ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη Συστήματος Πρόβλεψης Κίνησης του Χρήστη σε έναν Ιστοχώρο

Διονυσία - Μαρία Π. Αναγνωστοπούλου

Κωνσταντίνα Θ. Μεντή

Επιβλέποντες: Ευστάθιος Χατζηευθυμιάδης, Επικούρος Καθηγητής ΕΚΠΑ
Εμμανουήλ Σπανουδάκης, Υποψήφιος Διδάκτωρ ΕΚΠΑ

ΑΘΗΝΑ

ΔΕΚΕΜΒΡΙΟΣ 2007

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη Συστήματος Πρόβλεψης Κίνησης του Χρήστη σε έναν Ιστοχώρο

Διονυσία - Μαρία Π. Αναγνωστοπούλου

A.M.: 018200200012

Κωνσταντίνα Θ. Μεντή

A.M.: 018200200048

ΕΠΙΒΛΕΠΟΝΤΕΣ:

Ευστάθιος Χατζηευθυμιάδης, Επικούρος Καθηγητής ΕΚΠΑ
Εμμανουήλ Σπανουδάκης, Υποψήφιος Διδάκτωρ ΕΚΠΑ

ΠΕΡΙΛΗΨΗ

Σε αυτή την πτυχιακή εργασία μελετούμε την εφαρμογή ενός συγκεκριμένου αλγορίθμου με σκοπό τη βελτίωση της αρχιτεκτονικής ενός Δικτύου Διανομής (CDN). Ειδικότερα, στόχος είναι να κατακερματιστεί όσο το δυνατόν με μεγαλύτερη «επιτυχία» ένας συγκεκριμένος ιστοχώρος, ώστε να χρησιμοποιηθεί ένα όσο το δυνατόν μικρότερο μέρος του και να επιτευχθεί όσο μεγαλύτερο ποσοστό επιτυχίας (hit rate).

Η μελέτη βασίζεται σε ένα στατικό μοντέλο γραφήματος του ιστοχώρου, και για τον κατακερματισμό χρησιμοποιήθηκε ο αλγόριθμος Edge Betweenness Clusterer. Ολοκληρώνοντας την υλοποίηση, εκτελέστηκαν σενάρια χρήσης του αλγορίθμου προκειμένου να εξετάσουμε την αποτελεσματικότητα και απόδοσή του. Μελετώντας τα αποτελέσματα που προέκυψαν, το συμπέρασμα που προκύπτει είναι πως ο συγκεκριμένος αλγόριθμος χρήζει βελτίωσης. Πιθανότατοι τρόποι για τη βελτίωση της αποτελεσματικότητας του παρατίθενται μαζί με τα συμπεράσματα.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Τεχνολογίες Διαδικτύου

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: ενδιάμεσος εξυπηρέτης, ομαδοποίηση, εξισορρόπηση φόρτου, αντιγραφή πληροφορίας, μηχανισμός ενδιάμεσης μνήμης

ΕΥΧΑΡΙΣΤΙΕΣ

Θα θέλαμε να ευχαριστήσουμε θερμά τον κ. Ε. Χατζηευθυμιάδη για την ανάθεση της πτυχιακής και για τη βοήθεια και καθοδήγηση που μας παρείχε καθ' όλη τη διάρκεια της εκπόνησής της. Επιπλέον, θα θέλαμε να ευχαριστήσουμε τον Ε. Σπανουδάκη που με τη γνώση που κατέχει σχετικά με το θέμα που διαπραγματεύεται η πτυχιακή και με τις χρήσιμες οδηγίες και συμβουλές του συνέβαλλε αποτελεσματικά στη διεκπεραίωση της παρούσας εργασίας.

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΡΟΛΟΓΟΣ

Η παρούσα πτυχιακή εργασία εκπονήθηκε για το τμήμα Πληροφορικής και Τηλεπικοινωνιών του Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών το ακαδημαϊκό έτος 2006-2007, με τίτλο «Ανάπτυξη Συστήματος Πρόβλεψης Κίνησης του

Χρήστη σε έναν Ιστοχώρο». Επιβλέπων καθηγητής του τμήματος ήταν ο κ. Ευστάθιος Χατζηευθυμιάδης τον οποίο ευχαριστούμε θερμά.

Τέλος, αφιερώνουμε την παρούσα πτυχιακή εργασία στον υποψήφιο διδάκτορα του τμήματος Εμμανουήλ Σπανουδάκη καθώς η πολύτιμη βοήθεια και στήριξή του ήταν πολύ σημαντικές για την εκπόνησή της.

Διονυσία – Μαρία Αναγνωστοπούλου
Κωνσταντίνα Μεντή

ΚΕΦΑΛΑΙΟ 1

ΔΙΚΤΥΑ ΔΙΑΝΟΜΗΣ ΠΕΡΙΕΧΟΜΕΝΟΥ (CDNs)

1.1. Γενικά

Τα τελευταία χρόνια, το Διαδίκτυο (Internet) έχει εξελιχθεί από ένα μηχανισμό μεταφοράς κειμένου και πολυμεσική πληροφορίας απλών εικόνων σε μια πλατφόρμα για πολυμέσα και άλλες εφαρμογές με μεγάλες απαιτήσεις σε εύρος ζώνης(bandwidth), οι οποίες απαιτούν υψηλότερα επίπεδα υπηρεσίας από αυτά που μπορεί να προσφέρει το συμβατικό Διαδίκτυο. Αν σε αυτούς τους παράγοντες, προσθέσουμε και το γεγονός

ότι οι πάροχοι περιεχομένου διακρίνουν ότι η απόδοση των δικτυακών τόπων αποτελεί βασικό παράγοντα για την επιτυχημένη παρουσία τους στο ηλεκτρονικό εμπόριο, μπορούμε να κατανοήσουμε την μεγάλη ώθηση που δίνεται στις υπηρεσίες Δικτύων Διανομής Περιεχόμενου (Content Delivery Networks).

Οι υπηρεσίες που παρέχονται από τα δίκτυα αυτά, αποτελούν μια πολλά υποσχόμενη λύση για να αντιμετωπιστεί η μεγάλη και διαρκώς αυξανόμενη κίνηση στο Διαδίκτυο. Τα Δίκτυα Διανομής Περιεχομένου αποτελούνται από ομάδες ενδιάμεσων εξυπηρετών (proxy-servers) τοποθετημένες σε καίριες θέσεις στο Διαδίκτυο, έτσι ώστε να εξασφαλίζεται πως μια αίτηση για ανάκτηση περιεχομένου (download), θα εξυπηρετείται πάντα από τον πιο «κοντινό» εξυπηρέτη (server).

Τα Δίκτυα αυτά, είναι ουσιαστικά ένα ενδιάμεσο επίπεδο ανάμεσα στους εξυπηρέτες και στους πελάτες (middleware) και χρησιμοποιούν τεχνικές ενδιάμεσης αποθήκευσης (caching), εξισορρόπησης φόρτου (load balancing) και διατήρησης αντιγράφων της πληροφορίας (replication). Τελευταία, παρατηρείται στην αγορά μια αυξανόμενη ζήτηση για υπηρεσίες παροχής υπηρεσιών Δικτύων Διανομής Περιεχομένου και στον τομέα αυτό έχουν δραστηριοποιηθεί μεγάλες εταιρείες. Η ραγδαία αύξηση αυτή δηλώνει μια δυναμική στην περιοχή των δικτύων αυτών, γεγονός που αποτελεί κίνητρο για ερευνητική δουλειά πάνω σε ζητήματα που παραμένουν ακόμα ανοιχτά ή φαίνεται να χρήζουν βελτιστοποίησης.

Η βελτίωση της αρχιτεκτονικής που προτείνεται, στην ουσία επιτρέπει στον πάροχο της υπηρεσίας να προβλέψει τους πόρους που θα ζητήσει ο χρήστης, και να τους προωθήσει στους κατάλληλους εξυπηρέτες, πριν αυτοί ζητηθούν από τον χρήστη, ώστε να μειωθεί ο χρόνος απόκρισης. Για να επιτευχθεί αυτό όμως, θα πρέπει να γίνει όσο το δυνατόν καλύτερη πρόβλεψη των πόρων που θα ζητήσουν οι χρήστες.

Ο σκόπος, λοιπόν, της συγκεκριμένης πτυχιακής εργασίας είναι να σχεδιαστεί και να υλοποιηθεί ένας τέτοιος αλγόριθμος, ο οποίος θα πρέπει να ικανοποιεί τις παρακάτω απαιτήσεις :

- Μείωση του ποσοστού του συνολικού ιστοχώρου που θα αντιγραφεί στους περιφερειακούς εξυπηρέτες (surrogates)
- Αύξηση του ποσοστού επιτυχίας, στην πρόβλεψη των πόρων που θα ζητηθούν από τους χρήστες.

Επίσης θα υλοποιηθεί ένα υποσύστημα εξομοίωσης, ώστε να αξιολογηθεί η υλοποίηση που προτείνεται.

1.2. Περιγραφή Αρχιτεκτονικής Δικτύων Διανομής Περιεχομένου

Στις επόμενες παραγράφους, παρουσιάζονται οι βασικές τεχνικές, στις οποίες στηρίζονται τα Δίκτυα Διανομής Περιεχομένου, η αρχιτεκτονική τους και τα βασικά υποσυστήματά τους.

1.2.1. Ενδιάμεσοι Εξυπηρέτες (Proxy-Servers)

Ο ενδιάμεσος εξυπηρέτης είναι ένα είδος μνήμης ενδιάμεσης αποθήκευσης (buffer) ανάμεσα στον υπολογιστή και στους πόρους του Διαδικτύου, στους οποίους επιθυμούμε πρόσβαση. Βρίσκεται δηλαδή ανάμεσα σε μια εφαρμογή του πελάτη (client application), όπως π.χ. είναι ένας φυλλομετρητής ιστού (web browser), και σε έναν πραγματικό εξυπηρέτη.

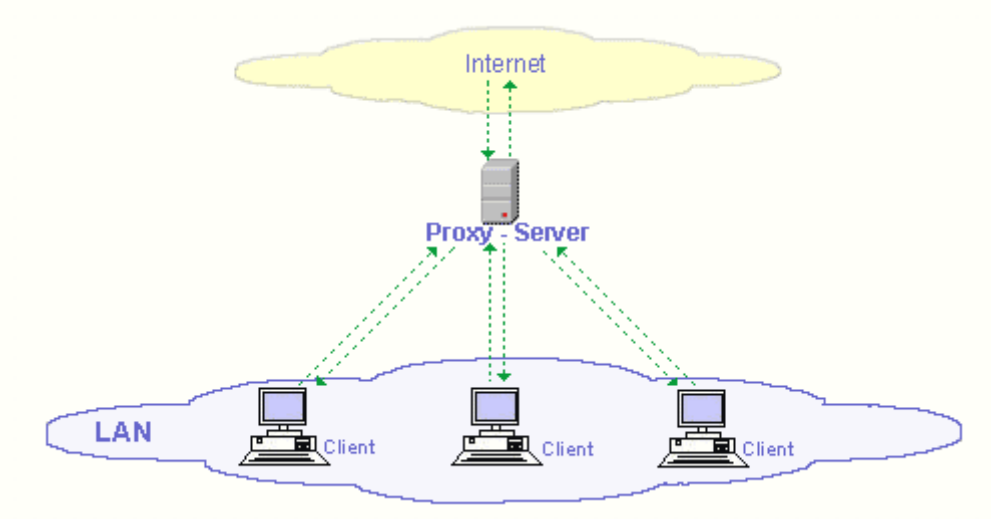
Συμπεριφέρεται απέναντι στον πραγματικό εξυπηρέτη ως πελάτης και απέναντι στον πελάτη ως εξυπηρέτης. Συγκεντρώνει και αποθηκεύει αρχεία, τα οποία ζητούνται πιο συχνά από χιλιάδες χρήστες του διαδικτύου, σε μια ειδική βάση δεδομένων, που ονομάζεται “cache” και που αποτελεί μια ενδιάμεση αποθηκευτική μνήμη. Στην συνέχεια, οι ενδιάμεσοι εξυπηρέτες έχουν την ικανότητα να αυξήσουν την ταχύτητα της σύνδεσης ενός χρήστη στο Διαδίκτυο. Η ενδιάμεση μνήμη των εξυπηρετών αυτών ενδέχεται, λοιπόν, να περιέχει ήδη πληροφορίες που χρειάζεται κάποιος χρήστης κατά την διάρκεια της αίτησης, καθιστώντας έτσι δυνατό για τον εξυπηρέτη να τις μεταφέρει άμεσα. Παρεμβαίνει, λοιπόν, σε όλες τις αιτήσεις προς τον πραγματικό εξυπηρέτη και ελέγχει ποιές από αυτές μπορεί να ικανοποιήσει ο ίδιος, ελέγχοντας την τοπική του ενδιάμεση μνήμη. Αν όχι, τότε προωθεί την συγκεκριμένη αίτηση στον πραγματικό εξυπηρέτη. Η συνολική αύξηση κατά την πράξη μπορεί να είναι αρκετά μεγάλη.

Επίσης, οι ενδιάμεσοι εξυπηρέτες μπορούν να βοηθήσουν και σε περιπτώσεις, που κάποιοι ιδιοκτήτες των πόρων του διαδικτύου επιθυμούν να επιβάλλουν κάποιους περιορισμούς στους χρήστες, που προέρχονται από συγκεκριμένες χώρες ή γεωγραφικές περιοχές, δηλαδή να χρησιμοποιούνται για έλεγχο πρόσβασης.

Επιπρόσθετα, ανάμεσα στους ενδιάμεσους εξυπηρέτες υπάρχουν και οι ονομαζόμενοι ανώνυμοι ενδιάμεσοι εξυπηρέτες, οι οποίοι κρύβουν την διεύθυνση του Διαδικτύου (IP address) του χρήστη, προστατεύοντας τον έτσι από πιθανά προβλήματα, αφού αποφεύγεται η χωρίς άδεια πρόσβαση στον υπολογιστή μέσω του Διαδικτύου.

Οι ενδιάμεσοι εξυπηρέτες έχουν, λοιπόν, δύο βασικούς σκοπούς:

- να αυξήσουν την απόδοση: οι ενδιάμεσοι εξυπηρέτες μπορούν να βελτιώσουν σημαντικά την απόδοση για ένα σύνολο χρηστών. Αυτό ισχύει γιατί, όπως έχουμε ήδη προαναφέρει, ένας ενδιάμεσος εξυπηρέτης αποθηκεύει τα αποτελέσματα όλων των αιτήσεων σε συγκεκριμένο χρονικό διάστημα. Ας θεωρήσουμε την περίπτωση που και ο χρήστης Χ και ο χρήστης Υ έχουν πρόσβαση στον Παγκόσμιο Ιστό (World Wide Web) μέσω ενός ενδιάμεσου εξυπηρέτη. Κατ'αρχήν ο χρήστης Χ ζητάει μια συγκεκριμένη ιστοσελίδα, που ονομάζουμε σελίδα 1. Κάποια στιγμή αργότερα, ο χρήστης Υ ζητάει την ίδια σελίδα. Αντί, λοιπόν, να αποστείλει την αίτηση στον δικτυακό εξυπηρέτη, στον οποίο ανήκει η σελίδα 1, η οποία διαδικασία μπορεί να είναι χρονοβόρα, ο ενδιάμεσος εξυπηρέτης απλά επιστρέφει την σελίδα 1, που την έχει φορτώσει ήδη για τον χρήστη Χ. Δεδομένου ότι ο εξυπηρέτης αυτός είναι στο ίδιο δίκτυο με τον πελάτη-χρήστη, αυτή είναι μια κατά πολύ γρηγορότερη διαδικασία. Πραγματικοί ενδιάμεσοι εξυπηρέτες υποστηρίζουν εκατοντάδες ή ακόμα και χιλιάδες χρηστών. Μεγάλες υπηρεσίες του Διαδικτύου, όπως η America Online, MSN, και Yahoo, για παράδειγμα, απασχολούν μια σειρά από ενδιάμεσους εξυπηρέτες.
- να φιλτράρουν αιτήσεις:οι ενδιάμεσοι εξυπηρέτες μπορούν επίσης να χρησιμοποιηθούν για φιλτράρισμα αιτήσεων. Για παράδειγμα, μια εταιρεία ενδέχεται να χρησιμοποιεί έναν ενδιάμεσο εξυπηρέτη για να εμποδίζει στους εργαζόμενους της την πρόσβαση σε ένα συγκεκριμένο σύνολο από ιστοσελίδες. Μπορεί, λοιπόν, ένας ενδιάμεσος εξυπηρέτης να αποτελέσει μέρος μιας πολιτικής προστασίας.

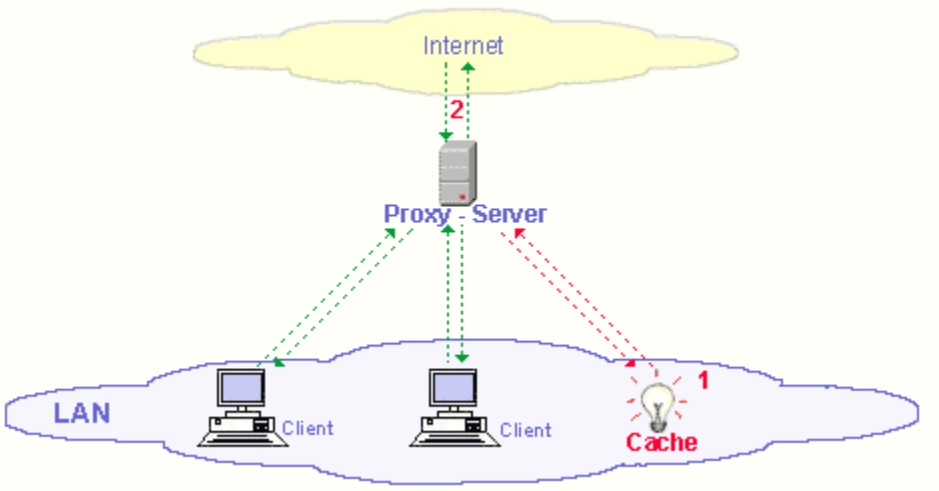


Εικόνα 1: Αρχιτεκτονική του συστήματος

1.2.2. Μηχανισμός Ενδιάμεσης Μνήμης (Caching)

Η ενδιάμεση μνήμη ορίζεται ως η τεχνική αποθήκευσης ιστοσελίδων και δικτυακών αρχείων με σκοπό την αργότερη επαναχρησιμοποίησή τους σε ένα σημείο του δικτύου, στο οποίο ο τελικός χρήστης έχει την δυνατότητα της πιο γρήγορης πρόσβασης.

Η τεχνική αυτή μπορεί να χρησιμοποιηθεί σε πολλά σημεία, συμπεριλαμβάνοντας και τους ενδιάμεσους εξυπηρέτες του Παρόχου Υπηρεσιών Διαδικτύου (Internet Service Provider) που χρησιμοποιεί ο χρήστης, αλλά και το τοπικό μηχάνημα του χρήστη.



Εικόνα 2: Λειτουργία ενός ενδιάμεσου εξυπηρέτη επιτελώντας την λειτουργία του μηχανισμού ενδιάμεσης μνήμης (caching).

Στην παραπάνω εικόνα, βλέπουμε την διαδικασία που ακολουθείται από έναν ενδιάμεσο εξυπηρέτη, ύστερα από την αίτηση ενός πελάτη. Ο εξυπηρέτης αυτός ελέγχει αν στην ενδιάμεση μνήμη του, που περιέχει παλαιότερες ανακτημένες σελίδες, υπάρχει η ζητούμενη σελίδα (1), αν ναι την επιστρέφει στον πελάτη πολύ πιο γρήγορα και χωρίς να επιβαρύνεται το δίκτυο επιπλέον, αν όχι ο εξυπηρέτης επικοινωνεί άμεσα με τον εξυπηρέτη που κατέχει το γνήσιο περιεχόμενο και ικανοποιεί την αίτηση του πελάτη (2).

Ο αντικειμενικός σκοπός είναι να γίνει αποτελεσματική και αποδοτική η χρήση των πόρων και να αυξηθεί η ταχύτητα της μεταφοράς του περιεχομένου στον τελικό χρήστη. Έτσι, λοιπόν, ανακτημένα δικτυακά αρχεία αποθηκεύονται για κάποιο χρονικό διάστημα έτσι ώστε η πρόσβαση σε αυτά να είναι πιο εύκολη σε όποια επόμενη αίτηση των χρηστών.

Ένα γνωστό πρόβλημα του μηχανισμού ενδιάμεσης μνήμης (caching) αποτελεί η ανάκτηση του πιο πρόσφατου αντιγράφου ενός αρχείου (cache consistency). Ο τρόπος με τον οποίο αυτό μπορεί να επιλυθεί εναπόκειται στο μηχανισμό της ενδιάμεσης μνήμης, η οποία κάνοντας ένα σύντομο έλεγχο, συγκρίνει την ημερομηνία του γνήσιου αρχείου με αυτήν του αρχείου που η ίδια έχει αποθηκευμένο. Αποτέλεσμα της διαδικασίας αυτής είναι η παροχή του πιο πρόσφατου αντιγράφου στο χρήστη.

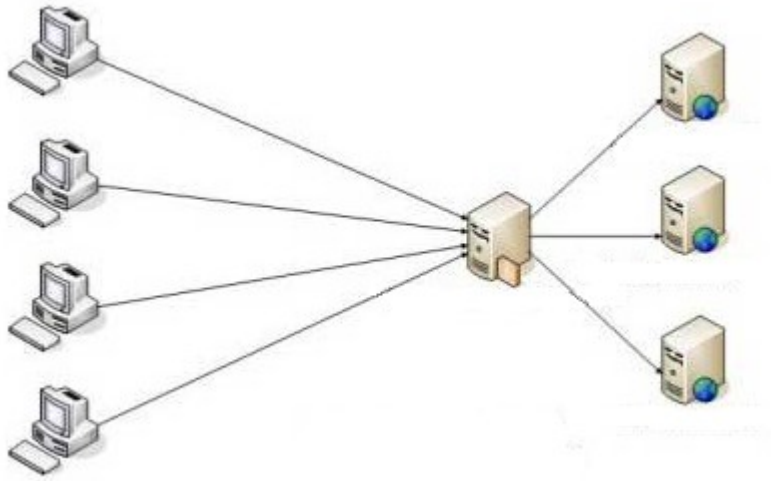
Η τεχνική της ενδιάμεσης μνήμης χρησιμοποιείται και από τους φυλλομετρητές ιστού (web browsers), προκειμένου να διατηρούν μνήμη με ανακτημένες σελίδες, σε περίπτωση πιθανών μελλοντικών ίδιων αιτήσεων. Αν η αίτηση δεν ικανοποιηθεί από αυτή την ενδιάμεση μνήμη, η αίτηση κατευθύνεται στον εξυπηρέτη της μνήμης αυτής και αν ούτε και σ'αυτό το σημείο δεν υπάρχει κάποιο θετικό αποτέλεσμα, τότε η αίτηση οδηγείται στον εξυπηρέτη, που περιέχει τις αυθεντικές πληροφορίες.

1.2.3. Εξισορρόπηση Φόρτου (Load Balancing)

Σήμερα, στην εποχή της υψηλής τεχνολογίας, το Διαδίκτυο (Internet) γίνεται πολύ γρήγορα η αγαπημένη πηγή συλλογής γνώσεων ακόμα και διασκέδασης. Ιδιοκτήτες δημοφιλών ιστοχώρων αντιμετωπίζουν προβλήματα επειδή οι εξυπηρέτες τους δεν μπορούν να ικανοποιήσουν τον μεγάλο αριθμό επισκεπτών του χώρου τους, που τον επισκέπτονται την ίδια χρονική στιγμή. Το πρόβλημα αυτό αντιμετωπίζεται με την τεχνική της εξισορρόπησης φόρτου (load balancing).

Ο ορισμός της εξισορρόπησης φόρτου είναι η διαδικασία διαμοιρασμού της ποσότητας εργασίας που ένας υπολογιστής οφείλει να επιτελέσει ανάμεσα σε έναν ή ακόμα και περισσότερους υπολογιστές, έτσι ώστε περισσότερες εργασίες να ολοκληρωθούν στην ίδια ποσότητα χρόνου και γενικά όλη η διαδικασία να ολοκληρωθεί πιο γρήγορα. Με τον τρόπο αυτό αυξάνεται η αποδοτικότητα του συστήματος.

Στην παρακάτω εικόνα δίνεται μια σχηματική αναπαράσταση της λειτουργίας της εξισορρόπησης φόρτου, στην οποία οι αιτήσεις των χρηστών δρομολογούνται σε έναν κεντρικό εξυπηρέτη-εξισορροπητή φόρτου, ο οποίος στην συνέχεια διαμοιράζει τις αιτήσεις προς εξυπηρέτηση στους διάφορους εξυπηρέτες ανάλογα με τον φόρτο που έχει ο καθένας την συγκεκριμένη χρονική στιγμή.



Εικόνα 3: Αρχιτεκτονική εξισορρόπησης φόρτου

Υπάρχουν πολλά πλεονεκτήματα από την χρήση της τεχνικής της εξισορρόπησης φόρτου. Τα δύο πιο αντιπροσωπευτικά είναι τα εξής:

- ο χρόνος απόκρισης – Όταν υπάρχει εξισορρόποηση φόρτου, δηλαδή όταν υπάρχουν παραπάνω από ένας υπολογιστής για την ικανοποίηση των πελατών, τότε ένα από τα πλεονεκτήματα είναι η μεγάλη διαφορά στην διάρκεια φόρτωσης της σελίδας.
- σταθερότητα – Με την εξισορρόπηση φόρτου επιτυγχάνεται σταθερότητα, διότι σε περίπτωση που η λειτουργία ενός υπολογιστή τερματιστεί πλήρως, τότε οι υπόλοιποι θα συνεχίσουν να λειτουργούν κανονικά, χωρίς οι επισκέπτες να αντιληφθούν το παραμικρό πρόβλημα.
- μείωση της κατανάλωσης των πόρων του συστήματος

Ο φόρτος (αριθμός αιτήσεων, αριθμός χρηστών, κτλ.) διαμοιράζεται μέσα στο δίκτυο έτσι ώστε καμία ανεξάρτητη συσκευή να μην επιβαρύνεται από την μεγάλη κίνηση του δικτύου, με πιθανό αποτέλεσμα την παύση λειτουργίας της.

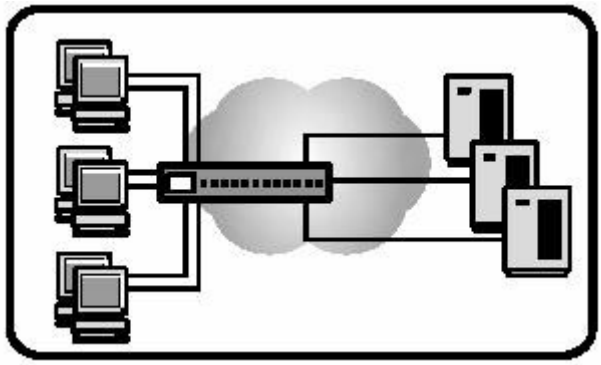
Η τεχνική της εξισορρόπησης φόρτου είναι αρκετά σημαντική για δίκτυα που είναι δύσκολο να υπολογιστεί εκ των προτέρων ο αριθμός των αιτήσεων που γίνονται σε έναν εξυπηρέτη. Τοποθετώντας, λοιπόν, δύο εξυπηρέτες προκειμένου να ικανοποιηθούν όλες οι αιτήσεις των χρηστών, σε περίπτωση αύξησης του φόρτου εργασίας στον έναν από τους δύο, γίνεται προώθηση των αιτήσεων στο δεύτερο

εξυπηρέτη. Αυτή η διαδικασία διαμοιρασμού των αιτήσεων σε μια ομάδα από εξυπηρέτες είναι γνωστή ως εξισορρόπηση φόρτου εξυπηρέτη (server load balancing).

Η εξισορρόπηση φόρτου περιλαμβάνει ακόμα και ανακατευθύνσεις αιτήσεων σε περίπτωση που ένας εξυπηρέτης σταματά να λειτουργεί.

Τα πλεονεκτήματα του είναι μεγάλης σημασίας στα δίκτυα:

- υψηλή αποδοτικότητα
- μεγάλη διαθεσιμότητα και ανάκαμψη προβλημάτων



Εικόνα 4: Λειτουργικότητα της εξισορρόπησης φόρτου

Σύμφωνα με την παραπάνω εικόνα, οι αιτήσεις των χρηστών στέλνονται σε μια συσκευή εξισορρόπησης φόρτου, η οποία αποφασίζει ποιος εξυπηρέτης είναι περισσότερο διαθέσιμος για την ικανοποίηση της αίτησης. Έπειτα προωθεί την αίτηση στον συγκεκριμένο εξυπηρέτη.

1.2.4. Αντιγραφή πληροφορίας (Replication)

Μια άλλη τεχνική που χρησιμοποιείται στα Δίκτυα Διανομής Περιεχομένου είναι αυτή της αντιγραφής (replication) πόρων. Δημιουργούνται δηλαδή πολλαπλά αντίγραφα της ίδιας πληροφορίας με στόχο τη βελτίωση της ταχύτητας πρόσβασης στους ιστοχώρους. Με τον τρόπο αυτό μειώνονται οι καθυστερήσεις πρόσβασης των χρηστών σε μια ιστοσελίδα και αυξάνεται η διαθεσιμότητα αυτής.

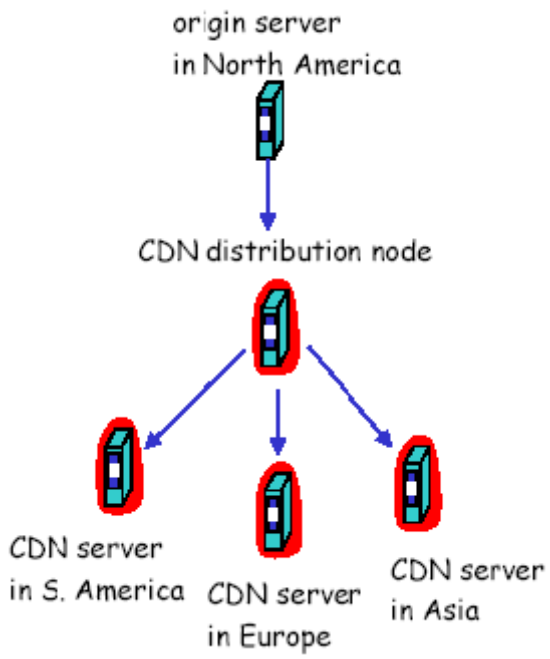
Μετά την εγκατάσταση στο διαδίκτυο ενός συνόλου εξυπηρετών των Δικτύων Διανομής Περιεχομένου από μια τέτοια εταιρεία, το δίκτυο αυτό αντιγράφει τα περιεχόμενα των πελατών του στους εξυπηρέτες αυτού του δικτύου. Όταν ο πελάτης μεταβάλλει και

ενημερώσει το περιεχόμενό του, τότε και το δίκτυο ενημερώνει το περιεχόμενο των εξυπηρετών. Ο στόχος της αντιγραφής είναι, όπως έχει προαναφερθεί, να αυξηθεί η αποδοτικότητα του δικτύου και η ταχύτητα εξυπηρέτησης της αίτησης ενός χρήστη, αφού πλέον οι αιτήσεις δεν κατευθύνονται στον εξυπηρέτη με το γνήσιο περιεχόμενο, αλλά στους εξυπηρέτες-αντιγραφείς.

1.3. Αρχιτεκτονική ενός Δικτύου Διανομής Περιεχομένου (CDN)

Τα Δίκτυα Διανομής Περιεχομένου (CDNs) είναι μια αποδοτική προσέγγιση για την βελτίωση της ποιότητας των υπηρεσιών του Διαδικτύου. Ένα τέτοιο δίκτυο αντιγράφει το περιεχόμενο από την γνήσια προέλευση στους αντιγραφείς-εξυπηρέτες, οι οποίοι είναι διασκορπισμένοι στο Διαδίκτυο, και εξυπηρετεί μια αίτηση από τον αντιγραφέα-εξυπηρέτη, ο οποίος θα είναι πιο κοντά στην προέλευση της αίτησης.

Ως Δίκτυο Διανομής Περιεχομένου (Content Delivery Network) ορίζεται ως ένα δίκτυο από εξυπηρέτες, οι οποίοι είναι παραταγμένοι με τέτοιο τρόπο στο διαδίκτυο, ώστε οι ιδιοκτήτες διακτυακού υλικού να είναι σε θέση να διανείμουν το περιεχόμενό τους στους χρήστες πιο γρήγορα και αποτελεσματικά. Είναι, στην ουσία, ένα δίκτυο τοποθετημένο κατάλληλα γεωγραφικά που καθιστά δυνατή την γρήγορη και αξιόπιστη ανάκτηση περιεχομένου από οποιοδήποτε τελικό χρήστη.



Εικόνα 5: Παράδειγμα λειτουργικότητας ενός Δικτύου Διανομής Περιεχομένου

Οι κόμβοι ενός τέτοιου δικτύου τοποθετούνται σε πολλαπλές θέσεις, συνεργάζονται μεταξύ τους για να ικανοποιήσουν αιτήσεις χρηστών για περιεχόμενο, επιτελούν την διανομή, βελτιώνοντας έτσι την διαδικασία μεταφοράς. Η βελτιστοποίηση αφορά την μείωση σε κόστος εύρους ζώνης, σε βελτίωση της απόδοσης του τελικού χρήστη ή ακόμα και στα δύο.

Ο αριθμός των κόμβων και των εξυπηρετών που αποτελούν ένα τέτοιο δίκτυο ποικίλει ανάλογα με την αρχιτεκτονική. Μερικές φορές αυτός ο αριθμός φτάνει τις χιλιάδες των κόμβων με δεκάδες χιλιάδες από εξυπηρέτες.

Τα Δίκτυα Διανομής Περιεχομένου χρησιμοποιούν τεχνολογίες υποδομής όπως μηχανισμούς ενδιάμεση μνήμη (caching) για να ωθούν το αντιγραμμένο περιεχόμενο κοντά στους τελικούς χρήστες.Η παγκόσμια εξισορρόπηση φόρτου έχει ως σκοπό οι χρήστες να οδηγηθούν στην βέλτιστη πηγή περιεχομένου.

Οι αιτήσεις για περιεχόμενο κατευθύνονται, με βάση κάποια κριτήρια που θα αναλυθούν στην συνέχεια, σε κόμβους που είναι οι βέλτιστοι, υπό κάποια έννοια.

Όταν ο στόχος είναι η βέλτιστη απόδοση, τότε οι θέσεις που θα εξυπηρετήσουν τον χρήστη ως προς κάποιες αιτήσεις περιεχομένου αρκετά γρήγορα και άμεσα επιλέγονται κάθε φορά. Η επιλογή κρίνεται σύμφωνα με τον ελάχιστο αριθμό των δικτυακών

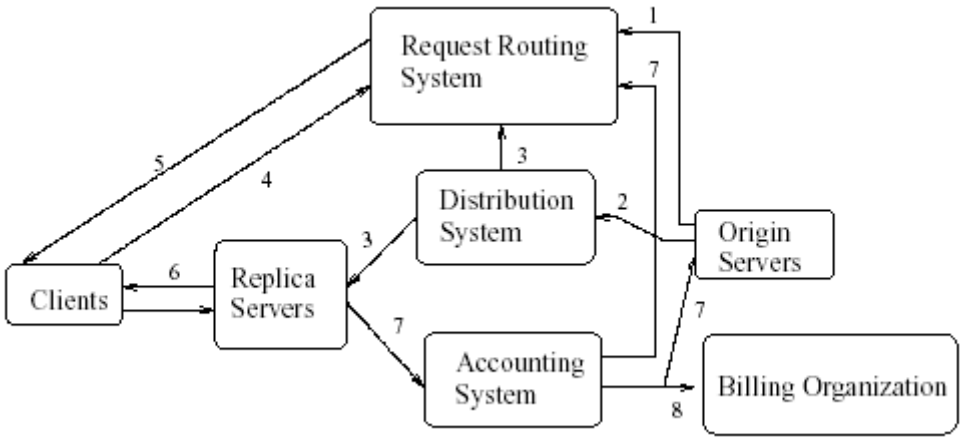
δευτερολέπτων που χωρίζουν τις θέσεις των εξυπηρετών ενός τέτοιου δικτύου από τον αιτούντα, έτσι ώστε να βελτιστοποιείται η μεταφορά. Όταν ο στόχος είναι το ελάχιστο κόστος, τότε οι θέσεις που είναι λιγότερο ακριβές θα εξυπηρετήσουν τις αιτήσεις.

Συχνά αυτοί οι δύο στόχοι τείνουν να ευθυγραμμιστούν, από την στιγμή που οι εξυπηρέτες που είναι κοντά στον τελικό χρήστη έχουν κάποιες φορές πλεονέκτημα όσον αφορά το κόστος, ίσως επειδή βρίσκονται στο ίδιο δίκτυο με τον τελικό χρήστη.

1.3.1. Υποσυστήματα Αρχιτεκτονικής Δικτύων Διανομής Περιεχομένου

Στην παράγραφο αυτή, θα γίνει αναλυτική παρουσίαση των υποστημάτων της αρχιτεκτονικής ενός Δικτύου Διανομής Περιεχομένου καθώς και του τρόπου με τον οποίο αυτή λειτουργεί.

Τα βήματα της λειτουργίας επεξηγούνται λεπτομερώς στην συνέχεια:



Εικόνα 6: Υποστήματα Αρχιτεκτονικής Δικτύου Διανομής Περιεχομένου

1. Ο αρχικός εξυπηρέτης (Origin Server) αναθέτει τον URI χώρο ονομάτων του για αντικείμενα που θα κατανεμηθούν και θα διαμοιραστούν από το δίκτυο διανομής, στο σύστημα δρομολόγησης αιτήσεων (Request Routing System).
2. Ο αρχικός εξυπηρέτης εκδίδει στο σύστημα διανομής (Distribution System) το περιεχόμενό του, το οποίο πρόκειται να διανεμηθεί και να μεταφερθεί.
3. Το σύστημα διανομής μεταφέρει το περιεχόμενο στους εξυπηρέτες-αντιγραφείς (Replica Servers). Επιπρόσθετα, το σύστημα αλληλεπιδρά με το σύστημα

δρομολόγησης αιτήσεων (Request Routing System) μέσω ανατροφοδότησης για να βοηθήσει στην διαδικασία επιλογής του εξυπηρέτη-αντιγραφέα για την ικανοποίηση της αίτησης ενός πελάτη (client).

4. Ο πελάτης (client) ζητά αρχεία/έγγραφα από αυτό που αυτός βλέπει ως αρχικό εξυπηρέτη. Παρ'όλ'αυτά, εξ'αιτίας της ανάθεσης του URI χώρου ονομάτων, η αίτηση στην ουσία κατευθύνεται στο σύστημα δρομολόγησης αιτήσεων (Request Routing System).
5. Το σύστημα δρομολόγησης αιτήσεων (Request Routing System) κατευθύνει την αίτηση στον κατάλληλο εξυπηρέτη, που περιέχει το αντίγραφο, μέσα στο δίκτυο διανομής.
6. Ο επιλεγμένος εξυπηρέτης-αντιγραφέας μεταφέρει το επιθυμητό περιεχόμενο στον πελάτη. Επιπλέον, ο εξυπηρέτης-αντιγραφέας στέλνει λογιστικές πληροφορίες για το μεταφερόμενο περιεχόμενο στο λογιστικό σύστημα (Accounting System).
7. Το λογιστικό σύστημα συναθροίζει, επεξεργάζεται και μετατρέπει τις λογιστικές πληροφορίες σε στατιστικά δεδομένα και λεπτομερείς εγγραφές περιεχομένου προς χρήση από τους αρχικούς εξυπηρέτες και από το σύστημα λογαριασμών (Billing Organisation). Τα στατιστικά δεδομένα χρησιμοποιούνται επίσης ως ανατροφοδότηση και στο σύστημα δρομολόγησης αιτήσεων (Request Routing System).
8. Το σύστημα λογαριασμών (Billing Organisation) χρησιμοποιεί τις λεπτομερείς εγγραφές περιεχομένου για να οργανώσει τα μέρη που έλαβαν μέρος στην διανομή του περιεχομένου και στην διαδικασία της μεταφοράς.

1.4. Χρήσεις και Παραδείγματα

Οι πάροχοι Δικτύων Διανομής Περιεχομένου, όπως η Akamai, η Speedera και η Digital Island έχουν συμβάλλει σε αυτήν την μεγάλη αλλαγή στο χώρο του Διαδικτύου. Αντιγράφουν το περιεχόμενο σε δικά τους δίκτυα με εξυπηρέτες ενδιάμεσης μνήμης, κατανεμημένους γεωγραφικά σε όλο το διαδίκτυο, φέρνοντας το, έτσι, πιο κοντά στους χρήστες και επιταχύνοντας την διαδικασία μεταφοράς δεδομένων.

Η Akamai, για παράδειγμα, έχει περίπου 12.000 εξυπηρέτες σε 62 χώρες, ενώ ο αριθμός των εταιρειών που χρησιμοποιούν αυτά τα δίκτυα έχει διπλασιαστεί σε 2.500

τον τελευταίο χρόνο. Στην συνέχεια, θα αναφέρουμε κάποιες εταιρείες που προσφέρουν τέτοιου είδους αρχιτεκτονική.

1.4.1 Akamai Technologies

Η Akamai Technologies είναι μια εταιρεία που προσφέρει μια κατανεμημένη πλατφόρμα για παγκόσμιο διαδικτυακό περιεχόμενο και την μεταφορά του. Στην ουσία, δρα σαν ένας διαχειριστής κίνησης και εξυπηρέτης περιεχομένου για το διαδίκτυο. Τοποθετούνται εξυπηρέτες σε όλο τον κόσμο, ενσωματωμένοι σε παρόχους διαδικτυακών υπηρεσιών (ISPs) σε πολλά σημεία, και προσφέρουν λογισμικό, εικόνες, videos και άλλα μεγάλα αρχεία, με πολύ πιο γρήγορο και αξιόπιστο τρόπο, σε σχέση με έναν κεντρικοποιημένο μοναδικό εξυπηρέτη.

Η Akamai, με διαφάνεια αντιγράφει το περιεχόμενο, που βρίσκεται αποθηκευμένο στους εξυπηρέτες των πελατών της. Παρόλο που το όνομα περιοχής (domain name) είναι το ίδιο, οι IP διευθύνσεις δείχνουν στον εξυπηρέτη της Akamai και όχι στου πελάτη.

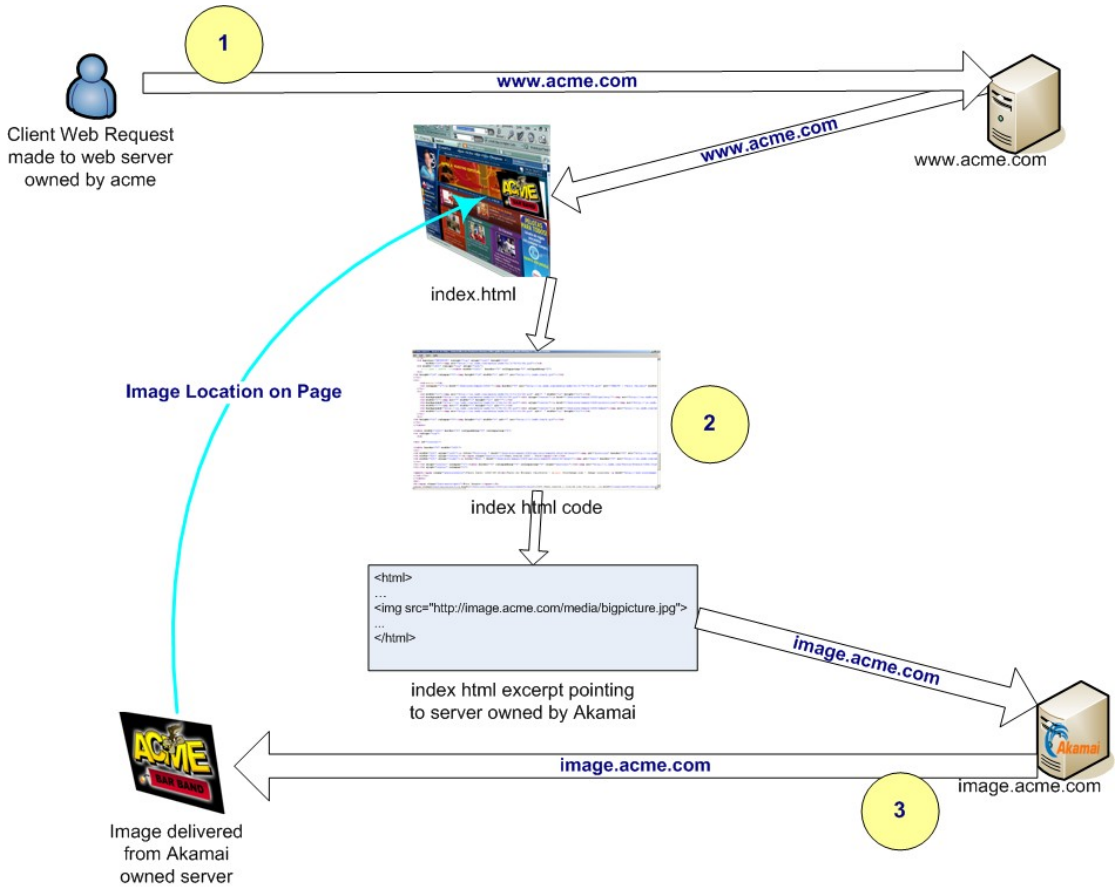
Η πλατφόρμα διανομής της Akamai είναι μία από τις μεγαλύτερες στον κόσμο. Μεταφέρει όλων των τύπων δικτυακό περιεχόμενο και εφαρμογές σε πάνω από 2.000 πελάτες και σε 20.000 περιοχές. Μεταφέρουν περιεχόμενο από πάνω από 20.000 εξυπηρέτες σε 2.800 τοποθεσίες, σε 660 πόλεις, πάνω από 70 χώρες. Οι εξυπηρέτες βρίσκονται τοπολογικά σε πάνω από 900 διαφορετικά δίκτυα.

Στην λίστα των πελατών της, περιλαμβάνονται η Yahoo, Google, Microsoft, Apple, Amazon, McAfee, Adobe, IBM, Xerox, FedEx, BBC News, Reuters, American Express, U.S. Air Force, MTV, White House.

Για να μπορέσει να χειριστεί την αυξημένη ζήτηση, η Akamai χρησιμοποίησε την τεχνική της εξισορρόπησης, όσον αφορά τους εξυπηρέτες, έτσι ώστε να εξασφαλιστεί ότι υπάρχει η απαραίτητη χωρητικότητα για το περιεχόμενο, εκεί που η ζήτηση είναι μεγαλύτερη.

Στο παρακάτω σχήμα, παρουσιάζεται ένα παράδειγμα χρήσης της αρχιτεκτονικής της Akamai. Πιο συγκεκριμένα, παρουσιάζεται η ιστοσελίδα κάποιου πελάτη της Akamai, στην οποία συγκεκριμένο περιεχόμενο μέσα σε αυτήν, δεν δείχνει στους εξυπηρέτες, που ανήκουν στον ίδιο τον πελάτη, αλλά σε κάποιους που ανήκουν στο δίκτυο της Akamai. Αυτό σημαίνει ότι αν ένας χρήστης του Διαδικτύου αποστείλει μια αίτηση

περιεχομένου προς τον πελάτη της Akamai, αυτή η αίτηση δεν θα εξυπηρετηθεί από τους εξυπηρέτες του πελάτη (οι οποίοι περιέχουν το γνήσιο περιεχόμενο), αλλά από τους εξυπηρέτες της Akamai, οι οποίοι έχουν αντιγράψει το περιεχόμενο των πελατών της.Είναι σημαντικό να τονίσουμε ότι παρόλο που το όνομα περιοχής είναι το ίδιο (www.acme.com), οι IP διευθύνσεις δείχνουν στην Akamai και όχι στην ACME.



Εικόνα 7: Παράδειγμα λειτουργίας ενός Δικτύου Διανομής Περιεχομένου

Βήμα1: Ο φυλλομετρητής του πελάτη κάνει μια αίτηση για μια ιστοσελίδα της ACME. Επιστρέφεται η ιστοσελίδα index.html.

Βήμα2: Εάν ο html κώδικας εξεταστεί, μπορούμε να δούμε ότι υπάρχει μια σύνδεση σε μια εικόνα, αποθηκευμένη στην Akamai, στην περιοχή image.acme.com.

Βήμα3: Καθώς ο φυλλομετρητής επεξεργάζεται τον html κώδικα, φέρνει την εικόνα bigpicture.jpg από την image.acme.com.

1.4.2 Digital Island

Η Digital Island ήταν ένας πλήρης πάροχος ιδιωτικής δικτύωσης και CDN. Από την δημιουργία της το 1996, η Digital Island φιλοξένησε με μεγάλη διαθεσιμότητα δικτυακές εφαρμογές σε όλο τον κόσμο. Είχε μεγάλη επιτυχία στην μεταφορά δικτυακών υπηρεσιών μεγάλη ποιότητα. Αγοράστηκε από την Cable & Wireless America. Τα κεφάλαια και οι πελάτες της Cable & America αγοράστηκαν έπειτα από την SAVVIS, που πούλησε την CDN αρχιτεκτονική στην Level 3 Communications τον Ιανουάριο του 2007.

1.4.3 Mirror Image

Η Mirror Image, θυγατρική εταιρεία της Xcelera Inc. παραδίδει υπηρεσίες υποδομής Διαδικτύου από το 1999. Η υποδομή του δικτύου διανομής περιεχομένου της Mirror Image, παρέχοντας απεριόριστη χωρητικότητα, εγγυάται διαθεσιμότητα και υψηλή απόδοση, ακόμη και κατά τη διάρκεια περιόδων υψηλού φόρτου. Έχει την δυνατότητα να αποθηκεύει στατικό ή δυναμικό περιεχόμενο σε ένα παγκόσμιο δίκτυο υψηλών αποδόσεων εξυπηρετών, με αποτέλεσμα να αποφορτώνεται ο γνήσιος εξυπηρέτης, να αποφεύγεται η συμφόρηση του Διαδικτύου και να επιταχύνεται η εξυπηρέτηση των χρηστών, μεταφέροντας με χαμηλό κόστος το περιεχόμενο του γνήσιου εξυπηρέτη σε εκατομμύρια χρήστες παγκοσμίως.

Η Mirror Image παρέχει ένα δίκτυο διανομής περιεχομένου για ψηφιακά δεδομένα και εξασφαλίζει την καλύτερη δυνατή εξυπηρέτηση των χρηστών επιτρέποντας στις επιχειρήσεις την ελαχιστοποίηση της επένδυσης σε υποδομή ιστοχώρου και την μείωση των λειτουργικών δαπανών τους.

1.4.3 RealNetworks

Η RealNetworks (πρώτα γνωστή ως Progressive Networks) δημιουργήθηκε από έναν πρώην συνεργάτη της Microsoft το 1995. Ο αρχικός στόχος της εταιρείας ήταν η παροχή ενός δικτύου διανομής πολιτικού προοδευτικού περιεχομένου. Γρήγορα εξελίχθηκε σε ένα τεχνολογικό εγχείρημα με σκοπό την εναλλακτική μετάδοση ηχητικών μέσων. Η Progressive Networks μετονομάστηκε σε RealNetworks τον Σεπτέμβρη του 1997 και εξελίχθηκε σε παροχέα υλικού (software) και υπηρεσιών.

ΚΕΦΑΛΑΙΟ 2

ΣΚΟΠΟΣ ΤΗΣ ΕΡΓΑΣΙΑΣ

2.1. Σχετικές εργασίες στα Δίκτυα Διανομής Περιεχομένου

Τα δίκτυα διανομής περιεχομένου (CDN) ανοίγουν τον ορίζοντα προς νέες βελτιώσεις της υπάρχουσας τεχνολογίας. Γι'αυτό τον λόγο, μέχρι σήμερα έχουν γίνει αρκετές θεωρητικές και πρακτικές μελέτες από έναν μεγάλο αριθμό επιστημόνων. Στη συνέχεια παρουσιάζονται ερευνητικές εργασίες στο γνωστικό αντικείμενο, που εστιάζουν σε βελτιώσεις της αρχιτεκτονικής, της εφαρμογής και της υλοποίηση των Δικτύων Διανομής.

2.1.1. Μέθοδοι ομαδοποίησης (Clustering) και Markov σε Δίκτυα Διανομής Περιεχομένου

Ο Thanh Vinh Nguyen στη διατριβή του [1], πρότεινε τη χρήση της ομαδοποίησης περιεχομένου, ομαδοποιώντας παρόμοια στοιχεία περιεχομένου σε ομάδες (clusters). Κατ' αυτό τον τρόπο, προτείνει ένα νέο ιεραρχικό σχέδιο ομαδοποίησης (clustering), όπου τα στοιχεία είναι συγκεντρωμένα, βασισμένα στις απαιτήσεις των πόρων και τη χωρική διανομή ζήτησης. Με την ομαδοποίηση παρόμοιων αντικειμένων περιεχομένου, που στην συνέχεια θεωρούνται σαν ενιαίο αντικείμενο, πετυχαίνεται μείωση της πολυπλοκότητας

Πρόσφατα, μια ερευνητική έκθεση έγινε από το IST [2], αναφερομένη σε έναν νέο μηχανισμό clustering που χρησιμοποιεί υβριδικό CDN - P2P σύστημα, το οποίο έχει δύο κύριους συντελεστές. Το πρώτο μέρος είναι η ομαδοποίηση των εξυπηρετών ενός δικτύου διανομής περιεχομένου, όσον αφορά το δίκτυό τους, και το δεύτερο μέρος είναι η ομαδοποίηση των ομότιμων όσον αφορά το δίκτυο από όπου συνδέονται. Το τελικό μέρος του μηχανισμού ομαδοποίησης είναι να υπάρχει συντονισμός μεταξύ των δικτύων και των ομοτίμων. Όταν μια σύνδεση αποτυγχάνει με έναν ομότιμο-πηγή κατά τη διάρκεια μιας συνόδου, το σύστημα εντοπίζει τον κοντινότερο εξυπηρέτη του δικτύου και υποστηρίζει την επικοινωνία με την παροχή των απαραίτητων δεδομένων. Επιτυγχάνεται με αυτόν τον τρόπο μείωση του φόρτου και αύξηση της αξιοπιστίας στα ομότιμα δίκτυα, λύνοντας παράλληλα προβλήματα εύρους ζώνης και χρήσης μνήμης.

Μια άλλη έκθεση με τον τίτλο «Cache Placement Methods Based on Client Demand Clustering» [3] από τους Paul Barford, Jin-Yi Cais και Jim Gast δημοσιεύεται και προτείνει την AS μέθοδο clustering, η οποία χρησιμοποιεί την πληροφορία από το καλύτερο μονοπάτι με βάση τους πίνακες δρομολόγησης Border Gateway Protocol (BGP) προκειμένου να δημιουργηθεί μια ιεραρχική ομαδοποίηση των αυτόνομων συστημάτων (AS's). Η μέθοδος προσθέτει επαναληπτικά τις μικρές ομάδες σε μεγαλύτερες βασιζόμενη στην ελαχιστοποίηση της απόστασης Hamming μεταξύ των γειτονικών συνόλων των ομάδων. Αυτή η μέθοδος οδηγεί σε ένα δάσος από AS δέντρα όπου κάθε ρίζα δέντρου ορίζεται ως ένας κύριος κόμβος του Διαδικτύου. Η απεικόνιση αυτή της AS συνδετικότητας είναι μια εξιδανίκευση της αληθινής δομής του Διαδικτύου. Ένα από τα "δυνατά" χαρακτηριστικά της AS μεθόδου ομαδοποίησης (clustering) είναι ότι είναι κατάλληλη για το πρόβλημα τοποθέτησης ενδιάμεσης μνήμης (cache).

Επιπλέον, στο πανεπιστήμιο California στο Berkeley, ο Yan Chen, ο Weiyu Chen, ο Luan Nguyen, ο Randy H.Katz και η Lili Qiu, στην Microsoft Research πρότειναν στην έκθεσή τους «Clustering Web Content for Efficient Replication» [4], την μη απευθείας και απευθείας αυξητική ομαδοποίηση, όπου η πρώτη υποθέτει ότι το ιστορικό πρόσβασης είναι διαθέσιμο ενώ η τελευταία προβλέπει το ιστορικό πρόσβασης βασισμένο στη δομή των συνδέσμων υπερ-κειμένου. Τα αποτελέσματα δείχνουν ότι η μη απευθείας μέθοδος ομαδοποίησης πλησιάζει την απόδοση της πλήρους επανα-ομαδοποίησης με πολύ λιγότερο κόστος. Η απευθείας αυξητική μέθοδος ομαδοποίησης και αντιγραφής πληροφορίας περιορίζουν το κόστος ανάκτησης από 4.6 - 8 φορές έναντι καμιάς αντιγραφής πληροφορίας και της τυχαίας αντιγραφής. Έτσι είναι ιδιαίτερα χρήσιμο να βελτιωθεί η διαθεσιμότητα εγγράφων κατά τη διάρκεια μεγάλου φόρτου.

Επίσης, ο Laurent Mathy, ο Steven Simpson, ο David Hutchison από το πανεπιστήμιο του Λάνκαστερ και ο Roberto Canonico από τον πανεπιστημίο Federico II παρέδωσαν έναν νέο αλγόριθμο ομαδοποίησης σε επίπεδο εφαρμογής στο βιβλίο τους «Scalable Adaptive Hierarchical Clustering» [5], ικανό για την δημιουργία ενός εκτιμένου δέντρου(spanning tree) μεταξύ των συμμετεχόντων σε συνόδους με πολλαπλές διανομές, χωρίς οποιαδήποτε συγκεκριμένη βοήθεια από τους δρομολογητές δικτύων. Αυτός ο αλγόριθμος είναι βασισμένος σε έναν μοναδικό καθορισμό των ζωνών γύρω από τους κόμβους και μια καινοτόμο προσαρμοστική διανομή των ομάδων. Η προτεινόμενη μέθοδος βρίσκει εφαρμογή σε πολλά πλαίσια όπου μεγάλης κλίμακας

δέντρα επικαλύψεων μπορούν να είναι χρήσιμα, π.χ. multicasting σε επίπεδο εφαρμογής, ομότιμα δίκτυα και δίκτυα διανομής περιεχομένου.

Το 2006, ένας νέος αλγόριθμος παρουσιάζεται από τους Oleksandr Grygorash, Yan Zhou και Zach Jorgensen από το πανεπιστήμιο της Νότιας Αλαμπάμα στο άρθρο τους «Minimum Spanning Tree Based Clustering Algorithms» [6]. Αυτός ο αλγόριθμος αναφέρεται στον αλγόριθμο ομαδοποίησης ελάχιστου εκτιμένου δέντρου(spanning tree), ο οποίος είναι ικανός να αναγνωρίζει ομάδες με ακανόνιστα όρια. Υπάρχουν δύο αλγόριθμοι ελάχιστου εκτιμένου δέντρου(spanning tree) που βασίζονται στην ομαδοποίηση. Ο πρώτος αλγόριθμος παράγει ένα κ-διαμέρισμα ενός συνόλου σημείων για οποιοδήποτε δεδομένο κ. Ο αλγόριθμος κατασκευάζει ένα ελάχιστο εκτιμένο δέντρο (spanning tree) του συνόλου σημείων και αφαιρεί ακμές που ικανοποιούν ένα προκαθορισμένο κριτήριο. Η διαδικασία επαναλαμβάνεται έως ότου παραχθούν κ συστάδες. Ο δεύτερος αλγόριθμος διαμερίζει ένα σύνολο σημείων σε ένα σύνολο ομάδων μεγιστοποιώντας την γενική μείωση σταθερής απόκλισης, χωρίς μια δεδομένη τιμή κ. Αυτοί οι αλγόριθμοι θα μπορούσαν να χρησιμοποιηθούν με μεγάλη επιτυχία στα δίκτυα διανομής περιεχομένου(CDNs).

2.1.2. HTTP Πακέτα

Ο Craig Wills, ο Gregory Trott και ο Mikhail Mikhailov από το τμήμα πληροφορικής στο πολυτεχνικό ίδρυμα του Worcester πρότειναν μια νέα αρχιτεκτονική στο πρωτόκολλο HTTP στο άρθρο τους «Using bundles for web content delivery» [7].

Τα πρωτόκολλα που χρησιμοποιούνται από την πλειοψηφία των συναλλαγών Ιστού είναι τα HTTP/1.0 και HTTP/1.1. Το HTTP/1.0 χρησιμοποιείται χαρακτηριστικά στις πολλαπλές ταυτόχρονες συνδέσεις μεταξύ του πελάτη και του εξυπηρέτη κατά τη διάρκεια της διαδικασίας της ανάκτησης ιστοσελίδας. Αυτή η προσέγγιση είναι μη αποδοτική λόγω του κόστους της εγκαταστάσης της σύνδεσης, των πολλαπλών TCP συνδέσεων αλλά και λόγω του φόρτου που επιβάλλεται στους εξυπηρέτες και στους δρομολογητές. Το HTTP/1.1 προσπαθεί να λύσει αυτά τα προβλήματα μέσω της χρήσης των διαρκών συνδέσεων, ιδιαίτερα με τη διοχέτευση(pipelining), από τους εξυπηρέτες του δικτύου, των πρακτόρων χρηστών και των μεσαζόντων. Επιπλέον, η χρήση των διαρκών συνδέσεων στο HTTP/1.1 δημιουργεί το πρόβλημα της μη-ντετερμινιστικής διάρκειας σύνδεσης. Οι φυλλομετρητές Ιστού συνεχίζουν να ανοίγουν

πολλαπλές ταυτόχρονες TCP συνδέσεις στον εξυπηρέτη. Η ιδέα είναι να ενσωματώσει το σύνολο των αντικειμένων που ενσωματώνονται σε μια ιστοσελίδα σε ένα ενιαίο αντικείμενο για την ανάκτηση από τους πελάτες. Με βάση τις μετρήσεις από τους πιο “δημοφιλείς” ιστοχώρους και από την εφαρμογή ενός τέτοιου μηχανισμού, φαίνεται ότι εάν τα αντικείμενα μιας ιστοσελίδας παραδίδονται στους πελάτες ως ένα ενιαίο αντικείμενο, ο χρόνος απόκρισης στους πελάτες είναι καλύτερος από αυτός που παρέχεται από τους ήδη χρησιμοποιούμενους μηχανισμούς. Τα αποτελέσματα δείχνουν ότι η χρήση των ενιαίων αντικειμένων παρέχει μικρότερους χρόνους ανάκτησης δεδομένων(download) και μειωμένες μέσες καθυστερήσεις αντικειμένου σε σύγκριση με το HTTP/1.0 και το HTTP/1.1. Αυτή η προσέγγιση μειώνει επίσης τον φόρτο στο δίκτυο και στους εξυπηρέτες. Η εφαρμογή του μηχανισμού δεν απαιτεί καμία αλλαγή στο HTTP πρωτόκολλο.

2.1.3. Κατανομή αιτήσεων χωρίς ανάγνωση του περιεχομένου, με μερική αντιγραφή πληροφορίας.

Υπάρχουν δύο τεχνικές που επιτρέπουν τη μερική αντιγραφή πληροφορίας ακόμα κι εκτελώντας την διανομή αιτημάτων αγνώστου περιεχομένου: η τεχνική των περιφερειακών εξυπηρετών (surrogates) και τα διανεμημένα συστήματα αρχείων.

Από την στιγμή που, η απαίτηση, ότι οποιοσδήποτε εξυπηρέτης στο σύνολο των αντιγράφων ενός ιστοχώρου πρέπει να είναι σε θέση να εκπληρώσει οποιοδήποτε αίτηση, είναι θεμελιώδης για ανακατεύθυνση του αιτήματος αγνώστου περιεχομένου, και οι δύο τεχνικές παρέχουν έναν τρόπο για έναν εξυπηρέτη να ικανοποιήσει την αίτηση ακόμα κι αν δεν έχει ένα αντίγραφο του ζητούμενου αντικειμένου.

2.1.3.1. Χρήση περιφερειακών εξυπηρετών ως εξυπηρέτες αντιγραφής πληροφορίας.

Η προσέγγιση με τους περιφερειακούς εξυπηρέτες (surrogates) διανέμει τις αιτήσεις μεταξύ των περιφερειακών εξυπηρετών (surrogates) που είναι διαφορετικοί από τους γνήσιους εξυπηρέτες. Ένας περιφερειακός εξυπηρέτης (surrogate) μπορεί να ικανοποιήσει οποιοδήποτε αίτηση. Εάν το ζητούμενο αντικείμενο είναι στην ενδιάμεση μνήμη του (cache), ο περιφερειακός εξυπηρέτης (surrogate) επεξεργάζεται την αίτηση

τοπικά. Διαφορετικά, ο περιφερειακός εξυπηρέτης (surrogate) ανακτά το αντικείμενο από τον γνήσιο εξυπηρέτη, το στέλνει στον πελάτη και επίσης το αποθηκεύει στην ενδιάμεση μνήμη του (cache) για μελλοντική χρήση εάν κριθεί απαραίτητο. Κατά συνέπεια, κάθε περιφερειακός εξυπηρέτης (surrogate) περιέχει μόνο ένα μερικό αντίγραφο του ιστοχώρου, το οποίο περιέχεται στην ενδιάμεση μνήμη του (cache). Επιπλέον, τα πιο δημοφιλή υποτμήματα ενός ιστοχώρου τείνουν να αντιγραφούν ευρύτερα στους περιφερειακούς εξυπηρέτες (surrogates). Όλοι οι μηχανισμοί ανακατεύθυνσης μπορούν να χρησιμοποιηθούν για να διανείμουν τις αιτήσεις μεταξύ των περιφερειακών εξυπηρετών (surrogates).

Το κύριο μειονέκτημα των περιφερειακών εξυπηρετών (surrogates) είναι ότι, όπως οι ενδιάμεσοι εξυπηρέτες, μπορούν συνήθως μόνο να διανείμουν το περιεχόμενο, που περιλαμβάνεται στην ενδιάμεση μνήμη τους (cache). Κατά συνέπεια, περιεχόμενο, το οποίο δεν περιλαμβάνεται στην ενδιάμεση μνήμη (cache), μπορεί να απαιτήσει άλλους τρόπους αντιγραφής. Επίσης, αν οι περιφερειακοί εξυπηρέτες (surrogates) δεν συνεργαστούν για να μοιραστούν τα δεδομένα στην ενδιάμεση μνήμη τους (cache), είναι δύσκολο να ελεγχθεί ο βαθμός αντιγραφής διάφορων αντικειμένων. Κάθε περιφερειακός εξυπηρέτης (surrogate) κανονικά τοποθετεί στην ενδιάμεση μνήμη του (cache) οποιοδήποτε αντικείμενο έχει ανακτηθεί (downloaded) ανεξάρτητα από τη “δημοτικότητά” του. Εντούτοις, από την στιγμή που το κόστος αποθήκευσης συνεχώς μειώνεται εντυπωσιακά, αυτός ο περιορισμός χάνει τη σημασία του.

2.1.3.2. Κατανεμημένα Υποστηρικτικά Συστήματα Αρχείων (Back-End Distributed File Systems)

Ένας απλός τρόπος να μπορεί κάθε αντιγραφέας-εξυπηρέτης να εκπληρώσει κάθε αίτηση χωρίς αποθήκευση ολόκληρου του ιστοχώρου είναι όλοι οι εξυπηρέτες δικτύου να τρέχουν πάνω από ένα διανεμημένο σύστημα αρχείων που παρέχει ένα κοινό περιβάλλον αρχείων και επιτρέπει σε κάθε εξυπηρέτη να έχει πρόσβαση σε κάθε αρχείο.

Ενώ ο Παγκόσμιος Ιστός (WWW) εμφανίζεται να είναι πραγματικά εξελικτικός μέσω της διανομής των αρχείων μεταξύ μιας σειράς αποκεντρωμένων εξυπηρετών, υπάρχουν περιπτώσεις όπου αυτή η μορφή διανομής φόρτου είναι και δαπανηρή και καταναλώνει πόρους. Σε τέτοιες περιπτώσεις μπορεί να είναι απαραίτητο να ορίσουμε έναν κεντρικά

τοποθετημένο HTTP εξυπηρέτη ως διαχειριστή. Λαμβάνοντας υπόψη την εκθετική αύξηση του Διαδικτύου γενικά, είναι όλο και περισσότερο πιο δύσκολο για τα πρόσωπα και τους οργανισμούς να προβλέπουν κατάλληλα τις μελλοντικές ανάγκες των HTTP εξυπηρετών τους, όσον αφορά το ανθρώπινο δυναμικό αλλά και τις απαιτήσεις υλικού. Αυτός είναι ο σκοπός του εγγράφου «A scalable Web server: The NCSA prototype», από Katz (1994) [8] για να περιγράψει τη μεθοδολογία που χρησιμοποιείται στο National Center for Supercomputing Applications για την οικοδόμηση ενός εξελικτικού εξυπηρέτη Παγκόσμιου Ιστού. Η υλοποίηση επιτρέπει τη δυναμική εξελιξιμότητα με την εναλλαγή μεταξύ μιας πληθώρας από HTTP εξυπηρέτες, που διαδοχικά προδιαγράφεται στο όνομα του www εξυπηρέτη.

Η μελέτη «Using wide-area file systems within the World-Wide Web», η οποία εκπονήθηκε από τον Spasojevic (1994) [9], προτείνει τη χρήση ενός συστήματος αρχείων για την αποθήκευση και την ανάκτηση εγγράφων. Παρουσιάζει ότι το μεγαλύτερο μέρος της λειτουργίας του Παγκόσμιου Ιστού(WWW), η παροχή πληροφορίας, μπορεί να παρασχεθεί με την αποθήκευση των εγγράφων σε AFS. Η προσέγγιση εξετάζει διάφορα προβλήματα απόδοσης στους WWW εξυπηρέτες αλλά και στους πελάτες, όπως ο αυξανόμενος δικτυακός φόρτος, η καθυστέρηση των δικτύων και η ανεπαρκής ασφάλεια. Επιπλέον, ο μηχανισμός παρουσιάζει την αξία ενός σφαιρικού, γενικού σκοπού, διαμοιραζόμενου συστήματος αρχείων.

Τα αντίγραφα-εξυπηρέτες σε αυτήν την προσέγγιση ανακτούν όλα τα αρχεία τους από το διανεμημένο σύστημα αρχείων (DFS), αντί των δικών τους τοπικών συστημάτων αρχείων. Δεδομένου ότι τα διανεμημένα συστήματα αρχείων παρέχουν μια προγραμματιζόμενη διεπαφή παρόμοια με το τυποποιημένο σύστημα αρχείων του λειτουργικού συστήματος Unix, ουσιαστικά καμία τροποποίηση του κώδικα του εξυπηρέτη δικτύου δεν απαιτείται.

Εννοιολογικά, αυτή η αρχιτεκτονική είναι κάπως παρόμοια με τους περιφερειακούς εξυπηρέτες (surrogates), με την κύρια διαφορά ότι οι περιφερειακοί εξυπηρέτες (surrogates) χρησιμοποιούν το πρωτόκολλο HTTP για να ανακτήσουν τα δεδομένα από τους κεντρικούς υπολογιστές και η DFS-βασισμένη αρχιτεκτονική στηρίζεται σε ένα συγκεκριμένο DFS πρωτόκολλο για να γίνει έτσι. Δεδομένου ότι ένας πελάτης DFS μπορεί να αποθηκεύσει στην ενδιάμεση μνήμη του (cache) οποιοδήποτε αρχείο για το οποίο έγινε ανάκτηση (download), όλο το περιεχόμενο (στην ουσία ως σύνολο αρχείων

προκειμένου να παραχθούν οι HTTP απαντήσεις) αποθηκεύεται στην ενδιάμεση μνήμη (cache). Αυτό περιλαμβάνει δυναμικό περιεχόμενο, για το οποίο ένας πελάτης DFS ανακτά το λογισμικό και τα στοιχεία που απαιτούνται για να παράγουν το δυναμικό περιεχόμενο τοπικά. Αφ' ενός, όλοι οι εξυπηρέτες-αντιγραφείς στην προσέγγιση DFS πρέπει να παρέχουν τα ίδια υπολογιστικά περιβάλλοντα και να είναι σε θέση να εκτελέσουν τους ίδιους υπολογισμούς.

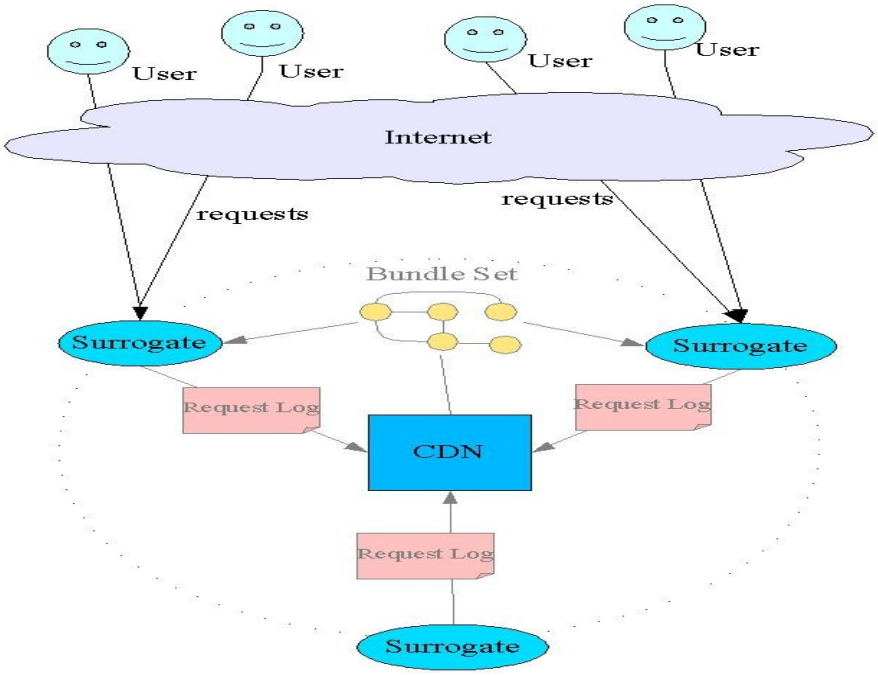
2.2. Προτεινόμενη Βελτίωση

Η προτεινόμενη αρχιτεκτονική συστήματος επεκτείνει την κλασική αρχιτεκτονική ενός δικτύου διανομής περιεχομένου, έτσι ώστε να υποστηρίξει την μερική αντιγραφή του περιεχομένου ενός τέτοιου δικτύου σε περιφερειακούς εξυπηρέτες (surrogates). Το προτεινόμενο σύστημα στοχεύει σε αυτό και υλοποιείται με την τμηματοποίηση του γράφου ενός τέτοιου δικτύου σε υπογράφους, αντιγράφοντας στην συνέχεια κάποιους από αυτούς τους υπογράφους στους περιφερειακούς εξυπηρέτες (surrogates). Ένας ειδικός αλγόριθμος, που θα ικανοποιεί τα επιθυμητά κριτήρια, θα αποφασίζει για την επιλογή των υπογράφων που θα αντιγραφούν. Περισσότερες λεπτομέρειες πάνω στους αλγορίθμους που θα χρησιμοποιηθούν παρουσιάζονται σε επόμενα κεφάλαια.

2.3. Αρχιτεκτονική

Μία από τις εργασίες-κλειδιά του συστήματος είναι η τμηματοποίηση της ιστοσελίδας με έναν αποδοτικό τρόπο, έτσι ώστε να επιτευχθεί ένα υψηλό ποσοστό επιτυχίας απόκτησης περιεχομένου, παράλληλα με την όσο το δυνατόν μείωση του χώρου αποθήκευσης. Για να επιτευχθεί αυτό, η υλοποίηση ενός πλήθους από λειτουργίες είναι απαραίτητη, όπως η τμηματοποίηση και η αντικατάσταση τμημάτων στους περιφερειακούς εξυπηρέτες (surrogates). Στην συνέχεια, ένα πλήθος από μηνύματα πρέπει να ανταλλαχτούν μεταξύ του δικτύου και των περιφερειακών εξυπηρετών (surrogates). Στην επόμενη εικόνα φαίνεται η προτεινόμενη αρχιτεκτονική του συστήματος και περιγράφεται η λειτουργία του.

2.3.1. Περιγραφή λειτουργίας



Εικόνα 8: Αρχιτεκτονική Συστήματος

Όπως φαίνεται από το παραπάνω σχήμα, το CDN στέλνει ένα μήνυμα, που περιέχει σύνολα υπογράφων (Bundles Set), οι οποίοι πρέπει να αντιγραφούν σε κάθε περιφερειακό εξυπηρέτη (surrogate). Πρέπει να τονίσουμε ότι κάθε υπογράφος (bundle) δεν είναι μια δομή που περιέχει τα πραγματικά αντικείμενα, αλλά μια λογική οντότητα, ένα σύνολο από αναφορές στα αντικείμενα που περιέχει. Το σώμα του μηνύματος, ενδέχεται να περιγράφει μόνο τα πραγματικά αντικείμενα που θα αντιγραφούν, ή να περιέχει αυτά τα αντικείμενα. Στην πρώτη περίπτωση, είναι υπευθυνότητα του περιφερειακού εξυπηρέτη (surrogate) να ανακτήσει αυτά τα αντικείμενα από το CDN. Αυτά τα μηνύματα στέλνονται περιοδικά, ή κατά την εκκίνηση του συστήματος, σύμφωνα με τον αλγόριθμο που χρησιμοποιείται.

Άλλος τύπος μηνυμάτων που ανταλλάσεται μεταξύ των περιφερειακών εξυπηρετών (surrogates) και του CDN και ενδέχεται να χρησιμοποιηθεί σε κάποιες περιπτώσεις είναι η καταγραφή των αιτήσεων (requests Logs στην παραπάνω εικόνα). Ανάλογα με τον αλγόριθμο που χρησιμοποιείται, τα δεδομένα που αφορούν τις αιτήσεις των χρηστών, ενδέχεται να χρησιμοποιηθούν για την υλοποίηση της τμηματοποίησης. Σε αυτήν την περίπτωση, κάθε περιφερειακός εξυπηρέτης (surrogate) στέλνει ένα αρχείο καταγραφής

που περιέχει τις αιτήσεις των χρηστών που ένας περιφερειακός εξυπηρέτης (surrogate) εξυπηρετεί. Το CDN, στην συνέχεια, εκτελεί τον αλγόριθμο τμηματοποίησης χρησιμοποιώντας τα δεδομένα από αυτά τα αρχεία καταγραφής σε συνδυασμό με προηγούμενα που έχει ήδη επεξεργαστεί, τμηματοποιεί ξανά τον CDN γράφο και στέλνει ένα μήνυμα στους περιφερειακούς εξυπηρέτες (surrogates), που περιέχει το ενημερωμένο σύνολο υπογράφων (bundles set). Όπως έχουμε ήδη αναφέρει, οι περιφερειακοί εξυπηρέτες (surrogates) ενδέχεται να πραγματοποιήσουν έναν αριθμό από αιτήσεις προς το CDN, έτσι ώστε να ανακτήσουν αντικείμενα που δεν περιέχουν ήδη.

Όπως προαναφέρθηκε, το προτεινόμενο σχήμα, σκοπεύει να επιτύχει ένα υψηλό ποσοστό επιτυχίας απόκτησης περιεχομένου, παράλληλα με την όσο καλύτερη μείωση αποθηκευτικού χώρου στον περιφερειακό εξυπηρέτη (surrogate). Η αίτηση ενός χρήστη χαρακτηρίζεται ως επιτυχημένη, όταν ικανοποιείται από τον περιφερειακό εξυπηρέτη (surrogate) τοπικά, δηλαδή ο πόρος είναι διαθέσιμος στους υπογράφους (bundles) του περιφερειακού εξυπηρέτη (surrogate). Στην περίπτωση που ο περιφερειακός εξυπηρέτης (surrogate) αποτυγχάνει να ικανοποιήσει την αίτηση τοπικά, πρέπει να επικοινωνήσει με το CDN και να αποκτήσει τον επιθυμητό πόρο. Σε αυτή την περίπτωση, υπάρχει μια μικρή επιβάρυνση στην επικοινωνία, αλλά αυτό δεν αναμένεται να συμβαίνει σχετικά συχνά.

2.4. Αλγόριθμοι Τμηματοποίησης

Οι αλγόριθμοι τμηματοποίησης που χρησιμοποιούνται για την δημιουργία των CDN υπογράφων (bundles), μπορούν να διαχωριστούν σε δύο κατηγορίες σύμφωνα με τα κριτήρια που χρησιμοποιούνται για την υλοποίηση της τμηματοποίησης. Η πρώτη κατηγορία περιέχει αυτούς τους αλγορίθμους που υλοποιούν την τμηματοποίηση βασισμένη στην δομή του ιστοχώρου, χρησιμοποιώντας μετρικές όπως την διασύνδεση των ιστοσελίδων, την “απόσταση” κτλ. Ο υπολογισμός των μετρικών αυτών είναι στατικός και συνεπώς δεν επηρεάζεται από άλλες παραμέτρους, αλλά από την δομή του ιστοχώρου.

Άλλοι αλγόριθμοι, που ανήκουν στην δεύτερη κατηγορία, χρησιμοποιούν δυναμικά δεδομένα, προκειμένου να υπολογίσουν τις κατάλληλες μετρικές. Ένας τέτοιος αλγόριθμος μπορεί να εκτελέσει μια ανάλυση των δικτυακών ιχνών έτσι ώστε να

προσδιορίσει την συσχέτιση των ιστοσελίδων που βασίζονται στα στατιστικά δεδομένα της πλοήγησης ενός χρήστη. Στην υλοποίηση μας, χρησιμοποιήσαμε έναν αλγόριθμο από την πρώτη κατηγορία με το όνομα «Ανάλυση Στατικού Ιστοχώρου» (Static Site Analysis), που αφορά τους στατικούς αλγορίθμους, και ο οποίος περιγράφεται σε επόμενο κεφάλαιο.

2.4.1 Ανάλυση Στατικού Ιστοχώρου

Η ιδέα του αλγορίθμου έγκειται στο να απομονώσει “κεντρικούς” κόμβους (πόρους) του ιστοχώρου, από τους οποίους κάθε ένας θα αποτελέσει το κεντρικό σημείο ενός υπογράφου (bundle). Κάθε κόμβος χρησιμοποιείται σαν το αρχικό σημείο για μια πρώτη διαδρομή στον γράφο κατά πλάτος. Κάθε κόμβος που επισκέπτεται, προστίθεται στο συγκεκριμένο υπογράφο (bundle), μέχρι να γεμίσει, ή μέχρι να μην υπάρχουν άλλοι προς επίσκεψη. Με περισσότερες λεπτομέρειες, τα βασικά βήματα του αλγόριθμου περιγράφονται με ψευδοκώδικα παρακάτω (Δείγμα Κώδικα Α) και θα συζητηθούν στις επόμενες παραγράφους.

```
SiteGraphGenerator(“www.di.uoa.gr”)
SiteGraph=SiteGraphHandler.readGraphXML(“XML_GraphFile”)
SortedVerticesList =SiteGraph.sortVerticesDegree()
Bundles=SiteGraph.generateBundles(SortedVelticesList)
```

Εικόνα 8: Δείγμα Κώδικα Α

Όπως μπορούμε να παρατηρήσουμε, ο αλγόριθμος χωρίζεται σε τέσσερα βασικά βήματα. Το πρώτο βήμα (μέθοδος “SiteGraphGenerator ()”), χρησιμοποιεί ένα αντικείμενο web spider προκειμένου να αναλυθεί ο ιστοχώρος-στόχος και να παραχθεί μια αναπαράσταση σε XML του γραφήματος της ιστοχώρου. Ο web spider που χρησιμοποιείται βασίζεται στην βιβλιοθήκη της WebSPHINX, που δημιουργήθηκε προκειμένου να ικανοποιήσει τις απαιτήσεις της περίπτωσης αυτής. Με περισσότερες λεπτομέρειες, η παραπάνω μέθοδος, επισκέπτεται κάθε πόρο (url) ενός ιστοχώρου και πράγει ένα XML αρχείο που αναπαριστά την δομή του ιστοχώρου σαν ένα κατευθυνόμενο γράφο, δηλαδή ένα σύνολο από κορυφές και ένα σύνολο από

κατευθυνόμενες ακμές. Ένα μέρος από το XML αρχείο του αποτελέσματος παρουσιάζεται παρακάτω (Δείγμα Κώδικα Β).

```
<?xml version="1.0" encoding="UTF-8" ?>
<SiteMap SiteUrl="http://www.di.uoa.gr">
  <VerticesList>
      <Vertex url="http://www.di.uoa.gr" size="2260" />
      <Vertex url="http://www.di.uoa.gr/images/menu-uoa.gif"
        size="9488" />
      <Vertex url="http://www.di.uoa.gr/gr/" size="5562" />
      . . .
  </VerticesList>
  <EdgesList>
      <DirectedEdge source="http://www.di.uoa.gr"
        target="http://www.di.uoa.gr/images/menu-uoa.gif" />
      <DirectedEdge source="http://www.di.uoa.gr"
        target="http://www.di.uoa.gr/index-items/di-
        grlft.jpg" />
      <DirectedEdge source="http://www.di.uoa.gr"
        target="http://www.di.uoa.gr/gr/" />
      <DirectedEdge source="http://www.di.uoa.gr"
        target="http://www.di.uoa.gr/index-
        items/choffgr.jpg" />
      . . .
  </EdgesList>
</SiteMap>
```

Εικόνα 9: Δείγμα Κώδικα Β

Το επόμενο βήμα είναι να διαβαστεί το XML αρχείο, που προέκυψε και να παραχθεί ένας γράφος-αντικείμενο, που θα χρησιμοποιηθεί για την δημιουργία των υπογράφων (bundles). Η λειτουργία αυτή υλοποιείται με την μέθοδο “SiteGraphHandler.readGraphXML()”.

Ο παραγόμενος γράφος-αντικείμενο πρέπει στην συνέχεια να περάσει από επεξεργασία πριν την τελική παραγωγή των υπογράφων (bundles). Όπως έχουμε ήδη προαναφέρει, ο αλγόριθμος απομονώνει τους πιο “κεντρικούς” κόμβους του γράφου και παράγει clusters-υπογράφους, ξεκινώντας με αυτούς.

Η μετρική της κεντρικότητας υπολογίζεται σε αυτό το βήμα, ως ο αριθμός των εξερχόμενων συνδέσεων από κάθε κόμβο του γράφου. Στην συνέχεια, όλοι οι κόμβοι ταξινομούνται με βάση την κεντρικότητά τους, σε φθίνουσα σειρά. Αυτή η λειτουργία

υλοποιείται με την μέθοδο “SiteGraph.sortVerticesDegree()”, που παρουσιάζεται παρακάτω (Δείγμα Κώδικα Γ).

Τέλος, το τελευταίο βήμα του αλγορίθμου είναι να παράγει τους υπογράφους (bundles), από τον Αλγόριθμο Στατικού Ιστοχώρου (Static Site Algorithm). Οι ταξινομημένοι κόμβοι του γράφου που πέρασαν από επεξεργασία στο προηγούμενο βήμα χρησιμοποιούντια για την παραγωγή των υπογράφων (bundles). Ξεκινώντας από τους πιο “κεντρικούς”, κάθε κόμβος χρησιμοποιείται σαν κόμβος-ρίζα για να εκτελεστεί μια πρώτη κατά πλάτος διαδρομή στον γράφο. Όλοι οι κόμβοι που έχουν επισκεφτεί κατά της διάρκεια αυτής της διαδρομής προστίθενται στον ίδιο υπογράφο (bundle) μέχρι να γεμίσει ή να μην υπάρχουν άλλοι κόμβοι προς επίσκεψη. Η μέθοδος ολοκληρώνεται, όταν για δεδομένο αριθμό από υπογράφους (bundles), που δίδεται ως όρισμα, γεμίσουν. Η μέθοδος “generateBundles”, που ο αντίστοιχος ψευδοκώδικας της, δίδεται παρακάτω (Δείγμα Κώδικα 3), εκτελεί την κατάλληλη επεξεργασία στους ταξινομημένους κόμβους, που παράχθηκαν στο προηγούμενο βήμα, και παράγει έναν πίνακα αντικειμένων, που περιέχει τους εξερχόμενους υπογράφους (bundles).

Πρέπει να σημειωθεί ότι λόγω της δομής του γράφου του ιστοχώρου, υπάρχουν κύκλοι και συνεπώς με μια κατά πλάτος επανάληψη, ξεκινώντας από διαφορετικό κόμβο κάθε φορά, ενδέχεται να επισκεφτούν κόμβοι που είχαν επισκεφτεί από προηγούμενες επαναλήψεις (και είχαν ξεκινήσει από διαφορετικό κόμβο-ρίζα). Ο υλοποιημένος αλγόριθμος δίνει ιδιαίτερη σημασία στο να αποφευχθεί η επανεμφάνιση κόμβων σε διαφορετικούς υπογράφους (bundles).

```
sortVerticesDegree()
{
/* Sorts the Vertices of a graph based on the number of outgoing
links from a node in descending order. */

      For each Node
          Degree=Node.outDegree
          NodesArray.add(Node,degree)
      End for
      NodesArray.sortDescending
}

/*-----------------------------------------------------------------*/

bundlesArray[] generateBundles()
{
  i=0
      while  i< NodesArray.size()
      {
       rootNode=NodesArray(i)
            currentBundleSize=0
            bundle = new Bundle
            nextNode=rootNode.breadthFirstIterator.Next()
            while ((nextNode != Null) and (nextNode.size +
                                          currentBundleSize <
                                                maxBundleSize))
            {
             bundle.add(nextNode)
             currentBundleSize += nextNode.size

            }
      }
}

/*-----------------------------------------------------------------*/

SiteGraphGenerator (the_site)
{
  siteGraph=MyCrawler.websphinx.Crawler(the_site)
      saveSiteXML(siteGraph)
}
```

Εικόνα 10: Δείγμα Κώδικα Γ

Πρέπει να σημειωθεί, επίσης, ότι αυτός ο αλγόριθμος χρειάζεται κάποια χρονική περίοδο, κατά την οποία όλες οι εισερχόμενες αιτήσεις χαρακτηρίζονται ως αποτυχημένες από τον "αναλυτή αιτήσεων", έτσι ώστε να αρχικοποιηθεί ο γράφος και να εκτελέσει την αρχική τμηματοποίηση του ιστοχώρου. Σημειώνεται ότι, κατά την διάρκεια της περιόδου αυτής, δεν υπάρχει αντιγραφή περιεχομένου του ιστοχώρου στους περιφερειακούς εξυπηρέτες (surrogates). Παρ'όλ' αυτά, αυτό μπορεί να αποφθεχθεί, έαν προηγούμενα δεδομένα αιτήσεων συλλεχθούν και αναλυθούν, πριν

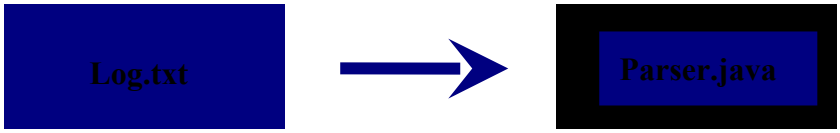
την εγκατάσταση του συστήματος και στην συνέχεια να εκτελεστεί η αρχική τμηματοποίηση του συστήματος.

ΚΕΦΑΛΑΙΟ 3

ΥΛΟΠΟΙΗΣΗ

3.1. Τεχνικές λεπτομέρειες

Αρχικά υλοποιήσαμε ένα συντακτικό αναλυτή (parser) προκειμένου να μπορέσουμε να κάνουμε την ανάλυση της δομής των αρχείων καταγραφής (log files) και τελικά να εξάγουμε τα δεδομένα που μας ενδιέφεραν. Το όνομα του αρχείου αυτού είναι Parser.java

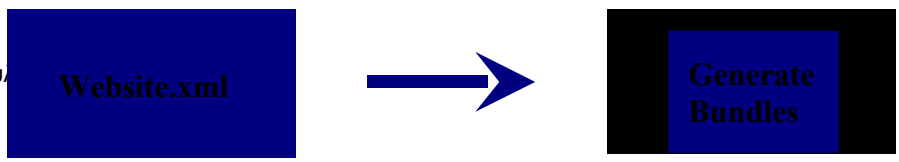


Εικόνα 11: Συντακτική ανάλυση αρχείων καταγραφής

Στη συνέχεια, το .xml αρχείο, το οποίο περιγράφει τη δομή του προαναφερθέντος ιστοχώρου, τροφοδότησε την κλάση GenerateBundles με σκοπό να κατακερματιστεί ο γράφος.

Το πλήθος των ακμών του γραφήματος που αφαιρέθηκαν κατά την προσομοίωση προκειμένου να επιτευχθεί ο κατακερματισμός, δεν είναι προκαθορισμένο αλλά δίνεται από το χρήστη στην εκκίνηση της προσομοίωσης.

Τα στοιχεία που ακολουθούν αναφέρονται σε εκτελέσεις του αλγορίθμου για αφαίρεση 100, 200 και 500 ακμών αντιστοίχως. Η αφαίρεση περισσοτέρων των 500 ακμών θεωρήθηκε μη ενδιαφέρουσα περίπτωση καθώς ο κατακερματισμός του γραφήματος θα ήταν πολύ μεγάλος.



Εικόνα 12: Δημιουργία πακέτων υπογράφων

3.1.1. Εργαλείο ANT

Το εργαλείο Apache Ant είναι ένα εργαλείο αυτόματης μεταγλώττισης κώδικα (automating software build). Είναι παρόμοιο με το μεταγλωττιστή Make αλλά είναι γραμμένο σε γλώσσα προγραμματισμού Java, απαιτεί πλατφόρμα Java και είναι καταλληλότερο για τη μεταγλώττιση πηγαίων αρχείων Java.

Η βασικότερη όμως διαφορά μεταξύ των δύο είναι πως το Ant χρησιμοποιεί XML (**E**xtensible **M**arkUp **L**anguage) για να περιγράψει τη διαδικασία της μεταγλώττισης και τις απαραίτητες εξαρτήσεις., ενώ το εργαλείο Make χρησιμοποιεί τη δική του Makefile μορφή. Εξ' ορισμού το XML αρχείο ονομάζεται build.xml

Το Ant είναι δημιούργημα της ASF (Apache Software Foundation) και αποτελεί προϊόν Ανοικτού Λογισμικού (Open Source Software).

Παρακάτω δίνουμε ένα δείγμα ενός αρχείου build.xml για μία απλή "Hello World" εφαρμογή.

Ορισμός Έργου

Κάθε αρχείο αναφέρεται σε ένα μόνο έργο (project). Ο ορισμός του έργου, αποτελείται από τρία γνωρίσματα: όνομα, εξορισμού κατάλογος και κατάλογος βάσης. Απαραίτητα πρέπει να οριστεί ο εξορισμού κατάλογος, που θα χρησιμοποιηθεί σε περίπτωση που δεν ορίζεται κατάλογος προορισμού στη συνέχεια του εγγράφου.

Ορισμός Ιδιοτήτων

Κάθε έργο, μπορεί να περιέχει ένα σύνολο από ιδιότητες (<*property*>). Μια ιδιότητα ορίζεται ως ένα ζευγάρι ονόματος-τιμής, και μπορεί να αναφερθεί μέσα στο αρχείο, με τη χρήση του συμβόλου '$' και το όνομα της ιδιότητας να περικλείεται ανάμεσα σε άγκιστρα '{}'. Ένα τέτοιο παράδειγμα φαίνεται αποτελεί το τμήμα κώδικα "<delete dir="${build.dir}">", όπου ορίζεται η διαγραφή του καταλόγου με όνομα την τιμή της ιδιότητας "build.dir", που έχει οριστεί πιο πάνω.

Ορισμός Στόχων

Σε κάθε έργο, μπορούν να περιέχονται ένας η περισσότεροι στόχοι (<*target*>), που έχουν πέντε γνωρίσματα: όνομα (*name*), εξαρτάται από (*depends*), εάν (*if*), εκτός αν (*unless*), και περιγραφή (*description*). Μόνο το πρώτο γνώρισμα από τα παραπάνω είναι απαραίτητο.

Ορισμός Εξαρτήσεων

Κάθε στόχος, ορίζει μια ή παραπάνω εργασίες (*task*), που πρέπει να εκτελεστούν. Όταν το εργαλείο αρχίσει την διαδικασία μεταγλώττισης, μπορεί να καθοριστεί ποιος στόχος από αυτούς που ορίζονται στο αρχείο εισόδου, θα εκτελεστεί. Ο ορισμός εξαρτήσεων, μπορεί να καθοριστεί με το γνώρισμα “depends” στον ορισμό του στόχου. Στο δικό μας παράδειγμα, ο στόχος jar περιέχει το στόχο compile σαν εξάρτηση (dependency). Το εργαλείο Ant αντιλαμβάνεται από αυτή τη δομή πως προτού προχωρήσει στο στόχο jar πρέπει να ολοκληρώσει το στόχο compile. Με αυτό τον τρόπο, ορίζεται η σειρά εκτέλεσης των στόχων.

Ορισμός Εργασιών

Οι εργασίες ορίζονται μέσα στους στόχους, και δίνουν την εντολή για την εκτέλεση κώδικα. Ο κώδικας που εκτελείται, μπορεί να είναι για οποιαδήποτε γλώσσα. Για παράδειγμα θα μπορούσε να είναι η εκτέλεση της εντολής του DOS “echo”, ή η εκτέλεση ενός προγράμματος Java με τη χρήση της εντολής “<java..>” Στο παράδειγμά μας, για τη μεταγλώττιση του στόχου compile, το εργαλείο Ant πρέπει πρώτα να δημιουργήσει ένα φάκελο (directory) με όνομα classes και στην συνέχεια να καλέσει τον μεταφραστή (compiler) Java. Επομένως οι εργασίες (tasks) που χρησιμοποιούνται είναι οι mkdir και javac. Η λειτουργία των δύο αυτών εργασιών είναι παρόμοια με αυτή των ομώνυμων εργασιών της γραμμής εντολών (command line).

Το Ant, παρέχει εξορισμού υποστήριξη για πολλές βασικές εργασίες (όπως μεταγλώττιση java κώδικα, εκτέλεση java κώδικα, δημιουργία συμπιεσμένων αρχείων, λειτουργίες καταλόγων), αλλά μπορεί να υποστηρίξει και ένα σύνολο από άλλες εργασίες με τις κατάλληλες προσθήκες.

Άλλη μία εργασία που περιέχεται στο παρακάτω παράδειγμα είναι η εργασία jar:

```
<jar destfile="hello.jar">
```

Αυτή η εργασία έχει την ίδια ονομασία με το εργαλείο JAR της γραμμής εντολών, αλλά στην πραγματικότητα είναι μία κλήση στην ενσωματωμένη υποστήριξη του εργαλείου Ant για συμπιεσμένα αρχεία (jar/zip).

```
<?xml version="1.0"?>
 <project name="Hello" default="compile">
 <property name="build.dir" location="build"/>
     <target name="clean" description="remove intermediate files">
         <delete dir="${build.dir}"> </delete>
     </target>
   <target name="compile"
      description="compile the Java source code to class files">
         <mkdir dir="classes"/>
         <javac srcdir="." destdir="classes"/>
     </target>
   <target name="jar" depends="compile"
      description="create a Jar file for the application">
         <jar destfile="hello.jar">
             <fileset dir="classes" includes="**/*.class"/>
             <manifest>
                 <attribute name="Main-Class" value="HelloProgram"/>
             </manifest>
         </jar>
     </target>
 </project>
```

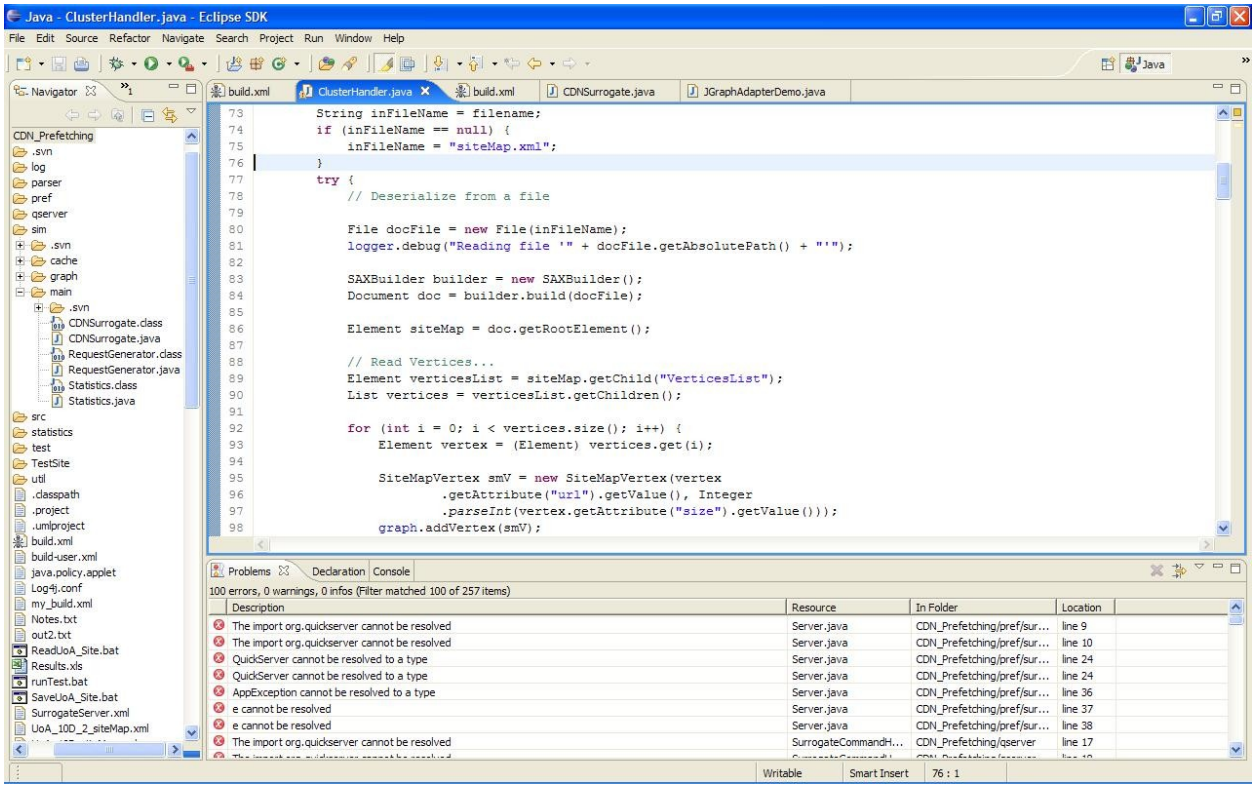
Ορισμός Αρχείου Εισόδου

Όπως αναφέρθηκε και πιο πάνω, εξ' ορισμού το Ant ψάχνει το αρχείο με το όνομα "build.xml" στον τρέχοντα κατάλογο και αν αυτό δεν βρεθεί, συνεχίζει την αναζήτηση προς τα πάνω μέχρι τον κατάλογο ρίζα του συστήματος αρχείων. Μπορεί όμως να οριστεί διαφορετικό αρχείο, για αρχείο εισόδου με τη χρήση κατάλληλης παραμέτρου.

3.1.2. Εργαλείο ανάπτυξης λογισμικού ECLIPSE

Η υλοποίηση του λογισμικού πραγματοποιήθηκε στο περιβάλλον Eclipse. Το Eclipse αποτελεί μία ανοιχτή πλατφόρμα ανάπτυξης λογισμικού η οποία προσφέρει εργαλεία εύκολα στη χρήση. Τα εργαλεία αυτά καλύπτουν όλα τα στάδια: από την υλοποίηση και τη μεταγλώττιση έως την εφαρμογή του λογισμικού που παράχθηκε.

Για την παρούσα εφαρμογή έγινε χρήση της έκδοσης Eclipse SDK 3.1.1



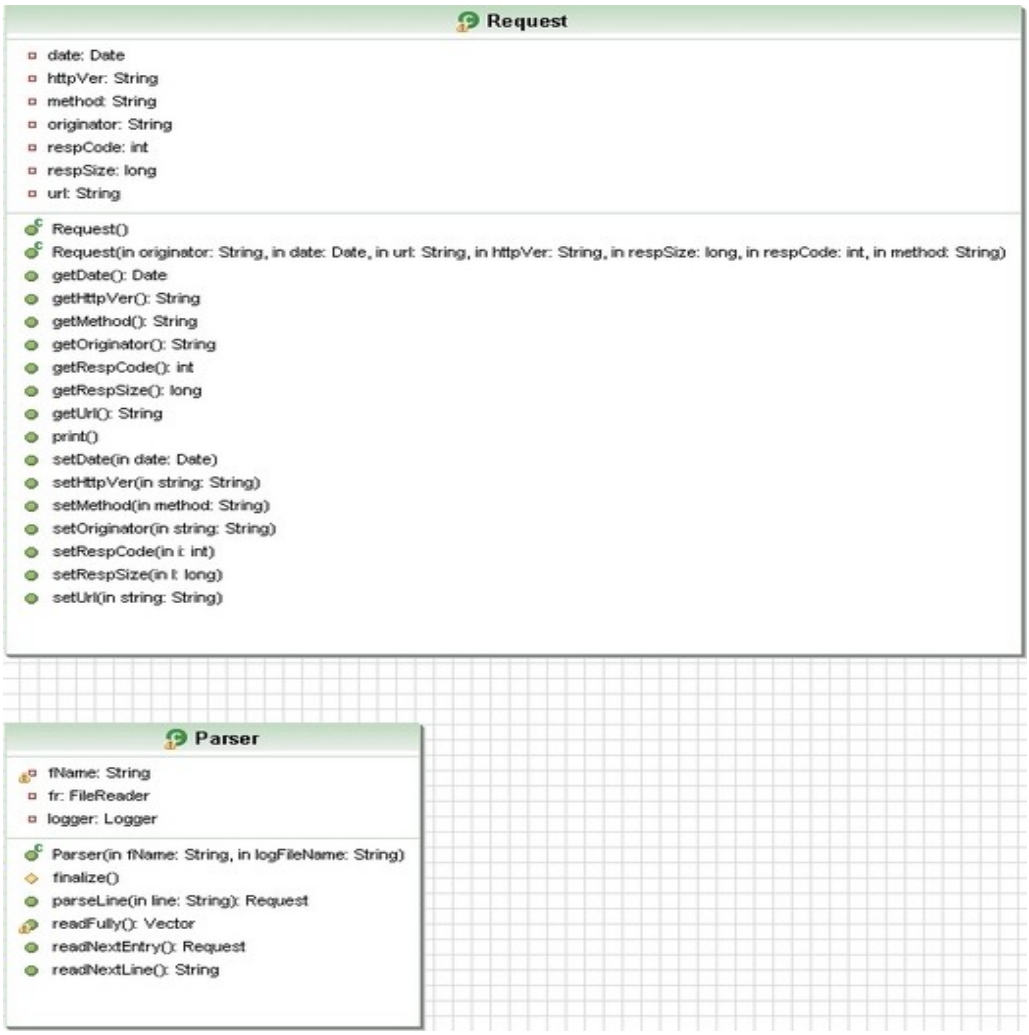
Εικόνα 13:Εικόνα από το περιβάλλον Eclipse SDK 3.1.1

3.2. Αντικειμενοστραφής Σχεδιασμός (Object-oriented design)

Κατά τον αντικειμενοστραφή σχεδιασμό που πραγματοποιήσαμε προέκυψαν τα παρακάτω αντικείμενα τα οποία και χρησιμοποιήθηκαν στο στάδιο της υλοποίησης.

3.2.1 Οντότητες Parser και Request

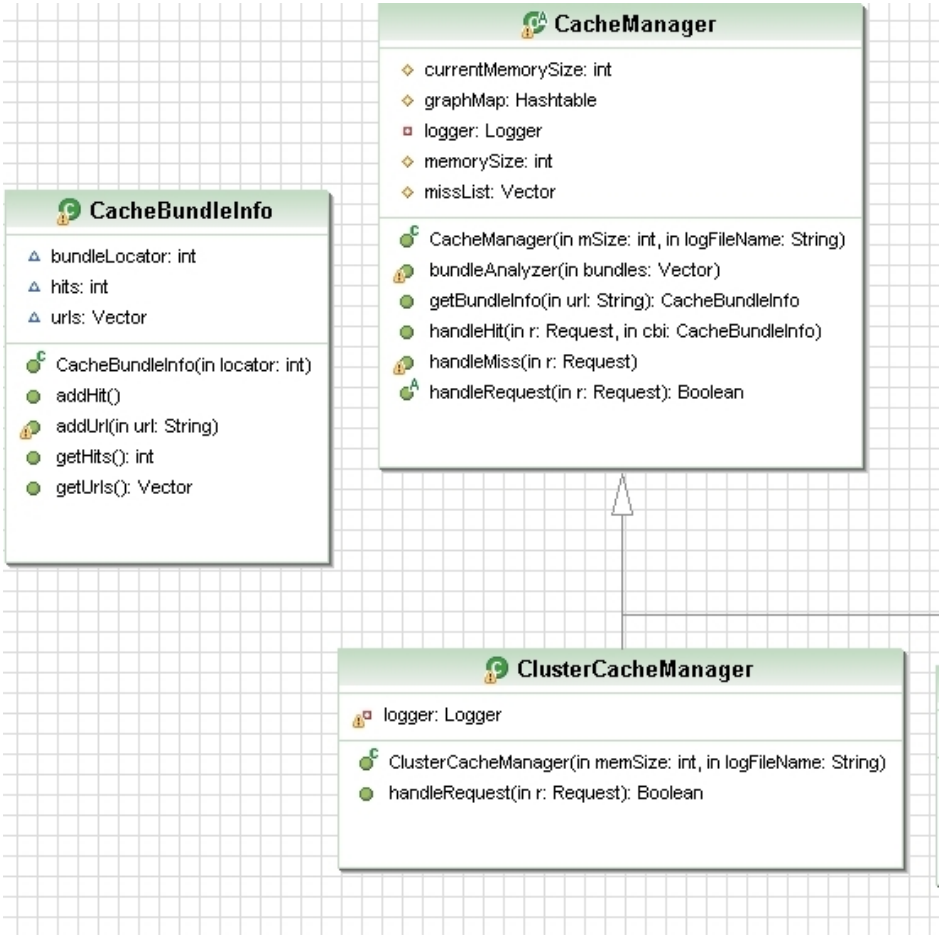
Παρακάτω φαίνεται το UML σχήμα της οντότητας Parser η οποία υλοποιήθηκε προκειμένου να αναλυθούν οι αιτήσεις των χρηστών του ιστοχώρου και να μπορέσουμε να εξάγουμε χρήσιμα για την υλοποίησή μας στοιχεία, όπως για παράδειγμα ποιας ιστοσελίδας την ανάκτηση επιχείρησε ο χρήστης και ποια χρονική στιγμή. Κάθε μία από τις αιτήσεις αυτές αντικατοπτρίζεται από την οντότητα Request, το UML σχήμα της οποίας απεικονίζεται επίσης στο σχήμα που ακολουθεί.



Εικόνα 14:UML Σχήμα οντοτήτων Parser και Request

3.2.2 Οντότητες CacheBundleInfo, CacheManager και ClusterCacheManager

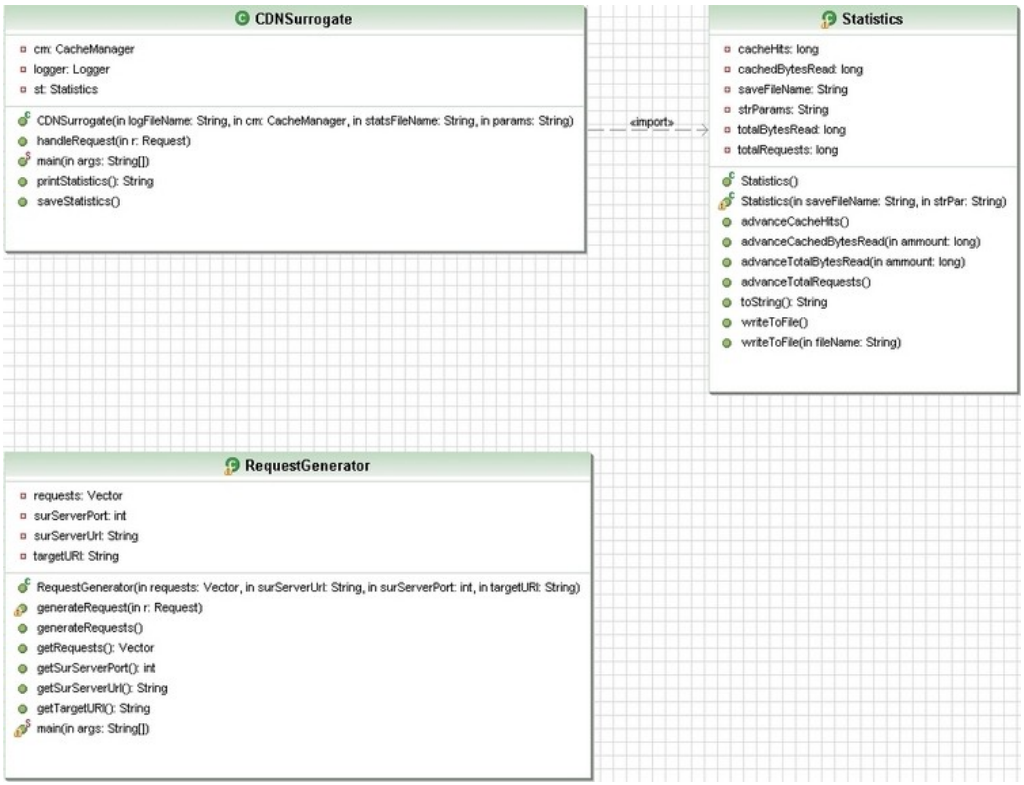
Οι οντότητες CacheBundleInfo, CacheManager και ClusterCacheManager υλοποιήθηκαν προκειμένου να εξυπηρετήσουμε τις ανάγκες του αλγορίθμου για διαχείρηση των διαθέσιμων πόρων κατά την επεξεργασία του αρχικού γραφήματος του ιστοχώρου. Το UML σχήμα και των τριών φαίνεται στην παρακάτω εικόνα.



Εικόνα 15:UML Σχήμα οντοτήτων CacheBundleInfo, CacheManager και ClusterCacheManger.

3.2.3 Οντότητες CDNSurrogate, Statistics και RequestGenerator

Τέλος παρακάτω παραθέτουμε τη σχηματική απεικόνιση σε UML μορφή των οντοτήτων RequestGenerator, CDNSurrogate, και Statistics, οι οποίες χρησιμοποιούνται κατ' αντιστοιχία για την «εξαγωγή» των αιτήσεων των χρηστών από το αρχείο καταγραφής, την «ενορχήστρωση» των επιμέρους οντοτήτων για την επεξεργασία του γραφήματος του ιστοχώρου και την εξαγωγή των αποτελεσμάτων της υλοποίησης στο στάδιο των δοκιμών.



Εικόνα 16:UML Σχήμα οντοτήτων CDNSurrogate, Statistics και RequestGenerator.

3.3. Αλγόριθμος Ομαδοποίησης (Clustering Algorithm)

Ο Αλγόριθμος που χρησιμοποιήθηκε για την υλοποίησή μας είναι ο Edge Betweenness Clusterer αλγόριθμος, ο οποίος υλοποιείται από την κλάση EdgeBetweennessClusterer της βιβλιοθήκης jung.

Πρόκειται για έναν αλγόριθμο υπολογισμού ομάδων (Clusters) σε γράφους, ο οποίος βασίζεται στην Κεντρικότητα των ακμών του γραφήματος (Edge Betweenness). Η Κεντρικότητα μιας ακμής ορίζεται ως ο βαθμός στον οποίο η ακμή αυτή συμμετέχει στα ελάχιστα μονοπάτια μεταξύ όλων των ζευγών που μπορούν να δημιουργήσουν οι κόμβοι του γράφου. Οι ακμές οι οποίες είναι περισσότερο «κεντρικές» σταδιακά αφαιρούνται από το γράφημα έως ότου οι ομάδες να είναι επαρκώς απομονωμένες μεταξύ τους.

Ο αλγόριθμος υλοποιείται ακολουθώντας αναδρομικά τα εξής δύο βήματα μέχρι να αφαιρεθεί το προκαθορισμένο πλήθος ακμών:

1. Υπολόγισε την Κεντρικότητα κάθε ακμής στο γράφημα.
2. Αφαίρεσε την ακμή με τη μεγαλύτερη Κεντρικότητα.

για i = 1 έως [προκαθορισμένο πλήθος ακμών]

για j = 1 έως [πλήθος ακμών γραφήματος]

Υπολόγισε Κεντρικότητα ακμής[j]

Αφαίρεσε από το γράφο την ακμή με max(Κεντρικότητα)

Εικόνα 17:Ψευδογλώσσα αλγορίθμου Edge Betweenness Clusterer

Η πολυπλοκότητα του αλγορίθμου αυτού είναι O(kmn), όπου k είναι το πλήθος των ακμών που θα αφαιρεθούν από το γράφημα, m το συνολικό πλήθος των ακμών του γραφήματος και n το πλήθος των κόμβων. Στις περιπτώσεις πολύ αραιών γραφημάτων, ο χρόνος εκτέλεσης πλησιάζει το O(kn^2) .

3.4. Πηγαίος Κώδικας (Source Code)

Ακολουθεί ο πηγαίος κώδικας της υλοποίησης.

3.4.1 Parser.java

```
package parser;

import java.io.FileReader;
import java.text.SimpleDateFormat;
import java.util.Locale;
import java.util.Vector;

import org.apache.log4j.Logger;
import org.apache.log4j.PropertyConfigurator;


public class Parser {

    private String fName;

    private FileReader fr;

    private Logger logger;

    public Parser(String fName, String logFileName) {
        this.fName = fName;
        logger = Logger.getLogger(this.getClass().getName());

        PropertyConfigurator.configure(logFileName);
```

```
        try {
            fr = new FileReader(fName);
        } catch (Exception e) {
            logger.error(e);
        }
    }

    public Request readNextEntry() {
        logger.debug("Method 'readNextEntry' invoked");
        Request l = new Request();

        String nextLine = readNextLine();
        if (nextLine.equalsIgnoreCase("endOffile"))
            return null;
        l = parseLine(nextLine);
        logger.debug("Method 'readNextEntry' terminated");
        return l;
    }

    public String readNextLine() {
        logger.debug("Method 'readNextLine' invoked");
        StringBuffer line = new StringBuffer("");
        try {

            int c;
            while (((c = fr.read()) != -1) && (c != '\n')) {
                line.append((char) c);
            }
            if (c == -1 && line.toString().equals("")) {
                logger.debug("Method readNextLine terminated");
                return "EndOfFile";
            }
            logger.debug("Method 'readNextLine' terminated");
        } catch (Exception e) {
            logger.error(e);
        }
        return line.toString();
    }

    public Request parseLine(String line) {
        logger.debug("Method 'parseLine' invoked");
        Request r = new Request();
        String originalLine = line;
        try {
            int index = line.indexOf(' ');
            r.setOriginator(line.substring(0, index));

            // read date information
            index = line.indexOf('[') + 1;
            line = line.substring(index);
```

```
index = line.indexOf(']');
String date = line.substring(0, index);
line = line.substring(index);

SimpleDateFormat sdf = new SimpleDateFormat(
        "dd/MMM/yyyy:HH:mm:ss Z", Locale.US);
sdf.setLenient(true);
r.setDate(sdf.parse(date));
logger.debug("Date is: " + sdf.parse(date).toString());

logger.debug("Line->" + line);
index = line.indexOf('\"');
line = line.substring(index);

logger.debug("Line->" + line);
// Method is GET or POST and followed by a ' '
index = line.indexOf(' ');
r.setMethod(line.substring(1,index));
// The first cahracter of line is ""
line = line.substring(index).trim();
logger.debug("Line->" + line);

index = line.indexOf(' ');
logger.debug("Line->" + line);
r.setUrl(line.substring(0, index));
line = line.substring(index).trim();

// get http version...
index = line.indexOf("HTTP");

line = line.substring(index).trim();
index = line.indexOf("");
r.setHttpVer(line.substring(0, index));
line = line.substring(index + 1).trim();

index = line.indexOf(' ');
logger.debug("Line->"+line + " index:" + index);

r.setRespCode(Integer.parseInt(line.substring(0, index)));

line = line.substring(index).trim();
index = line.indexOf(' ');
if (r.getRespCode() == 200) {
    try {
        r.setRespSize(Long.parseLong(line.substring(0, index)));
    }

    catch (NumberFormatException nfe) {
        logger.info("Number Format Exception in line:"
                + originalLine);
```

```
                r.setRespSize(0);
            }
        }
        logger.debug("Method parseLine terminated");
    } catch (Exception e) {
        logger.error(e);
        logger.error(originalLine);
        r.setRespCode(-12);
    }
    return r;
}

public Vector readFully() {
    logger.debug("Method 'readFully' invoked");
    Vector v = new Vector();
    try {

        Request r;
        while ((r = readNextEntry()) != null) {
            v.add(r);
        }
    } catch (Exception e) {
        e.printStackTrace();
        if (e.getClass().isInstance(OutOfMemoryError.class)) {
            logger.error("Too many records....");
        } else {
            logger.info(e.getClass().getName());
            logger.error(e);
        }
    }
    logger.info("Read " + v.size() + " lines");
    logger.debug("Method 'readFully' terminated");
    return v;
}

protected void finalize() {
    try {
        fr.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

}
```

3.4.2 Request.java

```
package parser;
```

```
import java.io.Serializable;
import java.util.Date;

public class Request implements Serializable{

    private String originator;
    private Date date;
    private String url;
    private String httpVer;
    private long respSize;
    private int respCode;
    private String method;


    /**
     * @param originator
     * @param date
     * @param url
     * @param httpVer
     * @param respSize
     * @param respCode
     * @param method
     */
    public Request(String originator, Date date, String url, String httpVer,
            long respSize, int respCode, String method) {
        this.originator = originator;
        this.date = date;
        this.url = url;
        this.httpVer = httpVer;
        this.respSize = respSize;
        this.respCode = respCode;
        this.method = method;
    }
    public Request(){};

    public String getMethod() {
        return method;
    }

    public void setMethod(String method) {
        this.method = method;
    }

    public String getOriginator() {
        return originator;
    }

    public int getRespCode() {
        return respCode;
    }
```

```
    public long getRespSize() {
        return respSize;
    }

    public String getUrl() {
        return url;
    }

    public void setOriginator(String string) {
        originator = string;
    }

    public void setRespCode(int i) {
        respCode = i;
    }

    public void setRespSize(long l) {
        respSize = l;
    }

    public void setUrl(String string) {
        url = string;
    }


    public void print()
    {
        System.out.println("\n------------LogEntry-----------");
        System.out.println("Originator:"+originator);
        System.out.println("url:"+url);
        System.out.println("method:"+method);
        System.out.println("httpVersion:"+httpVer);
        System.out.println("respCode:"+respCode);
        System.out.println("respSize:"+respSize);
        System.out.println("Date:"+date);
        System.out.println("--------- End LogEntry ----------");
    }

    public String getHttpVer() {
        return httpVer;
    }

    public void setHttpVer(String string) {
        httpVer = string;
    }

    public Date getDate()
    {
        return date;
```

```
    }

    public void setDate(Date date)
    {
        this.date = date;
    }

}
```

3.4.3 CDNSurrogate.java

```
package sim.main;

import java.util.TreeMap;
import java.util.Vector;

import org.apache.log4j.Logger;
import org.apache.log4j.PropertyConfigurator;

import dioNan.ClusterHandler;

import parser.Parser;
import parser.Request;

import sim.cache.CacheManager;
import sim.cache.ClusterCacheManager;
import sim.cache.MarkovCacheManager;
import sim.cache.StaticSiteCacheManager;
import sim.graph.MarkovGraphHandler;
import sim.graph.SiteGraphHandler;

public class CDNSurrogate {

    private CacheManager cm;
    private Statistics st;

    private Logger logger;

    public CDNSurrogate(String logFileName, CacheManager cm,String statsFileName,String
params) {
        logger = Logger.getLogger(this.getClass().getName());
        PropertyConfigurator.configure(logFileName);
        this.cm=cm;
        st=new Statistics(statsFileName,params);
    }

    public void handleRequest(Request r)
    {
        logger.debug("Invoking method handleRequest");
```

```
        // Forward the request to the Cache Manager

        Boolean hit=cm.handleRequest(r);

        if (hit==null)
        {
            // Warming up Period in case of Markov
            logger.debug("Warming up...");
            return;
        }

        //Update Statistics
        if (hit.booleanValue())
        {
            st.advanceTotalRequests(); // Advance Total requests by 1
            st.advanceCachedBytesRead(r.getRespSize()); // Advance Total Bytes Read
            st.advanceTotalBytesRead(r.getRespSize());
            st.advanceCacheHits();
        }
        else
        {
            st.advanceTotalRequests(); // Advance Total requests by 1
            st.advanceTotalBytesRead(r.getRespSize()); // Advance Total Bytes Read
        }

        logger.debug("Method handleRequest terminated");
    }

    public String printStatistics()
    {
        logger.info("Invoking method printStatistics");
        return st.toString();
    }

    public void saveStatistics()
    {
        logger.info("Invoking method saveStatistics");
        try
        {
            st.writeToFile();
        }
        catch (Exception e)
        {
            logger.error("Error while writing statistics...");
            logger.error("-----------Error Message---------");
            logger.error(""+ e.getMessage());
            logger.error("---------------------------------");
        }

        logger.info("Method saveStatistics terminated");
```

```
    }

    public static void main(String args[])
    {
        //Initialize parameters
        if (args.length < 7) {
            System.out.println("Usage: CDNSurrogate CacheManager MemorySize[MB]
BundleSize[MB] log4jConfigFileName TraceFileName StatisticsFileName
SavedGraphFileName");
            return;
        }

        String cacheManager=args[0];
        String parameters="\n";
        int cacheMemorySize;
        try {
            cacheMemorySize = Integer.parseInt(args[1]);
            parameters+="CacheSize=" +cacheMemorySize+"\n";
            cacheMemorySize=1024*1024*cacheMemorySize; // Convert in Bytes...
        } catch (NumberFormatException e1) {
            System.out.println("Invalid Number Format !" + args[1]);
            return;
        }

        int bundleSize;
        try {
            bundleSize = Integer.parseInt(args[2]);
            parameters+="BundleSize=" +bundleSize+"\n";
            bundleSize=1024*1024*bundleSize; // Convert in Bytes...
        } catch (NumberFormatException e1) {
            System.out.println("Invalid Number Format !" + args[2]);
            return;
        }

        String log4jConfigFileName = args[3];

        String traceFileName=args[4];
        parameters+="TraceFile:  " + traceFileName +"\n";
        String statisticsFileName=args[5];
        String savedGraphFileName=args[6];

        // Initialize CacheManager
        CacheManager cm=null;
        if (cacheManager.toLowerCase().equals("static"))
        {
            System.out.println("--------Processing Static site-----");
            SiteGraphHandler sgh=new SiteGraphHandler(log4jConfigFileName);
            sgh.readGraphXML(savedGraphFileName);
            TreeMap tm = sgh.sortVerticesDegree();
            Vector bundlesVector=sgh.generateBundles(tm, bundleSize,cacheMemorySize);
```

```
            cm=new StaticSiteCacheManager(cacheMemorySize,log4jConfigFileName);
            cm.bundleAnalyzer(bundlesVector);
        }
        else if (cacheManager.toLowerCase().equals("cluster"))
        {
            System.out.println("--------Processing Clustering -----");
            ClusterHandler ch=new ClusterHandler(log4jConfigFileName);
            ch.readGraphXML(savedGraphFileName);

            Vector bundlesVector=ch.generateBundles(5, bundleSize);

            cm=new ClusterCacheManager(cacheMemorySize,log4jConfigFileName);
            cm.bundleAnalyzer(bundlesVector);
        }

        else if (cacheManager.toLowerCase().equals("markov"))
        {
            int lookaheadWindow;
            try {
                lookaheadWindow = Integer.parseInt(args[7]);
                parameters+="LookaheadWindow=" +lookaheadWindow+"\n";
            } catch (NumberFormatException e1) {
                System.out.println("Invalid Number Format !" + args[7]);
                return;
            }
            catch (ArrayIndexOutOfBoundsException ex){
                System.out.println("In Case Of Markov usage is: CDNSurrogate CacheManager
MemorySize[MB] BundleSize[MB] log4jConfigFileName TraceFileName StatisticsFileName
SavedGraphFileName lookahead activeClientPeriod resetCount");
                return;
            }

            int activeClientPeriod;
            try {
                activeClientPeriod = Integer.parseInt(args[8]);
                parameters+="ActiveClientPeriod=" +activeClientPeriod+"\n";
            } catch (NumberFormatException e1) {
                System.out.println("Invalid Number Format !" + args[8]);
                return;
            }
            catch (ArrayIndexOutOfBoundsException ex){
                System.out.println("In Case Of Markov usage is: CDNSurrogate CacheManager
MemorySize[MB] BundleSize[MB] log4jConfigFileName TraceFileName StatisticsFileName
SavedGraphFileName lookahead activeClientPeriod resetCount");
                return;
            }

            int resetCount; // Indicates when to rebuild the Markov Graph
            try {
```

```
                resetCount = Integer.parseInt(args[9]);
                parameters+="ResetCount=" +resetCount+"\n";
            } catch (NumberFormatException e1) {
                System.out.println("Invalid Number Format !" + args[9]);
                return;
            }
            catch (ArrayIndexOutOfBoundsException ex){
                System.out.println("In Case Of Markov usage is: CDNSurrogate CacheManager
MemorySize[MB] BundleSize[MB] log4jConfigFileName TraceFileName StatisticsFileName
SavedGraphFileName lookahead activeClientPeriod resetCount");
                return;
            }


            MarkovGraphHandler mgh = new MarkovGraphHandler(log4jConfigFileName);
            mgh.readGraphXML(savedGraphFileName);

            TreeMap tm = mgh.sortVerticesWeight();
            Vector bundlesVector=mgh.generateBundles(tm,bundleSize,cacheMemorySize);
            cm=new
MarkovCacheManager(cacheMemorySize,log4jConfigFileName,lookaheadWindow,activeClient
Period,bundleSize,resetCount);
            cm.bundleAnalyzer(bundlesVector);
        }
        else
        {
            System.out.println("Undefined cache Manager:" + cacheManager);
            System.out.println("Valid values {static,markov}");
            return;
        }


        CDNSurrogate c=new
CDNSurrogate(log4jConfigFileName,cm,statisticsFileName,parameters);

        // Read the Log File
        try
        {
            Parser p =   new Parser(traceFileName,log4jConfigFileName);

            Request r;
            int count=0;
            while ((r=p.readNextEntry())!=null)
            {
                if (r.getRespCode()==200)
                // In case of a parse error the ResponseCode of the request is set to
                // -12
                {
                    c.handleRequest(r);
                    count++;
```

```
                if (count % 50000 ==0)
                {
                    System.out.println("------Statistics--------\n"+c.printStatistics());
                }
            }
        }
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }

    System.out.println("------Statistics--------\n"+c.printStatistics());
    c.saveStatistics();


  }
}
```

3.4.4 CacheManager.java

```
package sim.cache;

import java.util.Enumeration;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.Set;
import java.util.Vector;

import org.apache.log4j.Logger;
import org.apache.log4j.PropertyConfigurator;

import parser.Request;
import util.bundle.Bundle;
import util.graph.SiteMapVertex;

public abstract class CacheManager {

    protected int memorySize;
    protected int currentMemorySize;
    protected Hashtable graphMap;
    protected Vector missList;

    private Logger logger;

    public abstract Boolean handleRequest(Request r);

    public CacheManager(int mSize,String logFileName)
    {
```

```
        missList=new Vector();
        graphMap=new Hashtable();
        memorySize=mSize;
    logger = Logger.getLogger(this.getClass().getName());
        PropertyConfigurator.configure(logFileName);
    }

    public void handleMiss(Request r)
    /*
     * Handles a request for a url that doesn't exist in Cache
     */
    {
        logger.debug("Invoking Method handleMiss");
        missList.add(r.getUrl());
        logger.debug("Method handleMiss Terminated");
    }

    public void handleHit(Request r,CacheBundleInfo cbi)
    /*
     * Handles a request for a url that doesn't exist in Cache
     */
    {
        logger.debug("Invoking Method handleHit");
        cbi.addHit();
        logger.debug("Method handleHit Terminated");
    }

    public CacheBundleInfo getBundleInfo(String url)
    {
        logger.debug("Invoking Method getBundleInfo");
        logger.debug("Search for url->'"+url+"'");
        if (graphMap.containsKey(url))
        {
            CacheBundleInfo cbi=(CacheBundleInfo)graphMap.get(url);
            logger.debug("Method getBundleInfo Terminated");
            return cbi;
        }
        logger.debug("Method getBundleInfo Terminated");
        return null;
    }

    public void bundleAnalyzer(Vector bundles)
    /*
     * Kanei mia analusi twn bundles, kai ftiaxnei mia apeikonisi
     * (antikeimeno graphMap)
     * apo url->bundleInfo pou periexei plirofories gia to bundle
     * sto opoio anikei to url
     */
    {
        logger.info("Invoking Method bundleAnalyzer");
```

```
graphMap=new Hashtable();
logger.info("Found "+bundles.size() + " bundles");
for (int i=0; i<bundles.size(); i++)
{
    /*
     * Gia ka8e bundle
     */
    logger.info("Bundle "+i);
    Bundle b=(Bundle)bundles.get(i);
    /*
     * Pairnw ta graphs pou mporei na periexei to bundle...
     */
    Hashtable graphs=b.getGraphs();

    CacheBundleInfo cbi=new CacheBundleInfo(i);

    Enumeration e=graphs.keys();
    logger.info("Has "+graphs.size() + " graphs");
    /*
     * To ka8e kleidi einai ena sunolo apo tis
     * korufes tou upografou
     */

    /*
     * Sarwnetai mia fora gia na ftiaksw mia lista
     * me ta urls pou 8a periexontai sto cbi
     */
    while (e.hasMoreElements())
    {
        Set vertexSet=(Set)e.nextElement();
        logger.info("Has "+vertexSet.size() + " vertices");
        Iterator it = vertexSet.iterator();
        while (it.hasNext() )
        {
            SiteMapVertex vertex=(SiteMapVertex)it.next();
            logger.debug("url->" + vertex.getUrl());
            cbi.addUrl(vertex.getUrl());
        }
    }

    e=graphs.keys();
    while (e.hasMoreElements())
    {
        Set vertexSet=(Set)e.nextElement();

        Iterator it = vertexSet.iterator();
        while (it.hasNext() )
        {
            SiteMapVertex vertex=(SiteMapVertex)it.next();
            graphMap.put(vertex.getUrl(),cbi);
```

```
                }
            }
        }
        logger.info("Method bundleAnalyzer Terminated");
    }
}
```

3.4.5 ClusterCacheManager.java

```
package sim.cache;

import org.apache.log4j.Logger;
import org.apache.log4j.PropertyConfigurator;

import parser.Request;

public class ClusterCacheManager extends CacheManager {

    private Logger logger;

    public ClusterCacheManager(int memSize, String logFileName) {
        super(memSize, logFileName);
        logger = Logger.getLogger(this.getClass().getName());
        PropertyConfigurator.configure(logFileName);
    }

    public Boolean handleRequest(Request r) {

        /*
         * Check to see if the url exists in any of the bundles...
         */
        r.setUrl("http://www.di.uoa.gr" + r.getUrl());
        boolean out = false;
        CacheBundleInfo cbi = getBundleInfo(r.getUrl());
        if (cbi == null) // Indicates a Cache Miss
        {
            handleMiss(r);
            out = false;
        } else {
            handleHit(r, cbi);
            out = true;
        }
        return new Boolean(out);
    }

}
```

3.4.6 ClusterHandler.java

```
package sim.graph;

import java.io.File;
import java.util.HashSet;
import java.util.Iterator;
import java.util.List;
import java.util.NoSuchElementException;
import java.util.Set;
import java.util.SortedMap;
import java.util.TreeMap;
import java.util.Vector;

import org._3pq.jgrapht.alg.StrongConnectivityInspector;
import org._3pq.jgrapht.edge.DirectedEdge;
import org._3pq.jgrapht.graph.DirectedSubgraph;
import org._3pq.jgrapht.graph.SimpleDirectedGraph;
import org._3pq.jgrapht.traverse.BreadthFirstIterator;
import org.apache.log4j.Logger;
import org.apache.log4j.PropertyConfigurator;
import org.jdom.Document;
import org.jdom.Element;
import org.jdom.input.SAXBuilder;

import util.bundle.Bundle;
import util.graph.SiteMapVertex;

import edu.uci.ics.jung.algorithms.cluster.ClusterSet;
import edu.uci.ics.jung.algorithms.cluster.EdgeBetweennessClusterer;
import edu.uci.ics.jung.graph.DirectedGraph;
import edu.uci.ics.jung.graph.impl.DirectedSparseGraph;

public class ClusterHandler {

    private SimpleDirectedGraph graph;

    private Logger logger;

    // private Hashtable verticesTable;

    public ClusterHandler(String logFileName) {
        logger = Logger.getLogger(this.getClass().getName());
        PropertyConfigurator.configure(logFileName);
        // verticesTable=new Hashtable();
    }

    public void readGraphXML(String filename) {
        /*
         * Reads the XML file and creates a graph with vertices SiteMapVertex
```

```
 * objects. In this graph the edges connect SiteMapVertex objects...
 *
 */

logger.info("Entering readGraph method...");

graph = new SimpleDirectedGraph();
String inFileName = filename;
if (inFileName == null) {
    inFileName = "siteMap.xml";
}
try {
    // Deserialize from a file

    File docFile = new File(inFileName);
    logger.debug("Reading file '" + docFile.getAbsolutePath() + "'");

    SAXBuilder builder = new SAXBuilder();
    Document doc = builder.build(docFile);

    Element siteMap = doc.getRootElement();

    // Read Vertices...
    Element verticesList = siteMap.getChild("VerticesList");
    List vertices = verticesList.getChildren();

    for (int i = 0; i < vertices.size(); i++) {
        Element vertex = (Element) vertices.get(i);

        SiteMapVertex smV = new SiteMapVertex(vertex
                .getAttribute("url").getValue(), Integer
                .parseInt(vertex.getAttribute("size").getValue()));
        graph.addVertex(smV);
    }

    // Read Edges...
    Element edgesList = siteMap.getChild("EdgesList");
    List edges = edgesList.getChildren();

    for (int i = 0; i < edges.size(); i++) {
        // Create a DirectedEdge object and add it to the graph
        Element edge = (Element) edges.get(i);
        // String edgeSource=edge.getAttribute("source").getValue();
        // String edgeTarget=edge.getAttribute("target").getValue();

        // Find the vertices in the graph
        SiteMapVertex edgeSource = getSiteMapVertex(edge.getAttribute(
                "source").getValue());
        SiteMapVertex edgeTarget = getSiteMapVertex(edge.getAttribute(
                "target").getValue());
```

```
                graph.addEdge(new DirectedEdge(edgeSource, edgeTarget));
    }

        } catch (Exception e) {
            logger.error("readGraph-> Error reading graph...\n"
                    + e.getMessage());
        }
        logger.info("readGraph method terminated ");
    }

    private SiteMapVertex getSiteMapVertex(String value) {

        Iterator i = graph.vertexSet().iterator();
        SiteMapVertex smV = new SiteMapVertex(value, 0);
        while (i.hasNext()) {
            smV = (SiteMapVertex) i.next();
            if (smV.getUrl().equalsIgnoreCase(value)) {
                break;
            }
        }
        return smV;
    }

    public Vector generateBundles (TreeMap tMap, int numOfEdgesToRemove, int
maxCapacity)
    {
        Vector bundles = new Vector();
        DirectedSparseGraph dsg=new DirectedSparseGraph();
        cs.sort();
        Iterator clset_iter = new Iterator();
        Iterator cluster_iter = new Iterator();
        Set newVerticesSet = new HashSet();
        clset_iter = cs.iterator();
        Set cluster = new Set();
        Bundle b = new Bundle();
        float cur_size=0;
        int flag;
        SiteMapVertex smv = new SiteMapVertex();
        while ((clset_iter.hasNext() == true) && (cur_size < maxCapacity))
        {
            flag=0;
            newVerticesSet.clear();
            cluster = clset_iter.next();
            cluster_iter = cluster.iterator();
            while((cur_size<maxCapacity) && (cluster_iter.hasNext() == true))
            {
                smv = (SiteMapVertex)cluster_iter.next();
                cur_size+=smv.getSize();
                if (cur_size > maxCapacity)
```

```
                {
                    DirectedSubgraph subGraph = new DirectedSubgraph (graph, newVerticeSet,
null);
                    b.add(subGraph.vertexSet(), subGraph,size);
                    bundles.add(b);
                    flag=1;
                    break;
                }
                newVerticesSet.add(smv);
            }
            if(flag==0)
            {
                DirectedSubgraph subGraph = new DirectedSubgraph (graph, newVerticeSet, null);
                b.add(subGraph.vertexSet(), subGraph,cur_size);
                bundles.add(b);
            }


        }
        return bundles;
    }

    public List getSCC() {
        StrongConnectivityInspector sci = new StrongConnectivityInspector(graph);
        List subGraphs = sci.stronglyConnectedSubgraphs();

        Iterator i = subGraphs.iterator();
        int count = 0;
        while (i.hasNext()) {
            DirectedSubgraph d = (DirectedSubgraph) i.next();
            logger.info("Graph " + ++count + " is:" + d.toString());
        }
        return subGraphs;

    }

    public static void main(String[] args) {
        if (args.length < 2) {
            System.out.println("Usage: ClusterHandler savedfile maxSize(MB)");
            return;
        }

        String inFileName = args[0];

        int maxSize=Integer.parseInt(args[1])*1024*1024;
        System.out.println(maxSize);

        SiteGraphHandler sh = new SiteGraphHandler("Log4j.conf");

        sh.readGraphXML(inFileName);
```

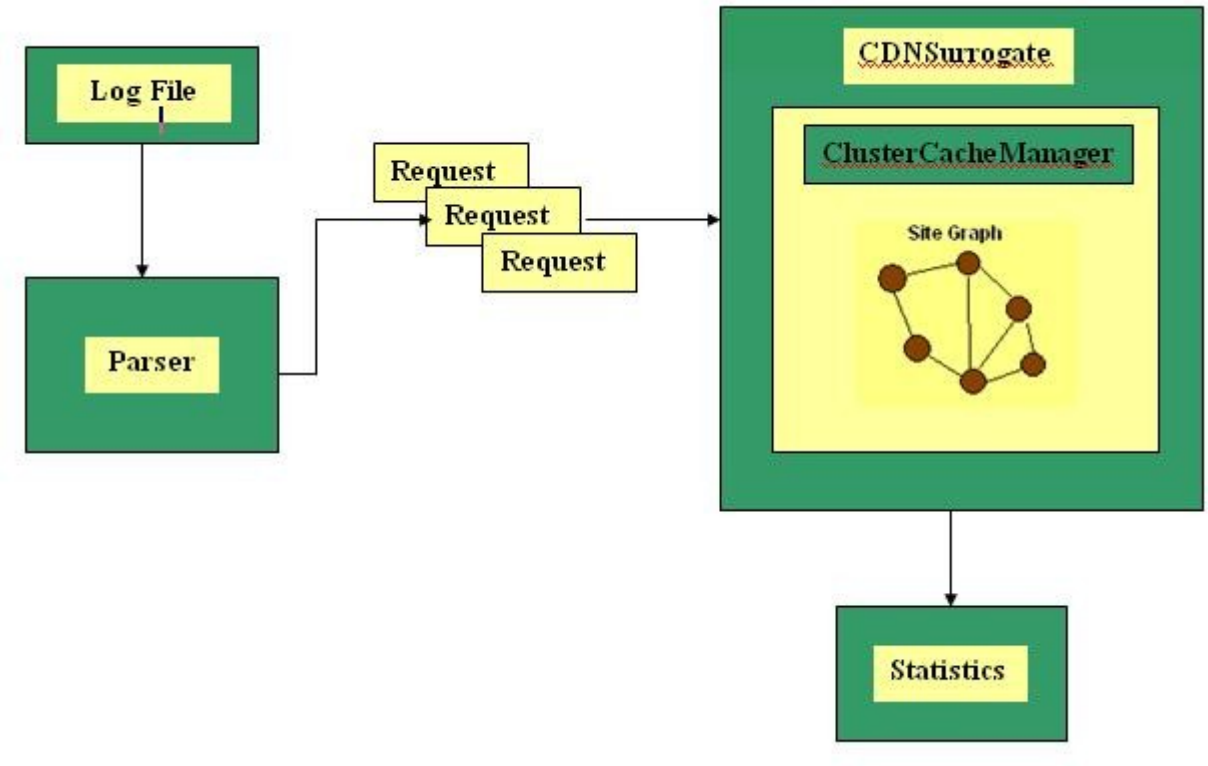
```
        sh.generateBundles(tm, 5000000,maxSize);
    }

}
```

ΚΕΦΑΛΑΙΟ 4

ΑΠΟΤΕΛΕΣΜΑΤΑ

4.1. Setup δοκιμών

Η προσομοίωση της εφαρμογής έγινε με βάση τις αιτήσεις (Requests) του ιστοχώρου του Τμήματος Πληροφορικής και Τηλεπικοινωνιών του Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών (www.di.uoa.gr). Είχαμε στη διάθεσή μας για το σκοπό αυτό τα αρχεία καταγραφής (log files) των μηνών Οκτωβρίου, Νοεμβρίου και Δεκεμβρίου του έτους **2005.**



Εικόνα 18:Συνολική απεικόνιση διαδικασίας

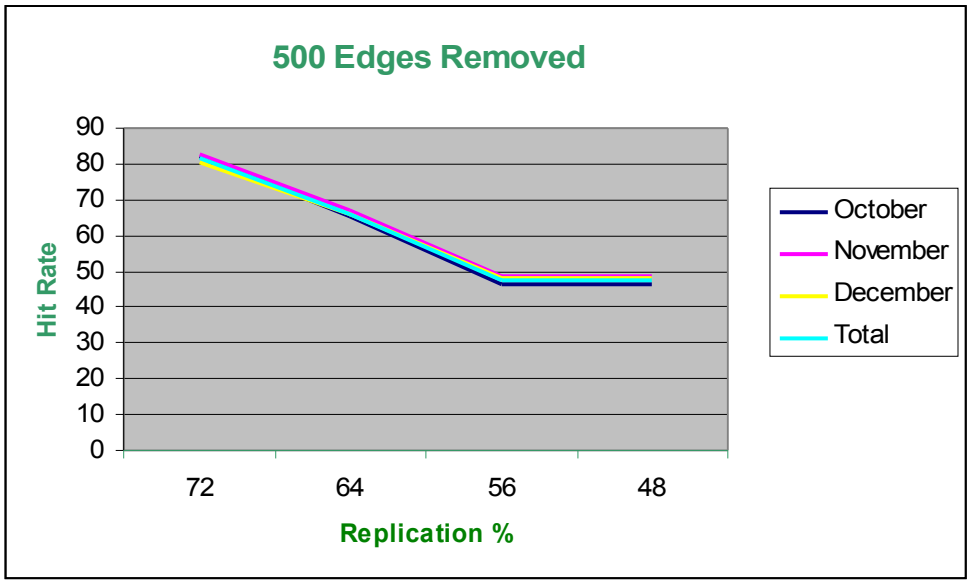
Όπως αναφέρθηκε και νωρίτερα, για την υλοποίηση χρησιμοποιήθηκε το εργαλείο Ant, το οποίο τροφοδοτήσαμε με το παρακάτω build.xml αρχειο:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE project [<!ENTITY buildfile SYSTEM "file:./build-user.xml">]>
<!-- WARNING: Eclipse autogenerated file.  Any modifications will be overwritten.
        Please edit build-user.xml instead.-->
<project basedir="." default="build" name="CDN_Prefetching">
```

```
  &buildfile;
  <path id="project.classpath">
    <pathelement location="."/>
    <pathelement location="D:/ProgrammingTools/eclipse/plugins/org.junit_3.8.1/junit.jar"/>
    <pathelement location="D:/java/logging-log4j-1.2.9/dist/lib/log4j-1.2.9.jar"/>
    <pathelement location="D:/java/QuickServer/dist/QuickServer.jar"/>
    <pathelement location="D:/java/QuickServer/dist/commons-pool.jar"/>
    <pathelement location="D:/java/commons-codec-1.3/commons-codec-1.3.jar"/>
    <pathelement location="D:/java/pljava/pljava.jar"/>
    <pathelement location="D:/java/jung-1.7.4/lib/jung-1.7.4.jar"/>
    <pathelement location="D:/java/jdom-1.0/build/jdom.jar"/>
    <pathelement location="D:/java/GraphLibs/jgrapht-0.6.0/jgrapht-0.6.0.jar"/>
    <pathelement location="D:/java/WebSPHINX/websphinx.jar"/>
    <pathelement location="D:/java/commons-httpclient/commons-httpclient-3.0-rc4.jar"/>
  </path>


  <property name="tracefile" value="E:\\dioNan\\Tracefile"/>


  <target name="init"/>
  <target name="clean"/>
  <target depends="init" name="build">
    <echo message="${ant.project.name}: ${ant.file}"/>
    <javac destdir=".">
      <src path="."/>
      <classpath refid="project.classpath"/>
    </javac>
  </target>


  <target name="CDNSurrogateCluster">
    <java classname="sim.main.CDNSurrogate" failonerror="true" fork="yes">
      <jvmarg value="-Xmx900M"/>
     <arg value="cluster"/>
      <arg value="150"/>
      <arg value="10"/>
      <arg value="E:\\dioNan\\implementation\\Log4j.conf"/>
      <arg value="${tracefile}"/>
      <arg value="ClusterStatsOut"/>
      <arg value="E:\\dioNan\\implementation\\UoA_10D_SiteMap.xml"/>
      <classpath refid="project.classpath"/>
    </java>
  </target>
</project>
```

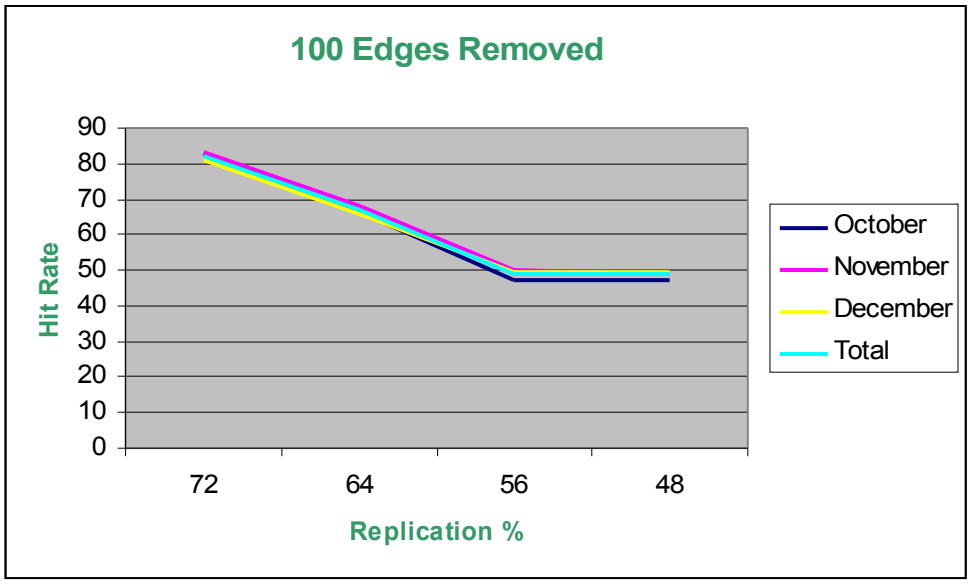
4.2. Αποτελέσματα Δοκιμών

Στις δοκιμές που πραγματοποιήσαμε εκτελέσαμε τον αλγόριθμο αφαιρώντας πεντακόσιες (500), διακόσιες (200) και εκατό (100) ακμές συνολικά. Το Hit Rate του αλγορίθμου συναρτήσει του ποσοστού Replication (%) για κάθε μία από τις περιπτώσεις αυτές, παρουσιάζεται στα διαγράμματα που ακολουθούν.



Εικόνα 19:Συνολική απεικόνιση διαδικασίας

Εικόνα 20:Συνολική απεικόνιση διαδικασίας



Εικόνα 21:Συνολική απεικόνιση διαδικασίας

Ακολουθεί αναλυτικός πίνακας με τα αποτελέσματα των δοκιμών.

Πίνακας 1: Αποτελέσματα Προσομοίωσης

Month Source File	Cache Size	Total Size	Replication %	Bundle Size	Removed Edges	Hits	Total Requests	Hit Rate	Hit Bytes	Total Bytes	ByteHitRate
10	180	250	72	5	500	393433	480152	81,94	3113839504	4780369211	65,14
10	180	250	72	5	200	393433	480152	81,94	3113850529	4780369211	65,14
10	180	250	72	5	100	396837	480152	82,65	3254222038	4780369211	68,07
10	180	250	72	10	500	393537	480152	81,96	3190532544	4780369211	66,74
10	180	250	72	10	200	393537	480152	81,96	3190532544	4780369211	66,74
10	180	250	72	10	100	396979	480152	82,68	3382768279	4780369211	70,76
10	160	250	64	5	500	315093	480152	65,62	2256616998	4780369211	47,21
10	160	250	64	5	200	315093	480152	65,62	2256628023	4780369211	47,21
10	160	250	64	5	100	318889	480152	66,41	2430002277	4780369211	50,83
10	160	250	64	10	500	315211	480152	65,65	2349211222	4780369211	49,14
10	160	250	64	10	200	315211	480152	65,65	2349211222	4780369211	49,14
10	160	250	64	10	100	319031	480152	66,44	2558548518	4780369211	53,52
10	140	250	56	5	500	223609	480152	46,57	1630251943	4780369211	34,10
10	140	250	56	5	200	223609	480152	46,57	1630262968	4780369211	34,10
10	140	250	56	5	100	227990	480152	47,48	1786087136	4780369211	37,36
10	140	250	56	10	500	223684	480152	46,59	1659320927	4780369211	34,71
10	140	250	56	10	200	223684	480152	46,59	1659320927	4780369211	34,71
10	140	250	56	10	100	228089	480152	47,50	1851108137	4780369211	38,72
10	120	250	48	5	500	223609	480152	46,57	1630251943	4780369211	34,10
10	120	250	48	5	200	223609	480152	46,57	1630262968	4780369211	34,10
10	120	250	48	5	100	227990	480152	47,48	1786087136	4780369211	37,36
10	120	250	48	10	500	223684	480152	46,59	1659320927	4780369211	34,71
10	120	250	48	10	200	223684	480152	46,59	1659320927	4780369211	34,71
10	120	250	48	10	100	228089	480152	47,50	1851108137	4780369211	38,72
11	180	250	72	5	500	289616	350938	82,53	2508903481	4410461321	56,89
11	180	250	72	5	200	289608	350938	82,52	2508867794	4410461321	56,88
11	180	250	72	5	100	292613	350938	83,38	2655695303	4410461321	60,21
11	180	250	72	10	500	289661	350938	82,54	2517824975	4410461321	57,09
11	180	250	72	10	200	289661	350938	82,54	2517824975	4410461321	57,09
11	180	250	72	10	100	292696	350938	83,40	2746361491	4410461321	62,27

11	160	250	64	5	500	236219	350938	67,31	1855596843	4410461321	42,07
11	160	250	64	5	200	236211	350938	67,31	1855561156	4410461321	42,07
11	160	250	64	5	100	239653	350938	68,29	2038434131	4410461321	46,22
11	160	250	64	10	500	236276	350938	67,33	1887240897	4410461321	42,79
11	160	250	64	10	200	236276	350938	67,33	1887240897	4410461321	42,79
11	160	250	64	10	100	239736	350938	68,31	2129100319	4410461321	48,27
11	140	250	56	5	500	171112	350938	48,76	1372434313	4410461321	31,12
11	140	250	56	5	200	171104	350938	48,76	1372398626	4410461321	31,12
11	140	250	56	5	100	174860	350938	49,83	1520355779	4410461321	34,47
11	140	250	56	10	500	171148	350938	48,77	1361233695	4410461321	30,86
11	140	250	56	10	200	171148	350938	48,77	1361233695	4410461321	30,86
11	140	250	56	10	100	174922	350938	49,84	1568177295	4410461321	35,56
11	120	250	48	5	500	171112	350938	48,76	1372434313	4410461321	31,12
11	120	250	48	5	200	171104	350938	48,76	1372398626	4410461321	31,12
11	120	250	48	5	100	174860	350938	49,83	1520355779	4410461321	34,47
11	120	250	48	10	500	171148	350938	48,77	1361233695	4410461321	30,86
11	120	250	48	10	200	171148	350938	48,77	1361233695	4410461321	30,86
11	120	250	48	10	100	174922	350938	49,84	1568177295	4410461321	35,56
12	180	250	72	5	500	355969	443004	80,35	5315550110	10689901613	49,72
12	180	250	72	5	200	355966	443004	80,35	5315587802	10689901613	49,73
12	180	250	72	5	100	357889	443004	80,79	5372360452	10689901613	50,26
12	180	250	72	10	500	356165	443004	80,40	5560331551	10689901613	52,01
12	180	250	72	10	200	356165	443004	80,40	5560331551	10689901613	52,01
12	180	250	72	10	100	358124	443004	80,84	5982992984	10689901613	55,97
12	160	250	64	5	500	290800	443004	65,64	4411549653	10689901613	41,27
12	160	250	64	5	200	290797	443004	65,64	4411587345	10689901613	41,27
12	160	250	64	5	100	294576	443004	66,50	4558502167	10689901613	42,64
12	160	250	64	10	500	291025	443004	65,69	4719073886	10689901613	44,15
12	160	250	64	10	200	291025	443004	65,69	4719073886	10689901613	44,15
12	160	250	64	10	100	294811	443004	66,55	5169134699	10689901613	48,36
12	140	250	56	5	500	213982	443004	48,30	3384169043	10689901613	31,66
12	140	250	56	5	200	213979	443004	48,30	3384206735	10689901613	31,66
12	140	250	56	5	100	218571	443004	49,34	3409155795	10689901613	31,89
12	140	250	56	10	500	214110	443004	48,33	3499900924	10689901613	32,74
12	140	250	56	10	200	214110	443004	48,33	3499900924	10689901613	32,74

Α.-Μ. Αν αγν ωσ τοπ ούλ ου, Κ. Με νπή

7

Ανά τπυξ η Σuστ ήματ ος Πρό βλε μης Κίνη σης του Χρή στη σε ένατ ίστο Χώρ ο

12	140	250	56	10	100	218709	443004	49,37	3827995975	10689901613	35,81
12	120	250	48	5	500	213982	443004	48,30	3384169043	10689901613	31,66
12	120	250	48	5	200	213979	443004	48,30	3384206735	10689901613	31,66
12	120	250	48	5	100	218571	443004	49,34	3409155795	10689901613	31,89
12	120	250	48	10	500	214110	443004	48,33	3499900924	10689901613	32,74
12	120	250	48	10	200	214110	443004	48,33	3499900924	10689901613	32,74
12	120	250	48	10	100	218709	443004	49,37	3827995975	10689901613	35,81
10+11+12	180	250	72	5	500	1039018	1274094	81,55	10938293095	19880732145	55,02
10+11+12	180	250	72	5	200	1039007	1274094	81,55	10938306125	19880732145	55,02
10+11+12	180	250	72	5	100	1047339	1274094	82,20	11282277793	19880732145	56,75
10+11+12	180	250	72	10	500	1039363	1274094	81,58	11268689070	19880732145	56,68
10+11+12	180	250	72	10	200	1039363	1274094	81,58	11268689070	19880732145	56,68
10+11+12	180	250	72	10	100	1047799	1274094	82,24	12112122754	19880732145	60,92
10+11+12	160	250	64	5	500	842112	1274094	66,09	8523763494	19880732145	42,87
10+11+12	160	250	64	5	200	842101	1274094	66,09	8523776524	19880732145	42,87
10+11+12	160	250	64	5	100	853118	1274094	66,96	9026938575	19880732145	45,41
10+11+12	160	250	64	10	500	842512	1274094	66,13	8955526005	19880732145	45,05
10+11+12	160	250	64	10	200	842512	1274094	66,13	8955526005	19880732145	45,05
10+11+12	160	250	64	10	100	853578	1274094	66,99	9856783536	19880732145	49,58
10+11+12	140	250	56	5	500	608703	1274094	47,78	6386855299	19880732145	32,13
10+11+12	140	250	56	5	200	608692	1274094	47,77	6386868329	19880732145	32,13
10+11+12	140	250	56	5	100	621421	1274094	48,77	6715598710	19880732145	33,78
10+11+12	140	250	56	10	500	608942	1274094	47,79	6520455546	19880732145	32,80
10+11+12	140	250	56	10	200	608942	1274094	47,79	6520455546	19880732145	32,80
10+11+12	140	250	56	10	100	621720	1274094	48,80	7247281407	19880732145	36,45
10+11+12	120	250	48	5	500	608703	1274094	47,78	6386855299	19880732145	32,13
10+11+12	120	250	48	5	200	608692	1274094	47,77	6386868329	19880732145	32,13
10+11+12	120	250	48	5	100	621421	1274094	48,77	6715598710	19880732145	33,78
10+11+12	120	250	48	10	500	608942	1274094	47,79	6520455546	19880732145	32,80
10+11+12	120	250	48	10	200	608942	1274094	47,79	6520455546	19880732145	32,80
10+11+12	120	250	48	10	100	621720	1274094	48,80	7247281407	19880732145	36,45

4.3. Σχόλια-προτάσεις

Όπως προκύπτει από τα αποτελέσματα των δοκιμών, ο αλγόριθμος Edge Betweenness Clusterer αποδίδει αρκετά καλά για μεγάλες τιμές Replication. Στην αντίθετη περίπτωση, όταν δηλαδή το ποσοστό είναι μικρό, δεν επιτυγχάνεται η ανάκτηση μεγάλων τμημάτων του γραφήματος και τα τμήματα που ανακτώνται δεν φαίνεται να επιτυγχάνουν καλή πρόβλεψη της κίνησης των χρηστών. Επίσης παρατηρούμε πως στην περίπτωση που το προκαθορισμένο πλήθος των προς αφαίρεση ακμών είναι μεγάλο, ο αλγόριθμος φαίνεται να είναι λιγότερο αποδοτικός. Η συμπεριφορά αυτή είναι είναι μάλλον αναμενόμενη καθώς όσο περισσότερες ακμές αφαιρούνται από το γράφημα, τόσο μικρότερη είναι η συνεκτικότητά του, με αποτέλεσμα να κατακερματίζεται ο γράφος και να μην ανακτώνται αρκετά συνεκτικά τμήματά του. Έτσι αρκετά τμήματα που δεν ανακτώνται λόγω του κατακερματισμού, φαίνεται ότι μπορεί να είναι «σημαντικά» και γιαυτό το λόγο μειώνεται η αποδοτικότητα του αλγορίθμου .

Οι παρατηρήσεις αυτές μας οδηγούν στο συμπέρασμα πως ο αλγόριθμος δημιουργεί τις ομάδες (clusters) με τρόπο ο οποίος δε φαίνεται να προσομοιώνει τη πραγματική κίνηση του χρήστη στον ιστοχώρο που μελετήσαμε. Η αποδοτικότητά του ασφαλώς αφήνει περιθώρια βελτίωσης.

Μία πιθανή κατεύθυνση μελέτης είναι η λειτουργία του αλγορίθμου πάνω σε κάποιο δυναμικό μοντέλο (Markov) του ιστοχώρου και όχι στο στατικό. Σε αυτή την περίπτωση όμως, θα πρέπει να μελετηθεί η αποδοτικότητα του αλγορίθμου, καθώς όπως έχει προαναφερθεί, ο αλγόριθμος ομαδοποίησης έχει μεγάλη πολυπλοκότητα κάτι που μπορεί να αποτρέπει την εκτέλεσή του σε πραγματικό χρόνο.

ΞΕΝΟΓΛΩΣΣΗ ΚΑΙ ΕΛΛΗΝΙΚΗ ΟΡΟΛΟΓΙΑ

Internet	Διαδίκτυο
Bandwidth	Εύρος ζώνης
Content Delivery Networks	Δίκτυα Διανομής Περιεχομένου
Download	Ανάκτηση περιεχομένου
Server	Εξυπηρέτης
Middleware	Ενδιάμεσο λογισμικό
Load Balancing	Εξισορρόπηση φόρτου
Caching	Μηχανισμός ενδιάμεσης μνήμης
Proxy Server	Ενδιάμεσος εξυπηρέτης
Replication	Αντιγραφή πληροφορίας
Buffer	Μνήμη ενδιάμεσης αποθήκευσης
Client Application	Εφαρμογή πελάτη
Web Browser	Φυλλομετρητής ιστού
IP address	Διεύθυνση διαδικτύου
World Wide Web	Παγκόσμιος Ιστός
Internet Service Provider	Πάροχος υπηρεσιών Διαδικτύου
Cache Consistency	Συνέπεια της ενδιάμεσης μνήμης
Server Load Balancing	Εξισορρόπηση φόρτου εξυπηρέτη
Origin Server	Αρχικός εξυπηρέτης
Request Routing System	Σύστημα δρομολόγησης αιτήσεων
Distribution System	Σύστημα διανομής
Replica Servers	Εξυπηρέτες-αντιγραφείς
Client	Πελάτης
Accounting System	Λογιστικό σύστημα
Billing Organisation	Σύστημα λογαριασμών
Domain Name	Όνομα περιοχής
Software	Υλικό
Cluster	Ομάδα
Clustering	Ομαδοποίηση
Border Gateway Protocol	Πρωτόκολλο πινάκων δρομολόγησης
Spanning Tree	Εκτιμένο δέντρο
Pipelining	Διοχέτευση
Bundles Set	Σύνολο υπογράφων
Cache	Ενδιάμεση μνήμη
Surrogate	Περιφερειακός εξυπηρέτης
Request Logs	Καταγραφή αιτήσεων
Static Site Analysis	Ανάλυση στατικού ιστοχώρου
Bundle	Υπογράφος
Static Site Algorithm	Αλγόριθμος στατικού ιστοχώρου
Automating Software Build	Αυτόματη μεταγλώττιση κώδικα
Open Source Software	Ανοικτό λογισμικό
Project	Έργο
Κατανεμημένα Υποστηρικτικά Συστήματα Αρχείων	Back-End Distributed File Systems
Directory	Φάκελος
Compiler	Μεταφραστής
Command Line	Γραμμή εντολών

Request	Αίτηση
Source Code	Πηγαίος Κώδικας
Object-oriented design	Αντικειμενοστραφής Σχεδιασμός
Clustering Algorithm	Αλγόριθμος Ομαδοποίησης
Log Files	Αρχεία καταγραφής

ΑΚΡΩΝΥΜΙΑ

ΕΚΠΑ	Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών
CDN	Content Delivery Network
HTTP	HyperText Transfer Protocol
IP	Internet Protocol
WWW	World Wide Web
ISP	Internet Service Provider
URI	Uniform Resource Identifier
HTML	HyperText Markup Language
IST	Information Society Technologies
P2P	Peer to Peer
BGP	Border Gateway Protocol
AS	Autonomous System
TCP	Transmission Control Protocol
AFS	Andrew File System
DFS	Distributed File System
XML	Extensible Markup Language
URL	Uniform Resource Locator
ASF	Apache Software Foundation
DOS	Disk Operating System
JAR	Java Archive
SDK	Software Development Kit

ΑΝΑΦΟΡΕΣ

1. Nguyen Thanh Vinh (2005), “Content Distribution Networks over Shared Infrastucture, A Paradigm for Future Content Network Deployment”, University of Wollongong.
2. iClass Consortium (2006), “Research on Rich Media Content Storage, Management and Distribution”, Information Society Technologies.
3. Barford P., Cai Jin-Yi and Gast J. (2002), “Cache Placement Methods Based on Client Demand Clustering”, University of Madison.
4. Chen Y., Qiu L., Chen W., Nguyen L. and Katz R.H. (2002), “Clustering Web content for efficient replication”, University of California, Berkeley.
5. Mathy L., Simpson S., Hutchison D. and Canonico R. (2002), “Scalable Adaptive Hierarchical Clustering”, Book “Lecture notes in computer science”.
6. Grygorash O., Zhou Y. and Jorgensen Z. (2006), "Minimum Spanning Tree Based Clustering Algorithms", 18th IEEE International Conference on Tools with Artificial Intelligence.
7. Wills C., Trott G. and Mikhailov M. (2003), “Using bundles for Web content delivery”, Department of Computer Science, Worcester Polytechnic Institute.
8. Katz E. D., Butler M. and McGrath R. (1994), “A scalable HTTP server: The NCSA prototype”, In Proceedings of the First International World-Wide Web Conference.
9. Spasojevic M., Bowman M. and Spector A. (1994), "Using a Wide-Area File System Within the World-Wide Web", Computer Networks and ISDN Systems.