CORE
Metadata, citation and similar papers at core.ac.uk
Provided by University of Thessaly Institutional Repository

ΠΑΝΕΠΙΤΣΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ
ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

Διπλωματική Εργασία

Μελέτη πολιτικών προσωρινής αποθήκευσης σε συστήματα δημοσίευσης/συνδρομής .

Ζαραφέτας Χρήστος-Γεώργιος

Επιβλέποντες :

ΚΑΤΣΑΡΟΣ ΔΗΜΗΤΡΙΟΣ, Λέκτορας

ΤΑΣΙΟΥΛΑΣ ΛΕΑΝΔΡΟΣ, Καθηγητής

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

Διπλωματική εργασία για την απόκτηση του διπλώματος του Μηχανικού Ηλεκτρονικού Υπολογιστών, Τηλεπικοινωνιών και Δικτύων του Πανεπιστημίου Θεσσαλίας, στα πλαίσια του Προγράμματος προπτυχιακών σπουδών του Τμήματος Μηχανικών Η/Υ, Τηλεπικοινωνιών και Δικτύων του Πανεπιστημίου Θεσσαλίας.

Ζαραφέτας Χρήστος-Γεώργιος

Διπλωματούχος Μηχανικός Ηλεκτρονικών Υπολογιστών, Τηλεπικοινωνιών και Δικτύων.

Copyright Zarafetas Xristos-Georgios ,2010

Με επιφύλαξη παντός δικαιώματος. All rights reserved. Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικό ή ερευνητικής χρήσης .

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

ΕΥΧΑΡΙΣΤΙΕΣ :

Ευχαριστώ πρωτίστως την οικογένειά μου για την οικονομική αλλά και ηθική στήριξη που μου προσέφερε όλα τα χρόνια που χρειάστηκα για την κατάκτηση της μόρφωσης που έχω.

Ευχαριστώ πάρα πολύ τον διδακτορικό φοιτητή Βασίλη Σούρλα που σχεδόν μαζί κάναμε αυτήν την διπλωματική εργασία .

Ευχαριστώ επίσης και τον συμβασιούχο καθηγητή Πάρη Φλέγκα για τις ιδέες του και την καθοδήγηση του που ήταν απαραίτητη για την ολοκλήρωση της εργασίας αυτής.

Τέλος ένα μεγάλο ευχαριστώ αξίζει σε όλους τους συμφοιτητές/συμφοιτήτριες που με την συνεργασία τους αλλά και την παρέα τους κατάφερα να φτάσω εδώ .

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

Πίνακας περιεχομένων

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

ΕΝΟΤΗΤΑ 1 .Εισαγωγή

Η χρήση του διαδικτύου στις μέρες μας έχει αυξηθεί σε πολύ μεγάλο βαθμό με αποτέλεσμα να γίνεται εύκολα αντιληπτή η αναγκαιότητα τις εξέλιξής του ώστε να είναι σε θέσει να αντιμετωπίσει τις απαιτήσεις που δημιουργούνται. Κεντρικοποιημένες εφαρμογές δεν μπορούν να ανταποκριθούν στα νέα δεδομένα για αυτό το λόγο το ενδιαφέρων εστιάζετε πλέον στην χρήση των κατανεμημένων εφαρμογών .Τα κατανεμημένα συστήματα πλέον αποτελούνται από χιλιάδες τμήματα που είναι διασκορπισμένα σε όλον τον κόσμο έχοντας την δυνατότητα να συμπεριφέρονται και ως ξεχωριστές οντότητες αλλά και ως ένα ενιαίο σύνολο. Ο διαμελισμός αυτός όμως, ναι μεν λύνει το πρόβλημα της κλιμάκωσης, αλλά επίσης απαιτεί την δημιουργία ενός ποιο ευέλικτου και δυναμικού μοντέλου επικοινωνίας.

Τα παραδοσιακά συστήματα επικοινωνίας έχουν κάποια μειονεκτήματα καίριας σημασίας όπως για παράδειγμα απαιτούν κατά την διάρκειας ανταλλαγής πληροφοριών να είναι και οι δύο εφαρμογές που αλληλεπιδρούν ενεργές , ή έχουν την δυνατότητα να ανταλλάσουν πληροφορίες μόνο μεταξύ δυο εφαρμογών και είναι βασισμένα σε κεντρικοποιημένες αρχιτεκτονικές με αποτέλεσμα, σε περίπτωση αύξησης των εξυπηρετούμενων, το σύστημα να «καταρρέει», πράγμα πολύ πιθανόν πλέον μιας και μιλάμε για ένα χαοτικό διαδικτυακό χώρο.

Ένα πρόσφατο αλλά αυξανόμενης προσοχής σύστημα επικοινωνίας είναι το δημοσίευσης/συνδρομής (publish/subscribe-pub/sub) .Τα νέα χαρακτηριστικά που παρέχει το κάνουν ένα από τα πιο υποσχόμενα μοντέλα σχεδιασμού κατανεμημένων εφαρμογών που θα εκτοπίσει τα προβλήματα του διαδικτύου .Στην παρούσα εργασία αρχικά θα κάνουμε μια εισαγωγή περιγράφοντας περιληπτικά την λειτουργία του συστήματος αυτού μαζί με τεχνικές που θα εξετάσουμε αναλυτικά στην συνέχεα **[ΕΝΟΤΗΤΑ 2]** . Έπειτα θα ακολουθήσει μια εποπτική περιγραφή όλων των πτυχών του συστήματος **[ΕΝΟΤΗΤΑ 2]** και τέλος θα επικεντρωθούμε στο σύστημα επικοινωνίας Server-Client βασιζόμενο σε ένα επεκταμένο σύστημα δημοσίευσης/συνδρομής **[ΕΝΟΤΗΤΑ 4]** το οποίο και θα προσομοιώσουμε εξετάζοντας τα αποτελέσματα **[ΕΝΟΤΗΤΑ 5]** .

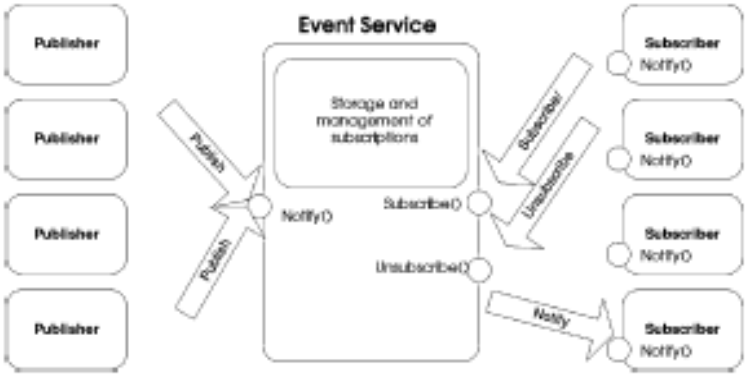
Σκοπός σε όλα τα παραπάνω στάδια της εργασίας μας είναι εφόσον αρχικά μελετήσουμε ένα τμήμα του συστήματος δημοσίευσης/συνδρομής, έπειτα να το συγκρίνουμε και να εξελίσσουμε αυτό ως προς την δυνατότητα εξυπηρέτησης μεγάλου αριθμού πελατών (κλιμάκωση συστήματος) την ανεξαρτητοποίηση των οντοτήτων που το αποτελούν και χρονικά (ασύγχρονα-σύγχρονα συστήματα) αλλά και χωρικά(π.χ. αλληλεπιδρόμενοι εξυπηρετητές τοποθετημένοι ανά τον κόσμο) και τέλος την αποδοτικότητα του ως προς τον χρόνο αναμονής (delay) και την πολυπλοκότητα που εισάγει (overhead).

Παρακάτω θα περιγράψουμε περιληπτικά τον βασικό τρόπο αλληλεπίδρασης του εν λόγο συστήματος και θα αναφέρουμε εν συντομία μερικές τεχνικές που βελτιώνουν ή αντιμετωπίζουν διάφορα μειονεκτήματα που προκύπτουν.

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

ΕΝΟΤΗΤΑ 2 .Γενική περιγραφή του συστήματος δημοσίευσης/συνδρομής

Στο σχήμα επικοινωνίας δημοσίευσης/συνδρομής οι εφαρμογές χωρίζονται σε δύο μεγάλες κατηγορίες, Α)αυτές που θα δημοσιεύουν μηνύματα με πληροφορίες για ένα γεγονός και Β)αυτές που θα κάνουν εγγραφή σε ένα γεγονός για το οποίο ενδιαφέρονται, με απώτερο σκοπό να παραλάβουν πληροφορίες για αυτό. Τέλος για να μπορέσει να γίνει εφικτό αυτό το μοντέλο επικοινωνίας χρειαζόμαστε ακόμα ένα κομμάτι :τον εξυπηρετητή γεγονότων. Ο ρόλος του είναι να συλλέγει τα μηνύματα που δημοσιεύονται από τις εφαρμογές τις κατηγορίας Α, να αναγνωρίζει το περιεχόμενο του κάθε μηνύματος χρησιμοποιώντας εκφράσεις(φίλτρα) και τέλος να μεταδίδει τα μηνύματα σε όποιες από τις εφαρμογές τις κατηγορίας Β είναι εγγραμμένες (έχουν ενεργή συνδρομή).Τέλος οι βασικές ενέργειες που υλοποιεί το σύστημα είναι το subscription() που εκτελείτε όταν ένας πελάτης θέλει να εγγραφή στον εξυπηρετητή γεγονότων και το publish() κατά το οποίο ένας πελάτης δημοσιεύει ένα μήνυμα. Στο Σχήμα 1 μπορούμε να δούμε μια εικονική αναπαράσταση του συστήματος που μόλις περιγράψαμε[3] .



Σχήμα 1 :Απεικόνιση συστήματος δημοσίευσης/συνδρομής .

Αν κατανοήσουμε την λειτουργία που παραπάνω συστήματος μπορούμε εύκολα να συμπεράνουμε και τα πλεονεκτήματα που προκύπτουν. Αρχικά δεν επιβάλουμε κανένα περιορισμό σε σχέση με τον συγχρονισμό των εφαρμογών αφού οι δημοσιευτείς όταν έχουν ένα καινούργιο μήνυμα σχετικά με ένα γεγονός το στέλνουν στον εξυπηρετητή γεγονότων και έπειτα μπορούν να συνεχίσουν την λειτουργία τους με οποιαδήποτε άλλη εργασία , γνωρίζοντας ότι ο εξυπηρετητής θα αναλάβει πλέον την δουλειά να το μεταδώσει στους ενδιαφερόμενους συνδρομητές. Επίσης, με την ίδια λογική και οι συνδρομητές δηλώνουν το ενδιαφέρον τους , συνεχίζουν στην λειτουργία τους περιμένοντας παράλληλα κάποια στιγμή να λάβουν από τον εξυπηρετητή μήνυμα/μηνύματα για το γεγονός/γεγονότα που ενδιαφέρονται.

Βασικό παράγοντα για την δημιουργία ενός συστήματος που να μπορεί να λειτουργήσει ικανοποιητικά στο περιβάλλον που έχουμε περιγράψει παραπάνω είναι η αρχιτεκτονική με την οποία θα στηθεί ο εξυπηρετητής γεγονότων .Αναλυτικότερα αντί να έχουμε έναν εξυπηρετητή που να αναλαμβάνει να χειριστεί μόνο αυτός όλο το πλήθος των δημοσιευτών/συνδρομητών μπορούμε

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

να χρησιμοποιήσουμε δύο η και περισσότερους έτσι ώστε να κατανέμετε ο φόρτος εργασίας και με αυτόν τον τρόπο να είναι εφικτό να εξυπηρετηθούν σε λογικό χρονικό περιθώριο ένα μεγάλο πλήθος πελατών .Έτσι καταλήγουμε στην χρήση και στην άμεση συνεργασία μεταξύ των εξυπηρετητών του διαδικτύου που όμως προσθέτει μια σειρά από καινούργια προβλήματα όπως ο τρόπος που θα συνεργάζονται-επικοινωνούν οι εφαρμογές (brokers) μεταξύ τους , σε ποιες από αυτές θα επιτρέπετε η εγγραφή – δημοσίευση και ποιες θα παίζουν τον ρόλο του μεσολαβητή, πώς θα γίνεται η μετάδοση των μηνυμάτων έτσι ώστε να μεταδίδονται μόνο σε όσους ενδιαφέρονται κ.α .

Στην βιβλιογραφία δεν υπάρχει ένας και μόνο τρόπος για την επίλυση των παραπάνω ερωτημάτων πράγμα που βεβαιώνετε από το πλήθος των εναλλακτικών τρόπων αντιμετώπισης που έχουν προταθεί. Συνεπώς θα εξετάσουμε διάφορους τρόπους χωρισμού του εξυπηρετητή γεγονότων σε εφαρμογές(brokers) και την ιεραρχία τους, τεχνικές δρομολόγησης και αναγνώρισης μηνυμάτων .

Ένας σημαντικός περιορισμός των συστημάτων δημοσίευσης/συνδρομής είναι ότι κάθε μήνυμα που δημοσιεύετε θα φτάσει σε όλους τους ενδιαφερόμενους συνδρομές που είναι ενεργοί εκείνη την χρονική στιγμή. Παραδείγματος χάρη ένας πελάτης κάνει μία εγγραφή σε ένα γεγονός και μετά από ένα χρονικό διάστημα αποχωρεί από το σύστημα, πράγμα που μπορεί να συμβεί σε ένα δυναμικό περιβάλλον όπου πελάτες φεύγουν και έρχονται χωρίς κανένα περιορισμό, και ξανασυνδέεται αμέσως μετά από μία δημοσίευση που τον ενδιέφερε και όπως είναι διαμορφωμένο το σύστημά μας δεν θα παραλάβει ποτέ. Συμπερασματικά στο μοντέλο pub/sub δεν παρέχεται η δυνατότητα ανάκλησης παλαιότερων μηνυμάτων τα οποία ταιριάζουν με την συνδρομή πελατών που δεν ήταν ενεργεί κατά την δημοσίευσή τους. Αυτό που μπορούμε να κάνουμε για να προσθέσουμε την λειτουργικότητα της ανάκλησης παλαιότερων μηνυμάτων είναι να κάνουμε το μοντέλο μας ικανό να «θυμάται» μηνύματα που δημοσιεύτηκαν στο παρελθόν περισσότερες πληροφορίες θα ακολουθήσουν στην **[ΕΝΟΤΗΤΑ 3]** .

Εδώ αξίζει να αναφέρουμε ότι μία τεχνική προσωρινής αποθήκευσης μνημάτων, με βάση το περιεχόμενο το οποίο μεταφέρουν, στους κόμβους που το αναμεταδίδουν θα είχε μεγάλη εφαρμογή στο διαδίκτυο και πιο συγκεκριμένα στο σύστημα επικοινωνίας Server-client.Διότι όταν ένας κόμβος αναζητά μία πληροφορία θα ρωτάει πρώτα τους γειτονικούς του κόμβους και εάν δεν το είχαν στην μνήμη αναδρομικά θα προωθούσαν το ερώτημα στους δικούς τους γείτονες μέχρι την χειρότερη περίπτωση να φτάσει στην πηγή της πληροφορίας (Server). Σε όλες τις άλλες περιπτώσεις θα είχαμε κέρδος και σε χρόνο και σε πόρους .Αν επίσης λάβουμε υπόψη μας ότι κατά το μεγαλύτερο ποσοστό τα ερωτήματα που κάνουν οι πελάτες αντιστοιχούν σε όμοια περιεχόμενα μηνυμάτων καταλαβαίνουμε πόσο ελκυστική είναι μια τέτοια προσέγγιση. Ήδη έχει γίνει μια τέτοια υλοποίηση που στο μέλλον ίσως αλλάξει όλη την δομή του διαδικτύου [12]. Παρακάτω **[ΕΝΟΤΗΤΑ 4]** **Θ**α ασχοληθούμε εκτενέστερα με αυτό το σύστημα επικοινωνίας.

Η πρώτη προσπάθεια υλοποίησης της λειτουργίας της ανάκλησης έγινε με την τοποθέτηση μιας βάσης δεδομένων χωρισμένης σε κομμάτια σε καθορισμένες θέσεις στο σύστημα με δυνατότητα επικοινωνίας με διάφορους brokers, στην οποία αποθηκεύονται τα μηνύματα που δημοσιεύονται με βάση το θέμα τους (topic based προσέγγιση)[1].Έτσι όταν ένας πελάτης εγγραφεί στο σύστημα μπορεί να κάνει μία ερώτηση στην βάση δεδομένων για παλαιότερα μηνύματα με το ίδιο θέμα που είχε η συνδρομή του .Η προσέγγιση αυτή στηρίζεται σε μεγάλες αποθηκευτικές μονάδες που θα πρέπει να τοποθετηθούν σε σταθερά σημεία πράγμα πού δεν είναι και τόσο χρήσιμο αν

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

αναλογιστεί κανείς ότι και οι brokers υλοποιούνται πάνω από ένα δυναμικό περιβάλλον οπότε η εξ αρχής τοποθέτηση πιθανόν μην είναι η ιδανική με αποτέλεσμα να χάνουμε αρκετό χρόνο για την απάντηση και από την άλλη αυξάνετε και το κόστος υλοποίησης.

Μια καλύτερη τεχνική είναι η τοποθέτηση μνήμης σε κάθε broker έτσι ώστε να μπορούν αυτοί να αποθηκεύουν τα μηνύματα που δημοσιεύονται .Στόχος είναι να βρούμε εάν θα χρησιμοποιήσουμε όλους τους brokers ως σημείο αποθήκευσης ή ένα υποσύνολο τους και με πια κριτήρια θα γίνει η επιλογή[5],[4]. Επίσης στην περίπτωση που γεμίσει η μνήμη ενός broker ποιο από αυτά θα αντικατασταθεί ,θα έχουμε μηχανισμούς που θα μας ξεχωρίζουν ένα δημοφιλές μήνυμα από ένα που δεν είναι; Επιπρόσθετα θα πρέπει να καθορίσουμε τον τρόπο με τον οποίο θα γίνεται η ανάκληση των παλαιών μηνυμάτων. Το ερώτημα ανάκλησης των πελατών που κάνουν την συνδρομή θα μεταδίδεται σε όλους τους brokers ή μόνο σε συγκεκριμένους και ποιοι θα είναι αυτοί; Μια αναλυτικότερη περιγραφή αυτού του επικοινωνιακού συστήματος και οι απαντήσεις σε αυτά τα ερωτήματα θα εξεταστούν εκτενέστερα στην συνέχεια **[ΕΝΟΤΗΤΑ 3]**.

Για την καλύτερη κατανόηση της εργασίας μας παρακάτω θα αναφέρουμε σύντομα ορισμένα βασικά στοιχεία που σχετίζονται με το κατανεμημένο σύστημα που μελετάμε .

2.1 Μοντέλα δημοσίευσης/συνδρομής

Οι συνδρομητές όπως έχουμε αναφέρει ενδιαφέρονται για ένα ή περισσότερα γεγονότα και όχι για κάθε γεγονός το οποίο προωθούν στο σύστημα οι δημοσιευτείς. Για να μπορέσει να επιτευχθεί αυτός ο διαχωρισμός υπάρχουν μηχανισμοί ταυτοποίησης γεγονότων .Επομένως οι διαφορετικοί τρόποι ταυτοποίησης ενός μηνύματος έχουν εισάγει και διαφορετικά μοντέλα δημοσίευσης/συνδρομής με ποιό ευρεία διαδεδομένα να είναι τα:

2.1.1 Topic base Publish/subscribe

Από τα πρώτα μοντέλα δημοσίευσης/συνδρομής που δημιουργήθηκαν είναι αυτό που βασίζεται στο θέμα(Topic base).Κάθε γεγονός αντιστοιχεί σε ένα μοναδικό θέμα(topic or subject) με ένα μοναδικό όνομα το οποίο είναι συνήθως ένα αλφαριθμητικό(string).Έτσι χρησιμοποιώντας αυτήν την ταυτοποίηση οι πελάτες δημοσιεύουν ή εγγράφονται σε ένα μήνυμα που ανήκει σε ένα από τα θέματα που υπάρχουν. Αυτή η υλοποίηση είναι παρόμοια με το σύστημα επικοινωνίας των ομάδων(groups communication). Αυτό οφείλεται στο γεγονός ότι τα πρώτα συστήματα που παρείχαν την δυνατότητα αλληλεπίδρασης με το σύστημα δημοσίευσης/συνδρομής βασίζονταν σε ένα επεκταμένο σύστημα groups communication[10]. Πιο συγκεκριμένα κατά την διαδικασία της συνδρομής, έστω του θέματος Α μεταφραζόταν ως την δημιουργία ενός καινούριου group Α, ενώ η δημοσίευση ενός μηνύματος που ανήκει στο θέμα Α αντιστοιχεί στην μετάδοση του μηνύματος στα μέλη του group Α .Αυτό το σύστημα επικοινωνίας έχει πραγματοποιηθεί στο Isis [11] group communication toolkit .Τώρα ως προς την υλοποίηση θεωρούμε ότι έχουμε έναν ξεχωριστό εξυπηρετητή γεγονότων για κάθε topic .Έτσι ένας publisher θα κάνει publish() στον εξυπηρετητή γεγονότων που αντιστοιχεί στο θέμα που δημοσιεύει και παρόμοια και ο subscriber.Τέλος επειδή μπορεί να χρειαστεί να δημιουργηθούν πολλά topic και ενδεχομένως μερικά από αυτά να αναφέρονται σε κοινά θέματα χρησιμοποιούμε την ιεραρχική οργάνωση θεμάτων έτσι ώστε να

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

μειώσουμε τον αριθμό τους .Σύμφωνα με αυτήν κατά την εγγραφή ενός πελάτη σε έναν κόμβο στο θέμα με την υψηλότερη ιεραρχία περιλαμβάνονται ταυτόχρονα και εγγραφές σε όλα τα επιμέρους θέματα αυτού του κόμβου. Τέλος να αναφέρουμε ότι αρκετά συστήματα όπως το TIBCO Rendezvous [9] προσφέρει την δυνατότητα ένας πελάτης να κάνει εγγραφή η δημοσίευση σε ονόματα θεμάτων που να ταιριάζουν με ένα συγκεκριμένο σύνολο λέξεων-κλειδιών όπως ένα υποδέντρο ή σε ένα συγκεκριμένο επίπεδο της ιεραρχίας .

2.1.2 Content base Publish/subscribe

Ακόμα και το ιεραρχικό μοντέλο του topic base μας παρέχει περιορισμένη λειτουργικότητα στο να κατατάσσουμε γεγονότα μόνο και μόνο από το όνομα του θέματος του ανήκουν. Έτσι το content base διαχωρίζει τα γεγονότα με βάσει το περιεχόμενο τους την στιγμή που το επεξεργάζεται και όχι με βάση προκαθορισμένα κριτήρια (πχ το όνομα του θέματος) .Τέτοια συστήματα που μεταφέρουν γεγονότα και χρησιμοποιούν δομές δεδομένων ώστε να προσδώσουν ιδιότητες και γνωρίσματα σε αυτά για να μπορούν αργότερα να τα ξεχωρίζουν είναι το SIENA [7] REDs[8]. Για να γίνει εφικτός ο διαχωρισμός των γεγονότων ο subscriber κατά την εγγραφή του παρέχει στο σύστημα ένα φίλτρο έτσι ώστε να επιστραφούν σε αυτόν όσα γεγονότα ταυτίζονται με το εν λόγο φίλτρο. Η ταύτιση γίνεται με σύγκριση γνωρισμάτων όπως τύπος, όνομα και κάποιες τιμές που χαρακτηρίζουν τα διάφορα γεγονότα με τις τιμές που έχει το φίλτρο. Για την σύγκριση των γνωρισμάτων χρησιμοποιούνται πράξεις σύγκρισης (πχ =,>,<) και ανάλογα με το αποτέλεσμα μπορούμε να αποφανθούμε εάν ένα γεγονός ταιριάζει με το φίλτρο που έχουμε. Τέλος ο subscriber μπορεί να δώσει στο σύστημα ειδοποίησης περισσότερα από ένα φίλτρα έτσι ώστε να καλύπτετε ένα μεγαλύτερο εύρος γεγονότων (pattern) [6].

2.1.3 Type base Publish/subscribe

Ένας άλλος τρόπος διαχωρισμού είναι αντί να ελέγχουμε το θέμα ή το περιεχόμενο του γεγονότος να φιλτράρουμε ανάλογα με την δομή, δηλαδή το τύπο του event.Αυτή η τεχνική μας παρέχει μια ποιο φυσική αναπαράσταση των φίλτρων σε σχέση με το content base.

Ανακεφαλαιώνοντας συμπεραίνουμε ότι έχουν σχεδιαστεί διάφορα συστήματα δημοσίευσης/συνδρομής τα οποία προσφέρουν το κάθε ένα διαφορετικό βαθμό εκφραστικότητας έτσι ώστε να μπορούν να γεγονότα να κατατάσσονται. Το topic base σύστημα δημοσίευσης/συνδρομής είναι στατικό και απλό αλλά παρόλα αυτά μπορεί να υλοποιηθεί αρκετά αποδοτικά .Από την άλλη πλευρά το content base έχει σε μεγάλο βαθμό εκφραστικότητας που μας επιτρέπει να χειριζόμαστε τα γεγονότα με μεγάλη ευχέρεια ,όμως απαιτεί πολύπλοκα πρωτόκολλα τα οποία προσθέτουν υψηλό φόρτο εργασίας κατά την εκτέλεσής τους .Έτσι ανάλογα με το τι εφαρμογή θέλουμε να υλοποιήσουμε θα επιλέξουμε και τον κατάλληλο μοντέλο. Παράδειγμα στην περίπτωση που θέλαμε να ένα σύστημα για stock ερωτήσεις/απαντήσεις τότε επειδή έχουμε ένα περιορισμένο σύνολο πιθανόν τιμών θα προτιμήσουμε το type base .

Παρακάτω θα περιγράψουμε την τοπολογία των servers δηλαδή τον τρόπο με τον οποίο συνδέονται αλλά και το είδος της σύνδεσης που καθορίζεται από το πρωτόκολλο που χρησιμοποιεί.

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

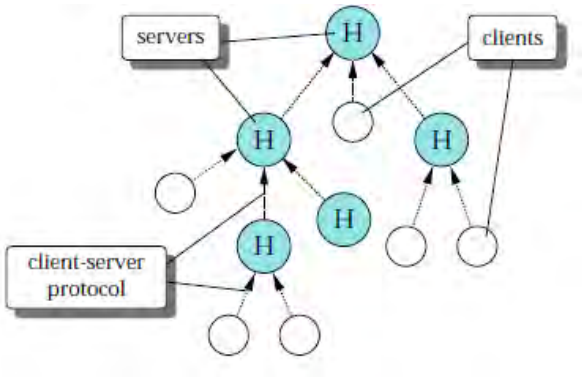
2.2 Αρχιτεκτονική: τοπολογία των servers και πρωτόκολλα

Όπως έχουμε αναφέρει και ποίο πάνω ο ρόλος του pub/sub συστήματος είναι να παρέχει την δυνατότητα στους δημοσιευτείς να στέλνουν τα μηνύματα τους ασύγχρονα στους συνδρομητές που έχουν κάνει εγγραφή. Για να το πετύχουμε αυτό χρησιμοποιούμε μια ενδιάμεση οντότητα τον εξυπηρετητή γεγονότων (event server) ο οποίος συλλέγει τα μηνύματα από τους παραγωγούς και τα στέλνει στους καταναλωτές που ενδιαφέρονται .Επειδή όμως οι σημερινές εφαρμογές έχουν να αντιμετωπίσουν ένα μεγάλο και συνεχώς αυξανόμενο διαδικτυακό χώρο είχε ως συνέπεια την δημιουργία ενός event server ο οποίος αποτελείτε από πολλούς ομότιμους servers τους λεγόμενους μεσολαβητές(brokers).Τώρα οι τοπολογία και ο τρόπος με τον οποίο επικοινωνούν οι brokers έτσι ώστε να επιτευχθεί η προκαθορισμένη λειτουργικότητα ονομάζετε αρχιτεκτονική του event server[7],[3] .

Από αρχιτεκτονικές που έχουν προταθεί εμείς θα αναφέρουμε περιληπτικά τρις βασικές :Την ιεραρχική την ισότιμη και την υβριδική .Δεν θα ασχοληθούμε με την κεντικοποιημένη αρχιτεκτονική που χρησιμοποιεί έναν μόνο broker διότι παρόλο που είναι απλή στην υλοποίηση και παρέχει πολύ καλή συνέπεια λειτουργικότητας δεν κλιμακώνει.

2.2.1 Ιεραρχική αρχιτεκτονική

Η τοπολογία των servers σε αυτήν την αρχιτεκτονική έχουν την διάταξή ενός γραφήματος δέντρου όπως φαίνετε και στο Σχήμα 1.Κάθε κόμβος (broker) είναι σημείο σύνδεσης πελατών και επίσης κάθε κόμβος έχει εάν 'master' broker.Ο broker που δεν έχει ΄master΄ είναι ο root broker.Έτσι έχουμε δύο ειδών συνδέσεις αυτές των πελατών με τους servers και τις συνδέσεις μεταξύ των servers.Παρόλα αυτά και στα δύο είδη συνδέσεων χρησιμοποιείται το Server-Client πρωτόκολλο .



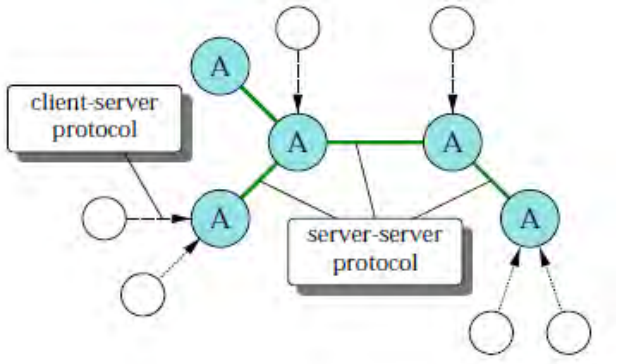
Σχήμα 1: Ιεραρχική τοπολογία εξυπηρετητών

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

Τώρα όταν ένας broker δεχτεί ένα subscription από έναν πελάτη που είναι συνδεδεμένος σε αυτόν πρέπει να το μεταδώσει στον master broker ο οποίος με την σειρά του θα κάνει το ίδιο μέχρι τελικά να φτάσει στον root broker.Αν δεχτεί ένα publish γεγονός τότε το μεταδίδει σε όποιον από τους πελάτες(σε αυτήν την διαδικασία πελάτες θεωρούνται και οι servers που έχουν master των broker που δέχεται το publish) των ενδιαφέρει και σίγουρα το μεταδίδει και στον master broker που έχει. Συμπεραίνουμε δηλαδή ότι ο κάθε broker αποτελεί ταυτόχρονα server για τους πελάτες που έχει και client για τον master broker για αυτό τον λόγο όπως είπαμε χρησιμοποιείτε το ίδιο πρωτόκολλο. Αυτή η υλοποίηση παρόλο που καταφέρνει να κατανέμει τον μεγάλο όγκο αποθήκευσης των στοιχείων των πελατών σε όλους τους brokers δημιουργεί μεγάλη συμφόρηση στους 'ψηλά' ιεραρχικά brokers μιας και όλες οι ενέργειες των πελατών καταλήγουν στον root broker .Επιπλέον σημαντικό πρόβλημα θα δημιουργηθεί σε περίπτωση που ένας broker πάθει μια βλάβη διότι θα κοπεί η επικοινωνία με τους brokers που συνδέονταν με αυτόν.

2.2.2 Ισότιμη(Peer to Peer) αρχιτεκτονική

Σε αυτήν την υλοποίηση όλοι οι κόμβοι που αποτελούν τον server event notification παίζουν ακριβώς τον ίδιο ρόλο έχουν δηλαδή την ίδια λειτουργικότητα (βλέπεται Σχήμα 2).Έτσι όταν ένας από αυτούς συνδεθεί με έναν πελάτη ο οποίος κάνει μια ενέργεια (subscription,advertisement, event) τότε ανάλογα με το πρωτόκολλο δρομολόγησης μεταδίδετε και τους κατάλληλους brokers .Για αυτόν τον λόγο έχουμε και δύο ειδών πρωτόκολλα ένα server-client που είναι ανάμεσα στον πελάτη και στον broker που θα συνδεθεί και το άλλο ανάμεσα στους brokers το Server-Server.Τέλος ανάλογα με της συνδέσεις που έχουν οι μεσολαβητές μπορούμε να χωρίσουμε το peer to peer σε δυο κατηγορίες το:



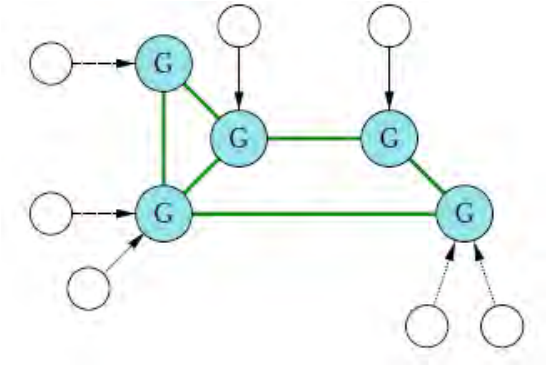
Σχήμα 2: Ισότιμη τοπολογία εξυπηρετητών.

Acyclic peer to peer στο οποίο κάθε server συνδέετε μόνο με έναν server και σε αυτήν την κατηγορία παραμένει το πρόβλημα που είχε και η ιεραρχική αρχιτεκτονική μια και σε περίπτωση βλάβης ενός server αποκόπτετε μέρος των brokers.

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

Και το general peer το peer στο οποίο κάθε server μπορεί να συνδεθεί με οποιοδήποτε άλλον(Σχήμα 3). Εδώ παρόλο που λύνουμε το πρόβλημα αποτυχίας ενός server εισάγουμε ένα καινούργιο , θα πρέπει να βρούμε ένα πρωτόκολλο δρομολόγησης των μηνυμάτων έτσι ώστε να αποφεύγονται οι κύκλοι που μπορούν να προκύψουν .

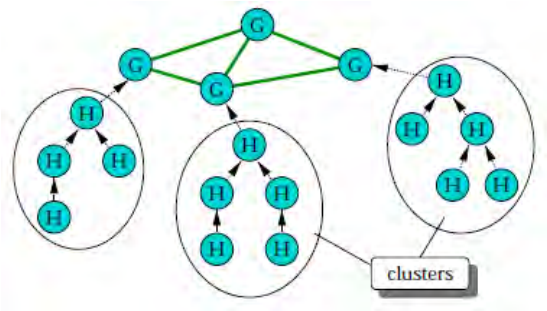
Φυσικά όποιες από τις δύο κατηγορίες και να υιοθετήσει κάποιος δεν πρόκειται να αλλάξει κάτι στην λειτουργικότητα με την οποία συμπεριφέρονται οι servers.



Σχήμα 3 : Γενικό γράφημα ισότιμης τοπολογίας εξυπηρετητών.

2.2.3 Υβριδική αρχιτεκτονική

Σε περίπτωση που απαιτείτε η δημιουργία ενός event server που προορίζετε για εξυπηρέτηση ενός πολύ μεγάλου αριθμού πελατών σε ευρεία χρήση ,τότε οι αρχιτεκτονική των brokers είναι προτιμότερο να χωριστεί σε επίπεδα(Σχήμα 4). Ποιο συγκεκριμένα servers που βρίσκονται μαζί σε τοπικό επίπεδο μπορούν να θεωρηθούν ως μια ομάδα (cluster) αποτελώντας το χαμηλό επίπεδο και να έχουν ιεραρχική αρχιτεκτονική .Αυτό είναι λογικό εφόσον τα μηνύματα που θα δημιουργούνται σε αυτό το cluster κατά πάσα πιθανότητα θα αφορούν το ίδιο οπότε δεν συντρέχει λόγος να ενημερώνονται για κάθε μήνυμα όλοι οι brokers.Επίσης θα χρειαστούν και brokers που θα συνδέουν τα διάφορα cluster μεταξύ τους(gateway brokers).Επειδή οι brokers αυτοί θα αποτελούν το υψηλό επίπεδο στης διάταξης χρειαζόμαστε μια ευέλικτη αρχιτεκτονική με την οποία θα



Σχήμα 4 : Υβριδική τοπολογία εξυπηρετητών.

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

συνδέονται και αυτή είναι η είναι general peer to peer. Τέλος οι gateway brokers θα πρέπει να είναι σε θέσει να εντοπίζουν μηνύματα που προέρχονται η προορίζονται από η προς τα cluster έτσι ώστε να προσαρμόζουν και το κατάλληλο πρωτόκολλο προώθησης που καθορίζει η αρχιτεκτονική που έχουν υιοθετήσει .Μία παρόμοια υλοποίηση ενός event server είναι μια πολύπλοκή διαδικασία σε σχέση με τις άλλες αρχιτεκτονικές όμως από την στιγμή που καταφέρεις και χωρίσεις αποδοτικά τους brokers και εφαρμόσεις και τα κατάλληλα για την περίσταση πρωτόκολλα θα έχεις ένα πολύ ποιο αποδοτικό σύστημα .

2.3 ΑΛΓΟΡΙΘΜΟΙ ΔΡΟΜΟΛΟΓΗΣΗΣ ΚΑΙ ΣΤΡΑΤΗΓΙΚΕΣ ΕΠΕΞΕΡΓΑΣΙΑΣ

Εφόσον ορίσουμε την αρχιτεκτονική που θα έχει ο event server μπορούμε πλέον να ασχοληθούμε με τον τρόπο που ένα μήνυμα που δημοσιεύεται θα φτάσει τελικά μέσο των brokers στους πελάτες που έχουν κάνει εγγραφή και σε μόνο αυτούς .Το πρόβλημα είναι ποια μονοπάτια από τον γράφο των brokers θα ακολουθήσει η ειδοποίηση έτσι ώστε να την λάβουν όλοι οι ενδιαφερόμενοι και χωρίς να σταλούν μηνύματα σε brokers που δεν ενδιαφέρονται έτσι ώστε να κάνουμε καλή διαχείριση πόρων[7]. Την απάντηση σε αυτό το ερώτημα έρχεται να το λύσει ένα πλήθος από αλγόριθμους δρομολόγησης .

Ένας πολύ απλός τρόπος είναι όταν ένα subscription γίνει σε έναν broker να αποθηκεύεται εκεί και όταν συμβεί ένα οποιοδήποτε publish γεγονός να διαδίδετε σε όλο το δίκτυο των brokers με την ελπίδα να υπάρχει τουλάχιστον ένας που να ενδιαφέρετε για αυτό .Όμως όπως μπορούμε να καταλάβουμε σε περίπτωση που έχουμε να εξυπηρετήσουμε πολλούς πελάτες ο αριθμός των μηνυμάτων ειδοποίησης θα είναι απαγορευτικός(σκεφτείτε πως για κάθε δημοσίευση θα πρέπει να σταλεί ειδοποίηση σε όλους τους brokers του συστήματος ακόμα και αν δεν υπάρχει κανένας πελάτης που να ενδιαφέρετε για αυτήν).

Μια άλλη τεχνική που έχει προταθεί και βελτιώνει λίγο την προηγούμενη που περιγράψαμε είναι: όταν γίνετε μία εγγραφή σε έναν μεσολαβητή να προωθείτε αυτή στους μεσολαβητές που είναι σημεία πρόσβασης πελατών έτσι ώστε αν κάποιος δημοσιευτής κάνει publish ένα γεγονός για το οποίο δεν υπάρχει στο σύστημα πελάτης που να ενδιαφέρεται τότε ο μεσολαβητής μπορεί να το γνωρίζει, εφόσον δεν θα του έχει μεταδοθεί καμία τέτοια εγγραφή και έτσι δεν θα το διαδώσει σε κανένα άλλον μεσολαβητή .Είναι μια τεχνική που ‘μπλοκάρει’ την διάδοση γεγονότων που δεν ενδιαφέρουν κανένα από τους μεσολαβητές που τα εισάγουν στο σύστημα .Όμως και αυτή η τεχνική έχει αυξημένο φόρτο γιατί παρόλο που κόβει τα γεγονότα που δεν ενδιαφέρουν κανένα πελάτη δεν μπορεί να παραδώσει τα γεγονότα στους μόνο στους μεσολαβητές που έχουν πελάτες να ενδιαφέρονται για αυτά. Για αυτό τον λόγο θα εξετάσουμε παρακάτω ποιο ενδιαφέρουσες λύσεις .

Το κύριο χαρακτηριστικό των στρατηγικών δρομολόγησης που θα περιγράψουμε είναι η προσπάθεια εύρεσης του ποιο σύντομου μονοπατιού που οδηγεί μόνο στους brokers που έχουν πελάτη/πελάτες που να ενδιαφέρονται για το μήνυμα που δημοσιεύτηκε.

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

2.3.1 Subscription forwarding

Η τεχνική που εφαρμόζετε για να πετύχουμε τον στόχο που μόλις είπαμε είναι αντί ο broker στον οποίο γίνετε μια εγγραφή να την αποθηκεύει μόνο αυτός να την προωθεί σε όλους τους άλλους brokers του συστήματος [7]. Οπότε σε οποιοδήποτε server και αν φτάσει το μήνυμα της ειδοποίησης μπορεί, χρησιμοποιώντας της καταχωρίσεις του πίνακα ST(subscription table) και τον αλγόριθμο minimum spanning tree, να βρει τα συντομότερα μονοπάτια προς τους brokers που έχουν πελάτη που ενδιαφέρετε για το μήνυμα που δημοσιεύτηκε, εφόσον πλέον όλοι οι brokers γνωρίζουν ποιοι από αυτούς ενδιαφέρονται για ποία μηνύματα .

Πιο αναλυτικά όταν ένας πελάτης κάνει μία εγγραφή σε έναν μεσολαβητή τότε αυτός καταχωρεί το αναγνωριστικό του πελάτη και το φίλτρο που ταυτίζει τα γεγονότα που τον ενδιαφέρουν στο ST .Έπειτα ο ίδιος ο μεσολαβητής παίζει πλέον τον ρόλο του πελάτη και διαδίδει την εγγραφή στους γείτονές τους. Αυτοί με την σειρά τους καταχαρούν το αναγνωριστικό του μεσολαβητή του τους έστειλε την εγγραφή και το φίλτρο στο δικό τους ST.Αυτή η διαδικασία συνεχίζει αναδρομικά μέχρις ότου όλοι οι μεσολαβητές ενημερωθούν για αυτήν την εγγραφή .Τώρα όταν γίνει σε έναν μεσολαβητή μια δημοσίευση ενός γεγονότος τότε με βάση το ST που έχει βρίσκει αν έχει κάποιον πελάτη-μεσολαβητή του να 'ταιριάζει' το γεγονός αυτό με το φίλτρο που είχε στείλει .Αν ναι τότε προωθεί το γεγονός σε αυτόν αν όχι δεν κάνει καμία απολύτως ενέργεια .Έτσι με τον τρόπο αυτόν το γεγονός θα φτάσει μόνο σε όσους μεσολαβητές ενδιαφέρονται για το γεγονός αυτό, στους υπόλοιπους δεν θα μεταδοθεί ποτέ .Κατά αυτόν τον τρόπο οι διαδρομές των γεγονότων προς τους πελάτες που ενδιαφέρονται για αυτά και μόνο σε αυτούς ,τίθενται από την διάδοση της εγγραφής για αυτό και ο αλγόριθμος αυτός έχει πάρει την ονομασία subscription forwarding.

Τέλος να προσθέσω ότι στον αλγόριθμο subscription forwarding χρησιμοποιώντας την τεχνική της επικάλυψης των subscriptions μπορούμε να μειώσουμε των αριθμό μετάδοσης αυτών σε ορισμένους brokers [2].Ένας εύκολος τρόπος για να το καταλάβουμε αυτό είναι να σκεφτούμε ότι ένας broker θα μετέδιδε ένα subscription στους γείτονές του μόνο εάν δεν έχει κανένα άλλο subscription που να επικαλύπτει το τρέχον. Κατά αυτόν τον τρόπο μπορούμε να μειώσουμε πολλές μεταδώσεις του subscription στο δίκτυο.. Ένα τέτοιο σύστημα υλοποιείται στο REDS[8] .

2.3.2 Advertisement forwarding

Σε αυτόν τον αλγόριθμο δρομολόγησης η τεχνική που χρησιμοποιείτε μοιάζει αρκετά με αυτή του subscription forwarding [7].Τώρα πλέον έχουμε ένα επιπλέον είδος μηνυμάτων τα advertisements τα οποία διαδίδονται σε όλο το δίκτυο με σκοπό να ενημερωθούν οι brokers για το τι πρόκειται να δημοσιευτεί και από ποιόν broker,είναι μπορούμε να πούμε το αντίστροφο του subscription.Κατά αυτόν τον τρόπο όταν σε έναν server κάνει μια εγγραφή ένας πελάτης τότε αυτός θα μεταδώσει το subscription όχι σε όλους στους servers όπως κάνει στο subscription forwarding αλλά πλέον μόνο σε όσους του είχαν στείλει advertisement δημοσίευσης μηνύματος που να ταιριάζει στην εγγραφή που μόλις του έγινε.

Παρακάτω θα ακολουθήσει μια εκτενές και λεπτομερή περιγραφή του συστήματος δημοσίευσης/συνδρομής. Επίσης θα αναλύσουμε και την επέκταση του συστήματος αυτού προσθέτοντας caching στους μεσολαβητές .

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

ΕΝΟΤΗΤΑ 3 .Εισαγωγή κρυφής μνήμης στο σύστημα Δημοσίευσης/Συνδρομης

3.1 Περιγραφεί λειτουργίας του συστηματος δημοσιευσης/συνδρομης

Έχοντας πλέον περιγράψει τα βασικά σημεία του κατανεμημένου συστήματος που μελετάμε μπορούμε να προχωρήσουμε στην ακριβή περιγραφή της λειτουργίας του συστήματος δημοσίευσης/συνδρομής. Θεωρούμε ότι έχουμε ένα event notification server με ισότιμη αρχιτεκτονική, subscription forwarding ως αλγόριθμο δρομολόγησης και βασισμένο στο content base.Τώρα ως προς την λειτουργικότητα, όταν κάποιος πελάτης θέλει να κάνει μια εγγραφή σε ένα broker, έστω τον ni , του στέλνει ένα μήνυμα της μορφής subscribe(fi,si) το οποίο περιέχει το κατάλληλο φίλτρο fi για την ταυτοποίηση των μηνυμάτων που θα φτάσουν στον broker μαζί με το αναγνωριστικό που έχει ως πελάτης. Έπειτα ο broker θα καταχωρίσει την εγγραφή αυτή στο Subscription Table (ST) όπου εκεί αποθηκεύει όλους τους πελάτες που έχει ο συγκεκριμένος broker μαζί με τα φίλτρα που έχουν στείλει κατά την εγγραφή. Στην συνέχεια θα αντιστραφούν οι ρόλοι και ο broker πλέον θα παίξει τον ρόλο του πελάτη και θα προωθήσει στους γείτονες του το ίδιο μήνυμα subscribe(fi,ni) με το ίδιο φίλτρο αλλά τώρα το δικό του αναγνωριστικό ni.Αυτό θα γίνει αναδρομικά μέχρι όλοι οι brokers που αποτελούν τον event server να ενημερώσουν το ST για την εγγραφή αυτή .Όπως έχουμε είδη αναφέρει για να μειώσουμε τον αριθμό των αναμεταδόσεων του μηνύματος subscribe(fi,ni) μπορούμε πριν ένας broker προωθήσει το μήνυμα να ελέγξει το ST και αν υπάρχει κάποια άλλη εγγραφή που να την καλύπτει (δηλαδή να είναι ειδίκευση μιας άλλης ποιο γενικής εγγραφής) να μην το προωθεί στους γείτονές του. Η αντίστροφη διαδικασία είναι η διαγραφή ενός πελάτη από το σύστημα και αυτό γίνετε παρόμοια με την εγγραφή απλά σε αυτήν την περίπτωση προωθείτε σε όλους τους brokers ένα μήνυμα της μορφής unsubscribe(fi,si), με τον ίδιο τρόπο που περιγράψαμε ποιο πάνω, και οι brokers που έχουν καταχωρίσει στο ST αυτήν την εγγραφή την διαγράφουν[3]. Τέλος μένει να περιγράψουμε την διαδικασία δημοσίευσης ενός μηνύματος. Σύμφωνα με αυτήν ένας πελάτης publisher θα δημοσιεύσει ένα μήνυμα σε κάποιον broker.Ο broker αυτός αφού εξετάσει το ST αν βρει μία εγγραφή του να 'ταιριάζει' με το μήνυμα αυτό τότε προωθεί το μήνυμα στον πελάτη της εν λόγο εγγραφής .Αυτό θα γίνει αναδρομικά μέχρις ότου όλοι οι πελάτες που είναι συνδεδεμένοι με το σύστημα αυτήν την χρονική στιγμή να πάρουν την δημοσίευση. Τώρα σε περίπτωση που δεν υπάρχει καταχωρημένη εγγραφή του να 'ταιριάζει' με το μήνυμα τότε ο broker δεν προωθεί κανένα μήνυμα.

3.2 Σημεία τοποθέτησης των cache

Τώρα πλέον μπορούμε να αρχίσουμε να περιγράφουμε το επεκταμένο pub/sub σύστημα. Αυτό που έχει γίνει είναι η προσθήκη cache στους brokers και η εισαγωγή δύο επιπλέον μηχανισμών επικοινωνίας το request/response έτσι ώστε να κάνουμε στους πελάτες που θα συνδέονται στο σύστημα εφικτή την ανάκληση πληροφοριών που είχαν δημοσιευτεί σε παλαιότερο χρόνο .

Το ενδιαφέρων στοιχείο αυτού του συστήματος είναι ότι οι brokers που θα αποθηκεύουν τα μηνύματα που δημοσιεύονται δεν είναι προκαθορισμένοι από την αρχή αλλά μεταβάλλονται κατά την ροή εκτέλεσης. Το κριτήριο για να είναι ένας broker υποψήφιος προς αποθήκευση (candidate

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

caching broker) για ένα συγκεκριμένο μήνυμα είναι να έχει στο Subscription Table τουλάχιστον έναν πελάτη που εγγραμμένος σε αυτό το μήνυμα. Η διαδικασία αλληλεπίδρασης του μηχανισμού αυτού είναι: κατά την δημοσίευση ενός μηνύματος από έναν publisher αυτό μεταδίδεται σε όλους τους brokers που έχουν πελάτες εγγεγραμμένους στο μήνυμα αυτό και επιπρόσθετα αποθηκεύετε στην cache τους. Έτσι όπως καταλαβαίνουμε μόνο οι υποψήφιοι brokers θα έχουν στο τέλος της διαδικασίας της δημοσίευσης το μήνυμα στην cache τους. Τέλος όταν ένας πελάτης κάνει subscription εκτός από αυτή την ενέργεια θα κάνει και ένα request για τυχόν παλιότερα μηνύματα που είναι αποθηκευμένα στους μεσολαβητές .Ο τρόπος διάδοσης του request είναι όμοιος με τον τρόπο διάδοσης ενός publish μηνύματος ακολουθεί το μονοπάτι των broker που είναι καταχωρημένη στο ST.

Πριν κλείσουμε αυτήν την ενότητα να αναφέρω ότι τον καθοριστικό παράγοντα για το ποιος broker θα έχει την δυνατότητα να κρατάει τα μηνύματα που δημοσιεύονται εξαρτάτε από την πολιτική caching που χρησιμοποιείτε και όχι μόνο από το αν είναι ή όχι candidate caching broker .Οι διάφορες πολιτικές θα περιγραφούν αναλυτικά στην ενότητα 3.5 .

3.3 Μηχανισμοί Request/Response

Οι δύο βασικοί μηχανισμοί για την ανάκληση παλαιότερων μηνυμάτων στο επεκταμένο σύστημα pub/sub είναι το Request() και το Response() [5].

Όταν ένας πελάτης s1 κάνει εγγραφή σε έναν broker εκτός από το μήνυμα subscribe(f1,s1) που έχουμε πει ότι θα στείλει τώρα θα πρέπει να στείλει και ένα ακόμα μήνυμα το request(f1,s1) το οποίο διαδίδεται όπως το publish μήνυμα αλλά κρατάει επιπρόσθετα και το αναγνωριστικό των broker από τους οποίους πέρασε. Συγκεκριμένα όταν το request(f1,s1) φτάσει σε έναν broker ni τότε ελέγχει το ST έτσι ώστε να βρει εάν υπάρχουν brokers που είναι εγγεγραμμένοι ST και το φίλτρο τους 'ταιριάζει' με το f1 .Σε όσους από αυτούς ικανοποιούν αυτήν την συνθήκη θα αναμεταδοθεί το request.Όμως όπως έχουμε πει στο request που θα μεταδώσει ο ni θα προστεθεί και το αναγνωριστικό του δηλαδή θα είναι της μορφής , Request(f1,ni ->s1) . Για παράδειγμα μετά από 3 μεταδώσεις το request θα είχε την μορφή Request(f1,n2->n1->ni->s1). Αυτή η διαδικασία θα εξακολουθεί να γίνετε αναδρομικά μέχρι να μεταδοθεί σε όλους τους candidate brokers. Σε περίπτωση που δεν υπάρχουν άλλοι brokers προς μετάδοση τότε το request αγνοείτε και δεν διαδίδετε σε κανέναν broker.

Όταν φτάσει το request σε έναν candidate broker τότε ψάχνει στην cache για μήνυμα που να ταιριάζει στο φίλτρο f2 .Σε περίπτωση που βρει ένα τέτοιο μήνυμα τότε ξεκινάει η διαδικασία της απάντησης .Το Response() μήνυμα δεν θα προωθηθεί προς όλους τους κόμβους αλλά μόνο σε αυτούς του είχαν μεταδώσει το request() και είχαν προσθέσει το αναγνωριστικό τους. Κατά αυτόν τον τρόπο το response κατευθύνετε κατευθείαν προς τον broker που είχε ξεκινήσει το request και μόνο αυτός θα παραλάβει παλαιότερα μηνύματα .Οι brokers που δέχονται ένα response μήνυμα διαγράφουν το αναγνωριστικό τους από την ακολουθία των αναγνωριστικών που είχε δημιουργηθεί κατά την μετάδοση του request και προωθούν το response στον επόμενο broker που υπάρχει στην απομένουσα ακολουθία. Τέλος να προσθέσω ότι οι candidate broker που είναι στην ακολουθία του response και δεν έχουν το μήνυμα στην cache τους το αποθηκεύουν.

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

Μια βελτίωση της διαδικασίας του response που έχει γίνει για την μείωση των μηνυμάτων που στέλνονται ως απάντηση είναι η διαδικασία απόρριψης μηνυμάτων(multiply responses). Ποιο συγκεκριμένα όπως έχουμε περιγράψει το σύστημά μας μέχρι τώρα, ένα μήνυμα που δημοσιεύεται θα αποθηκευτή σε πολλούς brokers και αυτό έχει ως αποτέλεσμα την δημιουργία πολλαπλών μηνυμάτων response για το ίδιο ακριβώς μήνυμα .Αυτό που θα ήταν ιδανικό θα ήταν να δημιουργείτε μία μόνο απάντηση. Για να μπορέσουμε να το πετύχουμε προστέθηκε ένας μηχανισμός απόρριψης μηνυμάτων .Κάθε candidate broker που δέχεται ένα response μήνυμα ελέγχει αν αυτό είναι ήδη στην cache του .Αν ναι τότε δεν προωθεί το response σε κανένα άλλων broker και το διαγράφη .Αν όχι τότε θα κάνει ακριβώς τις ίδιες ενέργειες που περιγράψαμε ποιο πάνω κατά την άφιξη ενός response μηνύματος σε έναν candidate broker.

Η διαδικασία του response όπως μπορούμε να καταλάβουμε προσθέτει ένα επιπλέον ποσό φόρτου στο σύστημα εξυπηρέτησης το οποίο όμως είναι αναπόφευκτο. Αυτό το διαπιστώνουμε αν σκεφτούμε ότι η cache των brokers είναι περιορισμένη χωρικά άρα δεν μπορούμε να ξέρουμε ακριβώς πόσο χρόνο θα παραμείνει ένα μήνυμα μέσα στις cache των διάφορων brokers που είχαν αποθηκεύσει το μήνυμα. Αυτό σημαίνει πως αν σε κάποια χρονική στιγμή το σύστημα έχει δυο candidate brokers που έχουν αποθηκεύσει ένα μήνυμα που αντιστοιχεί στο φίλτρο που έχουν στείλει οι subscribers τότε δεν είναι σίγουρο πως εάν κάνουνε ένα request σε ένα από τους δυο θα πάρουμε και θετική απάντηση διότι μπορεί να έχει διαγράψει το μήνυμα αυτό ενώ ο άλλος broker όχι. Για αυτό τον λόγο στέλνουμε το request μήνυμα σε όλους τους candidate brokers εις βάρος του αυξημένου φόρτου διάδοσης μηνυμάτων που προσθέτουμε στο σύστημα.

Βέβαια μίας και έχουμε υποθέσει ένα content base σύστημα είναι δυνατών να γίνουν ταιριάσματα μηνυμάτων τα οποία δεν είναι ακριβώς ίδια .Αυτό έχει ως αποτέλεσμα διαφορετικοί μεσολαβητές να έχουν μηνύματα που να ταιριάζουν στο φίλτρο που έχει το request και να μην είναι ίδια οπότε ρωτώντας όλους τους μεσολαβητές να συλλέγω τελικά διάφορα παλαιότερα μηνύματα πράγμα που θα ήταν αδύνατο εάν ρωτούσαμε μόνο έναν broker.Άρα βλέπουμε ότι αν θέλουμε να συλλέξουμε όλα τα παλιότερα μηνύματα ο επιπλέον φόρτος που εισάγουμε στο σύστημα είναι αναγκαίος .

3.4 Priority Policies

Η πολιτική προτεραιότητας καθορίζει σε ποία θέση της τοπικής μνήμης του broker θα τοποθετηθεί ένα καινούργιο μήνυμα και ποιο από αυτά θα διαγραφή στην περίπτωση που είναι γεμάτη.

Υπάρχουν διάφορες πολιτικές με ποιο διαδεδομένη την First in First out (FIFO).Σύμφωνα με αυτήν κάθε μήνυμα που φτάνει για πρώτη φορά θα αποθηκεύεται στην αρχή της μνήμης μετακινώντας όσα υπήρχαν κατά μια θέση δεξιά και κατά συνέπια το τελευταίο μήνυμα, αν υπάρχει, διαγράφετε. Είναι απλό ως προς την υλοποίηση αλλά δίνει την ίδια αξία σε όλα τα μηνύματα χωρίς να λαμβάνει υπόψη πως ένα από αυτά μπορεί να περιέχει πληροφορίες που είναι χρήσιμες σε πολλούς subscribers οπότε καλό θα ήταν να μείνει όσο περισσότερο γίνετε στην μνήμη.

Μια άλλη πολιτική που βελτιώνει την FIFO προσθέτοντας την παράμετρο κατά πόσο δημοφιλή είναι ένα μήνυμα είναι η regenarations και degradations. Χρησιμοποιώντας την όταν ένας broker δεχτεί ένα subscription-request και υπάρχει μήνυμα στην μνήμη που να αντιστοιχεί σε αυτό τότε μετακινούμε το μήνυμα στην αρχή της μνήμης(regenaration). Με αυτόν τον τρόπο μνήματα που δέχονται συχνά request θα βρίσκονται στις πρώτες θέσεις μνήμης έτσι διαγράφοντας από το τέλος

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

διαγράφουμε τα μη δημοφιλή. Αντίστοιχα όταν και ο τελευταίος πελάτης ενός broker κάνει unsubscribe τότε αν υπάρχει μήνυμα που να αναφέρεται σε αυτήν την εγγραφή τότε μετακινείται στο τέλος της μνήμης(degradation) ώστε σε περίπτωση που χρειαστεί να διαγραφεί ένα μήνυμα να είναι ένα που δεν ενδιαφέρει κανέναν από τους πελάτες μας .

Αν θέλαμε να αυξήσουμε ακόμα περισσότερο την απόδοση της μνήμης μας μπορούμε να χρησιμοποιήσουμε μια πολιτική προτεραιότητας που θα διαγράφει το μήνυμα με το μικρότερο ρυθμό request.Η πολιτική αυτή είναι γνωστή ως LFU (Least Frequently Used).Ως ρυθμό request ορίζουμε τον αριθμό των request που αντιστοιχούν σε ένα μήνυμα που έχει η μνήμη, διαιρεμένο με τον χρόνο παραμονής του στην μνήμη. Έτσι ένα μήνυμα που έχει δεχτεί μερικά request σε μικρό χρονικό διάστημα μπορεί να έχει μεγαλύτερο ρυθμό request από ένα άλλο το οποίο έχει δεχτεί πολλά request σε μεγαλύτερο όμως χρονικό διάστημα. Με άλλα λόγια όταν έρθει η στιγμή να διαγράψουμε ένα μήνυμα θα επιλέξουμε εκείνο που και λίγοι πελάτες το ζήτησαν αλλά και είναι την περισσότερη ώρα μέσα στην μνήμη .

Με βάση τον παραπάνω συλλογισμό μπορούμε να ορίζουμε ακριβώς τις ενέργειες που θα πρέπει να κάνει ένας broker που χρησιμοποιεί την πολιτική αυτή.

- Όταν δεχτεί ένα subscribe-request από έναν πελάτη τότε εάν υπάρχει αντίστοιχο μήνυμα του αυξάνει το count_request κατά ένα. Εάν δεν υπάρχει όμως, όπως ξέρουμε θα το αναζητήσει από τους υπόλοιπους brokers θα το τοποθετήσει στην μνήμη κρατώντας τον χρόνο εισαγωγής του start_time=current_time και θέτοντας το count_request=0.
- Όταν και ο τελευταίος πελάτης κάνει unsubscribe τότε σε αντίθεση με της προηγούμενες πολιτικές δεν χρειάζεται να κάνουμε καμία ενέργεια. Αυτό προκύπτει από το γεγονός ότι το μήνυμα μπορεί να έχει μεγάλο ρυθμό request οπότε είναι και πολύ πιθανό να έρθουν πάλι πελάτες που αναζητούν το ίδιο μήνυμα άρα αν το είχαμε βάλει στο τέλος μπορεί να χάναμε ένα σημαντικό μήνυμα. Δηλαδή με αυτόν τον τρόπο δίνουμε ένα χρονικό περιθώριο διαγραφής ενός μηνύματος που δεν έχει πλέον πελάτες ανάλογα με το πόσο δημοφιλές ήταν.
- Τέλος όταν ένας πελάτης κάνει publish ανάλογα με το policy caching που χρησιμοποιούμε οι brokers που θα επιλεχτούν πρέπει να τοποθετήσουν το μήνυμα στην μνήμη κρατώντας τον χρόνο εισαγωγής του start_time=current_time και θέτοντας το count_request=0 .

Στις παραπάνω περιπτώσεις όπου χρειάζεται τοποθέτηση μηνύματος στην μνήμη η διαδικασία είναι οι εξής :Εάν η τοπική μνήμη του broker δεν είναι γεμάτη τότε το μήνυμα αποθηκεύετε στην πρώτη άδεια θέση .Διαφορετικά θα πρέπει, όπως είχα αναφέρει και προηγουμένως συνοπτικά όμως, να βρούμε το μήνυμα με το μικρότερο ρυθμό request χρησιμοποιώντας τις μεταβλητές count_request και start_time του μηνύματος στην μνήμη που έχουμε αποθηκεύσει. Συνεπώς για κάθε μήνυμα υπολογίζουμε το rate_request=count_request/(current_time-start_time) και το μήνυμα με το μικρότερο διαγράφεται και στην θέση του μπαίνει το καινούργιο .

Τέλος για καλύτερη αξιολόγηση και σύγκριση των πολιτικών που αναφέραμε μελετάμε και την LRU (Least Recently Used) κατά την οποία όπως αναφέρουν και τα αρχικά της διαγράφετε το μήνυμα που έχει τον περισσότερο χρόνο να δεχτεί ένα request από πελάτη που να αναφέρετε σε αυτό. Ως προς την υλοποίηση οι ενέργειες που πρέπει να γίνουν είναι όμοιες με αυτές που περιγράψαμε για την LFU το μόνο που αλλάζει είναι ότι τώρα δεν χρειάζεται να μετράμε τον αριθμό των request που δεχτικέ ένα μήνυμα παρά μόνο να ανανεώνουμε τον χρόνο σε κάθε καινούρια προσπέλασή του

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

Έτσι χρειαζόμαστε έναν πίνακας που να κρατάει την χρονική στιγμή που ήρθε το τελευταίο request για κάθε μήνυμα ή τον χρόνο εισαγωγής του σε περίπτωση που μπαίνει στην μνήμη για πρώτη φορά .Στην περίπτωση που θα πρέπει να αντικαταστήσουμε ένα μήνυμα από την μνήμη τότε βρίσκουμε αυτό με το μέγιστο αριθμό της αφαίρεσης του τρέχοντος χρόνου με τον χρόνο τελευταίας ανανέωσης .Δηλαδή το μήνυμα που έχει τον περισσότερο χρόνο να δεχτεί ένα request άρα δεν είναι σημαντικό και δεν θα μας κοστίσει πολύ εάν το διαγράψουμε .

3.5 Caching policy

Ως cache policy ορίσουμε ένα σύνολο κανόνων που μας επιτρέπει να επιλέξουμε ποιοι brokers θα επιλέγονται για να αποθηκεύουν τα μηνύματα που δημοσιεύονται .Ο κυριότερος κανόνας όπως είχα αναφέρει είναι ότι ο μεσολαβητής που θα αποθηκεύει ένα μήνυμα που δημοσιεύετε θα πρέπει να έχει τουλάχιστον έναν πελάτη που να έχει κάνει εγγραφή για αυτή την δημοσίευση [4],[5].Αυτοί οι brokers που ικανοποιούν αυτή την συνθήκη ονομάζονται candidate brokers.

Παρακάτω θα περιγράψουμε τρις βασικές cache policy που έχουν προταθεί.

Α)Αποθήκευση στους ικανούς μεσολαβητές (candidate brokers)

Όταν δεχτεί ένας candidate broker μια δημοσίευση τότε αποθηκεύει το μήνυμα στην κρυφή μνήμη με βάση την place policy που έχει. Όταν ένας candidate broker δεχτεί ένα response μήνυμα και δεν το έχει στην cache του θα πρέπει να το αποθηκεύσει.

Β)Αποθήκευση στους μεσολαβητές των φύλλων(leaf brokers)

Το σύνολο των brokers που πλέον θα αποθηκεύουν τα δημοσιευμένα μηνύματα θα είναι μόνο οι brokers που έχουν τουλάχιστον ένα πελάτη , δεν συμπεριλαμβάνονται ως πελάτης ο μεσολαβητής του προώθησε το subscription.Δηλαδή μόνο οι brokers που αποτελούν τα φύλλα του γράφου και έχουν το λιγότερο έναν πελάτη θα αποθηκεύουν τα μηνύματα .Οι ενδιάμεσοι brokers ακόμα και αν έχουν πελάτες για αυτό το μήνυμα δεν θα ανήκουν στο σύνολο των leaf brokers .

Γ)Αποθήκευση σε όλους τους brokers

Είναι μια πολιτική που δεν υπακούει στο κανόνα ο broker να είναι candidate αλλά χρησιμοποιείτε στην προσομοίωση ως μέτρο σύγκρισης με τις άλλες πολιτικές. Σύμφωνα με αυτήν όλοι οι brokers που υπάρχουν στο σύστημα αποθηκεύουν το μήνυμα που δημοσιεύετε στην τοπική τους cache.Παρόλα αυτά επειδή το μήνυμα το έχουν όλοι οι brokers αυξάνετε το ποσοστό να απαντηθεί ένα request .Σε αντίθεση βέβαια αυξάνετε κατά πολύ και ο φόρτος που θα εισάγει στο σύστηνα μια τέτοια πολιτική caching .Τέλος η αντίθετη πολιτική με αυτήν που μόλις περιγράψαμε είναι η αποθήκευση μόνο σε έναν broker του συστήματος και την ονομάζουμε ατομική αποθήκευση (single caching).

Μέχρι στιγμής έχουμε μελετήσει σε μεγάλο βαθμό το σύστημα επικοινωνίας δημοσίευσης/συνδρομής και το επεκταμένο μοντέλο με εισαγωγή cache στους μεσολαβητές .Στην συνέχεια θα αναλύσουμε το σύστημα Server-client με το οποίο θα ασχοληθούμε σε όλο το υπόλοιπο της εργασίας μας και τελικά θα προσομοιώσουμε αξιολογώντας τα αποτελέσματα που θα προκύψουν.

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

ΕΝΟΤΗΤΑ 4 .Σύστημα επικοινωνίας Client-Server βασισμένο στο σύστημα δημοσίευσης/συνδρομής με κρυφή μνήμη .

4.1 Εισαγωγή

Εξετάζουμε ένα κατανεμημένο σύστημα επικοινωνίας content base με servers,clients και brokers προσπαθώντας να βρούμε πολιτικές αλληλεπίδρασης των οντοτήτων αυτών με των ποιο αποδοτικό τρόπο ως προς κάποιες παραμέτρους όπως την καθυστέρηση(delay) και το επιπλέον φόρτος(overhead).Ο βασικός τρόπος λειτουργίας αυτού του συστήματος είναι: οι πελάτες συνδέονται στους μεσολαβητές και αναζητούν μηνύματα που είναι αποθηκευμένα στους servers που είναι συνδεδεμένοι στο δίκτυο .Οι δύο κύριες διαδικασίες που πραγματοποιούνται είναι η request(“msg”,c1) κατά την οποία ένας πελάτης c1 αναζητεί από το δίκτυο μια πληροφορία με τίτλο “msg” από τον server που έχει αυτό το μήνυμα και το response(c1) κατά το οποίο ο server απαντάει στον πελάτη στέλνοντάς του αυτήν την πληροφορία που του ζήτησε .Προηγουμένως πριν αρχίσει η επικοινωνία Server-client ο Server έχει ενημερώσει το δίκτυο σε ποιόν broker βρίσκετε και για ποιο μήνυμα έχει πληροφορίες με μία διαδικασία όμοια με αυτήν του advertisement forwarding στο σύστημα επικοινωνίας publish-subscription.Λόγο αυτής της διαδικασίας μπορεί το request μόνο με το ‘τίτλο’ του μηνύματος να φτάσει στον κατάλληλο server και μόνο σε αυτόν ακριβώς σε αντιστοιχία με την διαδικασία του subscription.

Παρακάτω θα περιγράψουμε τέσσερες πολιτικές οι οποίες διαφοροποιώντας κυρίως την διαδικασία του request και προσθέτοντας κάποιες επιπλέον λειτουργικότητες στο σύστημα στοχεύουν στην μείωση του delay-overhead που εισάγουν οι δύο διαδικασίες(request,response) στο επικοινωνιακό σύστημα Server-client που μόλις περιγράψαμε. Η πολιτικές αυτές είναι Server , Server-Cache , Server-one hop away και Server-subscription .

4.2 Server σύστημα

Η πρώτη πολιτική, Server,είναι η ποιο απλή και ως προς τον τρόπο λειτουργίας αλλά και ως προς την υλοποίηση .Πιο συγκεκριμένα όταν ένας πελάτης, που είναι συνδεδεμένος σε έναν broker, κάνει ένα request για αναζήτηση μίας πληροφορίας τότε μέσο του AT(Advertisement Table) βρίσκει το μοναδικό μονοπάτι των μεσολαβητών προς τον server του συγκεκριμένου μηνύματος και αρχίζει η αναμετάδοση του request μέχρι να φτάσει τελικά στον Server. Στην συνέχεια ξεκινάει η διαδικασία του response κατά την οποία περνώντας από τους ίδιους brokers που είχε διασχίσει και το request επιστρέφεται η απάντηση στον πελάτη .

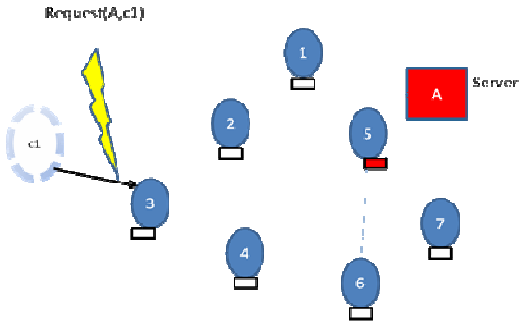
Ο παραπάνω τρόπος επικοινωνίας server-client όπου το request φτάνει από τον πελάτη μέχρι τον server και η απάντηση ακολουθεί το αντίστροφο μονοπάτι δεν είναι αρκετά αποδοτικός .Αυτό το συμπεραίνουμε από το γεγονός ότι κάθε φορά θα πρέπει να διανύεται η διαδρομή client-server και server-client που είναι η μέγιστη σε απόσταση άρα και σε καθυστέρηση(delay), ενώ θα δούμε στην συνέχεια πολιτικές που θα περιορίζουν αυτήν την απόσταση με άμεση συνέπεια την μείωση του delay και του overhead που εισάγει η διαδικασία του response .

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

4.3 Server-cache σύστημα

Ένας τρόπος να μειώσουμε την απόσταση που διανύει το response είναι να δώσουμε την δυνατότητα στους brokers που αποτελούν το δίκτυο να θυμούνται τα μηνύματα που αναμεταδίδουν στους πελάτες. Αυτό μπορεί να γίνει πολύ απλά βάζοντας μια cache σε κάθε broker (Σχήμα 1).Βέβαια δεν είναι αρκετό μόνο αυτό, θα πρέπει να αλλάξουμε και την λειτουργικότητα του συστήματος.

Ποιο συγκεκριμένα όταν συνδεθεί ένας πελάτης σε έναν broker και κάνει ένα request για κάποιο μήνυμα τότε η πρώτη ενέργεια που θα πραγματοποιήσει ο μεσολαβητής είναι ο έλεγχος της τοπικής cache που διαθέτη μήπως και είναι αποθηκευμένο το μήνυμα του ψάχνουμε εκεί. Ο έλεγχος αυτός γίνετε με βάση το περιεχόμενο των μηνυμάτων όπως έχουμε περιγράψει και στο content base σύστημα δημοσίευσης/συνδρομής που σημαίνει ότι θα πρέπει να έχει τα κατάλληλα εργαλεία όπως φίλτρα για κάθε μήνυμα. Στην συνέχεια εάν το μήνυμα υπήρχε τελικά στην cache τότε η διάδοση του request τερματίζεται και εκκινείτε η διαδικασία του response.Αν όμως δεν υπάρχει τότε το request θα πρέπει να αναμεταδοθεί βρίσκοντας με βάση έναν πίνακα σαν το AT(Advertisement Table) ,το μοναδικό μονοπάτι από brokers που οδηγεί στον server.Αυτός ο έλεγχος και ανάλογα οι αναμεταδώσεις γίνονται αναδρομικά μέχρι να βρεθεί η απάντηση σε κάποιον broker πριν τον server η στην χειρότερη στον server.

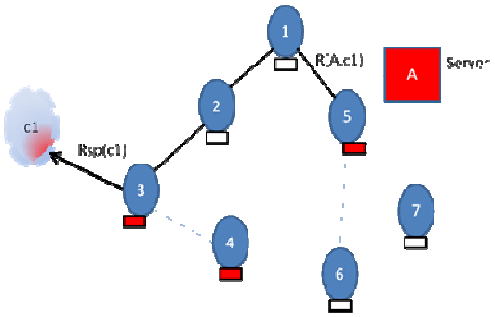


Σχήμα 1: Server-cache system at request time

Στο παράδειγμα που βλέπουμε στο σχήμα 1 ο πελάτης c1 συνδέεται στον broker 3 και κάνει ένα ερώτημα για την πληροφορία A. Με βάση την διαδικασία που έχουμε περιγράψει ο broker 3 θα ελέγξει εάν το μήνυμα που αναζητά είναι στην cache του, το οποίο δεν ισχύει οπότε θα πρέπει να το μεταδώσει στον επόμενο μεσολαβητή. Χρησιμοποιώντας το AT που έχει είδη, ο μεσολαβητής 3 μπορεί να υπολογίσει ότι η διαδρομή προς τον server που είναι οι brokers 2-1-5 οπότε και μεταδίδει το request(A,c1) στον πρώτο broker του μονοπατιού τον 2. Έπειτα καθώς το request διαδίδεται με την σειρά στους brokers πριν μεταδοθεί στον επόμενο γίνεται έλεγχος στην cache μήπως και είναι αποθηκευμένο το μήνυμα του ψάχνουμε στην τοπική cache όπως έχουμε πει. Όμως μίας και δεν υπάρχει σε κανένα από αυτούς θα αποτυγχάνει συνεχώς μέχρι το request να φτάσει τελικά στον broker 5 στον οποίο έχει συνδεθεί ο server και συνεπώς εκεί θα υπάρχει σίγουρα η απάντηση.

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

Τότε αρχίζει η διαδικασία του response προς τον client c1 ακολουθώντας την ανάποδη διαδρομή που είχε διανύσει το request και αποθηκεύοντας στην τοπική cache κάθε κόμβου που αναμεταδίδεται το response την πληροφορία που περιέχει (en routing caching).Όπως μπορούμε να παρατηρήσουμε και στο *σχήμα 2* όλοι οι brokers που μετέδωσαν το response(c1) έχουν στο τέλος της διαδικασίας στις τοπικές caches αποθηκευμένη την πληροφορία A .



Σχήμα 2:Server Cache system at response time

Όπως καταλαβαίνουμε στην χειρότερη περίπτωση το request θα φτάσει στο τέρμα του μονοπατιού δηλαδή στον server ο οποίος έχει σίγουρα την απάντηση όμως σε οποιαδήποτε άλλη περίπτωση που το μήνυμα βρεθεί σε κάποιον broker πριν από τον server τότε το response θα διανύσει μικρότερη απόσταση άρα μικρότερο delay.Αυτό μπορεί να τεκμηριωθεί εύκολα αν στο *σχήμα 2* συνδεθεί πάλι ένας πελάτης στον broker 3 και ζητήσει την πληροφορία A τότε αυτομάτως θα πάρει την απάντηση από την cache του χωρίς να διανύσει ούτε ένα hop απόσταση! Εάν δεν υπήρχαν οι caches τότε θα έπρεπε να διανύσει 3 hops όπως έκανε και την πρώτη φορά που περιγράψαμε λίγο ποιο πάνω.Οπότε προσθέτοντας caching στους ενδιάμεσους κόμβους έχουμε στο σύνολο των περιπτώσεων καλύτερη απόδοση σε σχέση με το απλό σχήμα επικοινωνίας Server στο οποίο το response διανύει πάντα απόσταση ίση με το μήκος του μονοπατιού server-client.

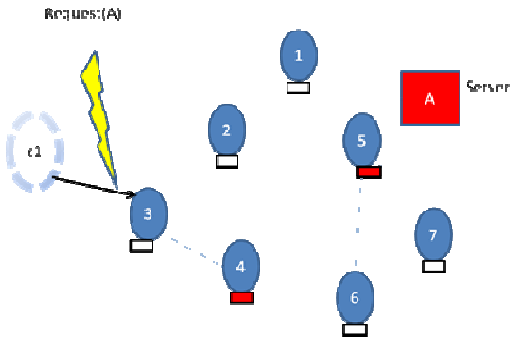
Τώρα έχοντας το σχήμα επικοινωνίας Server- cache(SC) μπορούμε να δοκιμάσουμε να επεκτείνουμε την στρατηγική διάδοσης του request έτσι ώστε να αυξήσουμε τις πιθανότητες να βρούμε το μήνυμα όσο ποιο κοντά στον client και κατά συνέπια να μειώσουμε την απόσταση που διανύει και το response βελτιώνοντας το delay. Το αποτέλεσμα είναι να εισάγουμε και να μελετήσουμε δύο ακόμα πολιτικές την Server one hop away(SOHA) και την Server subscription(SS) .

4.4 Server-one hop away σύστημα

ΤΟ SOHA λειτουργεί όπως περιγράψαμε το SC στο μόνο που διαφέρει είναι ότι το request δεν μεταδίδεται μόνο στους μεσολαβητές που ανήκουν στο μονοπάτι προς τον server αλλά και στους ενεργούς γείτονές τους. Οι ενεργοί γείτονες ενός broker είναι το σύνολο των brokers που είναι ένα hop απόσταση από αυτόν και δεν ανήκουν στο μονοπάτι προς τον Server .Εάν ένας από τους ενεργούς γείτονες δεν έχει την απάντηση τότε το request αποτυγχάνει και τερματίζεται η διαδικασία χωρίς να αναμεταδοθεί πουθενά. Εάν όμως έχει την απάντηση τότε ξεκινάει την

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

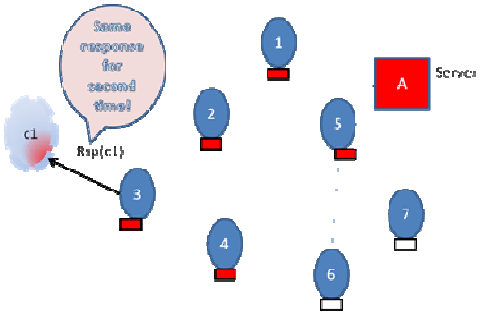
διαδικασία του response.Να επισημάνω πως το request σταματάει να μεταδίδετε μόνο έναν βρεθεί στην cache ενός μεσολαβητή που ανήκει στο μονοπάτι προς τον server . Έχοντας υπόψη αυτό προβλέπουμε ότι θα υπάρχουν διπλές η και παραπάνω απαντήσεις προς τον client.Αυτό μπορεί να προκύψει στην περίπτωση όπου από το σύνολο των brokers που έγινε η αναμετάδοση του request η απάντηση βρεθεί σε broker που ανήκει στο σύνολο των ενεργών γειτόνων με αποτέλεσμα να σταλεί response αλλά να μην σταματήσει η αναμετάδοση του request.Έτσι καθώς συνεχίζει να διαδίδεται το request στο μονοπάτι προς τον server μπορεί να βρεθεί και σε άλλους brokers με αποτέλεσμα όπως είχα προαναφέρει να έχουμε περισσότερες από μια απαντήσεις .



Σχήμα 3:Server-one hop away

Για να γίνει ποιο κατανοητή αυτή η λειτουργικότητα θα εξετάσουμε το παράδειγμα του Σχήματος 3. Έχουμε έναν πελάτη c1 στον broker 3 ο οποίος κάνει ένα request για την A πληροφορία. Βλέπουμε ότι 3 δεν έχει το μήνυμα του request που φτάνει σε αυτόν στην cache του οπότε υποχρεούται να το μεταδώσει στον επόμενο. Από το AT ξέρει ότι οι brokers που ανήκουν στο μονοπάτι προς τον server είναι οι 2-1-5. Μόνο που τώρα δεν θα μεταδώσει το μήνυμα μόνο στον 2 που είναι ο επόμενος, αλλά και σε όλους τους ενεργούς γείτονες. Στην περίπτωση μας στο σύνολο αυτό ανήκει ο 4 οπότε το request θα μεταδοθεί και σε αυτόν. Στην συνέχεια ο broker 2 εφόσον δεν έχει το μήνυμα στην cache του θα το αναμεταδώσει στον επόμενο του μονοπατιού τον 1 και σε κανένα άλλον εφόσον και οι δυο brokers που απέχουν ένα hop από αυτόν ο 1,3 ανήκουν στο μονοπάτι προς τον server. Παράλληλα με τις ενέργειες που κάνει ο broker 2 και ο 4 έχει δεχτεί το ίδιο request και μάλιστα έχει την απάντηση στην τοπική του cache οπότε σύμφωνα με αυτά που έχουμε πει θα αρχίσει την διαδικασία του response.Έτσι καθώς το response που δημιούργησε ο 4 κατευθύνετε προς τον πελάτη ο broker 1 συνεχίζει την διάδοση του request που δέχτηκε μιας και ο ίδιος δεν έχει την απάντηση στον broker 5 που είναι συνδεδεμένος ο Server.Αυτός με την σειρά του θα βρει το μήνυμα στην cache του και θα εκκινήσει την διαδικασία ενός ακόμα όμοιου response .Συνεπώς κατά την ολοκλήρωση της αναζήτησης όπως βλέπουμε και στο Σχήμα 4 ο πελάτης c1 παίρνει δύο φορές το ίδιο response.Επιπρόσθετα να σημειώσω ότι το σχήμα αυτό φαίνετε ότι οι brokers 3,2,1 που αναμετέδωσαν ένα response έχουν στην cache τους την πληροφορία που μετέδιδε .Τέλος μια λεπτομέρεια είναι ότι ο broker 3 που δέχεται δύο φορές το ίδιο response στην cache θα το αποθηκεύσει μόνο μια φορά.

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7



Σχήμα 4:Server one hop away at response time

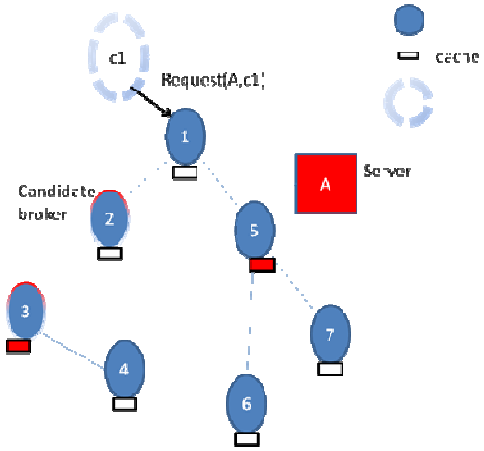
Συνεπώς αυτή η πολιτική αυξάνει το πλάτος που ψάχνει προς απάντηση ,αυξάνοντας παράλληλα και την πιθανότητα να την βρει πριν διανύσει όλη την διαδρομή προς τον server δηλαδή μειώνει την καθυστέρηση της απάντησης(delay),αλλά ως τίμημα πληρώνει το κόστος επιβάρυνσης του δικτύου με αποστολή όμοιων απαντήσεων (overhead) πράγμα πού δεν υπήρχε την προηγούμενη πολιτική Server-Cache.

4.5 Server-subscription σύστημα

Η τελευταία πολιτική που θα μελετήσουμε είναι η SS που είναι και αυτή επέκταση στης SC αλλά και του συστήματος δημοσίευσης/συνδρομής .Η σημαντικότερη διαφορά που υπάρχει στον τρόπο λειτουργίας είναι η διάδοση του request .

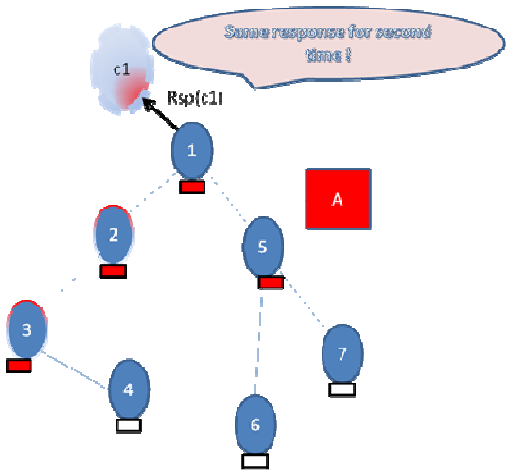
Ειδικότερα το request εκτός από τους μεσολαβητές που ανήκουν στο μονοπάτι προς τον server μεταδίδετε και σε όσους έχουν τουλάχιστον ένα πελάτη που αναζητά το ίδιο μήνυμα .Τώρα για να βρει ποιοι brokers ικανοποιούν την δεύτερη συνθήκη χρησιμοποιεί έναν πίνακα όμοιο με τον ST(Subscription Table) όπως ακριβώς γίνεται και στο σύστημα επικοινωνίας Publish/Subscribe .Επίσης και η λογική που γίνεται αυτό είναι παρόμοια ,εφόσον ένας broker έχει έναν πελάτη για κάποιο μήνυμα που ψάχνω πολύ πιθανόν να έχει και το μήνυμα στην cache του από προηγούμενη αναζήτηση .Έτσι προσπαθούμε να εκμεταλλευτούμε αυτήν την επιπλέον πληροφορία ώστε να φτάσουμε ποιο γρήγορα στην εύρεση του μηνύματος που συνεπάγεται και γρήγορο response που όπως έχουμε πει είναι ο στόχος μας. Όμως όπως και στην SOHA έχουμε και σε αυτήν την πολιτική πολλαπλά response αφού το request δεν διαδίδετε πάντα προς μία και μόνο κατεύθυνση άρα μπορεί να βρεθούν απαντήσεις από διάφορα σημεία του δικτύου με αποτέλεσμα έχουμε ένα σημαντικό αριθμό μηνυμάτων που επιβαρύνων χωρίς ουσία το σύστημά μας (overhead). Τέλος και σε αυτήν την πολιτική κάθε broker που αναμεταδίδει ένα response αποθηκεύει το μήνυμα που μεταφέρει στην cache του εκτός και αν το έχει είδη.

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7



Σχήμα 5:Server subscription base

Με ένα παράδειγμα θα μπορέσουμε να καταλάβουμε καλύτερα την παραπάνω λειτουργικότητα .Στο Σχήμα 5 έχουμε έναν πελάτης c1 που είναι συνδεδεμένος στον broker 1 και δύο candidate brokers των 2,3 που έχουν πελάτη που αναζητεί όμοια πληροφορία και τέλος τον Server που βρίσκεται στον broker 5.Όταν ο c1 ξεκινήσει ένα request για την πληροφορία A και εφόσον ελέγχει και δεν έχει την απάντηση στην cache του θα πρέπει να προωθήσει το request στο μονοπάτι προς τον server που το βρίσκει μέσω του AT ότι είναι ο 1,5 και σε όλους τους candidate brokers που μέσω του ST βρίσκει ότι είναι οι 2,3.



Σχήμα 6:Server subscription base at response time

Άρα στο σύνολο θα μεταδοθούν 3 request .Έτσι ένα από αυτά θα φτάσει στον broker 5 όπου θα βρει την απάντηση και θα αρχίσει το response, ένα άλλο θα πάει στον 2, δεν θα βρει στην cache την απάντηση οπότε και θα τερματιστεί εκεί η αναζήτηση και το τελευταίο θα φτάσει στον broker 3 όπου θα βρεθεί η πληροφορία A οπότε για δεύτερη φορά θα δημιουργηθεί ένα όμοιο response.Το

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

αποτέλεσμα είναι όπως βλέπουμε και στο Σχήμα 6 ο πελάτης c1 να λαμβάνει 2 απαντήσεις και οι brokers 2,1 που μετέδωσαν ένα response να έχουν στην τοπική cache τους την πληροφορία A.

Ένα θέμα που θα μπορούσε επίσης να μελετηθεί με περισσότερη έμφαση είναι ποιοι brokers μας συμφέρει να κρατάνε τα μηνύματα που κουβαλάνε τα response έτσι ώστε να τα κατανείμουμε όσο ποιο κοντά σε brokers που τα ζητάνε. Εμείς έχουμε επιλέξει για όλες της πολιτικές που αναλύσαμε παραπάνω να αποθηκεύονται σε όποιον broker αναμεταδώσει ένα response .Μια άλλη προσέγγιση μπορεί να ήταν ένας broker να αποθηκεύει στην cache του ένα μήνυμα μόνο εάν έχει περάσει ένα κατώφλι αναμετάδοσης .Εναλλακτικά θα μπορούσαμε να σκεφτούμε η αποθήκευση ενός response μηνύματος σε έναν μεσολαβητή να επιτρέπετε μόνο εάν έχει πελάτη που αναζητεί το συγκεκριμένο μήνυμα έτσι ώστε να μην επιβαρύνουμε τις caches πολλών brokers με τα ίδια μηνύματα .

Επίσης αν θέλουμε να αυξήσουμε την επιτυχία τις caches των μεσολαβητών θα μπορούσαμε με το που συνδέεται ένας server να αποθηκεύουμε την πληροφορία που παρέχει όχι μόνο στον broker που έχει συνδεθεί αλλά και σε ένα επιπλέον σύνολο broker. Το ένα κριτήριο για να ανήκει κάποιος broker σε αυτό θα μπορούσε να είναι οι brokers που δέχονται τα περισσότερα request δηλαδή έχουν αυξημένο αριθμό πελατών.

4.6 Πολιτική τοποθέτησης μηνυμάτων σε Server-Client σύστημα.

Σημαντικό ρόλο στην αποδοτική λειτουργία όλων των παραπάνω πολιτικών που χρησιμοποιούν cache παίζει ο μηχανισμός αντικατάστασης μηνυμάτων. Αυτή η διαδικασία αναφέρεται στους brokers των οποίων οι cache τους είναι γεμάτη και πρέπει να αποθηκευτή ένα καινούργιο μήνυμα . Οι πολιτικές place που θα μελετήσουμε είναι η FIFO,Regenerate, ,LRU,LFU και πάλι όπως και στο σύστημα δημοσίευσης/συνδρομής με caching που έχουμε περιγράψει είδη .Οπότε εδώ θα επικεντρωθούμε μόνο στην περιγραφεί της LFU και της LRU μιας και μόνο σε αυτές αλλάζουν σε κάποια σημεία οι ενέργειες που θα πρέπει να κάνει ένας broker.Οι πολιτικές FIFO και Regenerate λειτουργούν ακριβώς όπως έχουμε γράψει στην ενότητα 6.4 .

Στην LFU η διαδικασία έχει ως εξής :

- Όταν δεχτεί ένα request από έναν πελάτη τότε εάν υπάρχει αντίστοιχο μήνυμα στην cache αυξάνει το count_request κατά ένα. Εάν δεν υπάρχει τότε δεν χρειάζεται να κάνει καμία ενέργεια .
- Όταν δεχτεί ένα response τότε θα πρέπει να αποθηκεύσει το μήνυμα στην cache του θέτοντας το count_request=0 και το start_time=current_time . Αν έχει ελεύθερες θέσεις στην cache τότε το τοποθετεί σε μία από αυτές .Σε διαφορετική περίπτωση θα πρέπει να βρει το μήνυμα με την μικρότερη συχνότητα request και να το αντικαταστήσει με το καινούριο . Στην περίπτωση που δύο η παραπάνω μηνύματα στην cache του μεσολαβητή έχουν την ίδια συχνότητα χρησιμοποίησης τότε αντικαθιστούμε αυτό με τον ποιο πρόσφατο χρόνο εισαγωγής μιας και εάν ένα μήνυμα παραμείνει ένα μεγάλο διάστημα στην cache καλό είναι να το κρατήσουμε.

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

Στην LRU η διαδικασία είναι :

- Όταν δεχτεί ένα request από έναν πελάτη τότε εάν υπάρχει αντίστοιχο μήνυμα στην cache ανανεώνει τον χρόνο θέτοντας start_time=current_time διαφορετικά δεν κάνει καμία απολύτως ενέργεια.
- Όταν δεχτεί ένα response τότε θα πρέπει να αποθηκεύσει το μήνυμα στην cache του θέτοντας το start_time=current_time .Αν έχει ελεύθερες θέσεις τότε το τοποθετεί σε μία από αυτές .Σε διαφορετική περίπτωση θα πρέπει να βρει το μήνυμα που έχει τον περισσότερο χρόνο να δεχτεί request και να το αντικαταστήσει με το καινούριο .

Ανακεφαλαιώνοντας έχουμε 4 πολιτικές για να εξετάσουμε το σχήμα επικοινωνίας Server – client .Τις Server, Server-Caching ,Server- one hop away και την Server-subscription. Οι πολιτικές τώρα place που χρησιμοποιώ είναι η FIFO,Regenerate, ,LRU,LFU.

Εφόσον πλέον έχουμε αναλύσει την λειτουργικότητα του συστήματος επικοινωνίας Server-client μαζί με τις πολιτικές request και τις πολιτικές αντικατάστασης μηνυμάτων στην cache ενός μεσολαβητή, μπορούμε στην συνέχεια να προχωρήσουμε στην υλοποίηση-προσομοίωση .Τα αποτελέσματα και οι ανάλυσή τους ακολουθούν στις επόμενη ενότητα.

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

ΕΝΟΤΗΤΑ 5 .Προσομοίωση συστήματος

Σε αυτήν την ενότητα θα αξιολογήσουμε τον προτεινόμενο μηχανισμό που αναλύουμε στην ενότητα 4, χρησιμοποιώντας έναν discrete event simulation .Οι κύριες οντότητες που τον αποτελούν είναι οι μεσολαβητές N που οργανώνονται σε μια Acyclic peer to peer τοπολογία, οι πελάτες οι οποίοι δημιουργούνται δυναμικά και συνδέονται σε κάποιον μεσολαβητή και εκκινούν την διαδικασία των request με βάση έναν ρυθμό 'γένεσης ' request λ και τέλος οι εξυπηρετητές που συνδέονται σε έναν μεσολαβητή ακολουθώντας ένα ρυθμό μ 'γένεσης' μηνυμάτων προς εξυπηρέτηση χωρίς φυσικά να αποσυνδέονται στο μέλλον και σε αντίθεση με τους πελάτες .Επίσης κάθε μεσολαβητής έχει μια cache η οποία αποτελείτε από έναν συγκεκριμένο αριθμό κενών θέσεων K .Αρχικά όλες οι caches των μεσολαβητών είναι άδειες μέχρι να γίνει το πρώτο request .Κατά την εκτέλεση της προσομοίωσης υπάρχουν M διαφορετικά μηνύματα που εξυπηρετούν το σύνολο των servers του συστήματος και τα οποία οι πελάτες μπορούν να αναζητούν .
Αφήνουμε το σύστημά μας να λειτουργήσει στο πλαίσιο του δυναμικού περιβάλλοντος πελάτη-εξυπηρετητή χρησιμοποιώντας την πολιτική προτεραιότητας LFU και συγκρίνοντας τις 4 πολιτικές request(Server,Server-cache,Server-one hop away,Server-subscription) στο πρώτο μέρος των πειραμάτων και στο δεύτερο μέρος χρησιμοποιούμε την Server-Cache request πολιτική και συγκρίνουμε τις 4 πολιτικές προτεραιότητας (FIFO , Regenerate , LFU , LRU) .

5.1 Ορισμός μέτρων προς αξιολόγηση του συστήματος

Σκοπός τις προσομοίωσης είναι να αξιολογήσουμε το σύστημά μας με βάση τα παρακάτω μέτρα :

Καθυστέρηση απάντησης (delay response): ορίζουμε την απόσταση που διανύει το response από την στιγμή που βρεθεί η απάντηση του request στην cache κάποιου μεσολαβητή μέχρι να φτάσει στον πελάτη που εκκίνησε την διαδικασία . Αυτό το μέτρο μας δείχνει πόσο γρήγορα θα πάρει απάντηση ένας πελάτης που ψάχνει για μια πληροφορία και αυτό που προσπαθούμε να πετύχουμε είναι να μειώσουμε όσο τον δυνατών περισσότερο αυτόν τον χρόνο εισάγοντας της πολιτικές request που έχουμε περιγράψει .

Επιπλέον απαντήσεις (overhead): ορίζουμε τον επιπλέον φόρτο που δημιουργείτε στο δίκτυο επικοινωνίας λόγο διπλών οι περισσότερων της μιας απάντησης που θα ήταν το ιδανικό .Στην συγκεκριμένη εργασία τον φόρτο αυτόν τον μετράμε ως το άθροισμα των hops που θα κάνουν συνολικά όλα τα επιπλέον response .Έτσι δεν δίνουμε σημασία μόνο στον αριθμό των επιπρόσθετων response αλλά και πόσο επιβάρυναν το σύστημα με της αναμεταδόσεις τους .Σκοπός και εδώ είναι να πετύχουμε μια πολιτική request με τον μικρότερο άθροισμα hops των επιπλέον response .

Επιτυχία στην κρυφή μνήμη (Hit cache):ορίζουμε την κατάσταση κατά την οποία η απάντηση του request που εκκίνησε ο πελάτης βρέθηκε στην cache κάποιου μεσολαβητή που βρίσκετε ποιο κοντά στον πελάτη σε σχέση με την απόσταση που βρίσκετε ο Server .Δηλαδή αυτό το μέτρο μας δείχνει πόσο ικανό είναι το σύστημά μας να απαντάει σε ερωτήματα που δημιουργούν οι πελάτες χωρίς να χρειάζεται να επιφορτίσουμε τον server.Έτσι εκτός του ότι μειώνουμε την συμφόρηση που δέχεται

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

ο Server μειώνουμε και τον χρόνο απόκρισης της απάντησης το delay, εφόσον σε πολιτικές που το request διαδίδεται μόνο στο μονοπάτι προς τον server εάν βρεθεί η απάντηση πριν τον Server τότε είναι σίγουρο ότι θα φτάσει και στον πελάτη ποιο γρήγορα. Παράδειγμα μιας τέτοιας πολιτικής request είναι η Server-cache .Το μόνο που θα πρέπει να προσέχουμε εάν έχουμε καλό ποσοστό επιτυχίας στης caches των ενδιάμεσων μεσολαβητών είναι ότι μπορεί να δημιουργούνται και περισσότερα επιπλέον μηνύματα σε ορισμένες πολιτικές request όπως την Server-one hop away και την Server-subscription .Αυτό που μπορούμε να συμπεράνουμε εύκολα είναι ότι και τα τρία αυτά μέτρα που αναφέραμε παραπάνω είναι άμεσα αλληλοεξαρτώμενα.

Τα παραπάνω μέτρα αποτελούν τις τυχαίες μεταβλητές και θα προσπαθούμε να εκτιμήσουμε την σημασία τους πάνω στο σύστημά μας παρατηρώντας τα αποτελέσματα από της προσομοίωσης που θα πραγματοποιήσουμε .Σε κάθε προσομοίωση που θα διεξαχθεί θα παρατηρούμε της τιμές των μέτρων που αναφέραμε, σε σχέση με κάποιες παραμέτρους του συστήματος .Αυτές είναι ο αριθμός των brokers που αποτελούν το σύστημα μας , ο αριθμός των θέσεων του θα έχουν οι cache των μεσολαβητών , ο ρυθμός με των οποίο οι πελάτες θα κάνουν ερωτήσεις για εύρεση πληροφοριών και ο ρυθμός με των οποίο προσθέτονται καινούργιοι Servers .

5.2 Σχεδίαση συστήματος προσομοίωσης

Η σχεδίαση του συστήματος έχει γίνει στο Matlab δημιουργώντας έναν Discrete-Event Simulation που προσομοιώνει την λειτουργία του μοντέλου Server-Client.Οι δύο βασικές συναρτήσεις είναι η Simulation_SC() που υλοποιεί την λειτουργικότητα του συστήματος Server-client με την δυνατότητα να χρησιμοποιεί τρις πολιτικές request (Sever ,Server-cache ,Server-one hop away) και το Simulation() που υλοποιεί ένα επεκταμένο σύστημα δημοσίευσης/συνδρομής με την προσθήκη των Servers και πολιτική request την Server-subscription .Να αναφέρουμε εδώ ότι το σύστημα δημοσίευσης/συνδρομής λειτουργεί με caching policy στους candidate brokers, αλγόριθμο δρομολόγησης forwarding subscription και αρχιτεκτονική μεσολαβητών cycling peer to peer.Επίσης και οι δύο συναρτήσεις αυτές έχουν την δυνατότητα να χρησιμοποιούν και τις τέσσερες πολιτικές αντικατάστασης .

Πριν προχωρήσουμε να επισημάνω μερικές λεπτομέρειες στης προσομοίωσης :

1. Ο υπολογισμός του ρυθμού των request γίνεται με βάση το λ, που έχουμε πει είδη, αλλά επηρεάζεται και από τον αριθμό των μεσολαβητών (όσο περισσότεροι είναι τόσο ποιο πιθανό είναι να γίνει ένα request) και από τον αριθμό των διαφορετικών μηνυμάτων που εξυπηρετούν οι servers του συστήματος μεταβάλλοντας των αριθμό των request με τον ίδιο τρόπο όπως και οι μεσολαβητές .
2. Κατά την μέτρηση της καθυστέρησης(delay) αν φτάσει ένα request στον μεσολαβητή που είναι συνδεδεμένος ο server και δεν έχει το μήνυμα που εξυπηρετεί τότε θα πρέπει να το αντιγράψει από αυτόν, όμως επειδή θεωρούμε αυτήν την διαδικασία πολύ πιο γρήγορη από το συνολικό χρόνο απάντησης δεν το προσμετράμε ως απόσταση .
3. Ο χρόνος που θα αφήσουμε το σύστημά μας να αλληλεπιδρά θα τίθεται από την μεταβλητή MC θα έχει τιμή 5000 και βρίσκεται στην συνάρτηση main() η οποία εκκινεί την διαδικασία της προσομοίωσης και θέτει και τις τιμές των παραμέτρων. Επίσης η προκαθορισμένες τιμές των παραμέτρων θα είναι μ,λ=0.05 οι μεσολαβητές θα είναι 10 και ο αριθμός των

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

cache θέσεων 10.Οι τιμές είναι τέτοιες ώστε να γίνονται κατά μέσο όρο 400000 request και να δημιουργούνται 100 περίπου διαφορετικά μηνύματα σε όλους τους servers, ανά προσομοίωση .Τέλος να διευκρινίσω πως μία προσομοίωση θεωρούμε την μελέτη μίας πολιτικής είτε request είτε place με μια μεταβαλλόμενη παράμετρο και τις υπόλοιπες σταθερές .

5.3.Αποτίμηση αποτελεσμάτων

Όπως είχαμε αναφέρει και προηγουμένως στην πρώτη σειρά πειραμάτων θα εξετάσουμε τα μέτρα delay ,overhead και hit cache για τις 4 πολιτικές request αλλάζοντας κάθε φορά της παραμέτρους που σχολιάσαμε πιο πάνω και χρησιμοποιώντας σταθερά την LFU ως πολιτική αντικατάστασης. Στην δεύτερη σειρά θα ασχοληθούμε με τις τέσσερις πολιτικές αντικατάστασης έχοντας σταθερά την Server-cache ως πολιτική request και όπως και πριν θα ελέγχουμε τα μέτρα αλλάζοντας τις παραμέτρους του συστήματος .

5.3.1 Πρώτο μέρος πειραμάτων

Στο πρώτο πείραμα εξετάζουμε τον τρόπο με τον οποίο μεταβάλλονται τα μέτρα αξιολόγησης από την επίδραση των διάφορων πολιτικών request αυξάνοντας διαδοχικά των αριθμό των μεσολαβητών και κρατώντας σταθερές όλες τις άλλες παραμέτρους .Τα αποτελέσματα που προκύπτουν απεικονίζονται στο Σχήμα 1 και Σχήμα 2.

Όπως μπορούμε να παρατηρήσουμε(Σχήμα 1) η καθυστέρηση(delay), ανεξάρτητα από την πολιτική request που χρησιμοποιείτε αυξάνετε καθώς αυξάνονται και οι brokers.Αυτό οφείλετε στο γεγονός ότι όσο αυξάνονται οι μεσολαβητές δημιουργούν ένα μεγαλύτερο δίκτυο και έτσι είναι πιο πιθανό η απόσταση client-Server να είναι μεγαλύτερη, άρα και η απόσταση που διανύει το response να είναι μεγαλύτερη .Όσο αναφορά τις πολιτικές request βλέπουμε ότι η Server έχει τις μεγαλύτερες τιμές για την καθυστέρηση. Παράδειγμα για 50 μεσολαβητές, ένα response κάνει περίπου 6 hops απόσταση ανά request σε αντίθεση με την Server-one hop away που έχει ένα λιγότερο και άρα θα παίρνει τις απαντήσεις πιο άμεσα .Έτσι επαληθεύεται η πρόβλεψη που είχαμε κάνει πως με την προσθήκη cache στους μεσολαβητές θα μπορούμε να απαντάμε σε ερωτήματα πιο γρήγορα και χωρίς να επιβαρύνουμε τον server .

Ως προς το επιπλέον φόρτο (overhead) οι πολιτικές Server ,Server-cache δεν επιβαρύνουν το σύστημα καθόλου διότι η πρώτη δεν χρησιμοποιεί καν cache και οι δεύτερη στέλνει το request μόνο πάνω στο μονοπάτι προς τον server και έτσι και για τις δύο δεν υπάρχει τρόπος να πάρουν επιπλέον απάντηση .Δεν συμβαίνει το ίδιο όμως και για τις άλλες δύο πολιτικές και κυρίως για την Server-subscription η οποία εισάγει μεγάλο αριθμό σχεδόν 9 hops άθροισμα όλων των επιπλέον response ανά request στους 50 μεσολαβητές .Αυτό συμβαίνει γιατί διαδίδει το request σε πολλές κατευθύνσεις (και προς τον server και προς τους candidate brokers) και ενώ κάποιος θα περίμενε λόγο αυτού να βρίσκει απάντηση ποιο γρήγορα διαψεύδετε διότι έχει την δεύτερη χειρότερη καθυστέρηση από τις τέσσερες πολιτικές .Ο λόγος είναι ότι οι candidate brokers συχνά δεν βρίσκονται πιο κοντά σε απόσταση σε σχέση με τον Server από όπου και τελικά παίρνει την απάντηση ο πελάτης .Όλα αυτά σε αντίθεση με την Server-one hop away η οποία έχει ένα πολύ μικρό αριθμό επιπλέον response γύρο στο 0.2 όταν έχουμε 50 μεσολαβητές και όπως είπαμε την καλύτερη καθυστέρηση .

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

Από της μετρήσεις του Σχήματος 2 φαίνετε το ποσοστό επιτυχίας στης caches των brokers, εκτός αυτού που είναι συνδεδεμένος ο server ανά request μεταβάλλοντας των αριθμό των μεσολαβητών. Όπως παρατηρούμε σε όλες τις πολιτικές εκτός φυσικά την Server που δεν έχει cache καθώς αυξάνονται οι μεσολαβητές το ποσοστό επιτυχίας μεγαλώνει και μάλιστα στους 50 μεσολαβητές φτάνει και το 45% .Αυτό σημαίνει πρακτικά ότι σχεδόν το ένα στα δύο ερωτήματα απαντάτε από το σύστημα των μεσολαβητών και όχι από τους Servers .Όσο αναφορά τις πολιτικές ξεχωριστά βλέπουμε ότι το ψηλότερο ποσοστό επιτυχίας το πραγματοποιεί η Server-one hop away σε αντίθεση με την Server-subscription που έχει το μικρότερο .Αυτή η παρατήρηση είναι και μια επιβεβαίωση του γεγονότος ότι η Server-subscription έχει μεγάλο delay επειδή τις περισσότερες φορές παίρνει απάντηση από τον Server που βρίσκεται πιο κοντά στον πελάτη σε σχέση με τους candidate brokers οι οποίοι απαντούν καθυστερημένα επιβαρύνοντας το σύστημα με επιπλέον αχρησιμοποίητα μηνύματα .

Τα αποτελέσματα του επόμενου πειράματος φαίνονται στο Σχήμα 3 και Σχήμα 4.Σε αυτά εξετάζουμε την καθυστέρηση ,τον επιπλέον φόρτο μηνυμάτων προς το σύστημα και την επιτυχία στις cache ,μεταβάλλοντας τον αριθμό των θέσεων που έχει η cache των μεσολαβητών και κρατώντας τις υπόλοιπες παραμέτρους στις αρχικές τους τιμές.

Στο Σχήμα 3 από τις τιμές της καθυστέρησης βλέπουμε γενικά ότι καθώς αυξάνεται ο αριθμός χωρητικότητας μηνυμάτων των caches οι πολιτικές request που της χρησιμοποιούν καταφέρνουν να μειώσουν το ποσοστό hops ανά response περίπου κατά 1 hop όταν ο αριθμός των θέσεων είναι 50 σε σχέση με την Server πολιτική. Αυτή η μείωση είναι σημαντικότερη από όσο φαίνεται διότι οι προκαθορισμένη τιμή για τους μεσολαβητές είναι 10 και άρα οι αποστάσεις μεταξύ τους είναι μικρές έτσι αν είχαμε 50 μεσολαβητές τότε η μείωση της καθυστέρησης θα ήταν αρκετά μεγαλύτερη .Ως προς της πολιτικές ξεχωριστά παρατηρούμε ότι το προτιμότερο ποσοστό το πετυχαίνει η Server-one hop away πράγμα που μας οδηγεί προς το συμπέρασμα ότι είναι η προτιμότερη από της άλλες τέσσερις.

Συνεχίζουμε την μελέτη του Σχήματος 3 για το δεύτερο μέτρο προς εξέταση που είναι ο επιπλέον φόρτος εισαγωγής στο σύστημα. Πάνω σε αυτό παρατηρούμε ότι και εδώ οι πολιτικές Server, Server-cache δεν εισάγουν κανένα επιπλέον μήνυμα response και εξηγήσαμε τον λόγο προηγούμενα .Ως προς τις υπόλοιπες πολιτικές βλέπουμε ότι η Server-subscription έχει κατά πολύ αυξημένο ποσοστό (μέγιστη τιμή περίπου 5 φορές παραπάνω στα 50 slot σε σχέση με τις υπόλοιπες) overhead το οποίο αυξάνεται καθώς αυξάνονται οι θέσεις στην cache .Αυτό οφείλεται στο γεγονός ότι καθώς έχουμε περισσότερες θέσεις ανά cache μπορούμε να αποθηκεύσουμε περισσότερα μηνύματα και κατά συνέπια όποιες πολιτικές διασκορπίζουν σε πολλές κατευθύνσεις το request, όπως η Server-subscription, έχουν περισσότερες πιθανότητες να πάρουν παραπάνω απαντήσεις αφού θα υπάρχει το μήνυμα σε παραπάνω cache μεσολαβητών .Τέλος να προσθέσω ότι το ποσοστό overhead στο Σχήμα 1 όταν είχαμε 10 μεσολαβητές και με cache slot 10 είναι μικρότερο περίπου κατά 0.1 από το αντίστοιχο ποσοστό με 10 brokers και 50 cache slot που έχουμε στο Σήμα 3.Αυτο μας δείχνει ότι αν παράλληλα αυξάναμε και τους μεσολαβητές εκτός των cache slot το ποσοστό του overhead θα μεγάλωνε ακόμα ποιο πολύ.

Καλύτερη πολιτική και σε αυτές τις μετρήσεις αποδεικνύεται η Server-one hop away με μικρή ωστόσο διαφορά από την Server-Cache.

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

Το σχήμα 4 αναφέρεται στο ποσοστό επιτυχία στις cache ανά request, μεταβάλλοντας τον αριθμό των θέσεων που έχουν οι cache .Παρατηρούμε (Σχήμα 4) την λογική αύξηση του ποσοστού επιτυχίας καθώς αυξάνονται τα slots .Να τονίσουμε ότι το ποσοστό είναι περίπου 50% στα 50 slot στην πολιτική Server-one hop away .Αυτό το ποσοστό είναι πολύ μεγάλη βελτίωση διότι έχουμε μόνο 10 brokers και αν συγκρίνουμε το ποσοστό επιτυχίας που είχαν οι 10 brokers με 10 cache slots στο Σχήμα 2 θα δούμε ότι είναι μειωμένο παραπάνω από το μισό ,είναι περίπου 18% από την καλύτερη πολιτική Server-cache.Έτσι μπορούμε να προβλέψουμε ότι αν βάλουμε 50 μεσολαβητές με cache slots 50 μπορούμε να φτάσουμε σε εκπληκτικά ποσοστά όπως 70-85%. Τέλος βλέπουμε ότι η Server-subscription έχει τα χειρότερα ποσοστά επιτυχίας, μειωμένα κατά 20% περίπου σε σχέση με τις άλλες δυο πολιτικές .

Στο επόμενο πείραμα που ακολουθεί, με τα αποτελέσματα να παρατίθενται στο Σχήμα 5 και Σχήμα 6, μελετάμε την καθυστέρηση, το επιπλέον φόρτο και την επιτυχία στην cache που προκύπτει στο σύστημα μεταβάλλοντας την παράμετρο του ρυθμού άφιξης των request που κάνουν οι πελάτες στους μεσολαβητές .

Παρατηρώντας τις γραφικές (Σχήμα 5) παραστάσεις διαπιστώνουμε ότι και πάλι έχουμε μείωση των hops που κάνει κατά μέσο όρο ένα response με την Server πολιτική σε σχέση με τις υπόλοιπες που χρησιμοποιούν cache.Όμως δεν είναι τόσο μεγάλη, περίπου 0,3 hops λιγότερα , σε αντίθεση με την διαφορά που έφτανε το 1 hops όταν μεταβάλαμε τους μεσολαβητές ή τις θέσης των cache .Αυτό οφείλετε, όπως θα γίνει πιο κατανοητό και στην συνέχεια και συγκεκριμένα στο Σχήμα 6, στο ότι καθώς αυξάνετε ο ρυθμός των request και με τις caches να έχουν μόνο 10 θέσης φτάνουμε στο σημείο να ανανεώνονται συνεχώς τα μηνύματα που είναι αποθηκευμένα και όχι μόνο αυτό, αλλά να μην έχουμε και υψηλά ποσοστά επιτυχίας έχοντας ένα μεγάλο αριθμό request που ζητάνε κάθε φορά διαφορετικά μηνύματα .Το έμμεσο αποτέλεσμα αυτής της αστοχίας των caches είναι το request τις περισσότερες φορές να μην απαντάτε από τους brokers του συστήματος αλλά από τον Server .Κατά αυτόν τον τρόπο μειώνετε η διαφορά των hops τις απάντησης που κάνει η πολιτική Server σε σύγκριση με αυτές που χρησιμοποιούν cache .

Οι μετρήσεις του overhead επαληθεύουν αυτό που αναλύσαμε πάνω βλέποντας ότι η πολιτική που μέχρι τώρα είχε τα περισσότερα επιπλέον response, σε αυτό το πείραμα καθώς αυξάνονται τα request τα συνολικά hops των response μειώνονται περίπου κατά 0.5 όταν πάμε από λ=0.05 σε λ=1.5 .

Στις γραφικές παραστάσεις του Σχήματος 6 αποτυπώνετε η επιβεβαίωση του μικρού ποσοστού επιτυχίας στις cache των μεσολαβητών 18% καθώς αυξάνουμε τον ρυθμό των request , όταν με λ=0,05 είχαμε περίπου 50% επιτυχία, σε όλες τις πολιτικές .Τέλος να προσθέσουμε ότι καθώς αυξάνετε το λ δεν μειώνετε αντίστοιχα και το ποσοστό επιτυχίας, παραμένει δηλαδή σταθερό .Όσο είχε η κάθε πολιτική με λ=0.05 έχει και με λ=1.5 .Αυτό οφείλετε στο μικρό αριθμό θέσεων στις cache σε σχέση με την συνεχή και μεγάλη ζήτηση πολλών μηνυμάτων προς απάντηση .Έτσι μετά από έναν αριθμό λ και συγκεκριμένα μετά από λ>0.5 όσο και αυξάνουμε τον ρυθμό, cache με 10 θέσεις δεν μπορούν να αποδώσουν ούτε καλύτερα ούτε χειρότερα. Αν όμως αυξάναμε τις θέσεις τότε θα παρατηρούσαμε μια μεγαλύτερη μείωση της επιτυχίας καθώς αυξανόταν ο ρυθμός λ .

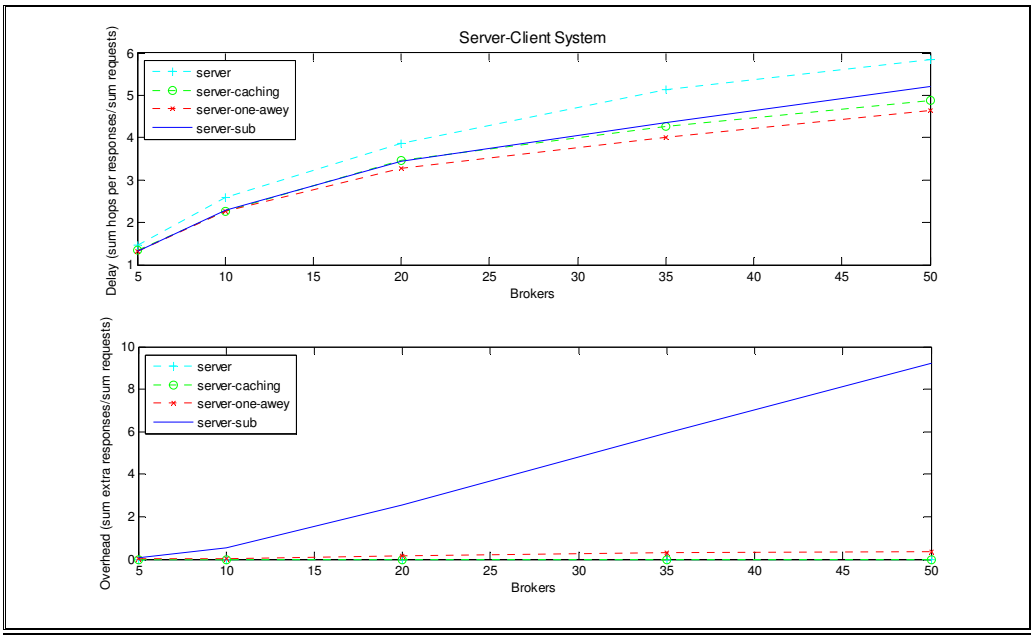
Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

Στο τελευταίο πείραμα αυτής της σειράς θα μελετήσουμε τον τρόπο με τον οποίο επιδρά η διαδοχική αύξηση των διαφορετικών μηνυμάτων που εξυπηρετούν οι severs στα μέτρα αξιολόγησης με βάση τις διάφορες πολιτικές request .Τα αποτελέσματα είναι αποτυπωμένα στα Σχήματα 7,8 .

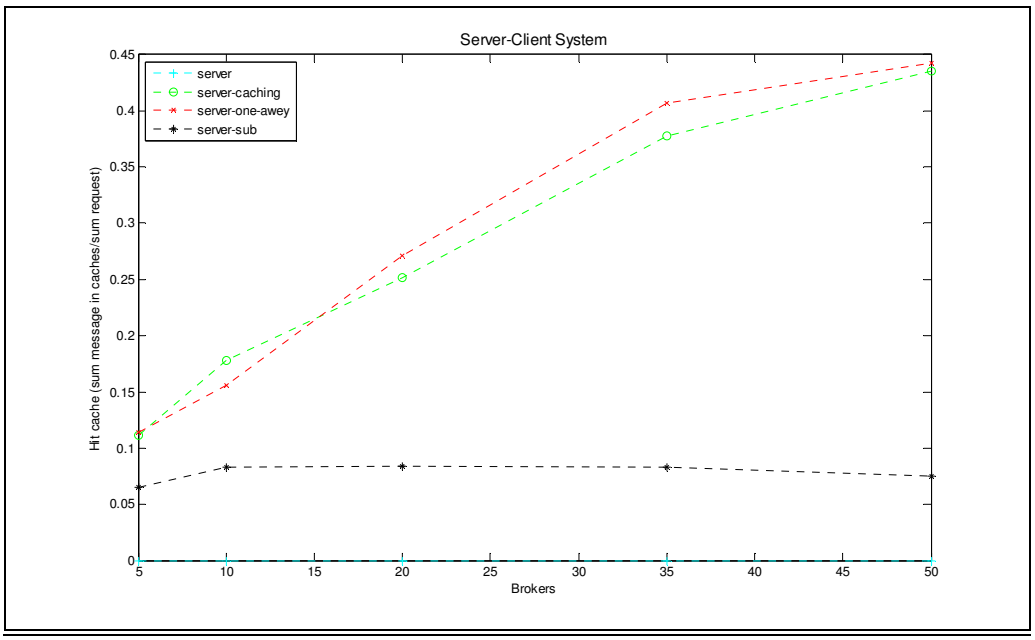
Στο Σχήμα 7 παρατηρούμε ότι όταν αυξηθούν κατά πολύ τα διαφορετικά μηνύματα που εξυπηρετεί το σύστημά μας τότε η καθυστέρηση που απαιτείτε από το response για να φτάσει στον πελάτη συγκλίνει στον ίδιο αριθμό ανεξάρτητα από την πολιτική και ανεξάρτητα από το αν το σύστημά μας χρησιμοποιεί η όχι cache .Αυτό οφείλεται στο γεγονός ότι με τόσο πολλά μηνύματα ουσιαστικά εκμηδενίζουμε την cache εφόσον μπορεί να κρατήσει μόνο ένα πολύ μικρό αριθμό μηνυμάτων .Για παράδειγμα για μ=0.5 τα συνολικά μηνύματα φτάνουν Μ=2500 και έτσι έχοντας μία cache με μόνο 10 θέσεις χωρητικότητα καλύπτει μόνο το 0.004 του Μ οπότε σχεδόν για κανένα request δεν βρίσκετε η απάντηση σε ενδιάμεσο μεσολαβητή και όλα καταλήγουν ανεξάρτητα αν έχουν cache η τον τρόπο διάδοσης του request στον server ,έτσι ακριβώς όπως και στην πολιτική Server .Αυτό μας το επιβεβαιώνει το Σχήμα 8 στο οποίο φαίνονται τα πολύ μικρά ποσοστά επιτυχίας που έχουν οι ενδιάμεσοι κόμβοι και στις τρις πολιτικές που χρησιμοποιούν cache καθώς αυξάνετε το μ .Συγκεκριμένα από τιμές του μ μεγαλύτερες του 0.3 σχεδόν αγγίζει το 0% .Φυσικά αν θέλαμε να επαναφέρουμε το σύστημά μας σε προηγούμενα επίπεδα λειτουργίας θα έπρεπε να αυξήσουμε της θέσεις των cache σε τουλάχιστον 250 αλλά και τον αριθμό των brokers .

Τέλος ,στο Σχήμα 8 βλέπουμε ότι ο επιπλέον φόρτος που επιβαρύνει το σύστημα λόγο των πολλαπλών response μειώνετε και σε μεγάλα μ σχεδόν μηδενίζετε .Όμως και σε συμφωνία με αυτά που έχουμε πει είδη αυτό δεν μας εκπλήσσει εφόσον είναι αιτία της μη αποδοτικής λειτουργίας των caches που έχουν οι μεσολαβητές και κατά συνέπια στις περισσότερες των περιπτώσεων η μόνη απάντηση που φτάνει στον πελάτη είναι από τον server .

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

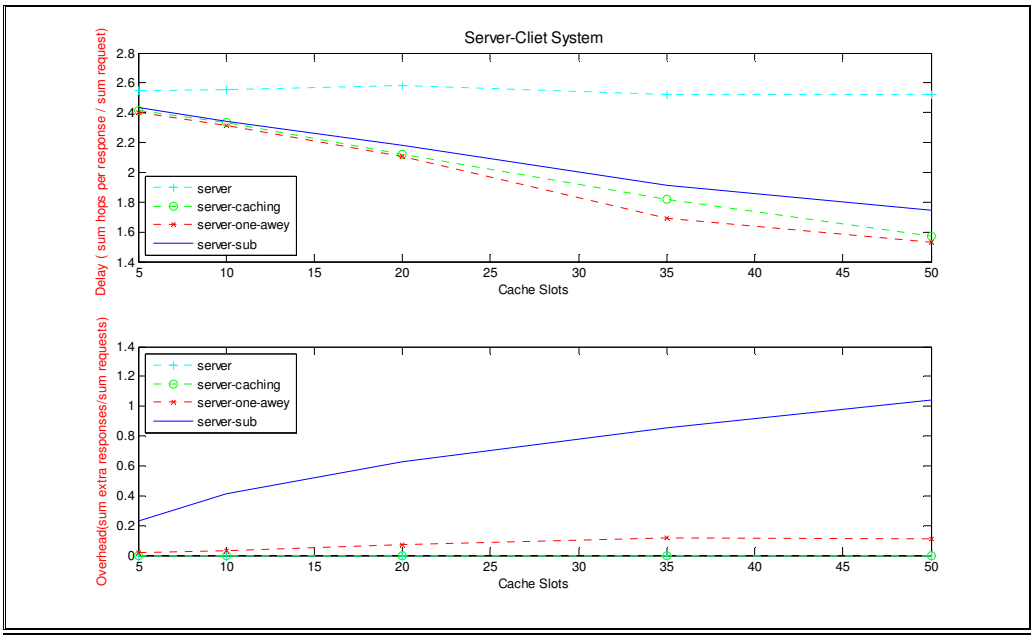


Σχήμα 1 :Μετρήσεις για delay και overhead για διάφορες τιμές μεσολαβητών

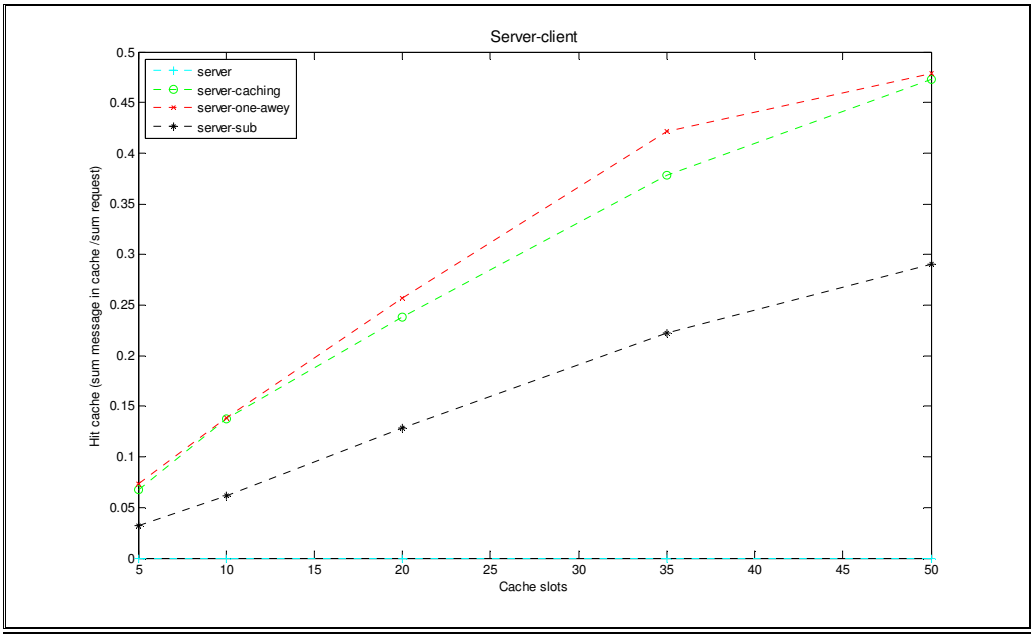


Σχήμα 2 :Μέτρηση επιτυχίας της caches για διάφορες τιμές των μεσολαβητών

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

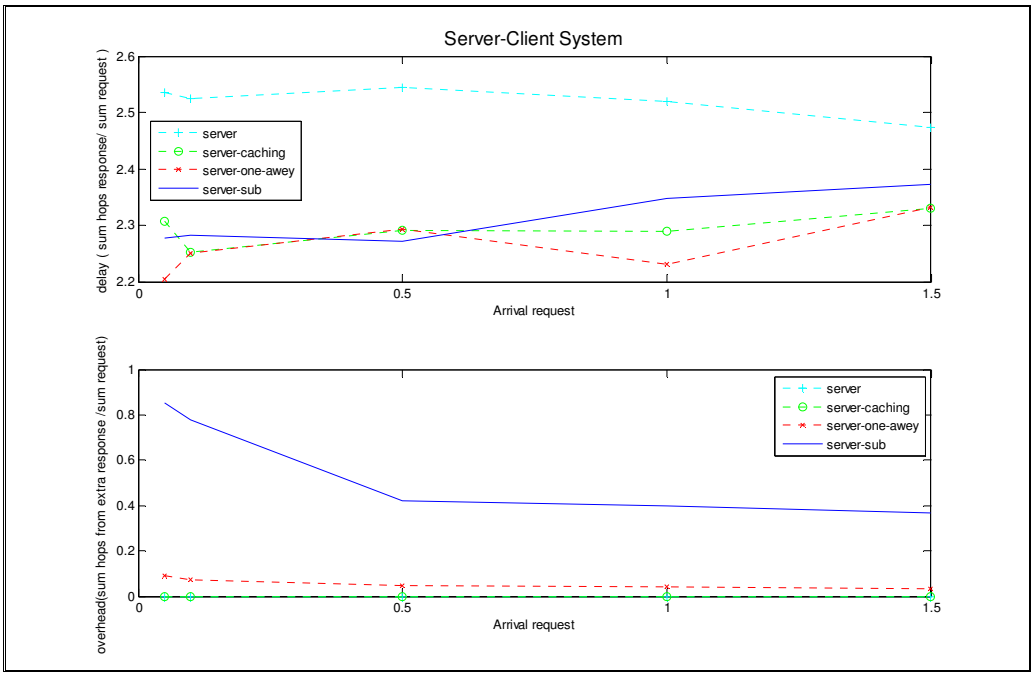


Σχήμα 3 :Μέτρηση delay και overhead για μεταβαλλόμενες τιμές caches slots.

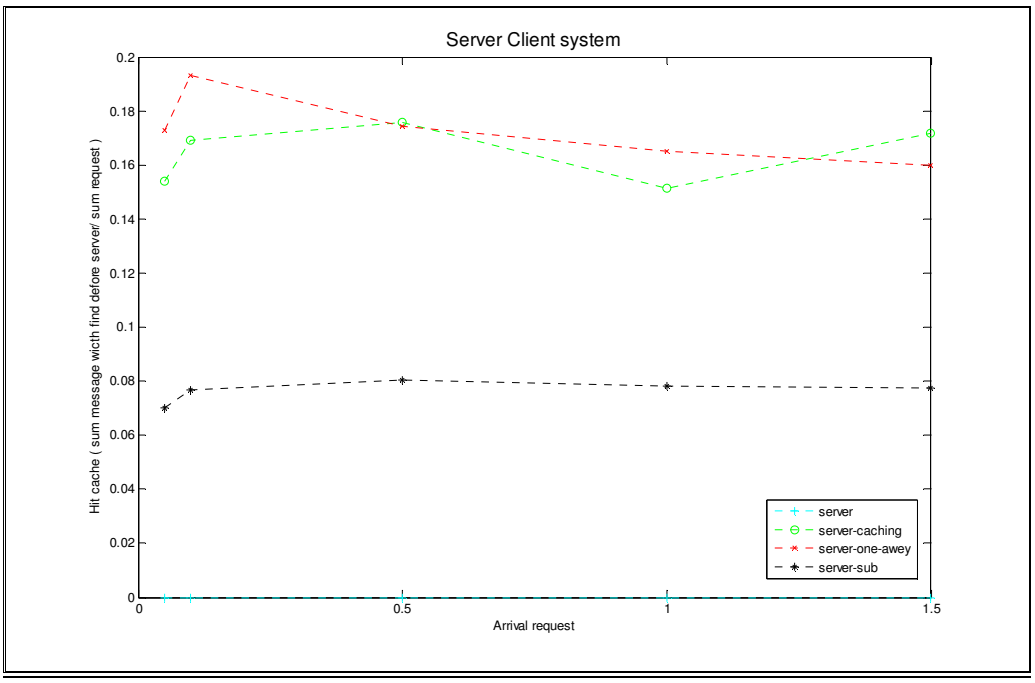


Σχήμα 4: Μέτρηση επιτυχίας τις caches για διάφορες τιμές των θέσεων των cache .

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

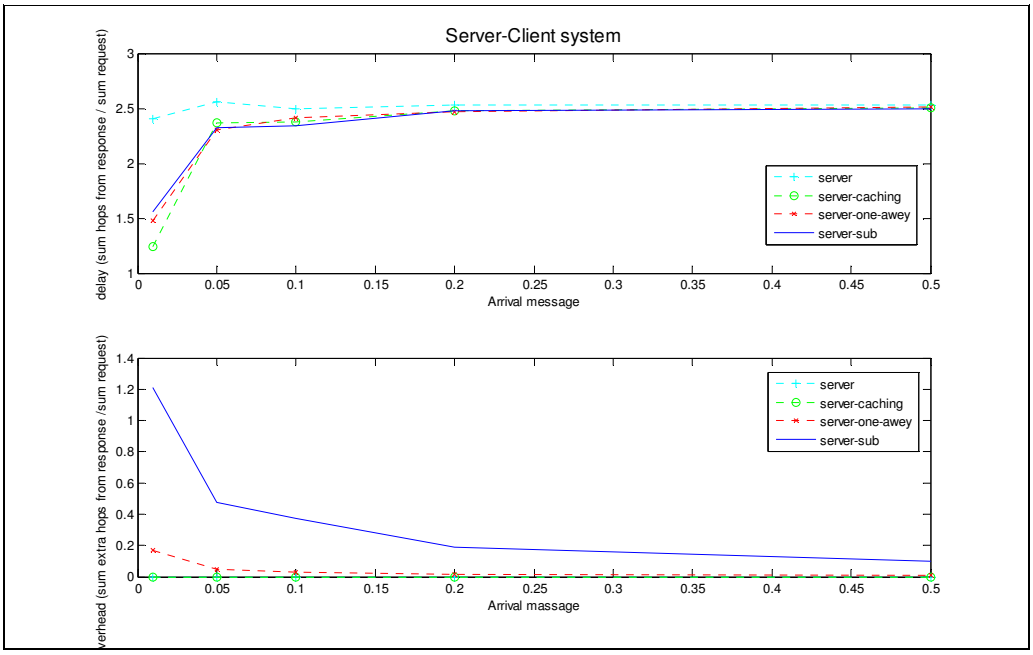


Σχήμα 5 : Μέτρηση delay και overhead για μεταβαλλόμενες τιμές του λ .

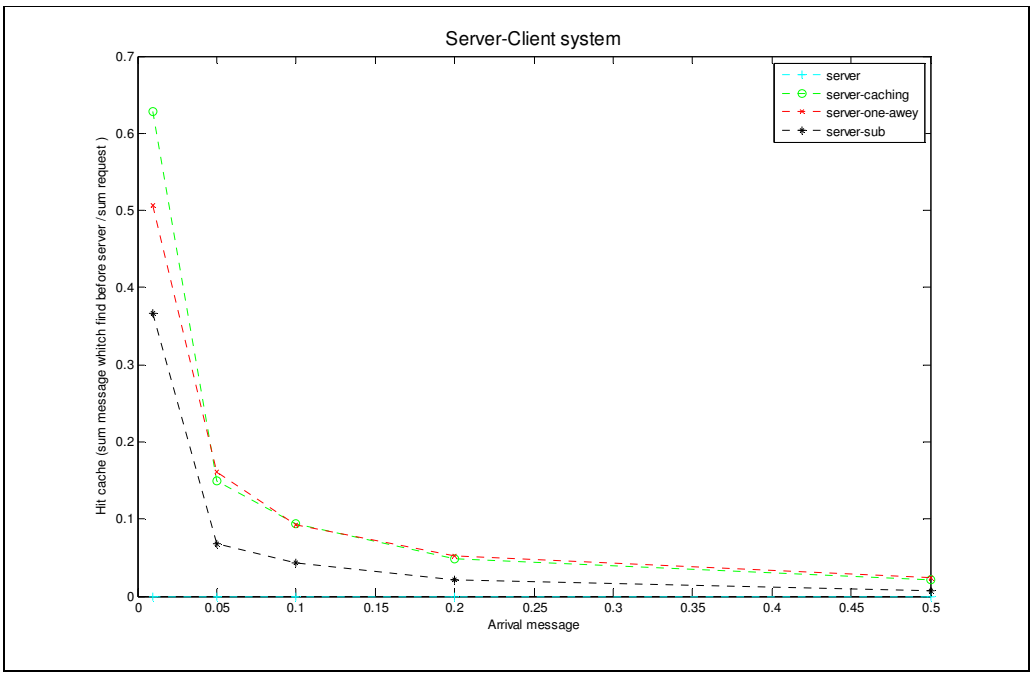


Σχήμα 6: Μέτρηση επιτυχίας τις caches για διάφορες τιμές του λ .

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7



Σχήμα 7 : Μέτρηση delay και overhead για μεταβαλλόμενες τιμές του μ.



Σχήμα 8 : Μέτρηση επιτυχίας τις caches για διάφορες τιμές του μ .

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

5.3.2 Δεύτερο μέρος πειραμάτων

Στην δεύτερη σειρά πειραμάτων εξετάζουμε τον τρόπο με τον οποίο μεταβάλλονται τα μέτρα αξιολόγησης, εκτός του επιπλέον φόρτου μηνυμάτων μιας και χρησιμοποιούμε ένα σύστημα Server-Cache που δεν έχει, από την επίδραση των διάφορων πολιτικών αντικατάστασης μεταβάλλοντας τις τιμές των παραμέτρων .

Στο πρώτο πείραμα που διεξάγουμε εξετάζουμε για κάθε μια από τις 4 πολιτικές αντικατάστασης (FIFO ,Regenerate ,LFU ,LRU) τι επίδραση έχει στα μέτρα προς αξιολόγηση, οι διαδοχική αύξηση του αριθμού των μεσολαβητών .Οι μετρήσεις που προκύπτουν παρουσιάζονται στα Σχήματα 9-10 .

Όπως μπορούμε να παρατηρήσουμε στο Σχήμα 9 για όλες τις πολιτικές η καθυστέρηση αυξάνετε καθώς αυξάνονται ο αριθμός των μεσολαβητών που είναι αναμενόμενο διότι αυξάνονται και οι αποστάσεις μεταξύ τους .Επίσης από τις γραφικές παραστάσεις προκύπτει ότι με όλες τις πολιτικές αντικατάστασης έχουμε σχεδόν την ίδια καθυστέρηση με μικρότερη αυτήν στης LFU (0.2 hops λιγότερα από τις άλλες για 50 μεσολαβητές).Θα περιμέναμε με βάση τον τρόπο λειτουργίας η LFU να είχε καλύτερα αποτελέσματα .Ο λόγος που δεν συμβαίνει αυτό είναι γιατί υποθέτουμε ότι όλα τα μηνύματα έχουν την ίδια πιθανότητα να επιλεγούν για request εκτός από έναν πολύ μικρό αριθμό μνημάτων που τους αυξάνουμε τον ρυθμό ζήτησης (είναι ο λόγος που η LFU έχει λίγο καλύτερη απόδοση όπως βλέπουμε στο σχήμα 9).

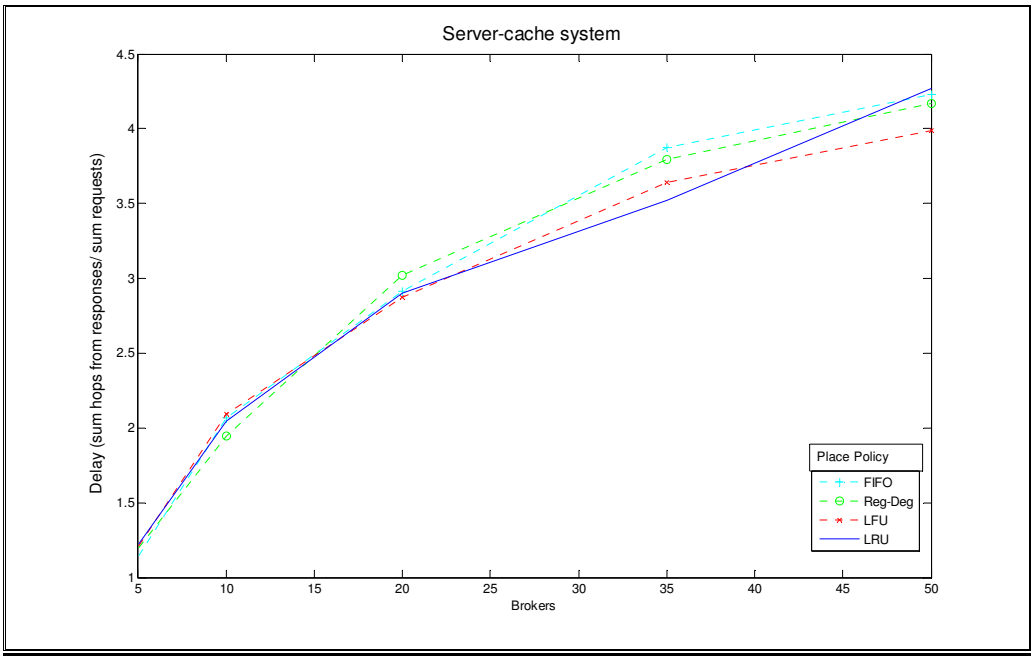
Στο Σχήμα 10 βλέπουμε πώς επηρεάζετε η επιτυχία στης cache των μεσολαβητών με βάση τις τέσσερες πολιτικές και αυξάνοντας των αριθμό των brokers.Τα αποτελέσματα είναι ανάλογα με αυτά που είχαμε επισημάνει με όταν μελετούσαμε το αντίστοιχο πείραμα με τις πολιτικές request το μόνο που θα πρέπει να τονίσουμε είναι ότι για τιμές μεσολαβητών από 45-50 οι πολιτική LFU είναι κατά ένα μικρό ποσοστό προτιμότερη από τις άλλες ,θα ήταν μεγαλύτερο το ποσοστό αυτό εάν είχαμε πιο συχνά request για συγκεκριμένα μηνύματα .

Στο επόμενο πείραμα, όπου τα αποτελέσματα απεικονίζονται στα Σχήματα 11-12 , συνεχίσουμε με τον ίδιο τρόπο μελέτης όπως και το προηγούμενο, με βάση της πολιτικές αντικατάστασης ελέγχουμε τον τρόπο που αλλάζουν οι τιμές των μέτρων αξιολόγησης, μόνο που τώρα θα μεταβάλλουμε των αριθμό των θέσεων στις cache των μεσολαβητών .

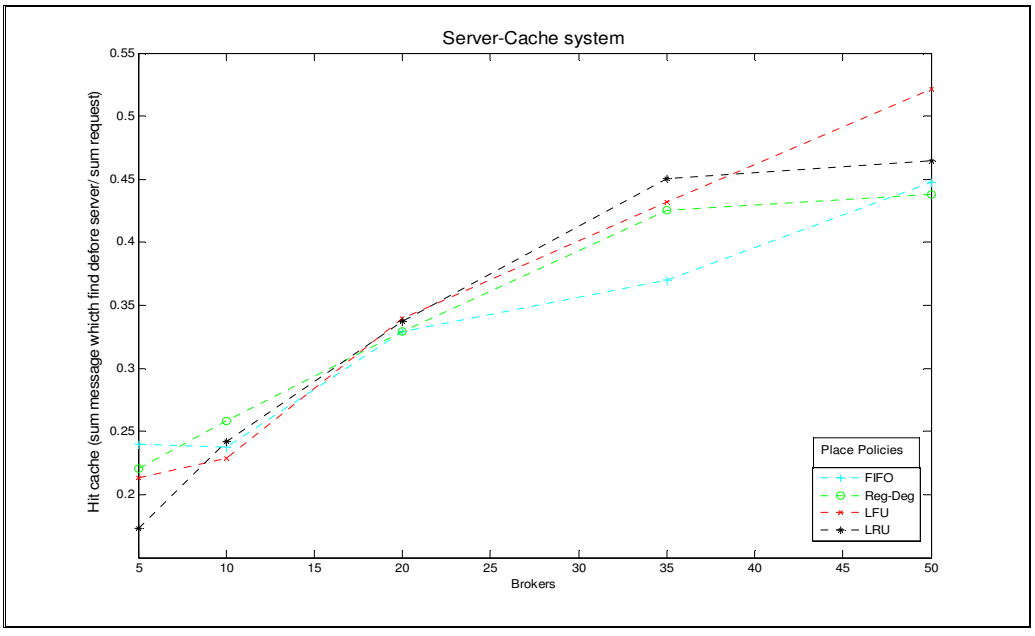
Στο σχήμα 11 φαίνετε μια μεγάλη μείωση της καθυστέρησης γενικά σε όλες της πολιτικές αυξάνοντας των αριθμό των θέσεων της cache όπως ακριβώς και με της πολιτικές request μόνο που εδώ αυξήσαμε ακόμα πιο πολύ τις χωρητικότητα(μέγιστο 50 στις πολιτικές request ,μέγιστο 80 τώρα) και το delay συνέχισε να μειώνετε μέχρι και 2 hops στην μέγιστη διαφορά. Δηλαδή με 80 slot η μείωση της καθυστέρησης ξεπερνάει το 65%.Ετσι διαπιστώνουμε αυτό που είχαμε αναφέρει ότι θα γίνει εάν αυξήσουμε και άλλο των χωρητικότητα των cache .Ως προς τις πολιτικές, οι τρις FIFO ,Regenerate ,LRU έχουν σχεδόν την ίδια επίδραση στο delay και η LFU φαίνετε πως είναι ελαφρώς καλύτερη .

Στα αποτελέσματα του Σχήματος 12 παρατηρούμε το μεγάλο ποσοστό επιτυχίας στις cache του συστήματος που φτάνει το 90% .Η διαφορά του ποσοστού αυτού σε σχέση με την προηγούμενη σειρά πειραμάτων οφείλετε μόνο στο γεγονός μεγαλύτερης αύξησης της χωρητικότητας των cache(ο μέγιστος αριθμός slot αυξάνεται από 50 σε 80).Ως προς τις πολιτικές που μελετάμε τώρα, ενδιαφέρων παρουσιάζει ότι η LFU που έχει το χαμηλότερο ποσοστό περίπου 0.05% λιγότερο από τις υπόλοιπες στα 80 slots .

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

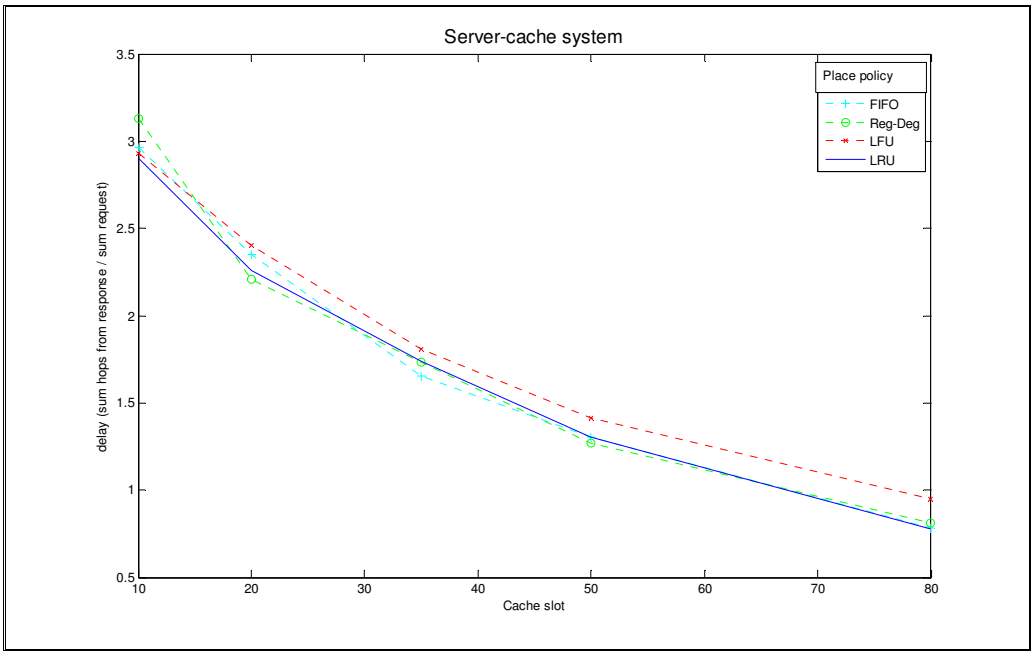


Σχήμα 9 : Μετρήσεις για delay για διάφορες τιμές μεσολαβητών

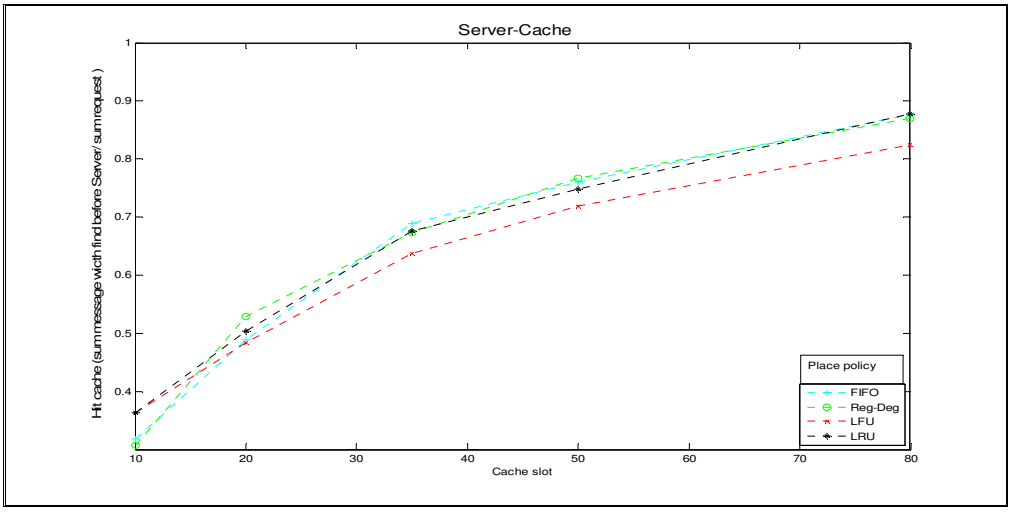


Σχήμα 10 : Μέτρηση επιτυχίας τις caches για διάφορες τιμές των μεσολαβητών .

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7



Σχήμα 11: Μετρήσεις για delay για διάφορες τιμές cache slot .



Σχήμα 12: Μέτρηση επιτυχίας τις caches για διάφορες τιμές cache slot

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

ΕΝΟΤΗΤΑ 6 .Συμπεράσματα και μελλοντική δουλειά

Το βασικό συμπέρασμα από την πρώτη σειρά πειραμάτων είναι ότι οι εισαγωγή cache στο σύστημα επικοινωνίας Server-client είναι ο βασικός λόγος για την μείωση της απόστασης που διανύει η απάντηση ενός request που κάνει ένας πελάτης, πράγμα που κάνει το σύστημα μας πιο γρήγορο στην εξυπηρέτηση πελατών .Επίσης εκτός της πολιτικής Server-subscription που προσθέτει στο σύστημα ένα υπολογίσιμο επιπλέον φόρτο με διπλά η περισσότερα response οι υπόλοιπες πολιτικές δεν εισάγουν καθόλου επιπλέον φόρτο όπως η Server-cache η εισάγουν ένα πολύ μικρό ποσοστό που μπορεί να θεωρηθεί και αμελητέο όπως η Server-one hop away.Επιπρόσθετα εάν αυξήσουμε την χωρητικότητα των cache στους μεσολαβητές σε 0.4*M από 0.1*M που είχαμε πει στην αρχή μπορούμε να πετύχουμε στης cache των brokers ποσοστό επιτυχία που να φτάνει το 80% .Δηλαδή δεν φτάνει να βάλουμε cache σε ένα σύστημα και εκεί τελειώνει η υπόθεση, αλλά είναι πολύ σημαντικό να ξέρεις τί τιμές cache slots πρέπει να έχεις σε ποίες τιμές των παραμέτρων του συστήματος ώστε να πετύχεις με το μικρότερο κόστος την καλύτερη απόδοση . Έτσι το όφελος του κερδίζουμε με μια καλή χρήση των cache στους μεσολαβητές είναι εκτός ότι εξυπηρετούμε τους πελάτες πιο γρήγορα καταφέρνουμε επίσης να αποδεσμεύουμε τους Servers από την διαδικασία της απάντησης των request .Τέλος μετά από μια σύγκριση των πολιτικών βλέπουμε ότι η Server-one hop away πετυχαίνει τα καλύτερα ποσοστά και στα τρία μέτρα αξιολόγησης εκτός από το overhead που όμως είπαμε είναι τόσο μικρό που μπορεί να αγνοηθεί.

Στην δεύτερη σειρά πειραμάτων προσπαθήσαμε να δούμε κατά πόσο επιδρά η πολιτική αντικατάστασης που θα χρησιμοποιήσεις στο σύστημα για την βελτίωση των μέτρων προς αξιολόγηση .Το συμπέρασμα που προέκυψε είναι ότι η LFU καταφέρνει να αυξήσει λίγο περισσότερο από τις υπόλοιπες πολιτικές τις επιδόσεις του συστήματος χωρίς όμως να φανεί ξεκάθαρα η αξία της για τον λόγο που είπαμε και πιο πάνω ότι δηλαδή δεν χρησιμοποιούμε δημοφιλή μηνύματα εκτός από ένα πολύ μικρό αριθμό μηνυμάτων που τους αυξάνουμε τον ρυθμό ζήτησης έτσι ώστε να φανεί έστω και λίγο η αξία της LFU .Για να αξιοποιήσουμε πλήρως τα ωφέλει μιας τέτοιας πολιτικής θα έπρεπε να έχουμε ένα μεγάλο πλήθος μηνυμάτων που να ζητούνται με διαφορετικό ρυθμό .

Ως μελλοντική δουλειά το πιο ενδιαφέρων είναι να προσομοιωθεί το σύστημα Server-Client σε πραγματικό περιβάλλον δημιουργώντας ένα σύστημα το οποίο να έχει την λειτουργικότητα που έχουμε περιγράψει (π.χ. υλοποίηση στο REDs).Έκτος όμως από αυτό υπάρχει ένα πολύ μεγάλο περιθώριο για περεταίρω βελτίωση προθέτοντας μια καινούργια πολιτική request η place.Θα μπορούσαμε ακόμη να αλλάξουμε και κάποια λειτουργικά θέματα όπως αντί να αποθηκεύουμε τα response σε κάθε μεσολαβητή που τα αναμεταδίδει να εισάγουμε μια πολιτική επιλεκτικής αποθήκευσης των response με βάση κάποια κριτήρια όπως αυτό που συζητήσαμε όταν λέγαμε για τους candidate brokers .Επίσης μια ακόμα βελτίωση θα μπορούσε να είναι η προσθήκη της διαδικασίας απόρριψης μηνυμάτων(multiply responses) έτσι ώστε να μειώσουμε ακόμα πιο πολύ τον επιπλέον φόρτο απαντήσεων .Τέλος, αν θέλουμε να αυξήσουμε την επιτυχία στις caches των μεσολαβητών θα μπορούσαμε με το που δημιουργείτε ένα νέο μήνυμα σε έναν server να αποθηκεύουμε την νέα πληροφορία που παρέχει όχι μόνο στον broker που έχει συνδεθεί αλλά και σε ένα επιπλέον σύνολο από brokers(per-caching).

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7

Πηγές

[1] Li G., Cheung A., Hou S., Hu S., Muthusamy V., Sherafat R., Wun A., Jacobsen H., and Manovski S., “Historic data access in publish/subscribe,”. Proceedings of the 2007 inaugural international Conference on Distributed Event-Based Systems (DEBS 2007), pp. 80–84, Toronto, Canada, 2007.

[2] Chand R. and Felber A., “A scalable protocol for content-based routing in overlay networks,” 2nd IEEE International Symposium on NetworkComputing and Applications, pp. 123–130, 2003.

[3] The Many Faces of Publish/Subscribe .PATRICK TH. EUGSTER PASCAL A. FELBER RACHID GUERRAOUI AND ANNE-MARIE KERMARREC

[4] Modeling the dynamics of caching in content-based publish/subscribe systems.Vasilis Sourlas12, Georgios S. Paschos12, Petteri Mannersalo3,Paris Flegkas12 and Leandros Tassiulas12

[5] Caching in content-based publish/subscribe systems.Vasilis Sourlas_, Georgios S. Paschosy, Paris Flegkas_ and Leandros Tassiulas__Department of Computer & Communication Engineering.University of Thessaly, Greece, yCERTH-ITI

[6] PUBLISH SUBSCRIBE CONTENT DELIVERY NETWORKSPart I: Intro to Publish/Subscribe Interaction Scheme

[7] Design and Evaluation of a Wide-Area .Event Notification Service . ANTONIO CARZANIGA University of Colorado at Boulder DAVID S. ROSENBLUM. University of California, Irvine and ALEXANDER L. WOLF University of Colorado at Boulder.

[8] REDS: A Reconfigurable Dispatching System .Gianpaolo Cugola and Gian Pietro Picco Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy

[9] TIBCO. TIB/Rendezvous. White paper. TIBCO, Palo Alto, CA.1999.

[10] Powell, D.: Group communication. Commun. ACM 39, 4 (Apr.), pp 50–97, 1996.

[11] Birman, K., Cooper, R., Joseph, T., Marzullo, K., Makpangou, M., Kane, K., Schmuck, F., and Wood, M.: The Isis System Manual. Dept. Of Computer Science, Cornell University, Ithaca, NY. 1990.

[12] Networking Named Content Van Jacobson D. K. Smetters James D. Thornton Michael Plass Nick Briggs Rebecca L.Braynard Palo Alto Research Center (PARC)

Institutional Repository - Library & Information Centre - University of Thessaly
08/12/2017 21:22:29 EET - 137.108.70.7