node->nxt;
                      free(node);
```

```
                        current_cache_size--;
                }

        /*      Δημιουργία ενός νέου κόμβου struct packet store */
                node = new (struct packet store);
                node->data_len = p->data_length;
                node->frequency = 1;
                memcpy(node->data, p->data(), p->data_length);
                memcpy(node->RID, p->RID, p->RID.length());

        /*      Αποθήκευση στον index_table   */
                struct index table index_node;
                index_node->RID = node->RID;
                index_node->address_node = node;
                hash_table_index.insert(p->RID, index_node);

        /*      Λειτουργία αλγορίθμων   */
                if(LRU)     {     list_add_node_end(node);      }
                else if(LFU){
                        list_add_node_begin(node);
                        list_sort_node(node);
                }
                else{ list_add_node_end(node);      }/*FIFO*/

                current_cache_size++;
        }
        output(port).push(p);
}
```

Το element Cache, έχει την δυνατότητα να πάρει διάφορες παραμέτρους κατά την εκκίνηση του Click, ανάλογα με την συμβολοσειρά διαμόρφωσης (Configuration String) που θα του δοθεί σαν είσοδος κατά την εκκίνηση του λογισμικού Blackadder στον δρομολογητή. Το Element Cache μπορεί να πάρει σαν είσοδο του:

- Τον αριθμό πακέτων που μπορεί να κάνει αποθήκευση, *«MAXSIZE»*. Υπάρχει προεπιλεγμένη τιμή που είναι ίση με 1000.
- Τον αλγόριθμο που θα τρέχει, *«ALGORITHM»*. Υπάρχουν 3 αλγόριθμοι, FIFO, LRU και LFU. Ο προεπιλεγμένος είναι ο LRU.
- Μια παράμετρος Boolean που μας λέει αν είναι ενεργοποιημένος ο μηχανισμός ή όχι *«ACTIVE»*. Η προεπιλεγμένη τιμή είναι true, άρα θα είναι

ενεργοποιημένος.
- Το Global Configuration του κόμβου για την λειτουργία του blackadder *«GC»*, το οποίο είναι αναγκαίο πεδίο και δεν έχει προεπιλεγμένη τιμή.

Η μορφή configuration string είναι η ακόλουθη:

Cache(MAXSIZE maxsize, GC globalconf, ALGORITHM algorithm, ACTIVE bool);

Για παράδειγμα, ένα στιγμιότυπο μπορεί να είναι:

Cache(MAXSIZE 100, GC globalconf, ALGORITHM LFU, ACTIVE 1);

3.3.2. Element Forwarder

Το element Forwarder είναι υπεύθυνο για την δρομολόγηση των πακέτων στο δίκτυο. Το τροποποιήσαμε προσθέτοντας κώδικα C++ στο αρχείο forwarder.cc

Ο σκοπός που επιλέξαμε το element Forwarder να κάνουμε τις κατάλληλες προσθήκες είναι επειδή μέσα στο element αυτό υπάρχει ο πινάκας προωθήσεως του κόμβου. Έτσι λιγοστέψαμε τον κώδικα από πλευράς υλοποίησης, παρά να δημιουργούσαμε πάλι τον πίνακα προώθησης στο element Cache. Ο πινάκας προωθήσεως στην υλοποίηση μας χρησιμεύει σε δυο διαφορετικές περιπτώσεις. Πρώτο σχετίζεται με την δημιουργία του ανάποδου μονοπατιού και δεύτερο χρησιμοποιείται από τα πακέτα δεδομένων που ανακτώνται από την Cache ώστε με βάση το ReverseFID που διαθέτουν να προωθηθούν πίσω σε αυτόν που έστειλε την αίτηση.

Για την δημιουργία του ανάποδου μονοπατιού, αναγκαία προϋπόθεση είναι:

1) Ο πινάκας προωθήσεως (forwarding table) που είναι αποθηκευμένος στην μνήμη του element Forwarder.
2) Το πακέτο αίτησης.

Έτσι κάθε φορά που εισέρχεται ένα πακέτο αίτησης στο Element Forwarder τότε να τρέχει ο αλγόριθμος που υλοποιήσαμε και κωδικοποιεί το ανάποδο μονοπάτι στο πεδίο ReverseFID που μεταφέρει η αίτηση.

Προσθέσαμε στο element Forwarder και μια τρίτη θύρα εισόδου, που θα είναι συνδεδεμένη με την τρίτη θύρα εξόδου του element Cache. Από εκεί εισέρχονται τα πακέτα δεδομένων που δημιουργούνται μετά από επιτυχημένη αναζήτηση κάποιου αιτήματος που έφτασε στην Cache.

Σε περίπτωση που παραλάβει πακέτο[3] από την Cache θα πρέπει να το δρομολογήσει στον κατάλληλο προορισμό. Επομένως με βάση τον πινάκα δρομολόγησης παίρνει την κατάλληλη απόφαση και έτσι τοποθετεί τα κατάλληλα headers IP ή ETHERNET. Υπάρχει η περίπτωση η αίτηση να έρχεται

[3] Το πακέτο έρχεται χωρίς κάποια headers (IP ή ETHERNET) αλλά μόνο με το ReverseFID.

από εφαρμογή που τρέχει στον ίδιο τον κόμβο. Σε αυτήν την περίπτωση ο Forwarder την προωθεί τοπικά στον κόμβο του και όχι στο δίκτυο.

3.3.3. Element GlobalConf

Στο element GlobalConf είναι οι καθολικές ρυθμίσεις για κάθε κόμβο ξεχωριστά. Προσθέσαμε κώδικα C++ στα αρχεία globalconf.cc και globalconf.hh.

Προσθέσαμε επίσης ένα προκαθορισμένο scope και το ονομάσαμε requestScope. Κάθε RID αίτησης θα αναρτάται κάτω από αυτό το requestScope. Δηλαδή τα RID των αιτήσεων θα έχουν ως πρόθεμα το requestScope. Το ορίσαμε στο αρχείο globalconf.hh ως:

String requestScope;

Και στο αρχείο globalconf.cc του δώσαμε την τιμή:

requestScope = "AAAAAAAAAAAAAAA"

κατά σύμβαση κάθε RID δεδομένων θα αναρτάται κάτω από το scope με *RID="DDDDDDDDDDDDDDD".*

Έτσι κάθε πακέτο δεδομένων θα έχει ως πρόθεμα του το *«DDDDDDDDDDDDDDD».*

3.3.4. Clients (Ταυτόχρονη λειτουργία publish και subscribe)

Οι υπάρχοντες Clients του Blackadder κάνουν μέχρι στιγμής μόνο μια από τις δυο βασικές λειτουργιές του μοντέλου publish/subscribe, δηλαδή κάνουν είτε publish είτε subscribe. Εμείς για τις ανάγκες του μοντέλου και την υλοποίηση του μηχανισμού αποθήκευσης, τροποποιήσαμε κατάλληλα τους clients ώστε να είναι και publisher και subscriber ταυτόχρονα.

Έτσι δημιουργήσαμε 2 διαφορετικά ειδή εφαρμογών clients:

- Η μια εφαρμογή client είναι αυτή που περιέχει την πληροφορία και θα είναι publisher στο RID της πληροφορίας και subcriber στο RequestRID το αιτήματα της πληροφορίας. Την εφαρμογή αυτή την υλοποιήσαμε στο

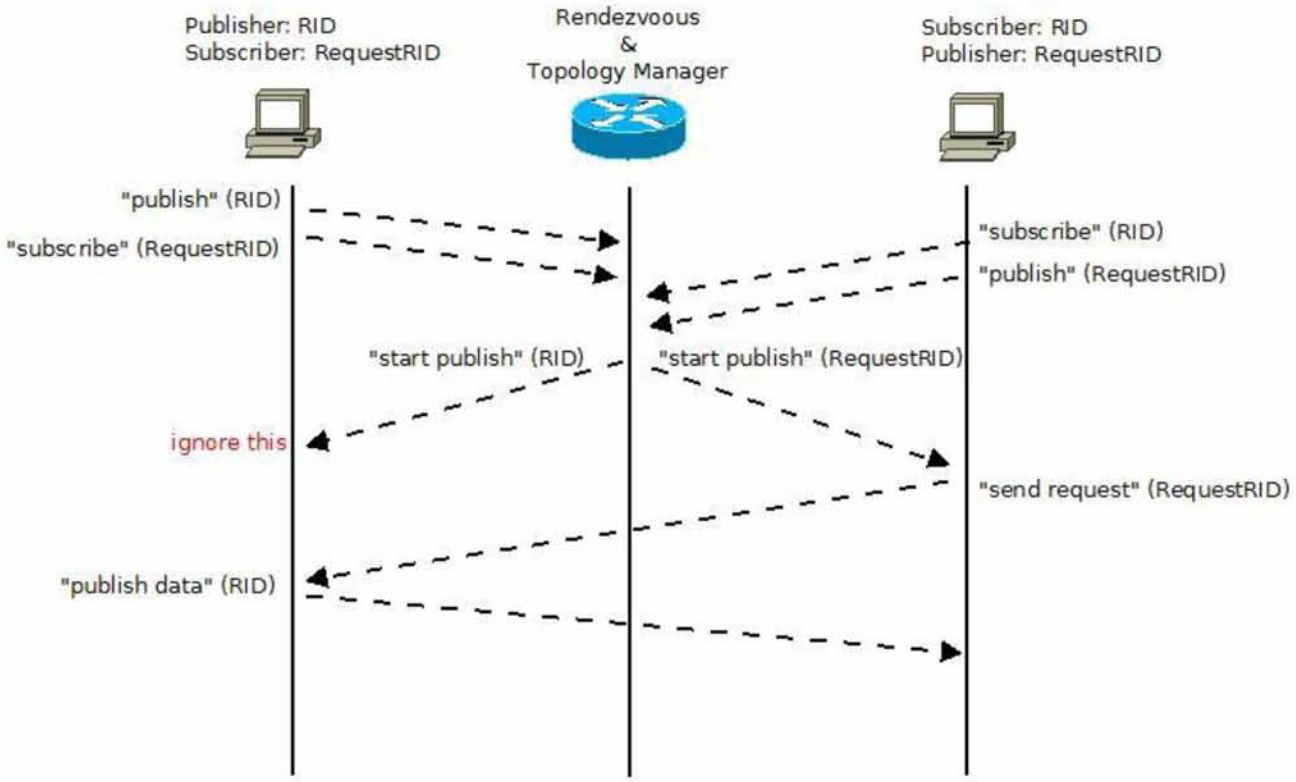
αρχείο pubD_subR.cpp.

- Η δεύτερη εφαρμογή client είναι αυτή που ζητά την πληροφορία και θα είναι subscriber στο RID της πληροφορίας και publisher στο RequestRID στο αίτημα της πληροφορίας. Την εφαρμογή αυτή την υλοποιήσαμε στο αρχείο subD_pubR.cpp.

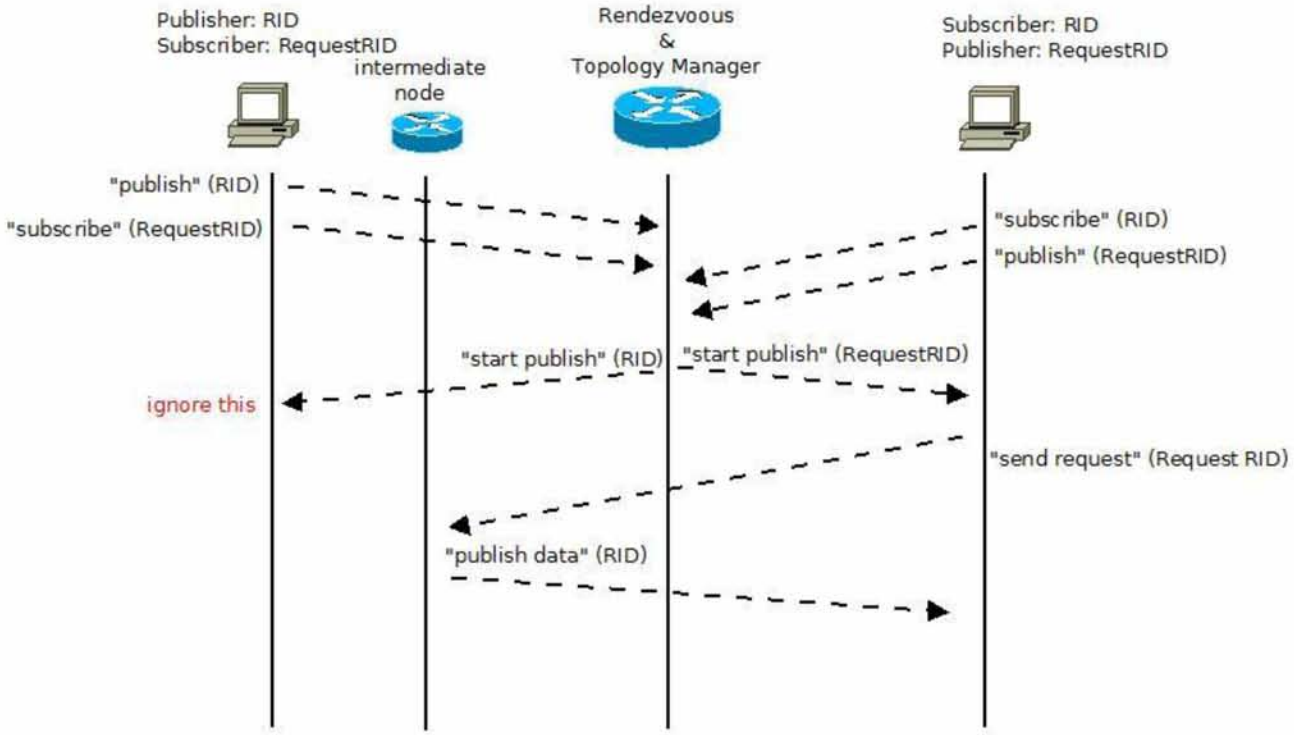
Στην περίπτωση της αποστολής της αίτησης δημιουργήσαμε μια νέα συνάρτηση στους clients που είναι παρόμοια με την ήδη υπάρχουσα publish_data(). Η συνάρτηση λέγεται sent_request(). Η συνάρτηση αυτή στέλνει την αίτηση προς το δίκτυο και συγκεκριμένα με δρομολόγιο προς τον Publisher.

*void sent_request(string &id, char strategy, char * LID, string &data_rid);*

Στα σχήματα που ακολουθούν, Σχήμα 22 και Σχήμα 23 δείχνουμε τον τρόπο που ανταλλάσσονται οι ενημερώσεις διαμέσω της Rendezvous λειτουργίας, ώστε να επιτευχθεί η παράδοση των δεδομένων. Στο Σχήμα 22 βλέπουμε τα δεδομένα να στέλνονται κανονικά από την εφαρμογή publisher μόλις παραλάβει το αίτημα (request). Ενώ στο Σχήμα 23 βλέπουμε πως τα δεδομένα ανακτούνται από έναν ενδιάμεσο κόμβο που παραλαμβάνει το αίτημα (request).



Σχήμα 22: *Αποστολή και παραλαβή των δεδομένων, μέσω ενημερώσεων. Τα δεδομένα ανακτούνται από την εφαρμογή που τα κάνει "publish".*



Σχήμα 23: *Αποστολή και παραλαβή των δεδομένων, μέσω ενημερώσεων. Τα δεδομένα ανακτούνται από κάποιον ενδιάμεσο κόμβο που τα είχε αποθηκευμένα στον μηχανισμό αποθήκευσης του.*

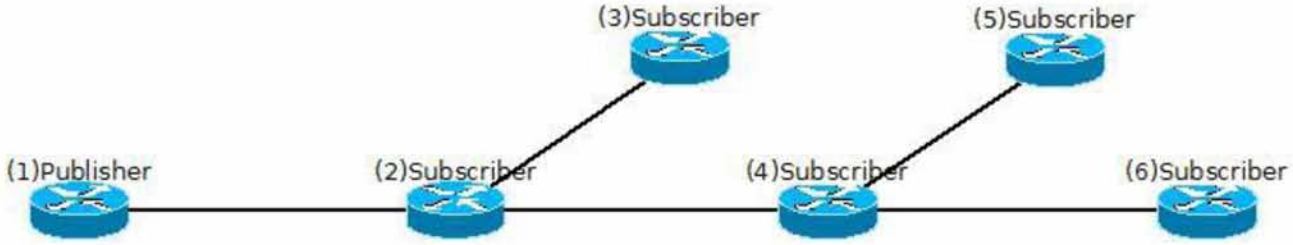
4. Πειράματα

Για να κάνουμε πράξη την παρούσα διπλωματική εργασία, κάναμε διάφορα πειράματα για να δούμε την λειτουργία του μηχανισμού αποθήκευσης. Κάναμε πειράματα για διάφορες τοπολογίες δικτύων στο planetlab και καταγράψαμε διάφορες μετρήσεις για χρόνους. Συγκεκριμένα λάβαμε μετρήσεις για τους τρεις αλγορίθμους ξεχωριστά FIFO, LFU και LRU καθώς και χωρίς την χρήση του μηχανισμού αποθήκευσης. Για όλες τις τοπολογίες δικτύων, υπήρχε μια εφαρμογή publisher και ένας αριθμός από εφαρμογές subscriber. Κατά την εκτέλεση των πειραμάτων αυξομειώναμε τον αριθμό της χωρητικότητα του μηχανισμού αποθήκευσης. Συγκρίναμε τα αποτελέσματα μεταξύ τους και βγάλαμε διάφορα συμπεράσματα μέσω των γραφικών παραστάσεων που προέκυψαν.

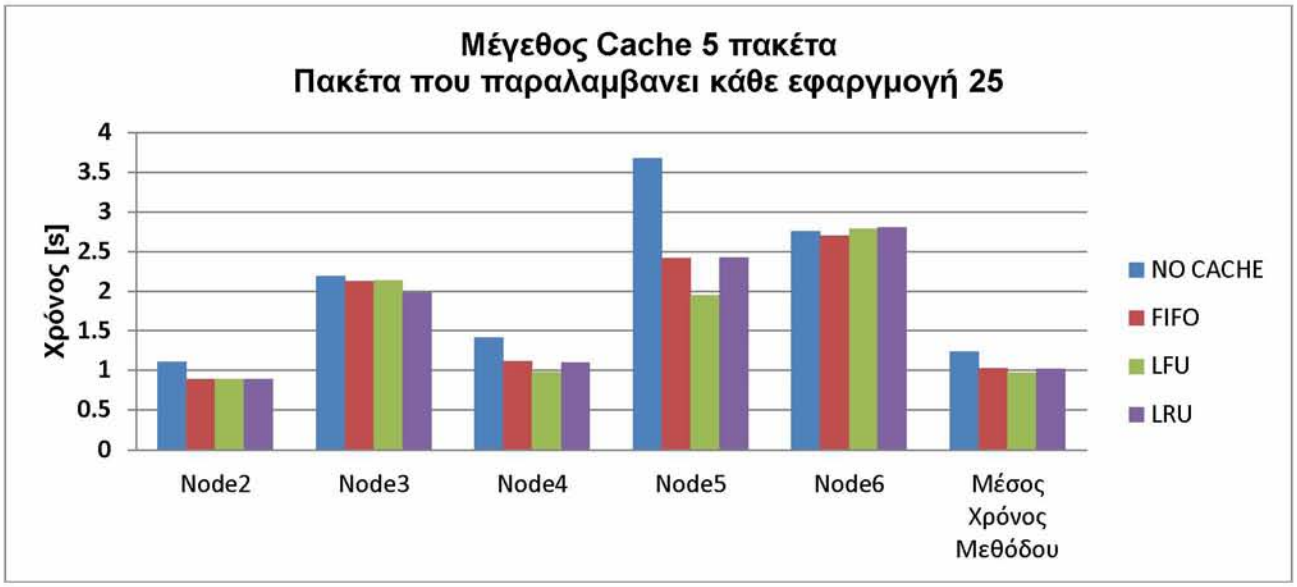
Για τα πειράματα χρησιμοποιήσαμε σαν κίνηση στο δίκτυο 5 διαφορετικά αντικείμενα τα οποία αποτελούνταν το κάθε ένα από 5 πακέτα. Έτσι συνολικά κάθε εφαρμογή που ζητούσε δεδομένα, ζητούσε συνολικά 25 πακέτα. Οι εφαρμογές που ζητούσαν πακέτα ξεκινούσαν ταυτόχρονα την εκτέλεση τους ώστε να γίνεται μείξη των πακέτων στους δρομολογητές.

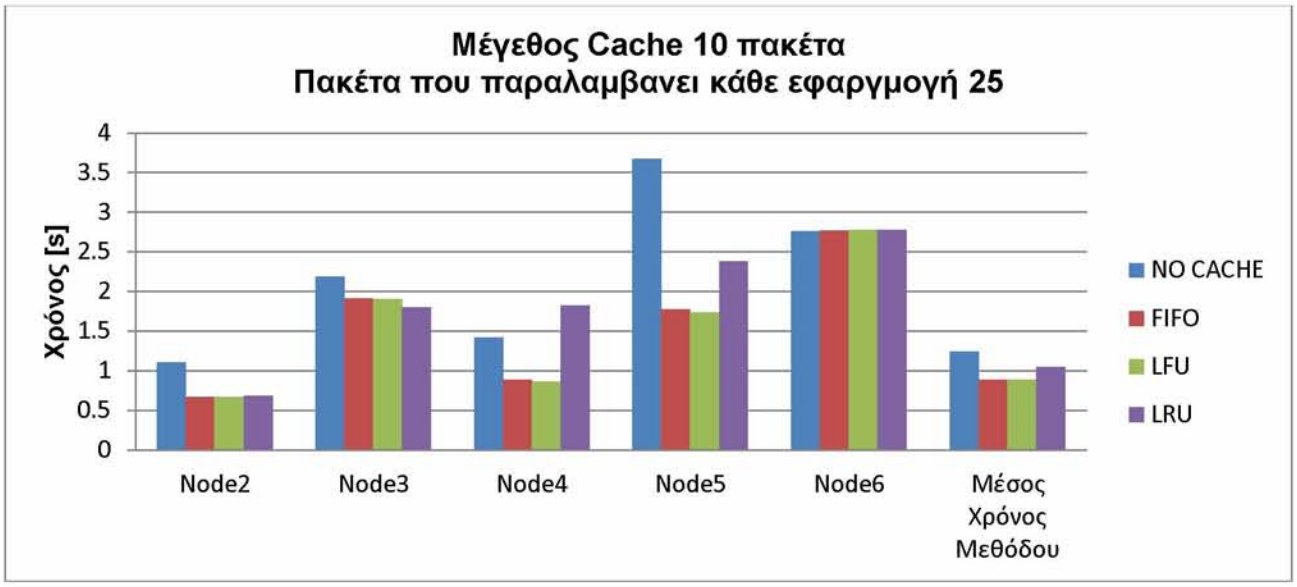
Πιο κάτω παρουσιάζονται τα αποτελέσματα των πειραμάτων σε γραφικές παραστάσεις και για τις δύο τοπολογίες δικτύων. Να αναφέρουμε ότι για κάθε τοπολογία παρουσιάζονται 4 γραφικές παραστάσεις. Στις οποίες σε κάθε μια, η κάθε εφαρμογή ζητάει σταθερό αριθμό πακέτων ίσο με 25, ενώ το μέγεθος της χωρητικότητας της μνήμης αυξάνεται σε 4 διαφορετικούς αριθμούς, όσες και οι γραφικές παραστάσεις. Συγκεκριμένα ο αριθμός χωρητικότητας πακέτων αυξάνεται ως εξής: 5, 10, 15 και 20 πακέτα.

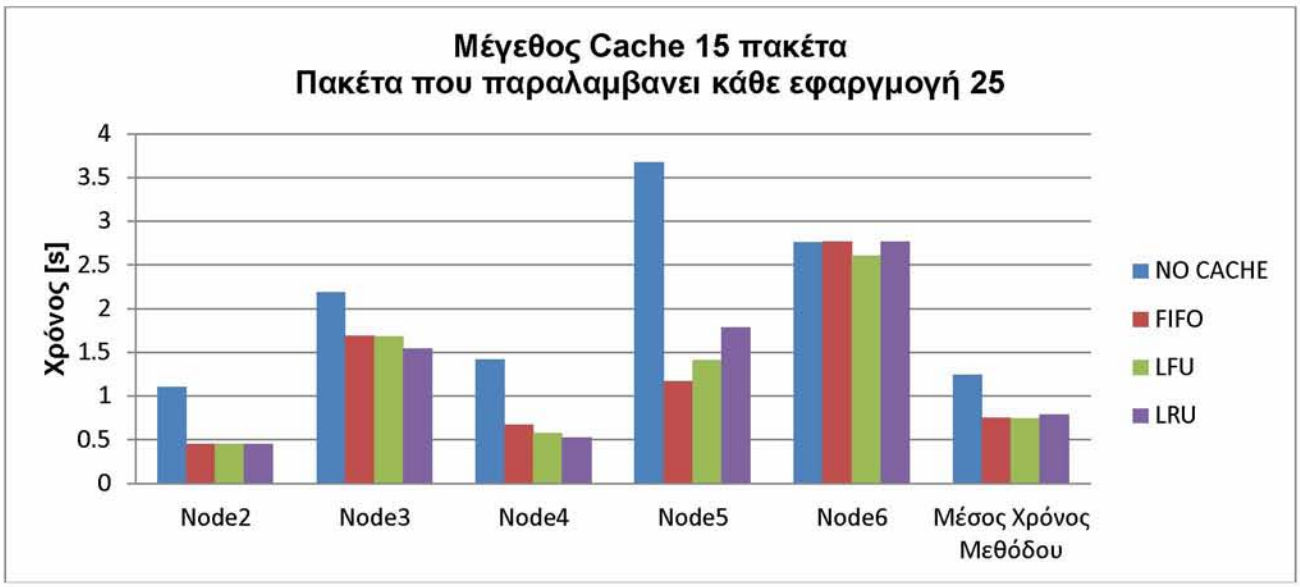
4.1 Συνδεσμολογία 6 Κόμβων

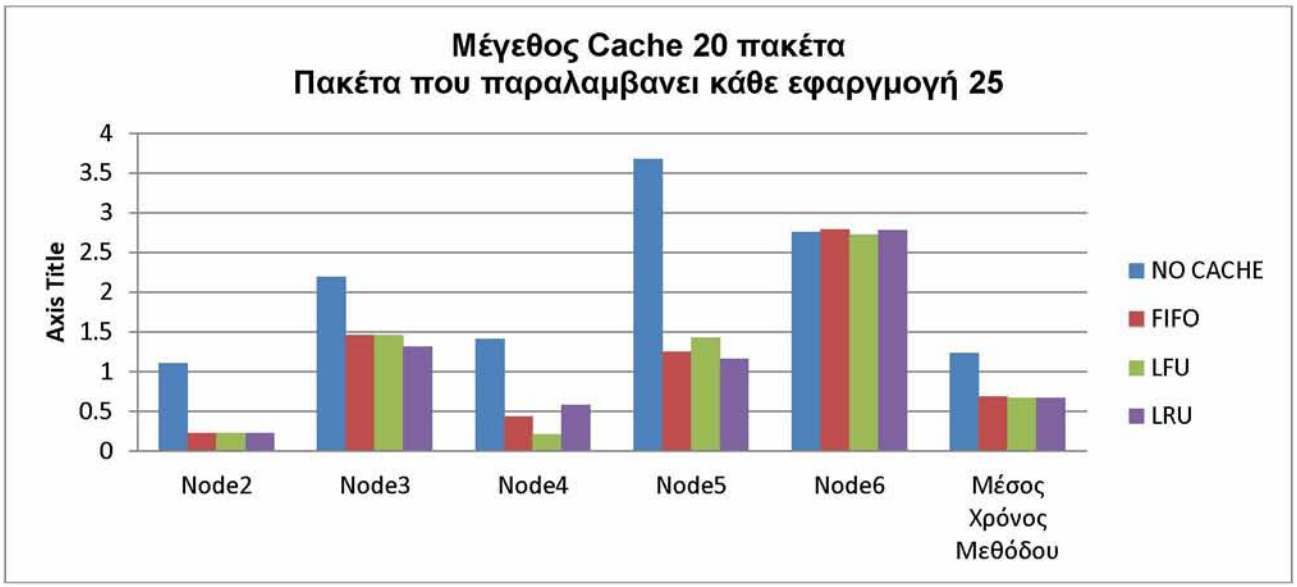


Σχήμα 24: *Τοπολογία με 6 κόμβους, 1 publisher και 5 subscribers, 1 σε κάθε κόμβο (Node).*





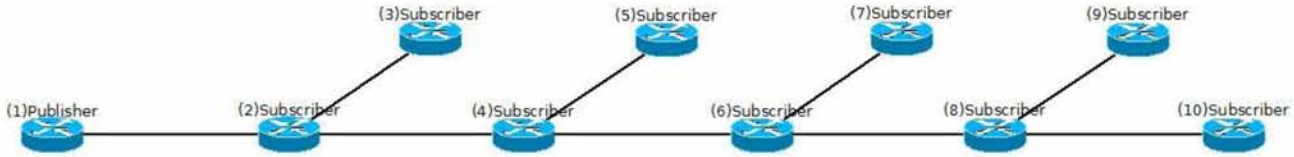




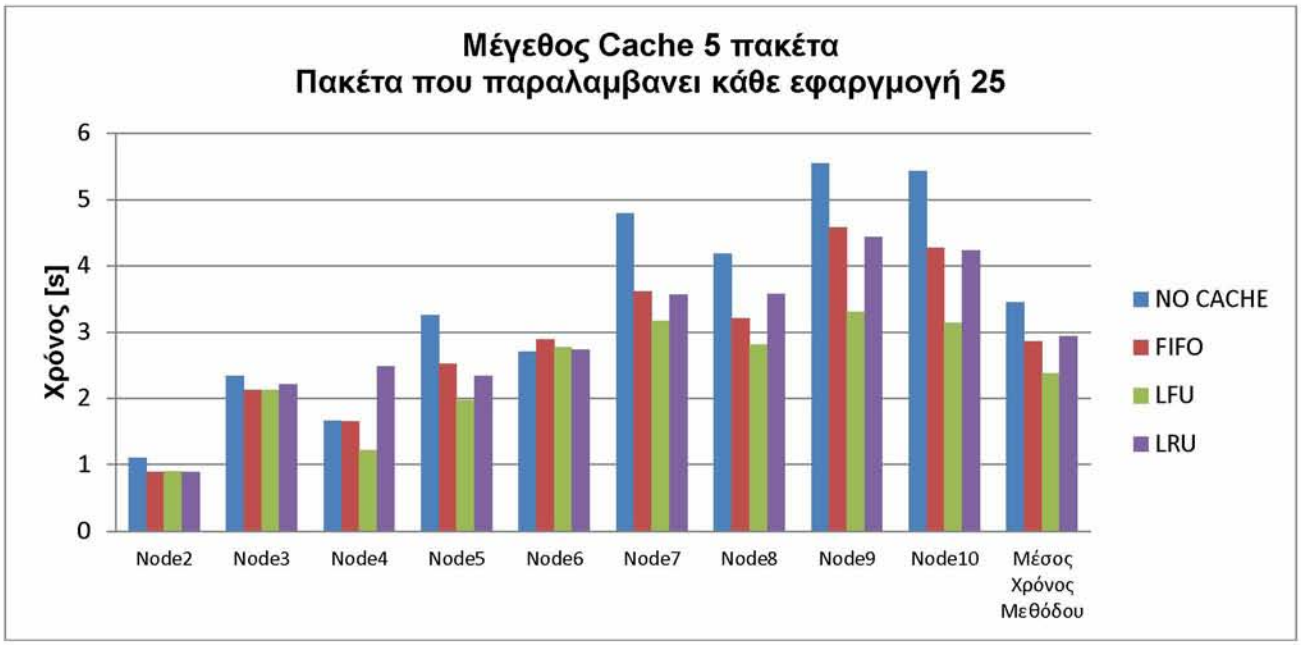
Αρχικά μπορούμε να πούμε ότι η τοπολογία των 6 κόμβων που φαίνεται στο Σχήμα 24 επαληθεύεται μέσω των χρόνων παραλαβής των πακέτων χωρίς την λειτουργία μηχανισμού αποθήκευσης (NO CACHE), που παρουσιάζεται με μπλε χρώμα στις γραφικές παραστάσεις. Βλέπουμε ότι οι χρόνοι παραλαβής των πακέτων από τις εφαρμογές που τρέχουν στους κόμβους είναι συμβατοί με βάση την τοπολογία. Για παράδειγμα οι εφαρμογές στους κόμβοι 5 και 6 λαμβάνουν σίγουρα σε μεγαλύτερους χρόνους τα πακέτα από τις εφαρμογές που τρέχουν στους κόμβους 2 και 4.

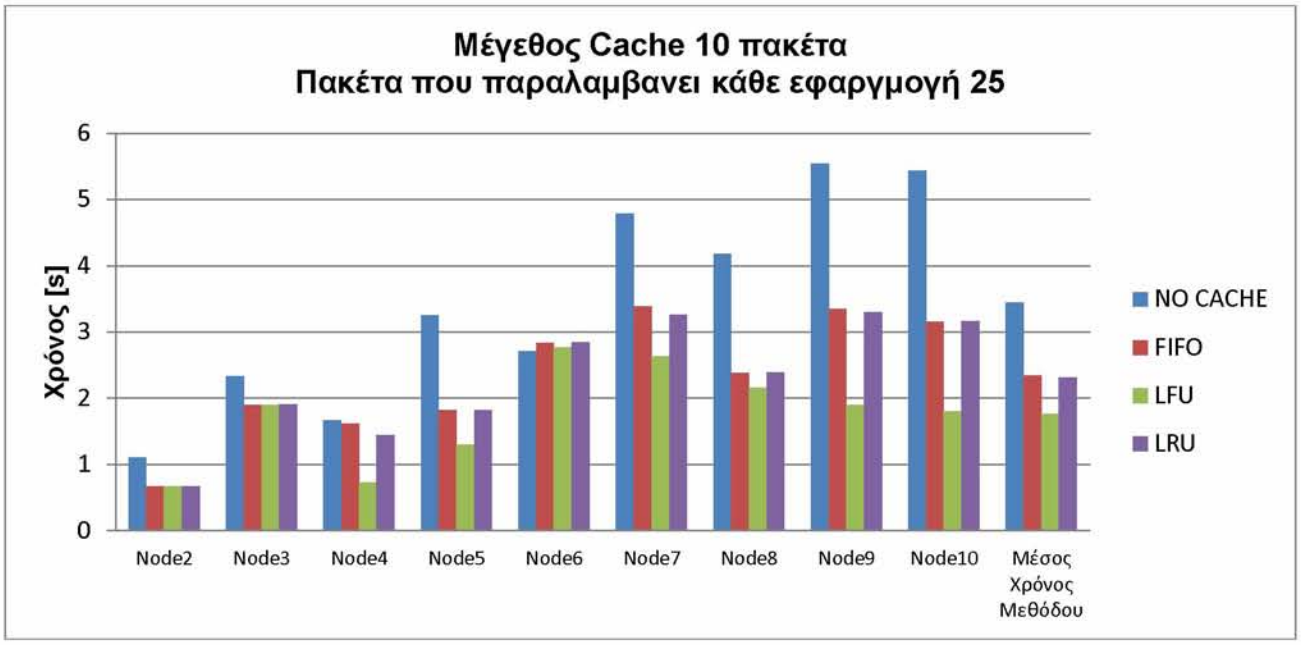
Τα αποτελέσματα των πειραμάτων είναι πολύ ενθαρρυντικά. Βλέπουμε ότι με βάση την τοπολογία στην περίπτωση που υπάρχει μηχανισμός αποθήκευσης οι χρόνοι παραλαβής των πακέτων είναι πολύ μικρότεροι σε σχέση με την μη χρησιμοποίηση μηχανισμού αποθήκευσης. Αυτό ισχύει για όλα τα μεγέθη της cache και βελτιώνεται όσο αυξάνεται η χωρητικότητα.

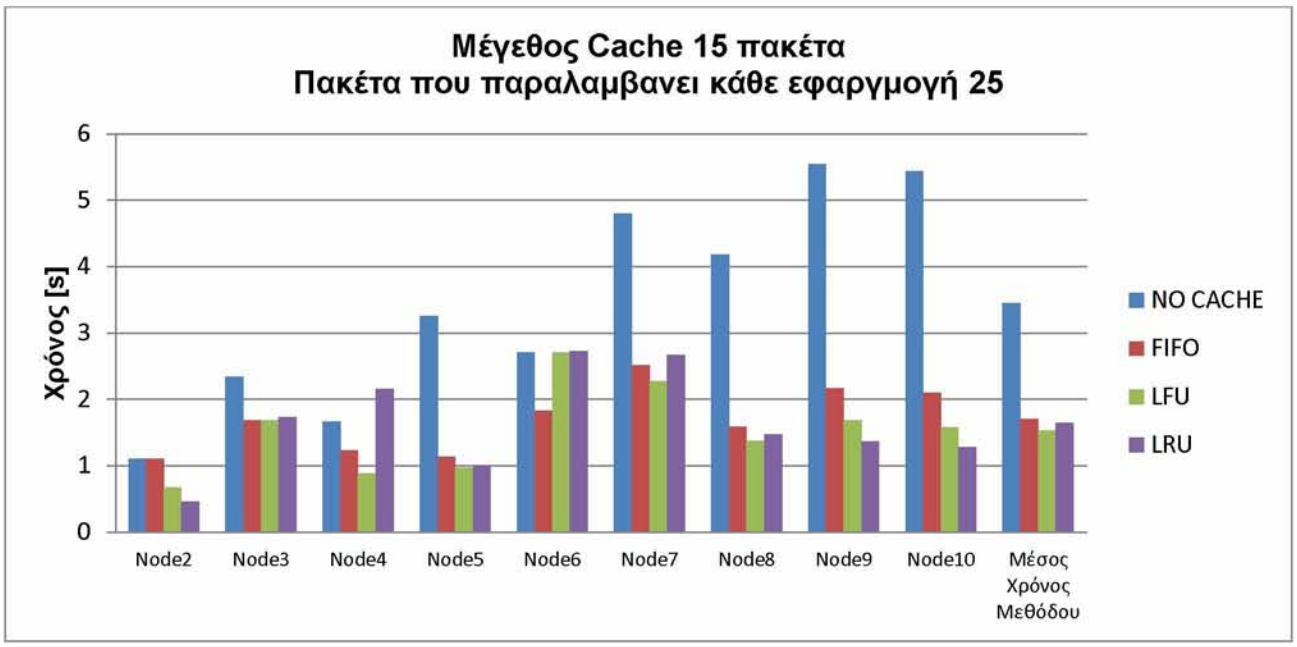
4.2 Συνδεσμολογία 10 Κόμβων

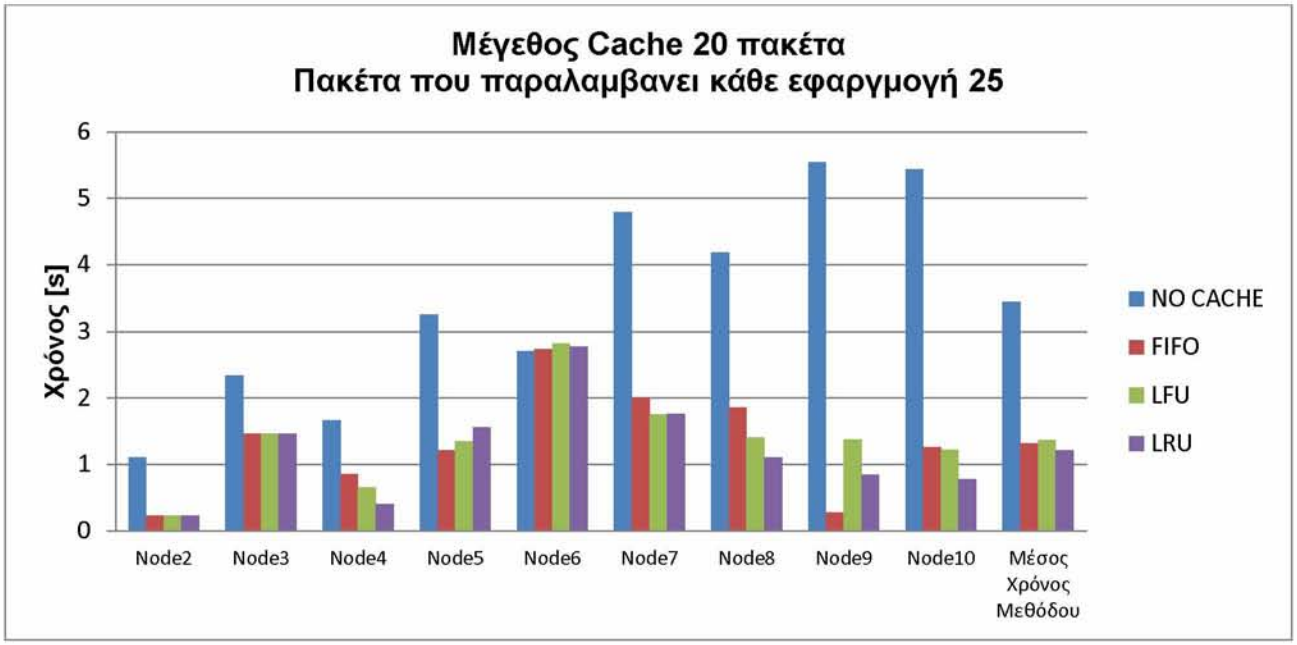


Σχήμα 25: *Τοπολογία με 10 κόμβους, 1 publisher και 9 subscribers, 1 σε κάθε κόμβο (Node).*









Τα αποτελέσματα των πειραμάτων για την τοπολογία 10 κόμβων είναι πάρα πολύ ενθαρρυντικά. Αρχικά βλέπουμε ότι και σε αυτήν την περίπτωση πάλι επαληθεύεται η τοπολογία που δημιουργήσαμε. Επίσης βλέπουμε στην περίπτωση που δεν υπάρχει μηχανισμός αποθήκευσης οι χρόνοι παραλαβής των πακέτων είναι πολύ πιο μεγαλύτερη σε σχέση με την χρήση μηχανισμού αποθήκευσης. Η μείωση του χρόνου επιτυγχάνεται σε όλες τις στρατηγικές αντικατάστασης.

5. Συμπέρασμα

Αντικείμενο της παρούσας διπλωματικής εργασίας ήταν η υλοποίηση μηχανισμού αποθήκευσης σε επίπεδο πακέτων για δίκτυα βασισμένα στο περιεχόμενο και πιο συγκεκριμένα η ενσωμάτωση του μηχανισμού στο λογισμικό του Blackadder. Σκοπός της υλοποίησης μηχανισμού αποθήκευσης ήταν η μείωση του χρόνου ανάκτησης δεδομένων που ζητά κάποιος χρήστης, σε σχέση με την ανάκτηση δεδομένων χωρίς χρήση μηχανισμού αποθήκευσης.

Το λογισμικό Blackadder λειτούργει μέσα από το εργαλείο Click Modular Router, για τον λόγο αυτό κληθήκαμε να υλοποιήσουμε κάποιο Click element και να τροποποιήσουμε κάποια υπάρχοντα elements. Σκοπός της δημιουργίας και τροποποίησης των elements ήταν η ενσωμάτωση του μηχανισμού μέσα στο λογισμικό Blackadder. Συγκεκριμένα έγινε η κατάλληλη συνδεσμολογία του νέου element Cache με τα ήδη υπάρχοντα. Επιπλέον με κάποιες νέες ιδέες που προτείναμε και κάποιες επεκτάσεις πάνω στο προηγούμενο μοντέλο πετύχαμε την ενσωμάτωση του μηχανισμού και τη ορθή ως προς την σχεδίαση λειτουργία του.

Καταλήγοντας είμαστε σε θέση να πούμε ότι επιτελέσαμε τον στόχο της διπλωματικής εργασίας. Ο μηχανισμός αποθήκευσης είναι φτιαγμένος σύμφωνα με τις προδιαγραφές που ορίζονται από τις αρχιτεκτονικές που στηρίζεται ο Blackadder. Επίσης είναι ένα ολοκληρωμένο Click element εύκολο ως προς τη διαχείριση του. Με το τέλος της εργασίας μπορούμε να πούμε ότι πετύχαμε την βέλτιστη συγγραφή κώδικα, τόσο από πλευράς μνήμης όσο και από πλευράς ταχύτητα. Η συνεισφορά που έδωσε αυτός ο μηχανισμός αποθήκευσης στον χώρο της δικτύωσης φάνηκε από τα αποτελέσματα των πειραμάτων.

Τα αποτελέσματα των πειραμάτων που πήραμε έδειξαν πολύ ξεκάθαρα την δυνατότητα του μηχανισμού. Βλέποντας ότι και για την πιο μικρή χωρητικότητα στους δρομολογητές πετυχαίνουμε καλυτέρα αποτελέσματα στον χρόνο παράδοσης της πληροφορίας στους χρήστες και ειδικά στον μέσο χρόνο παράδοσης των πακέτων προς όλους τους χρήστες. Από τα περάματα φαίνεται καθαρά ότι η χρήση μηχανισμού αποθήκευσης στο λογισμικό του Blackadder επιφέρει πολύ μεγάλο όφελος στους χρήστες (εφαρμογές) και γενικά στο δίκτυο.

5.1. Μελλοντικές Ιδέες

Το πεδίο εφαρμογής της παρούσας διπλωματικής εργασίας, που είναι η αποθήκευση δεδομένων σε ενδιάμεσους κόμβους πάνω σε δίκτυα βασισμένα στο περιεχόμενο, είναι ένα ανοιχτό πεδίο έρευνας και μπορεί να ξεχωρίσει σε διάφορες συνιστώσες. Είτε μπορεί κάποιος να επεκτείνει τον παρόν μηχανισμό αποθήκευσης είτε να αλλάξει κάποιες από τις βασικές σχεδιαστικές που προτείναμε και να πετύχει την δημιουργία παραπλήσιου μηχανισμού με επιπλέον δυνατότητες.

Μια μελλοντική επέκταση που μπορεί να γίνει στον Blackadder βασιζόμενη στον μηχανισμό που υλοποιήσαμε είναι να βρεθεί κάποιος τρόπος ώστε να ακυρώνονται τα δεδομένα που έχουν παλαιωθεί σε δρομολογητές του δικτύου.

Επιπλέον μια δεύτερη βελτίωση του μηχανισμού είναι η αποφυγή του RequestRID. Για παράδειγμα να υπάρχει μια ενημέρωση (notification) που να ενσωματωθεί στον μηχανισμό του Blackadder. Μια ενημέρωση παραπλήσια με την "Start Publish". Έτσι να ξέρει το λογισμικό του Blackader ότι φθάνει ένα πακέτο αιτήματος το οποίο ζητάει δεδομένα για το συγκεκριμένο RID. Με αυτό τον τρόπο υπάρχει περίπτωση να μην χρειάζεται το RequestRID, έτσι αποφεύγεται κάποιο μέρος της πληροφορίας στο πακέτο αίτησης.

Επιπλέον μια τρίτη βελτίωση του μηχανισμού είναι η ιδέα κάποιο αίτημα να ζητά πληροφορία για συγκεκριμένο scope παρά για κάθε συγκεκριμένο στοιχειό πληροφορίας. Για παράδειγμα ένας ενδιάμεσος κόμβος να στείλει όλα τα στοιχεία πληροφορίας που έχει αποθηκευμένα κάτω από το συγκεκριμένο scope. Έτσι θα μειώνεται και το συνολικό πλήθος των πακέτων αιτημάτων που θα στέλνονται στο δίκτυο.

6. Παραπομπές

[1] 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

[2] Mikko Sarela, Teemu Rinta-alho, Sasu Tarkoma. "RTFM: Publish/Subscribe Internetworking Architecture", ICT Mobile Summit, 2008

[3] Petri Jokela, Andras Zahemszky, Christian Eseve Rothenberg, Somaya Arianfar, Pekka Nikander. "LIPSIN: Line Speed Publish/Subscribe Iner-Networking", ACM SIGCOM, August 2009.

[4] Somaya Arianfar, Pekka Nikander, Jorg Ott. "On Content-Centric Router Design and Implications". In ReArch 2010

[5] Somaya Arianfar, Pekka Nikander. "Packet-level Caching for Information-centric Networking". ICT-SHOK Future Internet, June, 2010

[6] Eddie kohler. "The Click Modular Router". At http://pdos.csail.mit.edu/papers/click:kohler-phd

[7] Stefan Podliping, Laszio Boszormenyi. "A Survey of Web Cache Replacement Strategies". ACM Computing Surveys, Vol.35, No. 4, December 2003, pp. 374-398

[8] "Blackadder" project, at http://www.fp7-pursuit.eu/PursuitWeb/?page_id=338, 2012

[9] "Click Modular Router", at http://read.cs.ucla.edu/click, 2012

[10] "PATS Click Modular Router lessons", at http://www.pats.ua.ac.be/software/click, 2012