ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Νέοι τρόποι CACHING στον Παγκόσμιο Δικτυακό Ιστό

Γκανάς Λάμπρος

Μεταπτυχιακή Εργασία

Ηράκλειο, Ιούνιος 1997

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Νέοι τρόποι CACHING στον Παγκόσμιο Δικτυακό Ιστό

Εργασία που υποβλήθηκε από τον
Λάμπρο Γκανά
ως μερική εκπλήρωση των απαιτήσεων
για την απόκτηση
ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΕΙΔΙΚΕΥΣΗΣ

Συγγραφέας:

Γκανάς Λάμπρος
Τμήμα Επιστήμης Υπολογιστών

Εισηγητική Επιτροπή:

Χρήστος Νικολάου, Αναπληρωτής Καθηγητής, Επόπτης

Γεώργιος Σταμούλης, Επίκουρος Καθηγητής, Μέλος

Ευάγγελος Μαρκάτος, Επίκουρος Καθηγητής, Μέλος

Σαράντος Καπιδάκης, Ερευνητής ΙΠ/ΙΤΕ, Μέλος

Δεκτή:

Πάνος Κωνσταντόπουλος
Πρόεδρος Επιτροπής Μεταπτυχιακών Σπουδών

Ηράκλειο, Ιούνιος 1997

Στην οικογένεια μου
και
στην Αναστασία

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή Χρήστο Νικολάου για την καθοδήγησή του και τις πολύτιμες συμβουλές του στην διάρκεια εκπόνησης της εργασίας αυτής. Ευχαριστώ τον καθηγητή Σαράντο Καπιδάκη για την εποικοδομητική συνεργασία μας σε ολόκληρη την διάρκεια των σπουδών μου, τόσο σε προπτυχιακό όσο και σε μεταπτυχιακό επίπεδο. Ακόμη, θα ήθελα να ευχαριστήσω τους καθηγητές μου Γιώργο Σταμούλη και Ευάγγελο Μαρκάτο για την συνεισφορά τους στην τελική διαμόρφωση της εργασίας και την συμμετοχή τους στην επιτροπή εξέτασης.

Ευχαριστίες θέλω να εκφράσω και στους φίλους μου Σαράντη Τούλη, Ηλία Γκρίνια και Μιχάλη Αρταβάνη για τις πολύωρες και γόνιμες συζητήσεις που είχαμε κατά τη διάρκεια της διαμόρφωσης της παρούσας εργασίας. Επίσης, θα ήθελα να ευχαριστήσω τα μέλη της ομάδας των ΠΛΕΙΑΔΩΝ του Ινστιτούτου Πληροφορικής (ΙΠ/ΙΤΕ) για την υποστήριξη τους σε όλη την διάρκεια της εργασίας μου.

Θα ήθελα να ευχαριστήσω επίσης και την οικογένεια μου για την ηθική υποστήριξη που μου προσέφερε σε όλη την διάρκεια των σπουδών μου. Η ηθική και ψυχική συμπαράσταση που βρήκα στο πρόσωπο της Αναστασίας Μπάλια ήτανε ιδιαίτερα σημαντική και με βοήθησε να αντιμετωπίσω κάθε δυσκολία στην πορεία εκπόνησης της εργασίας μου.

Ευχαριστώ τέλος το Τμήμα Επιστήμης Υπολογιστών του Πανεπιστημίου Κρήτης για το υψηλό επίπεδο σπουδών που μου προσέφερε καθώς και το Ινστιτούτο Πληροφορικής του Ιδρύματος Τεχνολογίας και Έρευνας για την υλικοτεχνική υποδομή που μου παρείχε.

Νέοι τρόποι CACHING στον Παγκόσμιο Δικτυακό Ιστό

Γκανάς Λάμπρος
Μεταπτυχιακή εργασία

Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

Περίληψη

Ο παγκόσμιος δικτυακός ιστός, το WWW, αποτελεί έναν από τους ταχύτερα εξελισσόμενους κλάδους της επιστήμης υπολογιστών. Ο αυξανόμενος ρυθμός ανάπτυξης του τείνει να τον καταστήσει θύμα της ίδιας του της επιτυχίας. Η εξυπηρέτηση των χρηστών του WWW συνεχώς επιβαρύνεται από την συμφόρηση που προκαλεί η αυξανόμενη ζήτηση του.

Η παρούσα εργασία πραγματεύεται μεθόδους αντιμετώπισης της συμφόρησης στο WWW, βασισμένους στο caching (αποθήκευση δεδομένων με γρήγορη πρόσβαση). Για να εφαρμόσουμε τις νέες μεθόδους, σχεδιάσαμε και υλοποιήσαμε έναν proxy server (διαμεσολαβητής εξυπηρέτησης αιτήσεων) ο οποίος διαδραματίζει μεσολαβητικό ρόλο στην εξυπηρέτηση των αιτήσεων των χρηστών από τους WWW servers. Οι proxy servers που προτείνονται μπορούν να συνδέονται (σε κάποια συνδεσμολογία δικτύου) και να συνεργάζονται μεταξύ τους με σκοπό την αποδοτικότερη λειτουργία τους. Στην παρούσα εργασία έχει υλοποιηθεί ένα πρωτόκολλο επικοινωνίας το οποίο διέπει τις συναλλαγές τους. Μπορούν να αναζητήσουν αρχεία τα οποία δεν υπάρχουν στην δική τους cache στην cache κάποιου άλλου proxy server. Για να είναι γνωστό εκ των προτέρων ποιος proxy server μπορεί να εξυπηρετήσει τις αιτήσεις αυτές, ανταλλάσσουν μεταξύ τους λίστες με τα πολυτιμότερα αρχεία (πχ. με μεγάλη ζήτηση) των cache τους.

Επιπλέον, σχεδιάσαμε και υλοποιήσαμε αποδοτικούς αλγορίθμους διαχείρισης της χωρητικότητας της cache. Κύριος σκοπός των προτεινόμενων αλγορίθμων είναι η βελτίωση του μέσου χρόνου απόκρισης που αντιμετωπίζουν οι χρήστες στην εξυπηρέτηση των αιτήσεών τους. Η συγκριτική δοκιμή τους με άλλους ήδη καθιερωμένους αλγορίθμους (πχ. Least Recently Used-LRU, Least Frequently Used-LFU) πραγματοποιήθηκε σε έναν προσομοιωτή ο οποίος εξομοιώνει την λειτουργία του proxy server λαμβάνοντας υπόψιν τον πραγματικό φόρτο του WWW (WWW traces). Παρατηρείται βελτίωση της τάξης του 10-15% του χρόνου εξυπηρέτησης των αιτήσεων που είναι ένα αρκετά σημαντικό ποσοστό.

Επόπτης : Χρήστος Νικολάου
Αναπληρωτής Καθηγητής
Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης.

New Caching ways in the World Wide Web

Ganas Labros

Master of Science Thesis

Department of Computer Science
University of Crete

Abstract

The World Wide Web, WWW, is one of the faster evolved computer applications. Its growth rate tends to make it a victim of its own success. Users serving is overloaded from the causing congestion of WWW's growth rate.

The present work deals with solution methods of congestion in the WWW, based in **caching**. In order to apply the new methods, we design and implement a new proxy server which plays a mediator role in serving users requests by the WWW servers. The suggested proxy servers can be connected and cooperate in order to achieve better performance. In the present work, a new communication protocol for their cooperation has been implemented. Proxy servers can seek documents not existing in their cache to a cache of some other proxy server. In order to know which proxy server can service these requests, they interchagne catalogs with the more valuable documents in their caches (e.g. documents with big demand).

Additionally, we design and implement efficient algorithms for managing cache's capacity. The main goal of the suggested algorithms is the improvement of the average response time corresponding to servicing users requests. The comparative testing with some other classical algorithms (e.g. Least Recently Used-LRU, Least Frequently Used-LFU) has been made in a simulator which simulates the operation of proxy's cache having as input real load from the WWW (WWW traces). An improvement of 10-15% of the average response time is observed which is a very important percent.

Advisor : Christos Nikolaou
Associate Professor
Department of Computer Science
University of Crete.

Περιεχόμενα

1 Εισαγωγή **1**

1.1 Ορισμός του προβλήματος 1

1.2 Αντιμετώπιση του προβλήματος 3

1.3 Τοποθεσία της Cache 4

1.4 Στόχοι της εργασίας 5

1.5 Συνεισφορά της εργασίας 6

1.6 Το περιεχόμενο της εργασίας 8

2 Σχετική Εργασία **9**

2.1 CERN proxy server 9

2.1.1 Αρχεία που κρατούνται στην cache 9

2.1.2 Συνεργασία proxy servers 10

2.1.3 Λειτουργία του proxy server 10

2.1.4 Σύγκριση 10

2.2 Netscape proxy server 10

2.2.1 Αρχεία που κρατούνται στην cache 10

2.2.2 Συνεργασία proxy servers 11

2.2.3 Λειτουργία proxy server 11

2.2.4 Σύγκριση 11

2.3 Harvest proxy server 12

2.3.1 Αρχεία που κρατούνται στην cache 12

2.3.2 Συνεργασία proxy servers 12

2.3.3 Λειτουργία proxy server 12
2.3.4 Σύγκριση 13
2.4 Από το caching στο replication (caching goes replication) 13
2.4.1 Λειτουργία του μοντέλου 14
2.4.2 Σύγκριση 15
2.5 Intelligent caching for World-Wide-Web objects 15
2.5.1 Αρχεία που κρατούνται στην cache 16
2.5.2 Σύγκριση 16

3 Προτεινόμενη αρχιτεκτονική κατανεμημένου συστήματος 18

3.1 Ενεργές οντότητες της αρχιτεκτονικής 18
3.2 Γενική λειτουργία αποφάσεων 18
3.2.1 Περιγραφή συναλλαγών 20
3.2.2 Αλγόριθμος αντικατάστασης αρχείων. 20
3.3 Πρωτόκολλο συνεργασίας Proxy Servers 21
3.4 Παραδείγματα 22
3.5 Λεπτομέρειες του πρωτοκόλλου 24
3.5.1 Κατάλογος μηνυμάτων 24
3.5.2 Συναλλαγές συντονιστών δικτύου 24
3.5.3 Συναλλαγές συντονιστή δικτύου και proxy server ενός συστήματος caching 27
3.5.4 Συναλλαγές μεταξύ των proxy servers 28
3.5.5 Εξήγηση των παραμέτρων 28
3.5.6 Αρμοδιότητες κάθε ενεργή οντότητας της αρχιτεκτονικής 29
3.5.7 Αρμοδιότητες του συντονιστή δικτύου 29
3.5.8 Αρμοδιότητες και λειτουργίες του proxy server 29

4 Υλοποίηση 33

4.1 Υλοποίηση του proxy server 33
4.1.1 Υλοποίηση του εξυπηρετητή αιτήσεων 33

4.1.2 Βασικές δομές του εξυπηρετητή αιτήσεων 35
4.1.3 Αντιμετώπιση ταυτόχρονων αιτήσεων για το ίδιο αρχείο 36
4.1.4 Αντιστοίχιση URL σε όνομα αρχείου 38
4.1.5 Αντιμετώπιση περιπτώσεων βλάβης του δικτύου 38
4.2 Υλοποίηση του διαχειριστή της cache 38
4.2.1 Λεπτομέρειες υλοποίησης 39
4.3 Υλοποίηση του συντονιστή δικτύου 39

5 Μελέτη της κίνησης του WWW **42**
5.1 Χαρακτηριστικά του WWW 43
5.1.1 Μελέτη του size 44
5.1.2 Μελέτη του Latency 44

6 Αλγόριθμοι διαχείρισης της cache **47**
6.1 Εισαγωγή 47
6.2 Καθιερωμένοι αλγόριθμοι από τα λειτουργικά συστήματα 47
6.2.1 Αλγόριθμοι LRU, LFU 47
6.3 Αλγόριθμοι από το WWW 48
6.3.1 Αλγόριθμοι LRU-THOLD, LRU-MIN 48
6.3.2 Αλγόριθμος LNC-R-W3 48
6.3.3 Αλγόριθμος PER 49
6.4 Προτεινόμενοι αλγόριθμοι 50
6.4.1 Προτεινόμενος αλγόριθμος (LAT) 50
6.4.2 Προτεινόμενος αλγόριθμος HIT 51
6.4.3 Επιπλέον αλγόριθμοι που δοκιμάστηκαν 53
6.5 Σύγκριση των αλγορίθμων 54
6.6 Αξιοπιστία του κριτηρίου LSR 62
6.7 Σύγκριση με το Harvest Cache 62
6.8 Σύγκριση συνεργασίας proxy servers 64
6.9 Αλγόριθμοι δικτύου 64

6.9.1 Προτεινόμενος αλγόριθμος δικτύου 65
6.9.2 Πειραματικά αποτελέσματα 66

7 Συμπεράσματα **68**

7.1 Μελλοντική εργασία 69

A Το πρωτόκολλο HTTP **71**

A.1 Γενική περιγραφή αίτησης 71
A.1.1 Επικεφαλίδα της αίτησης 71
A.1.2 Πληροφορία του πελάτη 72
A.2 Γενική περιγραφή απάντησης 72
A.2.1 Πεδία απάντησης 73

B Επιπλέον μετρήσεις στο WWW **75**

C Ορολογία **78**

Βιβλιογραφία **81**

Κατάλογος Σχημάτων

1.1 Αρχιτεκτονική του WWW 2

1.2 Ρόλος του Proxy Server στο WWW 5

1.3 Συνεργασία proxy servers στο σύστημα 7

2.1 Ιεραρχική συνεργασία των Harvest proxy servers 13

2.2 Διάδοση της WLIS πληροφορίας. 14

2.3 Αλληλεπίδραση των ενεργών οντοτήτων της αρχιτεκτονικής. 16

3.1 Αλληλεπίδραση των ενεργών οντοτήτων της αρχιτεκτονικής 19

3.2 Αρχιτεκτονική Συστήματος 31

4.1 Δομές του εξυπηρετητή αιτήσεων 36

4.2 Κλήσεις συναρτήσεων του εξυπηρετητή αιτήσεων. 37

4.3 Κλήσεις συναρτήσεων του διαχειριστή της cache. 39

4.4 Δομές του συντονιστή δικτύου. 41

4.5 Κλήσεις συναρτήσεων του συντονιστή δικτύου 41

5.1 Γραφική παράσταση όπου απεικονίζεται η σχέση του μέγεθους των αρχείων με την ζήτηση τους. Τα μικρά σε μέγεθος αρχεία παρουσιάζουν μεγαλύτερη ζήτηση 44

5.2 Γραφική παράσταση όπου απεικονίζεται η σχέση του μέγεθους των αρχείων με το HR. Η αποθήκευση μικρών σε μέγεθος αρχείων βελτιώνει το Hit Ratio . . . 45

5.3 Γραφική παράσταση όπου απεικονίζεται η σχέση του Latency των αρχείων με την ζήτηση τους. Το μεγαλύτερο ποσοστό των αιτήσεων των χρηστών στρέφεται σε αρχεία με μικρό χρόνο απόκτησης 46

5.4 Γραφική παράσταση όπου απεικονίζεται η σχέση του Latency των αρχείων με το HR. Η αποθήκευση αρχείων με μικρό χρόνο απόκτησης σταδιακά βελτιώνει το Hit Ratio 46

6.1 Γραφική παράσταση της απόδοσης των εναλλακτικών αλγορίθμων που δοκιμάστηκαν για την βελτίωση του μέσου χρόνου εξυπηρέτησης των αιτήσεων των χρηστών. Το μέγεθος της cache είναι 131072 Kbytes στα CSD και FORTHNET και 65536 στο FORTH 53

6.2 Το Hit Ratio που έχει ο αλγόριθμος LRU-THOLD για διάφορες τιμές του THRESHOLD. Το μέγεθος της cache είναι 132 MB για το CSD και το FORTHNET και 32 MB για το FORTH 56

6.3 Το Latency Saved Ratio που έχει ο αλγόριθμος LRU-THOLD για διάφορες τιμές του THRESHOLD. Το μέγεθος της cache είναι 132 MB για το CSD και το FORTHNET και 32 MB για το FORTH 56

6.4 Ο μέσος χρόνος εξυπηρέτησης των αιτήσεων στα CSD traces. Ο LAT παρουσιάζει την καλύτερη απόδοση ενώ πολύ καλή απόδοση έχουν και οι LRU-MIN, HIT 57

6.5 Ο μέσος χρόνος εξυπηρέτησης των αιτήσεων στα FORTH traces. Ο LAT παρουσιάζει την καλύτερη απόδοση ενώ πολύ καλή απόδοση έχουν και οι LRU-MIN, HIT 57

6.6 Ο μέσος χρόνος εξυπηρέτησης των αιτήσεων στα FORTHNET traces. Ο LAT παρουσιάζει την καλύτερη απόδοση ενώ πολύ καλή απόδοση έχει και ο LRU-MIN 58

6.7 Ο μέσος χρόνος εξυπηρέτησης των αιτήσεων στα GRNET traces. Ο LAT παρουσιάζει αρκετά καλή απόδοση 58

6.8 Ο μέσος χρόνος εξυπηρέτησης των αιτήσεων στα NTUA traces. Ο LAT παρουσιάζει αρκετά καλή απόδοση 59

6.9 Το Hit Ratio στα CSD traces. Παρατηρείται η ανωτερότητα των HIT και LRU-MIN 59

6.10 Το Hit Ratio στα FORTH traces. Ο HIT παρουσιάζει την καλύτερη απόδοση . . . 60

6.11 Το Hit Ratio στα FORTHNET traces. Ο HIT παρουσιάζει την καλύτερη απόδοση ενώ πολύ καλή απόδοση έχει και ο LRU-MIN 60

6.12 Το Hit Ratio στα GRNET traces. Ο HIT παρουσιάζει αρκετά καλή απόδοση . . . 61

6.13 Το Hit Ratio στα NTUA traces. Ο HIT παρουσιάζει αρκετά καλή απόδοση 61

6.14 Ο μέσος χρόνος εξυπηρέτησης των αιτήσεων όταν συνεργάζονται οι proxy servers. Παριστάνεται γραφικά συναρτήσει του χρονικού διαστήματος που στέλνονται οι κατάλογοι των πολυτιμότερων αρχείων. Στην θέση 1, παριστάνεται η απόδοση του θεωρητικά βέλτιστου αλγορίθμου. Η βελτίωση που παρατηρείται είναι αρκετά σημαντική 67

B.1 Γραφική παράσταση όπου απεικονίζεται η σχέση του μέγεθους των αρχείων με το Latency Saved Ratio. Τα μικρά σε μέγεθος αρχεία βελτιώνουν σταθερά το LSR ενώ δεν παρατηρείται η ίδια σταθερότητα όσο μεγαλώνει το μέγεθος . . . 76

B.2 Γραφική παράσταση όπου απεικονίζεται η σχέση του μέγεθους των αρχείων με το Size Saved Ratio. Και στην περίπτωση αυτή, τα μικρά σε μέγεθος αρχεία βελτιώνουν το SSR αλλά με αργό ρυθμό 76

B.3 Γραφική παράσταση όπου απεικονίζεται η σχέση του Latency των αρχείων με το Latency Saved Ratio. Τα αρχεία με μικρό χρόνο απόκτησης είναι πολύτιμα για το LSR αλλά δεν αρκούν από μόνα τους 77

B.4 Γραφική παράσταση όπου απεικονίζεται η σχέση του Latency των αρχείων με το Size Saved Ratio. Τα αρχεία με μικρό χρόνο απόκτησης συγκεντρώνουν το μεγαλύτερο ποσοστό του SSR 77

Κατάλογος Πινάκων

1.1 Αύξηση του WWW 2

3.1 Λίστα μηνυμάτων μεταξύ Συντονιστών δικτύου 32

3.2 Λίστα μηνυμάτων μεταξύ Εξυπηρετητή αιτήσεων - Διαχειριστή cache 32

3.3 Λίστα μηνυμάτων μεταξύ Proxy server - Συντονιστή δικτύου 32

4.1 Λίστα συναρτήσεων για την διαχείριση των μηνυμάτων του συντονιστή δικτύου. 40

5.1 Χαρακτηριστικά των WWW Traces 43

6.1 Λίστα εναλλακτικών αλγορίθμων για την βελτίωση του μέσου χρόνου απόκρισης στην εξυπηρέτηση των αιτήσεων των χρηστών 52

6.2 Αποτελέσματα των αλγορίθμων για μέγεθος της cache 131072 Kbytes στα CSD, FORTHNET, GRNET, NTUA και 32768 στο FORTH. Δίνεται η βελτίωση του μέσου χρόνου εξυπηρέτησης των αιτήσεων 55

6.3 Αποτελέσματα των αλγορίθμων για μέγεθος της cache 131072 Kbytes στα CSD, FORTHNET, GRNET, NTUA και 32768 στο FORTH. Δίνεται το ποσοστό των αιτήσεων που εξυπηρετούνται από την cache 55

6.4 Η μεταβολή του LSR με την έλευση του χρόνου. Στην πρώτη στήλη δίνεται η έλευση του χρόνου (σε ώρες) από τη πρώτη μέτρηση ενώ στην δεύτερη το ποσοστό της βελτίωσης του μέσου χρόνου εξυπηρέτησης των αιτήσεων των χρηστών 62

6.5 Σύγκριση με το Harvest Cache 63

Κεφάλαιο 1

Εισαγωγή

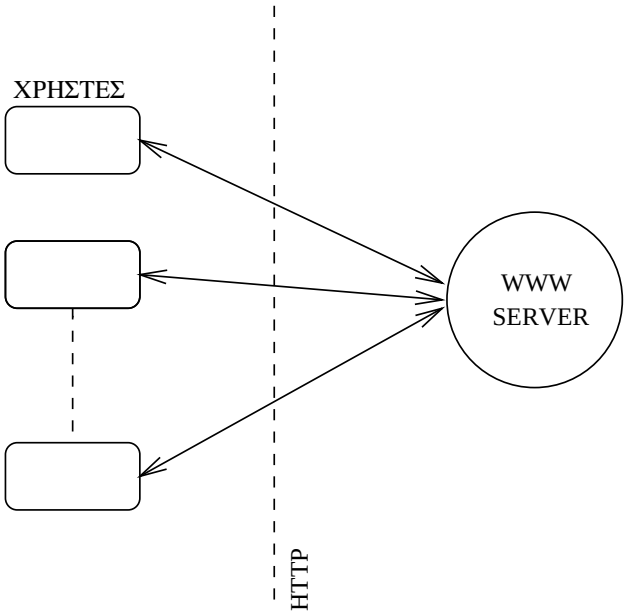
1.1 Ορισμός του προβλήματος

Ο παγκόσμιος δικτυακός ιστός, το WWW, αποτελεί μια από τις ταχύτερα εξελισσόμενες εφαρμογές της επιστήμης υπολογιστών. Αποτελεί το αποφασιστικό βήμα προς την ολοκλήρωση της αναζήτησης πληροφορίας μέσω του Internet.

Η λειτουργία του WWW είναι αρκετά απλή [Hug93]. Είναι δομημένη σε μια αρχιτεκτονική client/server (πελάτη/εξυπηρετητή, σχήμα 1.1). Οι WWW servers δέχονται τις αιτήσεις των clients, τις επεξεργάζονται και επιστρέφουν κάποια απάντηση σ'αυτούς (συνήθως οι clients στέλνουν αιτήσεις για αρχεία οπότε και η απάντηση που δέχονται έχει την μορφή αρχείων). Το πρωτόκολλο που χρησιμοποιείται για τις συναλλαγές των WWW servers και clients ονομάζεται HTTP (Hypertext Transfer Protocol) [BLFF96, FGM+96]. Οι διαδικασίες που επιτελούνται κατά την διάρκεια εξυπηρέτησης μιας αίτησης είναι εντελώς διαφανείς. Ο χρήστης δεν χρειάζεται να έχει ιδιαίτερες γνώσεις υπολογιστών, όπως για παράδειγμα σε ένα άλλο παρόμοιο σύστημα που χρησιμοποιεί το FTP. Δεν απαιτείται ακόμη να γνωρίζει την πραγματική τοποθεσία ενός WWW server. Το μοναδικό που απαιτείται είναι το "πάτημα του ποντικιού" σε κάποιον σύνδεσμο της πληροφορίας που τον ενδιαφέρει.

Τα δύο αυτά μεγάλα πλεονεκτήματα τα οποία προσφέρει η φιλοσοφία της λειτουργίας του WWW, διαθεσιμότητα τεράστιου όγκου πληροφορίας και ευκολία και διαφάνεια στην χρήση του, συνέβαλαν στην ανάπτυξη του [Bra]. Ο ρυθμός ανάπτυξης (σχήμα 1.1) του ήταν και συνεχίζει να είναι εκθετικός. Το 1993, η περίοδος διπλασιασμού της χρήσης του ήταν 3 μήνες ενώ το ίδιο εκτιμάται (δεν υπάρχουν ακριβή στοιχεία) ό,τι συνεχίζει και σήμερα (1997) [Gra96]. Η τεράστια αυτή αύξηση της λειτουργίας του WWW είχε σαν αποτέλεσμα την εμφάνιση συμφόρησης στο Internet. Επιπλέον, ο φόρτος εργασίας των WWW servers ολοένα και αυξάνεται. Όλα αυτά συμβάλλουν σε μεγάλους χρόνους απόκρισης που αντιμετωπίζουν οι χρήστες στην εξυπηρέτηση των αιτήσεων τους.

Χωρίς την λήψη αποτελεσματικών μεθόδων επίλυσης των παραπάνω προβλημάτων, το WWW κινδυνεύει να γίνει θύμα της ίδιας του της επιτυχίας. Είναι φανερό ότι το εύρος ζώνης (Bandwidth) του Internet δεν μπορεί να αυξηθεί με τον ίδιο ρυθμό με τον οποίο αυξάνεται



Σχήμα 1.1: Αρχιτεκτονική του WWW

Ημερομηνία	Πλήθος WWW servers
6/93	130
12/93	623
6/94	2738
12/94	10022
6/95	23500
1/96	100000
6/96 (Εκτίμηση)	230000
1/97 (Εκτίμηση)	650000

Πίνακας 1.1: Αύξηση του WWW

η ζήτηση και η χρήση του WWW διότι αφενός μεν είναι μια ιδιαίτερα δαπανηρή λύση, αφετέρου δε, δεν είναι επεκτάσιμη. Μια εναλλακτική λύση είναι να αυξηθεί η τοπικότητα των δεδομένων που ζητάει ο κάθε client.

1.2 Αντιμετώπιση του προβλήματος

Μια πρακτική λύση η οποία συμβάλλει στην αντιμετώπιση της συμφόρησης στο WWW είναι να διατηρούνται αντίτυπα των αρχείων των WWW servers σε άλλους servers (replication servers) οι οποίοι είναι πιο κοντά στους χρήστες. Η τεχνική αυτή ονομάζεται **Replication** (Mirroring) [BLM+97, BBM+, KsA] και χρησιμοποιήθηκε ευρύτατα σε προγενέστερα συστήματα (όπως σε αυτά που βασίζονται στο πρωτόκολλο File Transfer Protocol-FTP). Το πρόβλημα είναι ότι πρέπει να γνωρίζει ο κάθε χρήστης την ύπαρξη των servers αυτών καθώς και που υπάρχει το αντίτυπο του κάθε αρχείου. Η διαδικασία λειτουργίας του δεν είναι αυτόματη αλλά απαιτεί κάποια ανθρώπινη παρέμβαση (πχ. ο διαχειριστής του συστήματος) για την λήψη των αποφάσεων σχετικά με το κάθε πότε θα δημιουργεί αντίτυπα. Βέβαια, στο WWW, πολλά από τα προηγούμενα προβλήματα υπερσκελίζονται [BBMSRK] (όπως για παράδειγμα, δεν χρειάζεται ανθρώπινη παρέμβαση για την δημιουργία αντιτύπων αλλά μπορούν να δημιουργηθούν από την ζήτηση των αρχείων, τα πιο δημοφιλή αρχεία είναι αυτά που θα κρατηθούν σε αντίτυπα).

Η πρόβλεψη των αρχείων που θα ζητηθούν στο άμεσο μέλλον είναι μια άλλη μέθοδος αντιμετώπισης του προβλήματος [Bes95, Bes96]. Οι WWW servers κάθε φορά που εξυπηρετούν μια αίτηση στέλνουν επιπρόσθετα και τα αρχεία που προβλέπουν ότι ο client θα ζητήσει στο άμεσο μέλλον (πχ εικόνες που εμπεριέχονται σε ένα HTML έγγραφο).

Μια άλλη λύση η οποία προσεγγίζει καλύτερα το πρόβλημα είναι το **Caching**. Είναι μία τεχνική η οποία μπορεί να συμβάλλει στην αύξηση της τοπικότητας της πληροφορίας που διακινείται στο WWW. Η ιδέα είναι γνωστή από τα λειτουργικά συστήματα. Τα δεδομένα που ζητούν οι χρήστες φυλάσσονται σε κάποιον δίσκο ο οποίος είναι πιο κοντά σ'αυτούς. Έτσι, σε επόμενη ζήτηση της ίδιας πληροφορίας, η αίτηση μπορεί να εξυπηρετηθεί τοπικά, χωρίς να σταλεί στον WWW server.

Από τις τρεις δυνατές μεθόδους, προτιμήσαμε το caching γιατί μπορεί να χρησιμοποιηθεί σε εφαρμογές όπως σε ψηφιακή βιβλιοθήκη (πχ. Dienst). Στην περίπτωση αυτή, τα αρχεία είναι δυναμικού χαρακτήρα (αποτέλεσμα ερωτήσεων σε κάποια βάση δεδομένων) οπότε δεν μπορεί να εφαρμοστεί η ιδέα του **Replication** ή του **Prefetching**. Δεν έχει νόημα να κρατούνται αντίτυπα των αποτελεσμάτων των ερωτήσεων αφού υπάρχουν αρκετοί συνδυασμοί ερωτήσεων. Αντίθετα, στο caching, κάτι τέτοιο έχει νόημα αφού κρατούνται στην cache μόνο όσες ερωτήσεις έχουν γίνει από τους χρήστες. Επιπλέον, δεν έχει νόημα να διατηρούνται αντίγραφα της βάσης δεδομένων πάνω στις οποίες γίνονται ερωτήσεις διότι είναι ιδιαίτερα μεγάλη.

Η προσέγγιση του θέματος σε επίπεδο WWW είναι διαφορετική από ό,τι στο caching δίσκου στη μνήμη για τρεις κυρίως λόγους:

- Στο caching δίσκου σε μνήμη, η μονάδα caching είναι το Block ενώ στο WWW, είναι το αρχείο. Αυτό οδηγεί σε δυο ουσιαστικές διαφορές:

  -- Στο caching δίσκου σε μνήμη, γίνεται caching σταθερού μεγέθους Blocks σε αντίθεση με το WWW όπου κρατούνται στην cache αρχεία μεταβλητού μεγέθους.

  -- Στο caching δίσκου σε μνήμη μπορούμε να έχουμε μερικά από τα Blocks ενός αρχείου στην cache, ενώ στο WWW βρίσκεται στην cache ολόκληρο το αρχείο.

- Στο caching δίσκου σε μνήμη, ο χρόνος που απαιτείται για την εξυπηρέτηση κάποιας αίτησης που δεν υπάρχει στην cache είναι σταθερός ενώ στο WWW είναι μεταβλητός και εξαρτάται από την συμφόρηση και το εύρος ζώνης του δικτύου.

1.3 Τοποθεσία της Cache

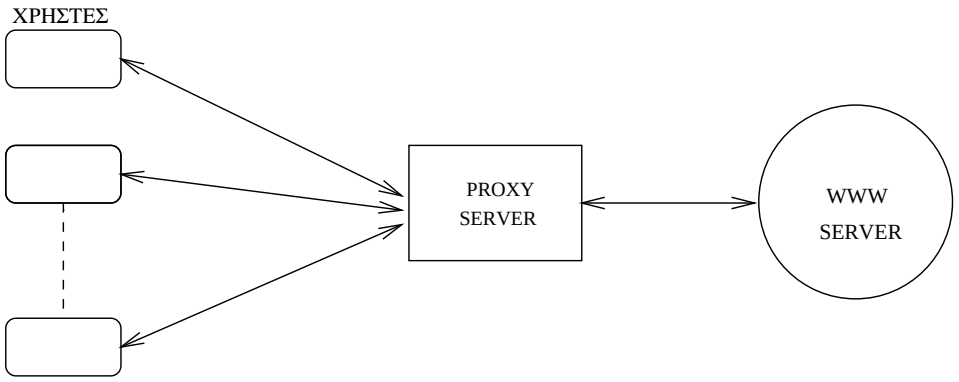
Στο WWW, η cache μπορεί να χρησιμοποιηθεί σε τρία επίπεδα:

- **Σε επίπεδο Server** [Mar96]: αν και όχι τόσο προφανής, ο λόγος που εξηγεί την ύπαρξη cache στο επίπεδο αυτό είναι απλός: να μειώσει τον φόρτο εργασίας των WWW servers. Οι δημοφιλέστεροι WWW servers του Internet αντιμετωπίζουν το πρόβλημα εξυπηρέτησης ενός τεράστιου αριθμού αιτήσεων από όλον τον κόσμο. Η λύση αυτή συμβάλλει στην επίλυση του προβλήματος αυτού.

- **Σε επίπεδο Client**: όλοι οι υπάρχοντες WWW browsers (λογισμικό που αποτελεί την σύνδεση των ανθρώπων με το WWW) διαθέτουν cache. Το μέγεθος της μπορεί να οριστεί από τον χρήστη του συστήματος. Για την εκπλήρωση μιας αίτησης, ο WWW browser προσπαθεί να εξυπηρετήσει την αίτηση αυτή από την cache του. Διαφορετικά στέλνει την αίτηση στον WWW server.

- **Σε επίπεδο δικτύου**: η μη επάρκεια της cache σε επίπεδο client διαφαίνεται όταν γειτονικοί χρήστες (πχ. φοιτητές ενός πανεπιστημίου) ζητάνε την ίδια πληροφορία. Τότε, ο καθένας στέλνει την αίτηση του στον WWW server χωρίς να εκμεταλλεύεται το γεγονός ότι η ζητούμενη πληροφορία πιθανόν να υπάρχει στην cache κάποιου διπλανού WWW browser. Η μονάδα η οποία διαχειρίζεται την cache σε επίπεδο δικτύου ονομάζεται **proxy server** [LA94].

  Το caching σε επίπεδο δικτύου (proxy server) είναι πιο αποτελεσματικό από ό,τι το caching σε κάθε χρήστη. Πρώτον, διότι εξοικονομείται χώρος στην cache γιατί μόνο ένα αντίτυπο της πληροφορίας αποθηκεύεται σ'αυτή και δεύτερον, διότι, μπορούν να εξυπηρετηθούν περισσότεροι του ενός χρήστες οι οποίοι παρουσιάζουν κάποια ομοιότητα στις προτιμήσεις τους (ζητούν συχνά ίδια αρχεία). Η τοποθεσία του proxy server βρίσκεται ανάμεσα στους χρήστες και τους WWW servers (σχήμα 1.2). Διαδραματίζει έναν μεσολαβητικό ρόλο στην εξυπηρέτηση των αιτήσεων των χρηστών. Αρχικά προσπαθεί να εξυπηρετήσει τις αιτήσεις από την cache. Στην περίπτωση αυτή λειτουργεί σαν WWW server μιας και παρέχει υπηρεσίες (αρχεία). Διαφορετικά, στέλνει την αίτηση στον WWW server, λειτουργώντας όπως ένας WWW χρήστης. Κατόπιν,



Σχήμα 1.2: Ρόλος του Proxy Server στο WWW

στέλνει την απάντηση του server στον χρήστη (ο οποίος έστειλε την αίτηση) και φυλάσσει ένα αντίτυπο στην cache. Μελλοντικές αιτήσεις για την ίδια πληροφορία θα εξυπηρετηθούν από την cache, εκτός και αν η πληροφορία αυτή έχει εκδιωχθεί από την cache, είτε επειδή έχασε την προτεραιότητα της, είτε λόγω της περιορισμένης διαθέσιμης χωρητικότητας της cache.

1.4 Στόχοι της εργασίας

Βασικό στόχο της εργασίας αυτής αποτελεί η μελέτη πολιτικών που συμβάλλουν στην αντιμετώπιση της συμφόρησης στο WWW. Για να επιτευχθεί αυτό, χρειάζεται:

- Επέκταση του caching σε κατανεμημένο caching. Αν κάποιο ζητούμενο αρχείο υπάρχει σε κάποιον γειτονικό proxy server είναι προτιμότερο να ζητηθεί το αρχείο αυτό από τον proxy server και όχι από τον WWW server [MLB95].

  Προτείνεται και υλοποιείται ένα πρωτόκολλο επικοινωνίας το οποίο διέπει τις συναλλαγές των proxy servers. Με αυτόν τον τρόπο προσδιορίζονται όλα τα πιθανά μηνύματα τα οποία ανταλλάσσουν οι proxy servers προκειμένου να συνεργαστούν και να υλοποιήσουν την λειτουργία ενός κατανεμημένου συστήματος.

- Συλλογή δεδομένων που απεικονίζουν τις λειτουργίες του WWW (www traces) και μελέτη των ιδιαίτερων χαρακτηριστικών που παρουσιάζει η κίνηση των αιτήσεων και οι προτιμήσεις των WWW χρηστών.

- Μελέτη και υλοποίηση κάποιων αλγορίθμων αντικατάστασης αρχείων λόγω της περιορισμένης διαθέσιμης χωρητικότητας της cache. Όπως έχει ήδη αναφερθεί, η σχεδίαση αλγορίθμων για τον σκοπό αυτό είναι δυσκολότερη υπόθεση σε σχέση με τα παραδοσιακά λειτουργικά συστήματα. Η μείωση του μέσου χρόνο απόκρισης που αντικρύζουν οι WWW χρήστες στην εξυπηρέτηση των αιτήσεων τους αποτελεί τον κύριο στόχο των προτεινόμενων αλγορίθμων. Το μεγαλύτερο ποσοστό των εργασιών με το ίδιο αντικείμενο παραβλέπουν τον σημαντικότατο αυτό παράγοντα προσπαθώντας να σχεδιάσουν αλγορίθμους οι οποίοι απλά βελτιώνουν το ποσοστό των αιτήσεων που

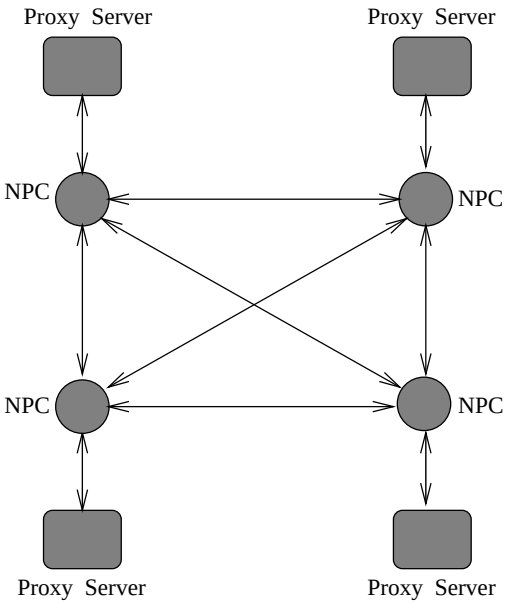
εξυπηρετείται από την cache και όχι τον μέσο χρόνο απόκρισης. Για παράδειγμα, είναι ενδεχομένως σημαντικότερο να διατηρείται στην cache ένα αρχείο του οποίου ο χρόνος απόκτησης του από κάποιον WWW server είναι μεγαλύτερος από τον αντίστοιχο χρόνο κάποιου άλλου αρχείου. Επίσης, προτείνονται αλγόριθμοι οι οποίοι προσπαθούν να βελτιώσουν το ποσοστό των αιτήσεων που εξυπηρετούνται από την cache.

1.5 Συνεισφορά της εργασίας

Για την εφαρμογή και υλοποίηση των παραπάνω ιδεών, έχει σχεδιαστεί και υλοποιηθεί:

- Ένας proxy server ο οποίος δέχεται και εξυπηρετεί τις αιτήσεις των χρηστών και διαχειρίζεται την διαθέσιμη χωρητικότητα της cache. Συνεργάζεται με τον συντονιστή δικτύου τον οποίον ρωτά για την θέση των ζητούμενων αρχείων. Τα ζητούμενα αρχεία μπορεί να υπάρχουν σε κάποιον συνεργαζόμενο proxy server. Έγινε μια πιλοτική υλοποίηση του Proxy Server σε γλώσσα Perl για να ελέγξουμε την αποδοτικότητα των ιδεών μας.
- Ένας συντονιστής των proxy servers του δικτύου (Network Proxy Coordinator, NPC) ο οποίος είναι υπεύθυνος για την συνεργασία των proxy servers με έναν κατανεμημένο τρόπο. Διατηρεί μια λίστα με αρχεία που βρίσκονται στις caches συνεργαζομένων proxy servers. Πληροφορεί τον proxy server σχετικά με τα αρχεία που κρατούν στην cache τους οι άλλοι proxy servers. Υπάρχει ένας για κάθε proxy server.
- Ένα πρωτόκολλο ανταλλαγής μηνυμάτων μεταξύ του proxy server και του συντονιστή δικτύου όσο και μεταξύ των συντονιστών δικτύου για την κατανεμημένη συνεργασία τους.
- Ένα προσομοιωτής (simulator) ο οποίος μπορεί να προσομοιώσει διάφορες πολιτικές αντικατάστασης αρχείων της cache. Σε αυτόν τον προσομοιωτή έχει γίνει η συγκριτική εκτέλεση των προτεινόμενων αλγορίθμων καθώς και αλγορίθμων από τα λειτουργικά συστήματα για να μπορέσουμε να αναπαράγουμε σενάρια με μικρό υπολογιστικό κόστος.

Η συνεισφορά της εργασίας αυτής διαφαίνεται από τον κατανεμημένο τρόπο λειτουργίας των proxy servers. Στην μεγάλη πλειονότητα των ήδη υπαρχόντων proxy servers, είναι δυνατή η συνεργασία τους σε ιεραρχικά επίπεδα. Οι αιτήσεις οι οποίες απέτυχαν να εξυπηρετηθούν από την cache τους στέλνονται στους proxy servers οι οποίοι βρίσκονται στο ανώτερο επίπεδο. Έτσι ανταλλάσσεται μεγάλη πληθώρα μηνυμάτων για την εξυπηρέτηση μιας αίτησης προκαλώντας επιπλέον φόρτο στο δίκτυο. Αν, για παράδειγμα, οι συνεργαζόμενοι proxy servers συνεργάζονται σύμφωνα με την ιεραρχική δομή ενός δένδρου βάθους n, τότε κάποιος proxy server που βρίσκεται στο τελευταίο (κατώτερο) επίπεδο μπορεί να προκαλέσει την ανταλλαγή n μηνυμάτων για κάθε αρχείο που δεν υπάρχει στην cache του. Επιπρόσθετα αν κάποιος proxy server σε ανώτερο επίπεδο δεν λειτουργεί (πχ είναι εκτός λειτουργίας εξαιτίας υπερβολικού φόρτου) η αίτηση αποτυγχάνει να εξυπηρετηθεί και επιστρέφεται



Σχήμα 1.3: Συνεργασία proxy servers στο σύστημα

ένα μήνυμα λάθους στον χρήστη. Αντίθετα, στην προτεινόμενη αρχιτεκτονική μας (σχήμα 1.3), κάθε proxy server ή συντονιστής δικτύου διαθέτει μια λίστα με τα αρχεία τα οποία βρίσκονται στις caches των άλλων συνεργαζομένων proxy servers (δεν απαιτείται ιεραρχική διαδικασία) και είναι απίθανο να εκδιωχθούν από αυτές στο άμεσο μέλλον. Έτσι, κάθε proxy server γνωρίζει εκ των προτέρων αν κάποια αίτηση η οποία απέτυχε να εξυπηρετηθεί από την δική του τοπική cache, δύναται να εξυπηρετηθεί από κάποιον συνεργαζόμενο proxy server. Μ'αυτόν τον τρόπο μειώνονται τα μηνύματα που ανταλλάσσουν οι proxy servers προκειμένου να εκπληρώσουν την συνεργασία τους. Επιπλέον, αντιμετωπίζεται και η περίπτωση που ένας συνεργαζόμενος proxy server είναι εκτός λειτουργίας. Στην περίπτωση όπου περισσότεροι του ενός proxy servers μπορούν να εξυπηρετήσουν την αίτηση, η απόφαση επιλογής εκείνου στον οποίο θα σταλεί λαμβάνεται με βάση τα εξής κριτήρια:

- Τον φόρτο εργασίας του κάθε proxy server. Είναι προτιμότερο να σταλεί η αίτηση στον proxy server που έχει λιγότερο φόρτο εργασίας αποφεύγοντας έτσι να υπερφορτώσουμε έναν ήδη φορτωμένο proxy server.
- Την καθυστέρηση την οποία επιβάλλει το δίκτυο. Είναι προτιμότερος κάποιος κοντινός proxy server παρά κάποιος μακρυνός. Έτσι μειώνεται ο χρόνος απόκρισης που αντικρύζει ο χρήστης στην εξυπηρέτηση της αίτησης του.

Και τα δύο αυτά, συνδυασμένα, μπορούν να μετρηθούν στατιστικά και χρησιμοποιούνται για την διαμόρφωση της απόφασης.

1.6 Το περιεχόμενο της εργασίας

Η εργασία αυτή ξεκινάει με την παρουσίαση κάποιων σχετικών εργασιών στο κεφάλαιο 2. Συνεχίζει με την περιγραφή της προτεινόμενης αρχιτεκτονικής στο κεφάλαιο 3. Στο κεφάλαιο 4 αναλύεται η υλοποίηση των δομικών στοιχείων της προτεινόμενης αρχιτεκτονικής, δηλαδή του proxy server και του συντονιστή δικτύου. Κατόπιν, στο κεφάλαιο 5, παρατίθενται κάποια συμπεράσματα τα οποία εξάχθηκαν από την στατιστική μελέτη της κίνησης του WWW τα οποία χρησιμοποιούνται στην σχεδίαση των προτεινόμενων αλγορίθμων την διαχείρισης της cache που παρουσιάζονται στο κεφάλαιο 6. Στο ίδιο κεφάλαιο γίνεται η συγκριτική δοκιμή των υπάρχοντων αλγορίθμων με τους προτεινόμενους αλγορίθμους. Η εργασία κλείνει εξάγοντας κάποια χρήσιμα συμπεράσματα στο κεφάλαιο 7.

Κεφάλαιο 2

Σχετική Εργασία

2.1 CERN proxy server

2.1.1 Αρχεία που κρατούνται στην cache

Όταν ο CERN proxy server [LBL] λάβει μια αίτηση, προσπαθεί αρχικά να την εξυπηρετήσει από την cache. Αν αυτό δεν είναι δυνατό, η αίτηση προωθείται στον WWW server, λαμβάνεται και αποθηκεύεται η απάντηση του στην cache και κατόπιν επιστρέφεται στον χρήστη. Διαφορετικά, αν η αίτηση μπορεί να εξυπηρετηθεί από την cache, ελέγχεται η εγκυρότητα του αντίτυπου εξετάζοντας την ημερομηνία λήξης του (expiry time). Υπάρχουν δυο περιπτώσεις:

- Το αντίτυπο της cache έχει λήξει. Τότε, στέλνεται μια υπό συνθήκη (*if-modified-since*) αίτηση στον WWW server. Αν ο server απαντήσει με την ένδειξη *Not-modified*, τότε ο proxy server επιστρέφει στον χρήστη το αντίτυπο του αρχείου της cache του. Αν όμως, το αρχείο έχει τροποποιηθεί, τότε ο WWW server στέλνει το αρχείο στον proxy server και κατόπιν αυτός πίσω στον χρήστη.
- Το αντίτυπο της cache δεν έχει λήξει. Τότε, ο proxy server στέλνει το αρχείο του πίσω στον χρήστη.

Για να αποθηκευτεί ένα αρχείο στην cache, πρέπει να πληρεί τις παρακάτω ιδιότητες:

- Η μέθοδος αίτησης του πρέπει να είναι η **GET**.
- Η αίτηση δεν είναι ερώτηση (πχ. δεν υπάρχει ? στο ζητούμενο URL). Αυτό, γιατί οι απαντήσεις σε αυτές τις αιτήσεις είναι δυναμικές και αλλάζουν κάθε φορά που εξυπηρετούνται.
- Το αρχείο δεν προστατεύται. Δεν περιέχει δηλαδή κάποιο **Authorization** πεδίο στην αίτηση του.
- Το αρχείο δεν λήγει αμέσως ή έχει πολύ μικρό χρόνο ζωής.

Επίσης, μπορούν να αποθηκευτούν αιτήσεις των οποίων το URL, συμφωνεί με κάποιο πρότυπο.

2.1.2 Συνεργασία proxy servers

Ο CERN proxy server μπορεί να συνεργάζεται με κάποιον άλλον proxy server. Έτσι, αιτήσεις οι οποίες αποτυγχάνουν να εξυπηρετηθούν τοπικά από την cache του, προωθούνται σε κάποιον άλλον proxy server. Το ελάττωμα του είναι ότι δεν ελέγχεται αν ο συνεργαζόμενος proxy server είναι σε λειτουργία ή έχει τεθεί εκτός λειτουργίας.

2.1.3 Λειτουργία του proxy server

Για κάθε αίτηση που λαμβάνεται από τον CERN proxy server, δημιουργείται μια καινούρια διεργασία για την εξυπηρέτηση της. Όταν τελειώσει η εξυπηρέτηση της αίτησης, η διεργασία αυτή καταστρέφεται. Η δημιουργία διεργασιών για την εξυπηρέτηση αιτήσεων είναι μια δαπανηρή διαδικασία. Έτσι, το σύστημα υπερφορτώνεται όταν υπάρχουν πολλαπλές αιτήσεις σε κάποια δεδομένη χρονική στιγμή.

2.1.4 Σύγκριση

Ο CERN proxy server δεν κρατά αποθηκευμένα αρχεία των οποίων το URL τους υποδηλώνει ερώτηση. Αντίθετα, στον δικό μας proxy server παρέχεται η δυνατότητα να κρατάμε αποθηκευμένα τα αρχεία αυτά στην cache. Ο λόγος είναι ότι πολλές φορές αξίζει να κρατάμε τα αρχεία αυτά, πχ. όταν το δίκτυο είναι πολύ αργό, όταν περιμένουμε οι απαντήσεις να μην αλλάζουν και τόσο πολύ (όπως για παράδειγμα στο Dienst). Κάποια άλλη διαφορά είναι η υπερφόρτωση στην λειτουργία του εξαιτίας της δημιουργίας διεργασίας για κάθε καινούρια αίτηση. Στον δικό μας proxy server γίνεται παράλληλη εξυπηρέτηση αιτήσεων κάνοντας χρήση της κλήσης συστήματος *select*. Τέλος, αυτό που το διαφοροποιεί είναι η συνεργασία του με άλλους proxy servers. Οι αιτήσεις οι οποίες προωθούνται σε κάποιον άλλον proxy server δεν έχουν μεγάλη πιθανότητα να εξυπηρετηθούν απ'αυτό μιας και δεν είναι γνωστά τα αρχεία που βρίσκονται στην cache του.

2.2 Netscape proxy server

2.2.1 Αρχεία που κρατούνται στην cache

Ο Netscape proxy server ακολουθεί την ίδια στρατηγική αποφάσεων με τον CERN proxy server για τα αρχεία που θα αποθηκευτούν στην cache. Πιο συγκεκριμένα, για να κρατηθεί ένα αρχείο στην cache, πρέπει να πληρεί τις παρακάτω ιδιότητες:

- Το αρχείο δεν προστατεύεται και δεν λήγει αμέσως.

- Το αρχείο δεν είναι αποτέλεσμα επεξεργασίας ερώτησης.
- Το αρχείο έχει μέθοδο αίτηση **GET**.

Η εξυπηρέτηση των αιτήσεων γίνεται με τον ίδιο τρόπο που συντελείται στον CERN proxy server. Και εδώ, υποστηρίζεται η υπό συνθήκη (if-modified-since) αίτηση για να διαπιστωθεί αν κάποιο αρχείο έχει τροποποιηθεί. Επίσης, προσφέρεται η δυνατότητα να κρατηθούν στην cache αρχεία που το URL τους συμφωνεί με κάποιο πρότυπο.

2.2.2 Συνεργασία proxy servers

Μπορεί να δημιουργηθεί μια αλυσίδα από proxy servers βασισμένη σε ένα πρότυπο URL. Δηλαδή, κάθε αίτηση που το URL της συμφωνεί με κάποιο πρότυπο και αποτυγχάνει να εξυπηρετηθεί από κάποιο τοπικό proxy server, προωθείται σε κάποιον συγκεκριμένο proxy server. Και εδώ το πρόβλημα είναι ότι αν κάποιος proxy server που βρίσκεται σε υψηλότερη ιεραρχική θέση τεθεί εκτός λειτουργίας, τότε οι χαμηλότερης ιεραρχίας proxy servers δεν αντιλαμβάνονται την κατάσταση αυτή και αποτυγχάνουν να εξυπηρετήσουν όσες αιτήσεις προωθούν σ'αυτούς.

2.2.3 Λειτουργία proxy server

Κατά την έναρξη της λειτουργίας του, ο proxy server δημιουργεί ένα σύνολο από διεργασίες για να εξυπηρετήσουν τις αιτήσεις των χρηστών. Όταν ο proxy server δέχεται μια αίτηση, ο πυρήνας του, αναθέτει την εξυπηρέτηση της σε κάποια από τις διαθέσιμες διεργασίες. Στον τερματισμό της εξυπηρέτησης της αίτησης, η διεργασία γίνεται ξανά διαθέσιμη για την εξυπηρέτηση μελλοντικών αιτήσεων. Έτσι, δημιουργείται ένα σύνολο διεργασιών οι οποίες ποτέ δεν καταστρέφονται. Ένας κύριος παράγοντας που επηρεάζει την απόδοση του proxy server και κατ' επέκταση τον μέσο χρόνο απόκρισης που αντικρίζουν οι χρήστες στις αιτήσεις είναι ο αριθμός των διεργασιών που αρχικά θα δημιουργηθούν. Μικρός αριθμός διεργασιών μεγαλώνει τον μέσο χρόνο εξυπηρέτησης των αιτήσεων χωρίς να αυξάνει όμως τον φόρτο εργασίας του proxy server. Το αντίθετο συμβαίνει με μεγάλο αριθμό διεργασιών.

2.2.4 Σύγκριση

Ο Netscape proxy server ανήκει στην επόμενη γενιά proxy servers, εν συγκρίσει με τον CERN proxy server. Προσφέρει ένα αρκετά καλό μοντέλο λειτουργίας προδημιουργώντας κάποιον αριθμό διεργασιών στην έναρξη της λειτουργίας του. Παρόλα αυτά, η επιλογή κάποιου ιδανικού αριθμού διεργασιών είναι δύσκολη απόφαση. Τέλος, πρόβλημα υπάρχει και με την συνεργασία σε ιεραρχικό επίπεδο όπου δεν μπορεί να ανιχνευθούν οι proxy servers που έχουν τεθεί εκτός λειτουργίας.

2.3 Harvest proxy server

2.3.1 Αρχεία που κρατούνται στην cache

Ο Harvest proxy server [CDCNW95] αποθηκεύει στην cache ότι αρχείο μπορεί να παραμείνει σ'αυτήν. Αυτό περιλαμβάνει αρχεία τα οποία δεν προστατεύονται και δεν λήγουν αμέσως ή δεν είναι αποτέλεσμα επεξεργασίας ερώτησης. Οι επόμενοι κανόνες χρησιμοποιούνται για να αποφασιστεί ποια αρχεία δεν θα παραμείνουν στην cache:

- Οποιοδήποτε αρχείο του οποίου η μέθοδος αίτησης του δεν είναι η **GET**.
- Οποιοδήποτε αρχείο που περιέχει ? στο URL της αίτησης του.
- Οποιοδήποτε αρχείο που περιέχει κάποιο authorization πεδίο στην αίτηση του.

Προς το παρών, δεν υποστηρίζεται η *υπό συνθήκη* αίτηση.

2.3.2 Συνεργασία proxy servers

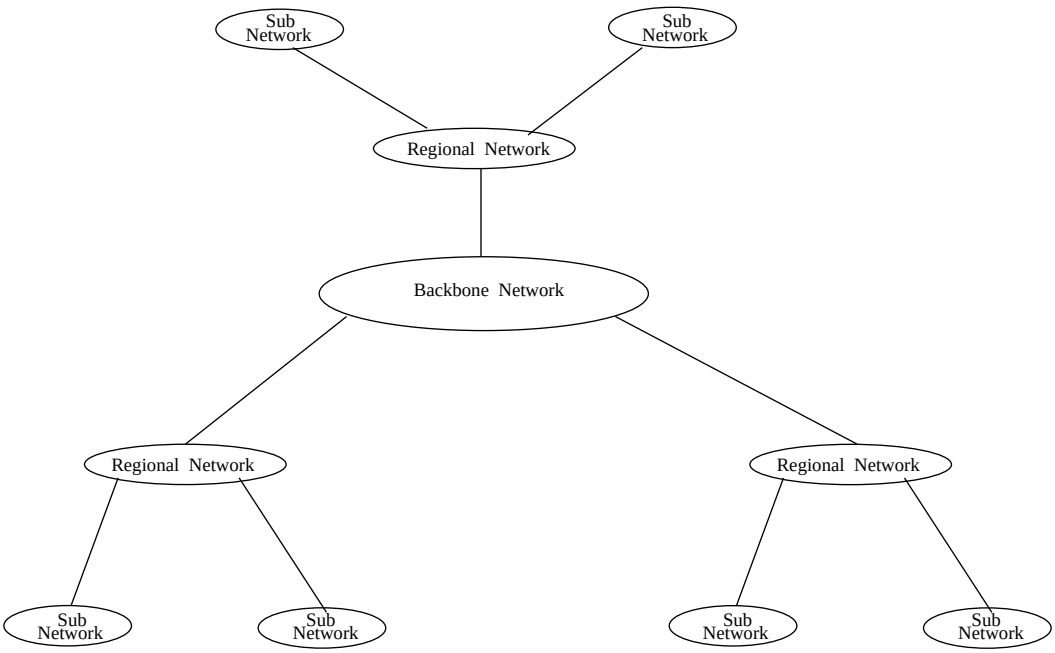
Για να μειωθεί ο φόρτος στο δίκτυο, οι proxy servers προωθούν τις αιτήσεις που δεν εξυπηρετούνται από την δική τους cache σε άλλους proxy servers πιο ψηλά στην ιεραρχία. Επιπρόσθετα, κάθε proxy server μπορεί να προωθήσει τις αιτήσεις του σε άλλους proxy servers στο ίδιο ιεραρχικό επίπεδο (σχήμα 2.1). Κάθε proxy server αποφασίζει αν θα εξυπηρετήσει τις αιτήσεις του από τον WWW server ή κάποιον άλλον proxy server σε κάποιο ιεραρχικό επίπεδο χρησιμοποιώντας ένα πρωτόκολλο που δουλεύει ως εξής.

Αν το ζητούμενο URL (που περιγράφει κάποιο αρχείο) συμφωνεί με κάποιο πρότυπο που έχει προκαθοριστεί να εξυπηρετείται κατευθείαν από τον WWW server, τότε προωθείται κατευθείαν σ'αυτόν, χωρίς να εκμεταλλεύεται την ιεραρχική συνεργασία. Αυτή η δυνατότητα διατίθεται για την εξυπηρέτηση αιτήσεων που είναι ερωτήσεις (πχ. cgi-bin script) κατευθείαν από τον WWW server χωρίς να ακολουθείται ολόκληρη αλυσίδα μηνυμάτων.

Διαφορετικά, όταν ο proxy server λαμβάνει μια αίτηση η οποία δεν μπορεί να εξυπηρετηθεί από την δική του cache, στέλνει ένα μήνυμα σε όλους τους proxy server της δικής του ιεραρχίας και των υψηλότερων ιεραρχιών για να ελέγξουν αν η αίτηση μπορεί να εξυπηρετηθεί απ'αυτούς. Ο proxy server λαμβάνει την απάντηση από εκεί όπου παρατηρείται μικρότερος χρόνος απόκρισης.

2.3.3 Λειτουργία proxy server

Για αύξηση της απόδοσης του, ο Harvest proxy server λειτουργεί σαν μια μοναδική διεργασία κάνοντας παράλληλη εξυπηρέτηση αιτήσεων (με την κλήση συστήματος *select*). Η επικοινωνία με τον δίσκο της cache και τους χρήστες είναι *non blocking*.



Σχήμα 2.1: Ιεραρχική συνεργασία των Harvest proxy servers

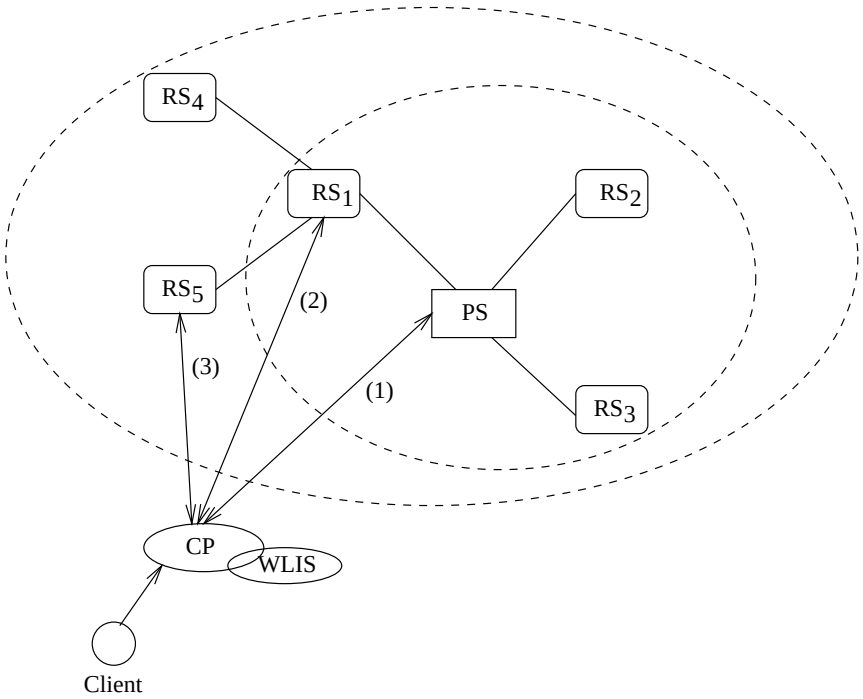
2.3.4 Σύγκριση

Το μοντέλο λειτουργίας του Harvest proxy server μοιάζει με το μοντέλο λειτουργίας του δικού μας proxy server. Η αύξηση της απόδοσης του στηρίζεται στην ταυτόχρονη εξυπηρέτηση πολλαπλών αιτήσεων. Επίσης, προσφέρει ένα αρκετά καλό μοντέλο συνεργασίας. Διαφέρει με το δικό μας μοντέλο συνεργασίας των proxy servers διότι κάθε ζητούμενο αρχείο που δεν υπάρχει στην cache αναζητάται στους συνεργαζόμενους proxy servers στο ίδιο ιεραρχικό επίπεδο ή στο ανώτερο ιεραρχικό επίπεδο. Αντίθετα, στην δική μας περίπτωση, το αρχείο αναζητάται σε κάποιον άλλον proxy server μόνο όταν υπάρχει μεγάλη πιθανότητα να βρεθεί εκεί.

2.4 Από το caching στο replication (caching goes replication)

Η βασική ιδέα [BBMSRK] της εργασίας αυτής είναι η συγχώνευση των λειτουργιών caching-replication. Βασίζεται σε ένα ενεργό σχήμα caching, όπου οι servers μπορούν να αποφασίσουν ποια αρχεία θα κρατήσουν στην cache και που θα γίνει αυτό. Κάτι τέτοιο απαιτεί την αλλαγή του ρόλου των (ήδη υπάρχοντων) παθητικών caching servers σε Replication Servers (RS). Οι RS μπορούν να κρατούν αντίγραφα από τον πρωταρχικό server (primary server, PS). Η δημιουργία μιας ιεραρχικής δομής από RS και PS βοηθάει στην βελτίωση της απόδοσης. ο PS πρέπει να ξέρει μόνο το σύνολο των RS του. Κάθε RS μπορεί να έχει δικούς του Replication Servers. Έτσι, δημιουργείται ένα δένδρο από Replication Servers.

Μια ερώτηση η οποία πρέπει να διευθετηθεί είναι το πώς οι χρήστες επιλέγουν τους WWW servers, τους RS και τους PS. Με μια πρώτη ματιά, αυτό οδηγεί σε μετατροπή των



Σχήμα 2.2: Διάδοση της WLIS πληροφορίας.

ήδη υπάρχοντων προγραμμάτων που χρησιμοποιούν οι χρήστες (WWW browsers). Για την αποφυγή του προηγούμενου, κάθε χρήστης μπορεί να επιλέγει να προωθεί τις αιτήσεις του σε έναν *client side proxy* (CP). Κάθε WWW browser παρέχει την δυνατότητα αυτή. Ο CP θα προωθήσει τις WWW αιτήσεις όχι μόνο στους PS αλλά και στους RS, αν αυτό είναι δυνατόν. Η θέση του είναι κοντά στους χρήστες, όπως υποδηλώνει και το όνομα του.

2.4.1 Λειτουργία του μοντέλου

Η λειτουργία του CP προσφέρει τον βασικό μηχανισμό στους χρήστες για να προσπελάσουν ένα σύνολο από servers. Αυτό που απαιτείται είναι κάποιος τρόπος μετάδοσης πληροφορίας σχετικά με το ποιοι RS υπάρχουν και τί αντίγραφα κρατάει ο καθένας εξ'αυτών. Για τον σκοπό αυτό, προτείνεται το WLIS (Web Location and Information Service). Ο κύριος σκοπός του είναι να διατηρεί ποια αρχεία είναι σε αντίγραφα και ποιοι servers ανήκουν σε λογικά σύνολα από RS και PS. Ένα απλό μέρος για την υλοποίηση της λειτουργίας αυτής είναι το CP.

Υποθέτουμε ότι έχουμε έναν πρωταρχικό (primary) server, τον PS ο οποίος αποτελεί την ρίζα ενός λογικού δένδρου από RS. Στο πρώτο επίπεδο αντιγράφων, ο PS γνωρίζει τους RS_1, RS_2, RS_3 ενώ στο δεύτερο επίπεδο ο RS_1 γνωρίζει τους RS_4, RS_5 (σχήμα 2.2).

Στην αρχή, ο CP προωθεί όλες τις αιτήσεις των χρηστών στον κατάλληλο PS (βήμα 1) μιας και δεν υπάρχει άλλη πληροφορία για το λογικό δένδρο. Ο PS απαντά στον CP με το ζητούμενο αρχείο και με μια επιπρόσθετη επικεφαλίδα στο μήνυμα HTTP, όπου περιέχεται η πληροφορία για τους RS_1, RS_2, RS_3. Έτσι, η αρχική πληροφορία για το WLIS φθάνει

στο CP. Για επικείμενες αιτήσεις, ο CP μπορεί να τις προωθήσει είτε στον ίδιο των PS είτε σε κάποιον από τους RS_1, RS_2, RS_3. Για την επιλογή αυτή, λαμβάνεται υπόψιν το εύρος ζώνης (bandwidth) με καθέναν απ'αυτούς. Αν επιλεγεί ο RS_1 (βήμα 2), τότε ο CP μαζί με το ζητούμενο αρχείο θα λάβει και την πληροφορία ύπαρξης των RS_4, RS_5. Στην συνέχεια, αιτήσεις μπορούν να σταλλούν και στους RS_4, RS_5 (βήμα 3).

2.4.2 Σύγκριση

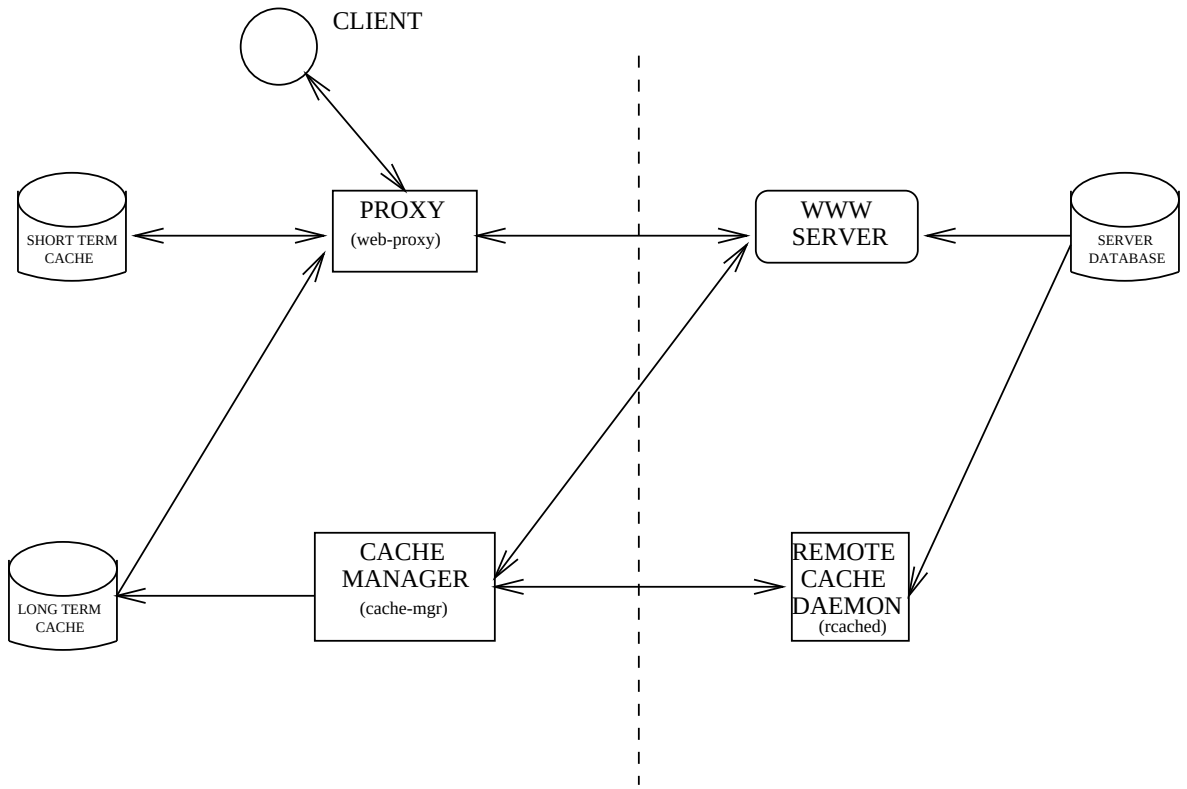
Ο δυναμικός τρόπος με τον οποίον γίνεται γνωστό το λογικό δένδρο συνεργασίας των proxy servers στην προσέγγιση μας παρουσιάζει αρκετές ομοιότητες με τον τρόπο που δουλεύει το WLIS. Η διαφορά έγκειται στο γεγονός ό,τι το WLIS χρησιμοποιείται για replication και όχι για caching. Βέβαια, στο WLIS, δεν υπάρχουν τα προβλήματα ανανέωσης των αντιτύπων των αρχείων λόγω της δυναμικής διάδοσης πληροφορίας που προσφέρει.

2.5 Intelligent caching for World-Wide-Web objects

Στην εργασία αυτή [Wes95] , παρέχεται η δυνατότητα να ελέγχεται αν κάποιο αρχείο μπορεί να κρατηθεί αποθηκεύμενο στην cache ρωτώντας τον ίδιο τον WWW server. Πιο αναλυτικά, τα δομικά στοιχεία της αρχιτεκτονικής στην οποία βασίζεται η ιδέα, είναι:

- **web-proxy**: είναι ο proxy server. Σχεδιασμένος να είναι μικρός (σε κώδικα), γρήγορος και αποδοτικός. Δεν δημιουργεί καινούριες διεργασίες για την εξυπηρέτηση των αιτήσεων αλλά κάνει παράλληλη εξυπηρέτηση με χρήση της κλήσης συστήματος *select*.
- **cache-mgr**: η οντότητα η οποία διαχειρίζεται την cache.
- **rcached**: τρέχει σε συνεργασία με τον WWW server. Το έργο του είναι να παρέχει άδεια σε όσους proxy servers θέλουν να αποθηκεύσουν στην cache τους αρχεία του WWW server. Επίσης, πληροφορεί τους proxy servers κάθε φορά που υπάρχει τροποποίηση αρχείων στον WWW server.

Χρησιμοποιείται cache δύο επιπέδων. Ο web-proxy αποθηκεύει τα αρχεία στην προσωρινή cache (short term cache). Τα αρχεία παραμένουν εκεί είτε μέχρι ο cache-mgr τα μετακινήσει στην μακράς διάρκειας cache (long term cache) είτε μέχρι να το διαγράψει από αυτή. Πριν την μετακίνηση ενός αρχείου στην μακράς διάρκειας cache, ο cache-mgr, προσπαθεί να πάρει άδεια από τον WWW server. Αυτό γίνεται στέλνοντας μια ερώτηση στον αντίστοιχο rcached. Η ερώτηση περιέχει και το χρονικό διάστημα που ο web-proxy επιθυμεί να κρατήσει το αρχείο στην cache του. Αν υπάρχει κάποια απάντηση, λαμβάνεται υπόψιν για την αποθήκευση του αρχείου. Διαφορετικά, το αρχείο αποθηκεύεται στην cache (όπως ούτως ή άλλως γίνεται στην κανονική περίπτωση).



Σχήμα 2.3: Αλληλεπίδραση των ενεργών οντοτήτων της αρχιτεκτονικής.

2.5.1 Αρχεία που κρατούνται στην cache

Για να κρατηθεί κάποιο αρχείο στην cache, πρέπει να πληρεί τις παρακάτω ιδιότητες:

- Η μέθοδος αίτησης του αρχείου είναι η **GET**.
- Η ημερομηνία λήξης του αρχείου δεν είναι αρκετά μικρή.
- Το αρχείο δεν προστατεύεται (δεν υπάρχει Authorization).
- Δεν συνέβηκε κάποιο διακοπή της σύνδεσης του δικτύου κατά την διάρκεια της απόκτησης του από τον WWW server.

2.5.2 Σύγκριση

Στην εργασία αυτή ακολουθείται η ίδια ιδέα υλοποίησης του proxy server (όπως και στον Harvest proxy server άλλωστε) με το δικό μας proxy server. Γίνεται ταυτόχρονη εξυπηρέτηση των αιτήσεων χρησιμοποιώντας την κλήση συστήματος *select*. Έτσι, ο web-proxy κάνει αποδοτική εξυπηρέτηση των αιτήσεων. Η διαφορά με το δικό μας σύστημα έγκειται στο γεγονός ότι δεν έχει ληφθεί ιδιαίτερα υπόψιν η συνεργασία των web-proxies. Επίσης, μια δευτερεύουσα διαφορά είναι ό,τι η λειτουργία του συστήματος βασίζεται και στην συνεργασία του με τους WWW servers. Βέβαια, κάτι τέτοιο, δεν καθιστά ιδιαίτερα ελκυστική

την λύση αυτή. Αντίθετα, στην δική μας προσέγγιση, οι λειτουργίες του proxy server είναι διαφανής τόσο για τους χρήστες όσο και για τους WWW servers.

Κεφάλαιο 3

Προτεινόμενη αρχιτεκτονική κατανεμημένου συστήματος

3.1 Ενεργές οντότητες της αρχιτεκτονικής

Οι ενεργές οντότητες οι οποίες και αποτελούν τα δομικά συστατικά της προτεινόμενης αρχιτεκτονικής είναι:

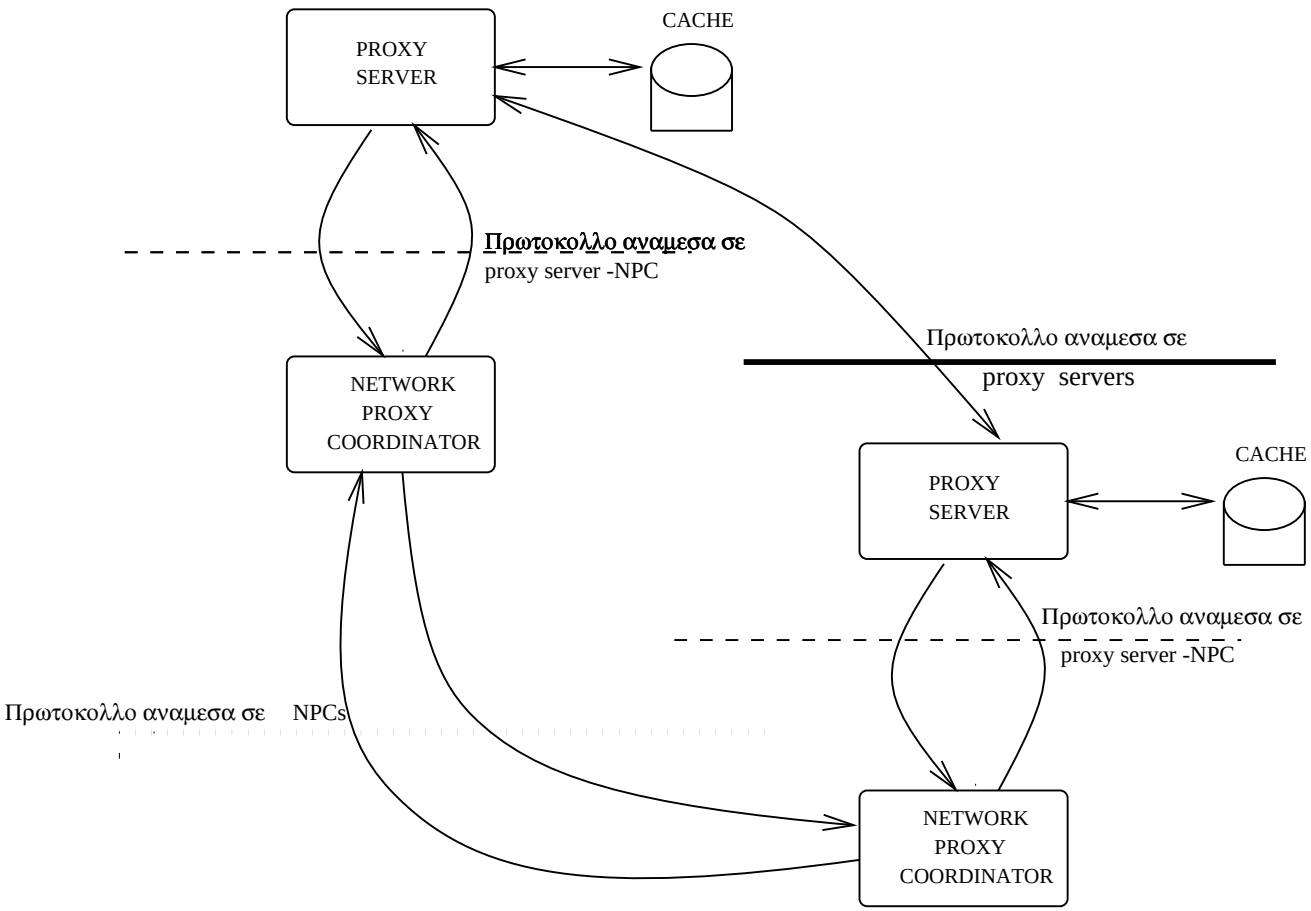
- Τα αρχεία που ζητάνε οι χρήστες του WWW οι οποίοι στέλνουν τις αιτήσεις και θέλουν να εξυπηρετηθούν όσο το δυνατό γρηγορότερα.
- Οι proxy servers οι οποίοι λαμβάνουν τις αιτήσεις αυτές και τις εξυπηρετούν.
- Οι συντονιστές δικτύου (Network Proxy Coordinators, NPC) οι οποίοι είναι αρμόδιοι για την επίτευξη της συνεργασίας των proxy servers.
- Οι WWW servers οι οποίοι είναι οι αρχικοί προμηθευτές των αρχείων.

Σύμφωνα με την αρχιτεκτονική, οι proxy servers που είναι οι μεσάζοντες στην επικοινωνία χρηστών-WWW servers, συνεργάζονται μεταξύ τους. Δέχονται αιτήσεις από τους χρήστες και εξυπηρετούν αυτές τις αιτήσεις. Αν το ζητούμενο αρχείο υπάρχει στην cache, λειτουργούν ως εξυπηρετητές επειδή εξυπηρετούν τους χρήστες. Αν δεν υπάρχει, λειτουργούν ως χρήστες (πελάτες) ζητώντας το από κάποιον άλλον proxy server ή τον WWW server και ακολούθως ως εξυπηρετητές παρέχοντας το πίσω στον χρήστη.

3.2 Γενική λειτουργία αποφάσεων

Για την αποτελεσματικότερη λειτουργία του συστήματος, χρειαζόμαστε αποφάσεις σε 2 επίπεδα:

- Σε κατανεμημένο επίπεδο, στην συνεργασία των proxy servers. Στην περίπτωση αυτή οι proxy servers διαδραματίζουν τον ρόλο του χρήστη γιατί ζητάνε αρχεία από άλλους proxy servers αλλά και τον ρόλο του εξυπηρετητή γιατί παρέχουν αρχεία σε άλλους proxy servers. Σημαντικό ρόλο στις συναλλαγές τους κατέχει το εύρος ζώνης (Bandwidth) του δικτύου που τους συνδέει. Για παράδειγμα, αν κάποιο ζητούμενο αρχείο υπάρχει σε δύο γειτονικούς proxy servers, ενδεχομένως είναι προτιμότερο να ληφθεί από εκείνον που απαιτεί λιγότερο χρόνο. Πρέπει να οριστεί ένα μοντέλο που θα διέπει τις συναλλαγές των proxy servers.
- Σε επίπεδο proxy server, για τον αλγόριθμο αντικατάστασης αρχείων στην cache λόγω της περιορισμένης διαθέσιμης χωρητικότητας της. Οι αποφάσεις της κατηγορίας αυτής αποτελούν υπερσύνολο της προηγούμενης κατηγορίας αποφάσεων γιατί λαμβάνουν υπόψιν τους το εύρος ζώνης του δικτύου για την απόκτηση ενός αρχείου (πχ. πολυτιμότερο αρχείο είναι εκείνο με τον μεγαλύτερο χρόνο απόκτησης). Για παράδειγμα, θα αποφασίζει ποια αρχεία συμφέρει να κρατήσει αποθηκευμένα όταν υπάρχουν νέες εισαγωγές και υπάρχει ανάγκη χώρου.



Σχήμα 3.1: Αλληλεπίδραση των ενεργών οντοτήτων της αρχιτεκτονικής

3.2.1 Περιγραφή συναλλαγών

Έστω $P_1, ..., P_N$ proxy servers και b_{ij} το εύρος ζώνης του δικτύου μεταξύ των P_i και P_j. Οι proxy servers ανταλλάσσουν μεταξύ τους λίστες που περιέχουν τα αρχεία που έχουν στις cache τους.

Όταν ο proxy server δέχεται μια αίτηση, προσπαθεί να την εξυπηρετήσει αρχικά από την cache του. Έτσι, αν το ζητούμενο αρχείο υπάρχει στην cache επιστρέφει το αρχείο αυτό. Διαφορετικά, πρέπει να το φέρει. Λαμβάνει υπόψιν του αν υπάρχει σε κάποιον άλλον proxy server. Έστω C το σύνολο των proxy servers που έχουν το αρχείο αυτό στην cache τους. Η αίτηση θα επιλεγεί να δρομολογηθεί σε εκείνον τον proxy server που μπορεί να την εξυπηρετήσει σε μικρότερο χρόνο. Έστω ότι η αίτηση λήφθηκε από τον P_i proxy server. Το παρακάτω μέγεθος χρησιμοποιείται για να καθοριστεί που θα σταλεί η αίτηση:

$$\mathbf{min}(\mathbf{WST_{aj}} + \frac{\mathbf{SIZE}}{\mathbf{b_{ij}}}) \text{ για κάθε } P_j \text{ στο } C$$

όπου $WST_{aj} + \frac{SIZE}{b_{ij}}$ η συνολική καθυστέρηση για την απόκτηση της απάντησης στην αίτηση και αναλύεται σε:

- WST_{aj} είναι ο μέσος χρόνος αναμονής και εξυπηρέτησης της αίτησης την χρονική στιγμή a στον P_j
- $SIZE$ είναι το μέγεθος της απάντησης.
- $\frac{SIZE}{b_{ij}}$ είναι ο συνολικός χρόνος που απαιτείται για να σταλεί η απάντηση από τον P_j στον P_i.

Το παραπάνω μέγεθος μετράται στατιστικά (πχ. κάθε φορά που συνεργάζονται οι proxy servers). Υπολογίζεται το συνολικό άθροισμα και όχι μόνο τα επιμέρους αθροίσματα αφού δεν μπορεί να γίνει διάκριση μεταξύ του μέσου χρόνου αναμονής της αίτησης σε κάποιο proxy server και του εύρους ζώνης του δικτύου.

3.2.2 Αλγόριθμος αντικατάστασης αρχείων.

Ο αλγόριθμος αντικατάστασης αρχείων είναι πολύ σημαντικός γιατί καθορίζει σε μεγάλο βαθμό την επίδοση του συστήματος. Η διαχείριση της cache και επομένως η αντικατάσταση των αρχείων πρέπει να γίνεται με γνώμονα την ικανοποίηση των κριτηρίων που θέτει ο διαχειριστής του συστήματος.

Τα πιθανότερα κριτήρια προσβλέπουν:

(1) Στην αύξηση του ποσοστού των αιτήσεων των χρηστών που εξυπηρετούνται από την cache. Περιγράφει την πιθανότητα με την οποία οι αιτήσεις των χρηστών εξυπηρετούνται από την cache.

(2) Στην αύξηση της ποσότητας της πληροφορίας (Bytes) που ζητάει ο χρήστης και εξυπηρετείται από την cache.

(3) Στην μείωση του μέσου χρόνου απόκρισης των αιτήσεων των χρηστών.

Τα κριτήρια συνήθως συντρέχουν αλλά δεν είναι ταυτόσημα, ενώ είναι ταυτόσημα στο caching δίσκου στη μνήμη όπου μονάδα caching είναι το Block του δίσκου. Στο WWW, ακόμα και για αρχεία ίδιου μεγέθους, οι χρόνοι απόκτησης τους είναι διαφορετικοί και μάλιστα γίνονται ιδιαίτερα αντιληπτοί από τους χρήστες σε αντίθεση με το caching δίσκου σε μνήμη που είναι σχεδόν ταυτόσημοι. Η απόδοση της cache εξαρτάται από το είδος της κίνησης (αιτήσεις). Μας ενδιαφέρει η λειτουργία της σε ένα πραγματικό περιβάλλον για αυτό και χρησιμοποιούμε στατιστικά δεδομένα τα οποία αποκτήθηκαν από την μελέτη της ήδη υπάρχουσας κίνησης στο δίκτυο. Πιστεύουμε πως είναι ιδιαίτερη σημαντική η ικανοποίηση του $3^{ου}_{=}$ κριτηρίου. Για τον λόγο αυτό, πρέπει να δοθεί μεγαλύτερη σημασία σε αρχεία τα οποία είναι απαντήσεις αιτήσεων με μεγάλο χρόνο απόκρισης. Βέβαια, τα αρχεία αυτά έχουν μικρότερη πιθανότητα επαναζήτησης.

Ως το σημείο αυτό, δόθηκε η θεωρητική περιγραφή των διαδικασιών και των αποφάσεων που απαιτούνται να ληφθούν κατά την διάρκεια εξυπηρέτησης μιας αίτησης. Ουσιαστικά, αναλύθηκε η αλληλεπίδραση των αρχείων (που όπως είπαμε είναι απαντήσεις στις αιτήσεις των χρηστών) με το σύστημα. Το επόμενο βήμα είναι η μελέτη της συνεργασίας του proxy server με τον συντονιστη δικτύου (NPC) όσο και η συνεργασία μεταξύ των συντονιστών δικτύου.

3.3 Πρωτόκολλο συνεργασίας Proxy Servers

Αφηρημένα, ορίζουμε σαν σύστημα caching το σύστημα το οποίο αποτελείται από:

- Proxy server, ο οποίος δέχεται, εξυπηρετεί τις αιτήσεις των χρηστών και διαχειρίζεται την διαθέσιμη χωρητικότητα της cache.
- Network proxy coordinator (συντονιστής δικτύου) ο οποίος αναλαμβάνει την συνεργασία των proxy servers.

Για τον ακριβή ορισμό της επικοινωνίας μεταξύ των συστημάτων caching, πρέπει να προσδιοριστεί ένα λεπτομερές πρωτόκολλο. Το κύριο σημείο είναι να οριστούν οι τύποι των μηνυμάτων που ανταλλάσσονται. Υπάρχουν δύο πιθανά είδη μηνυμάτων:

- Μηνύματα ανταλλαγής πληροφορίας (Information messages). Περιέχουν στοιχεία για τα αρχεία που διατηρούνται στις caches των συνεργαζομένων συστημάτων caching.
- Μηνύματα αιτήσεων. Τα συστήματα caching μπορούν να προωθήσουν τις αιτήσεις οι οποίες απέτυχαν να εξυπηρετηθούν από την δική τους τοπική cache σε άλλα συστήματα caching.

Το πρωτόκολλο δεν ορίζει έναν συγκεκριμένο αλγόριθμο, αλλά δίνει την δυνατότητα υλοποίησης διαφόρων αλγορίθμων χρησιμοποιώντας κατάλληλα τα μηνύματα. Έτσι, μπορούν να υλοποιηθούν διάφοροι αλγόριθμοι με βάση τις δυνατότητες και τις προτιμήσεις του διαχειριστή του συστήματος, όπως υπολογιστική ισχύ, εύρος ζώνης δικτύου και άλλα.

Σε κάθε σύστημα caching παρέχεται η δυνατότητα γνώσης της μεγαλύτερης πλειονότητας των αρχείων που βρίσκονται στις caches άλλων συνεργαζομένων συστημάτων caching. Ανταλλάσσουν μεταξύ τους καταλόγους των *πολυτιμότερων* αρχείων τους στην cache. Ένα θέμα προς συζήτηση είναι ο προσδιορισμός της έννοιας *πολύτιμο* αρχείο. Για τον λόγο αυτό ορίζονται κριτήρια με σκοπό να συμπεριλαμβάνονται στις λίστες αρχεία τα οποία έχουν πολύ μικρή πιθανότητα να διαγραφούν από την cache για κάποιο δεδομένο χρονικό διάστημα.

Όταν ένα σύστημα caching A λάβει μια αίτηση για κάποιο αρχείο το οποίο δεν υπάρχει στην τοπική του cache και γνωρίζει ότι υπάρχει στην cache κάποιου συνεργαζόμενου συστήματος caching B, προωθεί την αίτηση αυτή στον B. Αν ο B έχει όντως το ζητούμενο αρχείο στην cache του το στέλνει στο A. Διαφορετικά, (στην περίπτωση όπου το A δεν ήταν ενήμερο της διαγραφής του ζητούμενο αρχείου από την cache του B) υπάρχουν δύο πιθανές επιλογές:

- Το B στέλνει την αίτηση στον WWW server όπου βρίσκεται το ζητούμενο αρχείο. Κατόπιν, στέλνει την απάντηση (αρχείο) πίσω στο A.
- Το B στέλνει ένα μήνυμα στο A πληροφορώντας τον ότι το ζητούμενο αρχείο δεν υπάρχει στην cache του. Έτσι το A πρέπει να στείλει την αίτηση στον WWW server.

Στην προτεινόμενη προσέγγιση ακολουθήθηκε η πρώτη λύση διότι δεν υπάρχει λόγος πρόσθετης επιβάρυνσης του δικτύου με μηνύματα. Στην περίπτωση αυτή, δεν χρειάζεται να κρατηθεί στην cache του B το αρχείο αυτό δεδομένου ότι είναι μειωμένης σημασίας για το B (διαφορετικά δεν θα είχε διαγραφεί).

3.4 Παραδείγματα

Στην παράγραφο αυτή παρουσιάζονται ορισμένα παραδείγματα από τα οποία διαφαίνεται ο προσδιορισμός μηνυμάτων που απαιτούνται για την συνεργασία των συστημάτων caching. Το πρώτο βήμα είναι απλό. Ξεκινά η λειτουργία των συστημάτων caching.

Παράδειγμα 1

Περίπτωση όπου μόλις ξεκινά η λειτουργία του συστήματος caching A. Ας υποθέσουμε ότι είναι γνωστό κάποιο γειτονικό σύστημα caching B. Το σύστημα θέλει να μάθει άλλα συστήματα caching για να συνεργαστεί.

1. Στέλνει μια αίτηση στο σύστημα caching B για να αποκτήσει ένα κατάλογο των ήδη υπάρχοντων συστημάτων που συνεργάζονται με τον B.

2. Το Β απαντά με τον κατάλογο αυτό.
3. Το Α αποφασίζει ποιά συστήματα από τον κατάλογο είναι αποδοτικά (πχ. σχετικά κοντά) για συνεργασία (επόμενο παράδειγμα).

Στο παράδειγμα αυτό περιγράφηκε ο τρόπος με τον οποίο κάποιο σύστημα caching που μόλις ξεκίνησε γνωρίζει άλλα συστήματα caching. Στο επόμενο παράδειγμα, παρουσιάζεται ο τρόπος με τον οποίο επέρχεται η συνεργασία μεταξύ συστημάτων.

Παράδειγμα 2

Περίπτωση όπου το σύστημα Α θέλει να συνεργαστεί με το σύστημα Β. Ακολουθούνται τα παρακάτω βήματα:

1. Το Α στέλνει μια αίτηση στον Β για να διαπιστώσει τον μεταξύ τους χρόνο επικοινωνίας.
2. Το Β απαντά στην αίτηση του Α έτσι ώστε ο Α να χρονομετρήσει τον χρόνο επικοινωνίας τους.
3. Αν ο προηγούμενος χρόνος επικοινωνίας κριθεί ικανοποιητικός, το Α στέλνει μια αίτηση πρόσθεσης του στην λίστα του Β και το Β απαντά στέλνοντας καταλόγους με τα πολυτιμότερα αρχεία στην cache του. Εάν δεν είναι ικανοποιητικός, το Α τερματίζει την επικοινωνία.
4. Το Β απαντά στην αίτηση του Α.

Παράδειγμα 3

Ας υποθέσουμε την περίπτωση όπου το σύστημα caching Α θέλει να διακοπεί η συνεργασία με το σύστημα Β.

1. Το Α στέλνει την αίτηση που περιγράφει την επιθυμία διακοπής της συνεργασίας στο Β.
2. Αφού ο Β λάβει το μήνυμα, διαγράφει το σύστημα Α από την λίστα των συστημάτων στα οποία στέλνει κατάλογο αρχείων του στην cache.

Αφού δόθηκαν παραδείγματα τα οποία επιδεικνύουν την συνεργασία των συστημάτων caching, ακολουθούν κάποια άλλα που παρουσιάζουν την αλληλεπίδραση τους με τους χρήστες.

Παράδειγμα 4

Ο proxy server δέχεται αίτηση για κάποιο αρχείο το όποιο βρίσκεται στην cache του.

1. Ψάχνει για αρχείο στην cache το οποίο αντιστοιχεί στην αίτηση.
2. Στέλνει πίσω το αρχείο στον χρήστη.

Παράδειγμα 5

Ο proxy server δέχεται αίτηση για κάποιο αρχείο που δεν υπάρχει στην cache του.

1. Αποφασίζει αν υπάρχει το ζητούμενο αρχείο σε κάποιον συνεργαζόμενο proxy server.
2. Στέλνει αίτηση για το αρχείο αυτό στον WWW server ή σε κάποιον συνεργαζόμενο proxy server (προηγούμενο βήμα).
3. Λαμβάνει την απάντηση (δηλαδή το αρχείο) και αποκτά τον απαιτούμενο χώρο στην cache (εφαρμόζοντας τον αλγόριθμο διαχείρισης της cache) για να αποθηκεύσει το αρχείο για μελλοντικές ζητήσεις.
4. Στέλνει την απάντηση πίσω στον χρήστη.

3.5 Λεπτομέρειες του πρωτοκόλλου

Μέχρι το σημείο αυτό, παρουσιάσαμε τα βασικά θέματα που άπτονται της επικοινωνίας των συστημάτων caching σε θεωρητικό επίπεδο. Στις επόμενες παραγράφους δίνεται μια λεπτομερής εξήγηση των μηνυμάτων.

3.5.1 Κατάλογος μηνυμάτων

Έστω ότι έχουμε n συστήματα caching $C_1, ..., C_n$. Αναφερόμαστε στον proxy server του συστήματος C_i με CP_i και στον αντίστοιχο συντονιστή δικτύου με CN_i. Τα μηνύματα κατατάσσονται σε τρεις βασικές κατηγορίες:

- Μηνύματα που διέπουν τις συναλλαγές των συντονιστών δικτύου.
- Μηνύματα που διέπουν την επικοινωνία proxy server και συντονιστή δικτύου κάποιου συγκεκριμένου συστήματος caching.
- Μηνύματα που διέπουν την επικοινωνία μεταξύ των proxy servers.

3.5.2 Συναλλαγές συντονιστών δικτύου

Διαιρούμε τα μηνύματα που απαιτούνται για την συνεργασία των συντονιστών δικτύου σε τρεις βασικές κατηγορίες:

- Μηνύματα για την αρχικοποίηση της συνεργασίας μεταξύ δύο συντονιστών δικτύου.
- Μηνύματα για τον τερματισμό της συνεργασίας μεταξύ δύο συντονιστών δικτύου.
- Μηνύματα που ανταλλάσσονται κατά την διάρκεια της συνεργασίας δύο συντονιστών δικτύου.

Παραθέτουμε τα μηνύματα κατά κατηγορία:

- Τα μηνύματα τα οποία απαιτούνται για την έναρξη της συνεργασίας μεταξύ δύο συντονιστών δικτύου CN_i και CN_j (υποθέτοντας ότι ο CN_i στέλνει το μήνυμα στον CN_j) είναι:

  ALIVE cache version

  Η χρησιμότητα του μηνύματος αυτού είναι να διαπιστωθεί αν ο συντονιστής δικτύου CN_j βρίσκεται σε λειτουργία και να εκτιμηθεί ο χρόνος που απαιτείται για την μεταξύ τους επικοινωνία. Είναι ένα είδος μηνύματος *Ping*. Απόκριση στο μήνυμα αυτό είναι μια γενική απάντηση (περιγράφεται παρακάτω). Στέλνεται μια φορά, στην εκκίνηση του συστήματος. Είναι το πρώτο μήνυμα μιας ακολουθίας μηνυμάτων.

  ADDME cache version

  Ο CN_i πληροφορεί τον CN_j για την επιθυμία του να προστεθεί στην λίστα των συντονιστών δικτύου όπου ο CN_j στέλνει τον κατάλογο με τα *πολυτιμότερα* αρχεία στην cache του. Η χρησιμότητα του μηνύματος αυτού είναι η πρόσθεση ενός *νεοεισαγόμενου* συστήματος caching στην "κοινωνία" των ήδη υπάρχοντων συστημάτων caching. Με άλλα λόγια χρησιμοποιείται για την έναρξη της συνεργασίας μεταξύ δυο συντονιστών δικτύου. Απόκριση στο μήνυμα αυτό είναι μια γενική απάντηση. Στέλνεται μια φορά, μετά το μήνυμα ALIVE και αφού ο χρόνος της μεταξύ τους εποκοινωνίας (όπως προκύπτει από το μήνυμα ALIVE) είναι ικανοποιητικός.

  LISTCACHES cache version

  Η χρησιμότητα του μηνύματος αυτού είναι να αποκτηθούν τα συστήματα caching τα οποία γνωρίζει ο CN_j. Η απόκριση στο μήνυμα αυτό είναι μια λίστα από συντονιστές δικτύου. Κάθε γραμμή της λίστας αυτής περιέχει στοιχεία για τον συντονιστή δικτύου όπως το μηχάνημα στο οποίο δουλεύει. Στέλνεται στην αρχή, όταν γίνεται η πρώτη επικοινωνία με τον CN_j.

- Τα μηνύματα τα οποία ανταλλάσσονται κατά την διάρκεια της συνεργασίας δύο συντονιστών δικτύου CN_i και CN_j (υποθέτοντας ότι ο CN_i στέλνει το μήνυμα στον CN_j) είναι:

  LISTREMDOCS cache version

  Η χρησιμότητα του μηνύματος αυτού είναι για να σταλεί μια λίστα με τα

πιο *πολύτιμα* αρχεία που βρίσκονται στην cache του συστήματος caching C_i. Αποτελείται από μία λίστα από URL, ένα σε κάθε γραμμή, με κάποια πληροφορία για αυτά. Δεν υπάρχει απόκριση στο μήνυμα αυτό. Ο χρόνος αποστολής του μηνύματος αυτού δεν είναι σταθερός αλλά καθορίζεται από κάποια κριτήρια (για παράδειγμα, στέλνεται όταν έχει συνολικά διαγράφει από την cache πληροφορία ίση με το μισό του μεγέθους της cache).

PING cache version

χρησιμοποιείται για να διαπιστωθεί αν ο CN_j βρίσκεται σε λειτουργία. Η μόνη διαφορά με το ALIVE είναι ότι δεν χρονομετρά τον χρόνο επικοινωνίας, ούτε στέλνεται μόνο μια φορά, αλλά σε τακτά χρονικά διαστήματα. Απόκριση στο μήνυμα αυτό είναι μια γενική απάντηση.

- Τα μηνύματα τα οποία ανταλλάσσονται για τον τερματισμό της συνεργασίας των συντονιστών δικτύου CN_i και CN_j (υποθέτοντας πάλι ότι ο CN_i στέλνει το μήνυμα στον CN_j) είναι:

STOP cache version

Με το μήνυμα αυτό ο CN_j πληροφορείται ότι ο CN_i δεν επιθυμεί την περαιτέρω λήψη καταλόγων με τα πολυτιμότερα αρχεία της cache του. Ο CN_j διαγράφει τον CN_i από την λίστα των συντονιστών δικτύου στους οποίους στέλνει πληροφορία με τα αρχεία της cache του. Δεν απαιτείται απόκριση στο μήνυμα αυτό.

EXIT_COOP cache version

Ο CN_j πληροφορείται ότι πρέπει να σταματήσει κάθε είδους συνεργασία με τον CN_i (πιθανόν γιατί το σύστημα caching C_i θα τεθεί εκτός λειτουργίας). Έτσι δεν πρέπει να στέλνει πληροφορία στον CN_i, ούτε να περιμένει από αυτόν. Δεν υπάρχει απόκριση σε αυτό το μήνυμα.

Η γενική απάντηση είναι της μορφής:

code cache version:

Το code είναι ένας τριψήφιος αριθμός ο οποίος δηλώνει το αποτέλεσμα επεξεργασίας και εξυπηρέτησης μιας αίτησης. Υπάρχουν δυο τριψήφιοι κωδικοί αριθμοί:

- 200, η αίτηση λήφθηκε και εξυπηρετήθηκε επιτυχώς (θετική απάντηση).
- 400, η αίτηση λήφθηκε αλλά απέτυχε να εξυπηρετηθεί (αρνητική απάντηση).

3.5.3 Συναλλαγές συντονιστή δικτύου και proxy server ενός συστήματος caching

- Ο proxy server CP_i στέλνει τα ακόλουθα μηνύματα στον συντονιστή δικτύου CN_i με σκοπό:

  URL url version

  να ζητήσει πληροφορία για την θέση του αρχείου το οποίο περιγράφεται από το **url**. Το αρχείο μπορεί να υπάρχει σε κάποιο άλλο σύστημα caching ή σε WWW server. Η απόκριση στο μήνυμα αυτό είναι η θέση του αρχείου ή κάποιο ενδεικτικό ότι το αρχείο δεν υπάρχει σε κανέναν συνεργαζόμενο σύστημα caching. Το μήνυμα αυτό στέλνεται όποτε απαιτείται πληροφορία για αρχείο το οποίο δεν υπάρχει στην cache του.

  LISTLOCALDOCS cache version

  να στείλει έναν κατάλογο των πολυτιμότερων αρχείων που έχει στην cache του. Συνήθως περιλαμβάνει όλα τα αρχεία της cache του στο κατάλογο αυτό. Ο συντονιστής δικτύου χρειάζεται αυτόν τον κατάλογο για να τον στείλει στα άλλα συνεργαζόμενα συστήματα caching. Δηλαδή, αφού λάβει το μήνυμα αυτό, στέλνει ένα μήνυμα LISTREMDOCS στους συνεργαζόμενους συντονιστές δικτύου. Δεν απαιτείται απόκριση για το μήνυμα αυτό. Κάθε γραμμή του μηνύματος αποτελείται από ένα URL που περιγράφει ένα αρχείο καθώς και πιθανή πληροφορία για το αρχείο αυτό (όπως πχ. ο εκτιμώμενος χρόνος που θα παραμείνει στην cache και μετά θα διαγραφεί). Στέλνεται σε μη τακτές χρονικές στιγμές ανάλογα με το κριτήριο (πχ. ο proxy server στέλνει το μήνυμα αυτό στον συντονιστή δικτύου όταν συνολικά διαγράφεται από την cache πληροφορία ίση με το μισό του μεγέθους της cache).

  STOP_NPC cache version

  να τον πληροφορήσει ότι πρέπει να σταματήσει την λειτουργία του. Όταν ο συντονιστής δικτύου λάβει το μήνυμα αυτό πρέπει να στείλει το μήνυμα EXIT_COOP σε όλους τους συνεργαζόμενους συντονιστές δικτύου για να τερματίσει κάθε μορφή συνεργασίας τους. Δεν απαιτείται απόκριση στο μήνυμα αυτό. Στέλνεται όταν ο proxy server δεν θέλει να συνεργάζεται με κανέναν άλλον proxy server ή θέλει να τεθεί εκτός λειτουργίας με ομαλό τρόπο.

- Ο συντονιστής δικτύου CN_i στέλνει τα ακόλουθα μηνύματα στον proxy server CP_i με σκοπό:

  LISTDOCS cache version

να στείλει ένα κατάλογο με τα πολυτιμότερα αρχεία που βρίσκονται στην cache του συστήματος που περιγράφεται από την παράμετρο cache του μηνύματος. Κάθε γραμμή του μηνύματος αυτού αποτελείται από ένα URL που περιγράφει κάποιο αρχείο και πιθανόν πληροφορία για το αρχείο αυτό. Δεν χρειάζεται απόκριση στο μήνυμα αυτό. Στέλνεται όταν ο συντονιστής δικτύου λάβει κάποιο μήνυμα LISTREMDOCS από κάποιον συνεργαζόμενο συντονιστή δικτύου.

DOWN cache version

να πληροφορήσει τον proxy server ότι το σύστημα caching που περιγράφεται από την παράμετρο cache του μηνύματος είναι εκτός λειτουργίας. Ο proxy server αφού λάβει το μήνυμα αυτό, διαγράφει τον κατάλογο των αρχείων που βρίσκονται στην cache του εκτός λειτουργίας συστήματος και έτσι δεν προωθεί άλλες αιτήσεις σε αυτόν. Δεν χρειάζεται απόκριση στο μήνυμα αυτό. Χρησιμοποιείται όταν ο συντονιστής δικτύου στείλει κάποιο μήνυμα PING σε κάποιον άλλον συντονιστη δικτύου και δεν λάβει απάντηση.

3.5.4 Συναλλαγές μεταξύ των proxy servers

Οι proxy servers επικοινωνούν με τους ίδιους κανόνες που λαμβάνουν και εξυπηρετούν τις αιτήσεις των χρηστών. Με το URL που περιγράφει το ζητούμενο αρχείο. Επιπλέον, δίνεται η δυνατότητα, όταν κάποιος proxy server P_i προωθήσει κάποια αίτηση η οποία απέτυχε να εξυπηρετηθεί από την cache του, σε κάποιον άλλον proxy server P_j και αποτύχει και σε αυτόν να εξυπηρετηθεί, να μην επηρεάζεται η κατάσταση της cache του P_j proxy server. Δηλαδή, ο P_j, αφού αποτύχει να εξυπηρετήσει την αίτηση (από την cache του), στείλει την αίτηση στον WWW server και λάβει την απάντηση, δεν σώζει το αρχείο στην cache του. Αυτό γίνεται για τον λόγο ότι ο P_j διέθετε το ζητούμενο αρχείο στην cache του και μετά το διέγραψε. Δηλαδή ήταν περιορισμένης σημασίας γι'αυτόν. Έτσι, δεν σπαταλάται η διαθέσιμη χωρητικότητα της cache για την εξυπηρέτηση αιτήσεων οι οποίες δεν αντιπροσωπεύουν τις προτιμήσεις των χρηστών που εξυπηρετούνται από τον proxy server.

3.5.5 Εξήγηση των παραμέτρων

Κάθε μήνυμα έχει τρεις παραμέτρους με σκοπό να μοιάζει στα HTTP μηνύματα. Έτσι δεν απαιτείται να γραφτεί καινούριος κώδικας για την διάκριση των HTTP μηνυμάτων και των μηνυμάτων της αρχιτεκτονικής μας. Μόνο η παράμετρος **version** είναι αρκετή για τον σκοπό αυτό. Ακολουθεί λεπτομερέστερη εξήγηση των παραμέτρων.

- Σε όλα τα προηγούμενα μηνύματα, η πρώτη παράμετρος περιγράφει την ιδιότητα του μηνύματος. Αναγράφεται ρητά.
- Η παράμετρος cache περιγράφει την διεύθυνση ενός συστήματος caching.

- **version** είναι η έκδοση του πρωτοκόλλου. Η παρούσα έκδοση είναι η DOPS/1.0 (Distributed Oriented Proxy Server).

3.5.6 Αρμοδιότητες κάθε ενεργή οντότητας της αρχιτεκτονικής

Αφού έχει δοθεί μια λεπτομερής εξήγηση του πρωτοκόλλου επικοινωνίας μεταξύ των ενεργών οντοτήτων της αρχιτεκτονικής, ακολουθεί μια περιγραφή των αρμοδιοτήτων τους. Μερικές απ'αυτές έχουν αναφερθεί αλλά αναλύονται διεξοδικά για την πληρότητα του κειμένου.

3.5.7 Αρμοδιότητες του συντονιστή δικτύου

Κύρια αρμοδιότητα του συντονιστή δικτύου αποτελεί η διεκπεραίωση της συνεργασίας των proxy servers. Λειτουργούν με έναν διαφανή τρόπο. Οι χρήστες δεν γνωρίζουν την ύπαρξη τους. Οι συντονιστές δικτύου επικοινωνούν μεταξύ τους για την αρχικοποίηση της συνεργασίας τους, την ανταλλαγή καταλόγων που περιλαμβάνουν τα πολυτιμότερα αρχεία των cache τους και τον τερματισμό της συνεργασίας τους. Επίσης, παρέχεται η δυνατότητα, κάθε συντονιστής δικτύου να ελέγχει την ύπαρξη των συνεργαζομένων συντονιστών δικτύου σε τακτά χρονικά διαστήματα. Έτσι, δίνεται ένας δυναμικός χαρακτήρας στην μεταξύ τους συνεργασία. Ουσιαστικά, οι συντονιστές δικτύου συμβάλλουν στην υλοποίηση ενός κατανεμημένου (ή απομακρυσμένου, Remote) caching. Δηλαδή, στην επέκταση της χωρητικότητας της cache και της λειτουργίας των συστημάτων caching.

3.5.8 Αρμοδιότητες και λειτουργίες του proxy server

Κύρια αρμοδιότητα του proxy server είναι η λήψη και εξυπηρέτηση των αιτήσεων των χρηστών με όσο το δυνατόν οικονομικότερο τρόπο. Έτσι προσπαθεί να εξυπηρετήσει τις αιτήσεις από την δική του cache. Αν αποτύχει, εκμεταλλεύεται την δυνατότητα συνεργασίας του με άλλους proxy servers. Αρχικά, εξετάζει αν υπάρχει το ζητούμενο αρχείο σε κάποια άλλη cache. Σ'αυτήν την περίπτωση στέλνει την αίτηση στον proxy server που μπορεί να την εξυπηρετήσει με το μικρότερο κόστος. Διαφορετικά, η αίτηση εξυπηρετείται από τον WWW server. Παρέχονται δυο δυνατές μορφές συνεργασίας proxy server-συντονιστή δικτύου:

- **Συνεχούς επαφής**: Στην περίπτωση αυτή ο proxy server στέλνει μια αίτηση URL στον συντονιστή δικτύου για κάθε αίτηση του χρήστη που αποτυγχάνει να την εξυπηρετήσει από την cache του. Υποτίθεται ότι ο proxy server επικοινωνεί μόνο με τον συντονιστή δικτύου.
- **Μερικής επαφής**: Ο συντονιστής δικτύου στέλνει στον proxy server καταλόγους των πολυτιμότερων αρχείων στις caches των συνεργαζομένων συστημάτων caching (με την χρήση του μηνύματος LISTDOCS). Έτσι ο proxy server γνωρίζει την πλειονότητα των αρχείων που βρίσκονται στις άλλες caches. Δεν απαιτείται να ρωτάει τον συντονιστή δικτύου για κάθε αίτηση που απέτυχε να εξυπηρετηθεί τοπικά από την δική του cache.

Ο proxy server δύναται να λειτουργεί και χωρίς την ύπαρξη συντονιστή δικτύου. Έτσι δεν υπάρχει μεταξύ τους συνεργασία. Κάτι τέτοιο ακολουθείται όταν ο διαχειριστής του συστήματος δεν επιθυμεί ή δεν μπορεί να δικτυώσει τον δικό του proxy server αλλά θέλει να απολαμβάνει τους αποδοτικούς αλγορίθμους διαχείρισης της cache.

Τέλος, μια άλλη σημαντική αρμοδιότητα του proxy server είναι και η διαχείριση της διαθέσιμης χωρητικότητας της cache. Ο proxy server έχει διαιρεθεί σε δύο λογικές οντότητες (σχήμα 3.2):

- Τον εξυπηρετητή αιτήσεων (Request Manager, RM) ο οποίος εξυπηρετεί τις αιτήσεις των χρηστών.
- Τον διαχειριστή της cache (Cache Manager, CM) ο οποίος διαχειρίζεται την διαθέσιμη χωρητικότητα της cache.

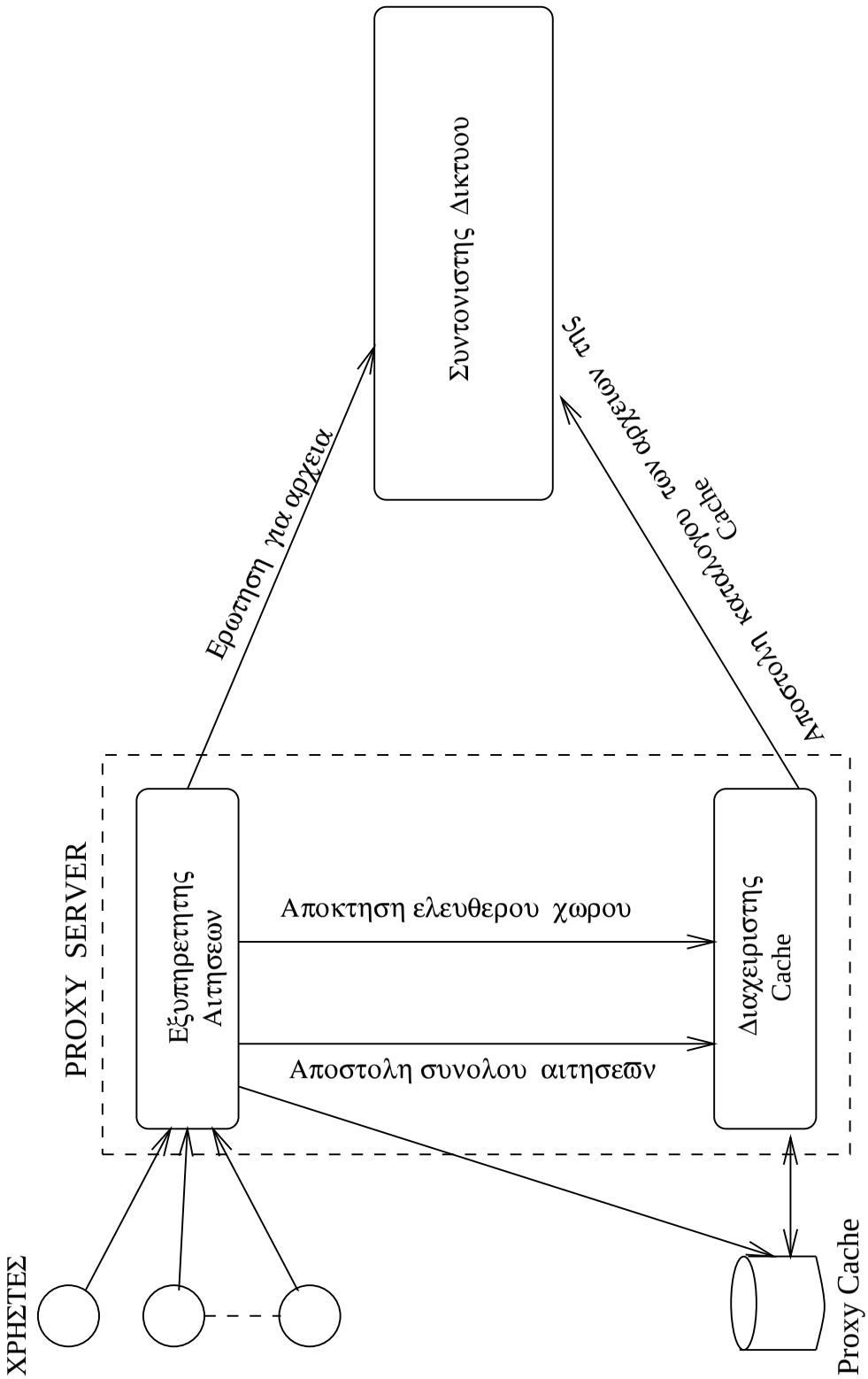
Ένα πρωτόκολλο που διέπει την συνεργασία του εξυπηρετητή αιτήσεων με τον διαχειριστή της cache πρέπει να δοθεί. Ο εξυπηρετητής αιτήσεων επικοινωνεί με τον διαχειριστή της cache για να του ζητήσει διαθέσιμο χώρο για αρχεία που είναι απαντήσεις σε αιτήσεις χρηστών. Δεν γνωρίζει τίποτα για την κατάσταση της cache. Η πληροφορία που ανταλλάσσουν (υποτίθεται ότι ο εξυπηρετητής αιτήσεων στέλνει το μήνυμα στον διαχειριστή της cache) είναι:

SPACE size paramaters
: Η χρησιμότητα του μηνύματος αυτού είναι για να πληροφορήσει τον διαχειριστή ότι απαιτεί χωρητικότητα ίση με **size** για να αποθηκεύσει ένα αρχείο. Η παράμετρος **parameters** περιέχει κάποια στοιχεία για το αρχείο αυτό ώστε να γίνεται αποτελεσματικότερη διαχείριση της cache.

HISTORY proxy version
: Απαιτείται για να πληροφορήσει τον διαχειριστή της cache για τις αιτήσεις που έλαβε. Ορίζεται κάποιος αριθμός ο οποίος καθορίζει κάθε πόσες αιτήσεις θα στέλνεται το μήνυμα στον διαχειριστή της cache. Έτσι, αν ο αριθμός αυτός είναι 100, θα στέλνεται κάθε 100 αιτήσεις.

Η συνεργασία του εξυπηρετητή αιτήσεων με τον διαχειριστή της cache γίνεται με διαφανή τρόπο. Οι χρήστες δεν γνωρίζουν τον καταμερισμό των αρμοδιοτήτων του proxy server. Γίνεται μόνο εσωτερικά, για την καλύτερη απόδοση του όλου συστήματος.

Έχοντας ορίσει την αρχιτεκτονική του όλου συστήματος, στο επόμενο κεφάλαιο ακολουθεί η υλοποίηση των επιμέρους ενεργών οντοτήτων της.



Σχήμα 3.2: Αρχιτεκτονική Συστήματος

Μήνυμα	Αποστολέας	Δέκτης	Απόκριση
ALIVE	Συντ. Δικτύου	Συντ. Δικτύου	Γενική απάντηση
ADDME	Συντ. Δικτύου	Συντ. Δικτύου	Γενική απάντηση
LISTCACHES	Συντ. Δικτύου	Συντ. Δικτύου	Λίστα από Συντ. Δικτύου
LISTREMDOCS	Συντ. Δικτύου	Συντ. Δικτύου	Λίστα από αρχεία
PING	Συντ. Δικτύου	Συντ. Δικτύου	Γενική απάντηση
STOP	Συντ. Δικτύου	Συντ. Δικτύου	-
EXIT_COOP	Συντ. Δικτύου	Συντ. Δικτύου	-

Πίνακας 3.1: Λίστα μηνυμάτων μεταξύ Συντονιστών δικτύου

Μήνυμα	Αποστολέας	Δέκτης	Απόκριση
SPACE	Εξυπ. Αιτήσεων	Διαχ. Cache	Χωρητικότητα
HISTORY	Εξυπ. Αιτήσεων	Διαχ. Cache	-

Πίνακας 3.2: Λίστα μηνυμάτων μεταξύ Εξυπηρετητή αιτήσεων - Διαχειριστή cache

Μήνυμα	Αποστολέας	Δέκτης	Απόκριση
URL	Proxy Server	Συντ. Δικτύου	Τοποθεσία του URL
LISTLOCALDOCS	Proxy Server	Συντ. Δικτύου	Λίστα από αρχεία
STOP_NPC	Proxy Server	Συντ. Δικτύου	-
LISTDOCS	Συντ. Δικτύου	Proxy Server	-
DOWN	Συντ. Δικτύου	Proxy Server	-

Πίνακας 3.3: Λίστα μηνυμάτων μεταξύ Proxy server - Συντονιστή δικτύου

Κεφάλαιο 4

Υλοποίηση

Ο κυριότερος λόγος που οδήγησε στην υλοποίηση ενός καινούριου proxy server και δεν χρησιμοποιήσαμε κάποιον από τους ήδη υπάρχοντες είναι η ευκολότερη εφαρμογή της προτεινόμενης συνεργασίας τους καθώς και των αλγορίθμων διαχείρισης της cache. Πρόκειται για μια πιλοτική υλοποίηση η οποία έχει ως σκοπό την δοκιμή των προτεινόμενων λύσεων για την αντιμετώπιση της συμφόρησης στο WWW.

4.1 Υλοποίηση του proxy server

Ο proxy server αποτελείται από δύο ενεργές οντότητες:

- Τον εξυπηρετητή αιτήσεων (Request Manager).
- Τον διαχειριστή της cache (Cache Manager).

Το κύριο χαρακτηριστικό της υλοποίησης είναι η ταυτόχρονη εξυπηρέτηση των αιτήσεων χωρίς να δημιουργείται καινούρια διεργασία για κάθε αίτηση. Αυτό γίνεται για να μην επιβαρύνεται το σύστημα με επιπλέον δαπανηρές κλήσεις συστήματος (πχ. το fork) εκτός από τις αναγκαίες. Κάθε ενεργή οντότητα του proxy server έχει υλοποιηθεί σαν μια διεργασία η οποία κάνει παράλληλη εξυπηρέτηση των αιτήσεων με την χρήση της κλήσης συστήματος **select** η οποία ελέγχει ποιοί περιγραφείς αρχείων είναι έτοιμοι για έγγραφη/ανάγνωση.

4.1.1 Υλοποίηση του εξυπηρετητή αιτήσεων

Στην έναρξη της λειτουργίας του δημιουργούνται οι δίαυλοι επικοινωνίας (sockets) μέσω των οποίων θα επικοινωνεί με τους χρήστες και τις άλλες ενεργές οντότητες της αρχιτεκτονικής. Συνοπτικά, δημιουργούνται οι ακόλουθοι δίαυλοι επικοινωνίας:

- Αυτό στο οποίο δέχεται τις αιτήσεις των χρηστών (USER_SOCKET).

- Ο απαραίτητος για την συνεργασία του με τον διαχειριστή της cache (CACHE_SOCKET).
- Ο απαραίτητος για την συνεργασία του με τον συντονιστή δικτύου (σε περίπτωση συνεργασίας του με αυτόν, NPC_SOCKET).

Στην συνέχεια, ο εξυπηρετητής αιτήσεων βρίσκεται σε ατέρμονα βρόγχο, περιμένοντας για την λήψη αιτήσεων. Κατά την διάρκεια της κάθε επανάληψης ελέγχει αν κάποιο socket είναι έτοιμο για εγγραφή/ανάγνωση. Έχει οριστεί εκ των προτέρων τί θα γίνει σε περίπτωση που κάποιο socket είναι έτοιμο. Για την εγγραφή και την ανάγνωση σε κάποιο socket υπάρχουν οι συναρτήσεις *write_data* και *read_data* αντίστοιχα οι οποίες διαβάζουν και γράφουν μόνο όσα δεδομένα είναι έτοιμα (είναι δηλαδή non blocking I/O).

Έτσι, αν το USER_SOCKET είναι έτοιμο για ανάγνωση, σημαίνει ότι ο εξυπηρετητής αιτήσεων δέχεται αίτηση από κάποιον χρήστη. Η συνάρτηση που καλείται είναι η *accept_connection* η οποία αρχικοποιεί την έναρξη της εξυπηρέτησης. Μετά, καλείται η *read_data* για να διαβαστεί η αίτηση του χρήστη. Αφού διαβαστούν τα πρώτα δεδομένα της αίτησης, ο εξυπηρετητής αιτήσεων είναι έτοιμος να διακρίνει το είδος της. Έτσι, το έργο της συνάρτησης *service_request* είναι να επεξεργαστεί την πρώτη γραμμή της αίτησης και να καλέσει την αντίστοιχη συνάρτηση:

- Την *service_match_request* για την επεξεργασία αιτήσεων οι οποίες ταιριάζουν σε κάποιο σχήμα (πχ αιτήσεις της ηλεκτρονικής βιβλιοθήκης του Dienst). Ουσιαστικά, η ρουτίνα *service_request* χρησιμοποιείται για το φιλτράρισμα των αιτήσεων ανάλογα με τις προτιμήσεις του διαχειριστή του συστήματος. Έργο της *service_match_request* είναι να διαβάσει τις επόμενες γραμμές της αίτησης και να κρατήσει κάποια χρήσιμα στοιχεία, όπως για το αν το ζητούμενο αρχείο είναι κατάλληλο να αποθηκευτεί στην cache. Στην συνέχεια αποφασίζει αν η εξυπηρέτηση της αίτησης θα γίνει από την δική του cache ή από το δίκτυο (κάποιον WWW server ή κάποιον άλλον proxy server). Αν η εξυπηρέτηση της αίτησης γίνει από την cache, καλείται η *open_cached_doc*. Διαφορετικά, αποφασίζει για την τοποθεσία του ζητούμενου αρχείου στο δίκτυο. Αν συνεργάζεται με τον συντονιστή δικτύου, καλεί μια από τις παρακάτω συναρτήσεις:

  -- Την *send_question* αν ρωτάει τον συντονιστή δικτύου για κάθε ζητούμενο αρχείο το οποίο δεν υπάρχει στην cache του (συνεργασία συνεχούς επαφής).

  -- Την *determine_position_url* αν ο συντονιστής δικτύου στέλνει καταλόγους με τα πολυτιμότερα αρχεία των άλλων συνεργαζομένων συστημάτων caching (συνεργασία μερικής επαφής). Απλά διατρέχει τις δομές του για να διαπιστώσει αν το αρχείο βρίσκεται στην cache κάποιου συνεργαζόμενου proxy server.

  Αφού αποφασιστεί η τοποθεσία του ζητούμενο αρχείου, καλείται η συνάρτηση *prepare_connection* για την έναρξη της επικοινωνίας με τον WWW server ή κάποιον άλλον proxy server.

- Την *service_simple_request* για την επεξεργασία αιτήσεων οι οποίες δεν ταιριάζουν στα πρότυπα των αιτήσεων που γίνονται caching. Διαβάζει τις επόμενες γραμμές της αίτησης και καλεί την συνάρτηση *prepare_connection*. Δεν κοιτάει αν το ζητούμενο αρχείο είναι στην cache.

- Την *read_listdocs* όταν ο συντονιστής δικτύου στέλνει στον εξυπηρετητή αιτήσεων κατάλογο με τα πολυτιμότερα αρχεία που βρίσκονται στην cache ενός συνεργαζόμενου συστήματος caching.
- Την *service_down_req* όταν ο συντονιστής δικτύου πληροφορεί τον εξυπηρετητή αιτήσεων ότι κάποιο συνεργαζόμενο σύστημα caching είναι εκτός λειτουργίας.

Μέχρι το σημείο αυτό, ο εξυπηρετητής αιτήσεων είτε έχει αρχίσει να εξυπηρετεί την αίτηση του χρήστη από την cache, είτε άρχισε την επικοινωνία με τον WWW server (ή τον proxy server) που μπορεί να εξυπηρετήσει την αίτηση. Τα επόμενα βήματα για την ολοκλήρωση της εξυπηρέτησης της αίτησης από την cache είναι:

1. Καλείται η συνάρτηση *read_data* για να διαβαστεί το αρχείο από την cache.
2. Κατόπιν, η συνάρτηση *write_data* για να δοθεί η απάντηση πίσω στον χρήστη.
3. Τέλος, η συνάρτηση *close_connection* η οποία τερματίζει την σύνδεση με τον χρήστη.

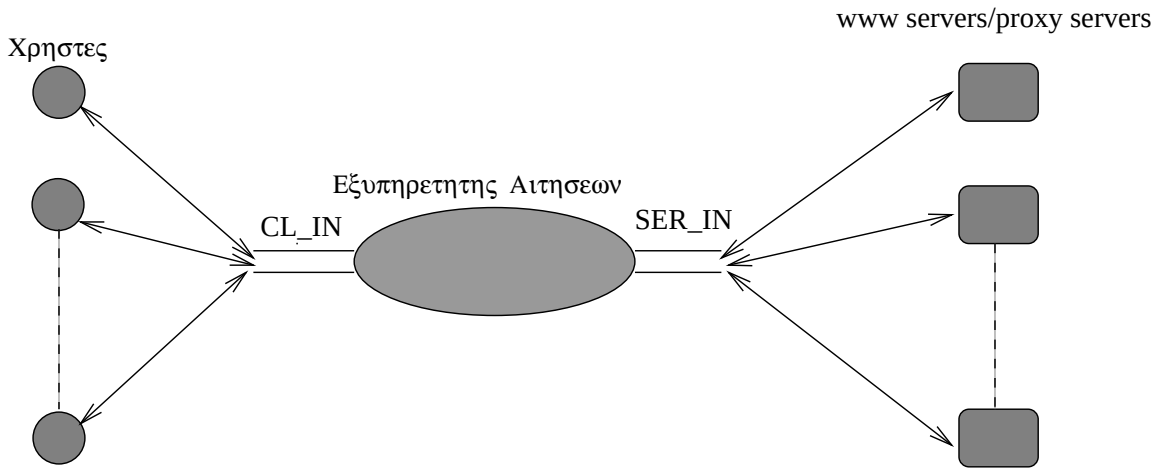
Τα βήματα για την συνέχιση της επικοινωνίας με τον WWW server (ή τον proxy server) είναι:

1. Η *write_data* στέλνει την αίτηση.
2. Η *read_data* διαβάζει την απάντηση του server.
3. Η *read_response_headers* για την επεξεργασία των επικεφαλίδων της απάντησης (πχ. αν η αίτηση πέτυχε ή απέτυχε να εξυπηρετηθεί).
4. Η *write_data* δίνει την απάντηση πίσω στον χρήστη.
5. Η *close_connection* για να τερματιστεί η επικοινωνία με τον χρήστη.
6. Η *send_alloc_req_msg* για να πληροφορήσει τον διαχειριστή της cache ότι απαιτείται ελεύθερος χώρος στην cache.
7. Τέλος, η *save_doc_to_cache* για να αποθηκεύσει το αρχείο στην cache.

4.1.2 Βασικές δομές του εξυπηρετητή αιτήσεων

Για κάθε δίαυλο επικοινωνίας που δημιουργείται για την εξυπηρέτηση αιτήσεων των χρηστών, αντιστοιχούν οι επόμενες δομές (σχήμα 4.1):

- Η *CL_IN* η οποία κρατά στοιχεία για τις αιτήσεις των χρηστών.
- Η *SER_IN* η οποία κρατά στοιχεία για τις απαντήσεις των WWW servers ή των proxy servers.



Σχήμα 4.1: Δομές του εξυπηρετητή αιτήσεων

Σε κάθε δομή κρατούνται αρκετά χρήσιμα στοιχεία. Τα πιο σπουδαία είναι η αίτηση του χρήστη (στην *CL_IN*) ή η απάντηση του server (στην *SER_IN*), το μέγεθος (σε bytes) των προηγούμενων δεδομένων, η επικεφαλίδα της αίτησης και της απάντησης.

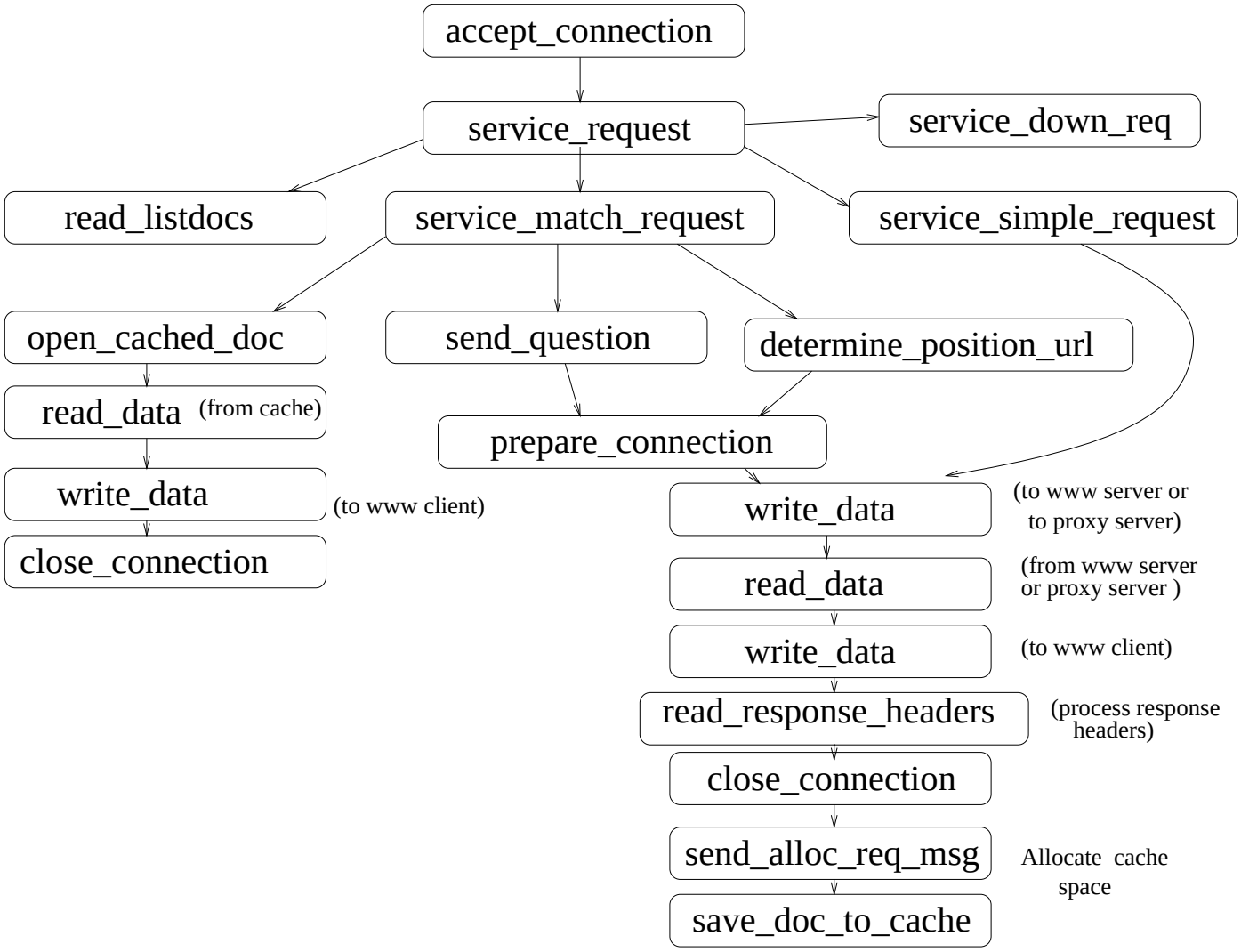
Επίσης, κρατούνται και κάποια στατιστικά στοιχεία για την εξυπηρέτηση μιας αίτησης, όπως:

- Ο χρόνος έναρξης και τερματισμού εξυπηρέτησης της αίτησης.
- Ο χρόνος της επεξεργασίας της αίτησης (χρόνος στην *service_simple_request* ή *service_match_request*).
- Ο χρόνος που απαιτείται για την εγκατάσταση της επικοινωνίας με τον WWW server ή τον proxy server.

4.1.3 Αντιμετώπιση ταυτόχρονων αιτήσεων για το ίδιο αρχείο

Μερικές φορές, ο εξυπηρετητής αιτήσεων συμβαίνει να δέχεται την ίδια χρονική στιγμή (ή πολύ κοντά), πολλαπλές αιτήσεις για το ίδιο αρχείο. Αν το αρχείο αυτό υπάρχει στην cache, τότε, δεν υπάρχει κανένα πρόβλημα. Οι αιτήσεις εξυπηρετούνται απ'αυτήν. Το πρόβλημα εμφανίζεται στην αντίθετη περίπτωση, όταν και αναγκαζόμαστε να κάνουμε πολλαπλές συνδέσεις με τον ίδιο WWW server ή proxy server για το ίδιο αρχείο. Προφανώς και καταναλώνεται το διαθέσιμο εύρος ζώνης (Bandwidth) του δικτύου για άσκοπη χρήση ενώ θα μπορούσαμε να στείλουμε μόνο μια αίτηση στον WWW server και οι υπόλοιπες να εξυπηρετηθούν κατόπιν από την cache.

Ο εξυπηρετητής αιτήσεων που υλοποιήσαμε λαμβάνει υπόψιν του την περίπτωση αυτή. Έτσι, αν ληφθούν δύο αιτήσεις την ίδια χρονική στιγμή για το ίδιο αρχείο (πχ. το http://www.some.net/file.html) τότε στέλνεται μόνο μια αίτηση ενώ η δεύτερη θα εξυπηρετηθεί από την cache. Για την πρόληψη τέτοιων περιπτώσεων κάθε φορά που ξεκινά η εξυπηρέτηση μιας αίτησης, ελέγχεται αν υπάρχει όμοια αίτηση η οποία ήδη εξυπηρετείται.



Σχήμα 4.2: Κλήσεις συναρτήσεων του εξυπηρετητή αιτήσεων.

4.1.4 Αντιστοίχιση URL σε όνομα αρχείου

Για να αποθηκευτεί κάποιο αρχείο στην cache, πρέπει να γίνει κάποια αντιστοίχιση του URL που το περιγράφει (οι HTTP αιτήσεις περιέχουν URL και όχι ένα απλό όνομα αρχείου) σε κάποιο όνομα με το οποίο θα αποθηκευτεί στον δίσκο της cache. Το όνομα αυτό προέρχεται από το URL και αποτελείται από τρία μέρη:

- Το πρώτο μέρος είναι το πρωτόκολλο, δηλαδή το http.
- Το δεύτερο μέρος είναι ο WWW server στον οποίον βρίσκεται το αρχείο.
- Το τρίτο μέρος είναι το όνομα του αρχείου στον WWW server.

4.1.5 Αντιμετώπιση περιπτώσεων βλάβης του δικτύου

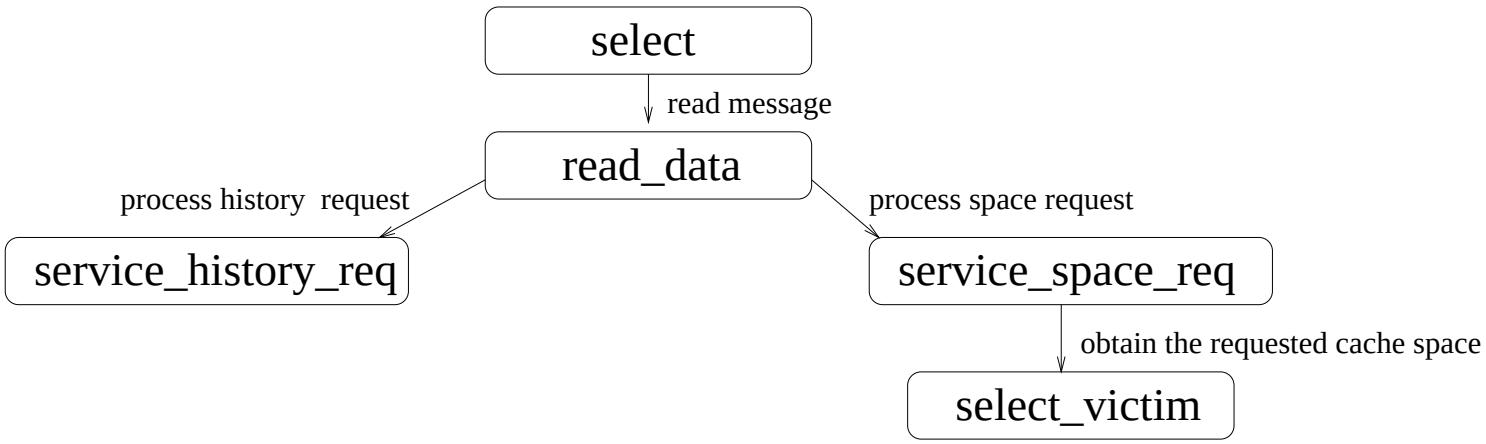
Μια άλλη πολύ σημαντική περίπτωση η οποία λήφθηκε υπόψιν κατά την σχεδίαση του εξυπηρετητή αιτήσεων, είναι η αντιμετώπιση περιπτώσεων όπου το δίκτυο έχει βλάβη. Μπορούμε να διακρίνουμε δυο περιπτώσεις:

- Χάνεται η επικοινωνία με τον χρήστη. Το ερώτημα που προκύπτει στην περίπτωση αυτή είναι αν θα λάβουμε όλη την απάντηση από τον WWW server για την αίτηση αυτή καθώς και αν θα αποθηκευτεί στην cache. Προσφέρεται η δυνατότητα να ακολουθηθούν και οι δυο πιθανές λύσεις.
- Χάνεται η επικοινωνία με τον WWW server ή τον proxy server, όσο περιμένουμε την απάντηση τους. Στην περίπτωση αυτή, ενημερώνεται ο χρήστης με ένα μήνυμα λάθους.

4.2 Υλοποίηση του διαχειριστή της cache

Ο σκοπός του διαχειριστή της cache είναι η αποδοτικότερη λειτουργία της cache. Οι δύο κυριότερες λειτουργίες του είναι:

- Η **διαχείριση** της cache. Λαμβάνει και εξυπηρετεί αιτήσεις του εξυπηρετητή αιτήσεων για εύρεση ελεύθερου χώρου. Αν υπάρχει διαθέσιμος χώρος το έργο του γίνεται απλό. Διαφορετικά, διαγράφει αρχεία από την cache μέχρι να αποκτήσει τον απαιτούμενο ελεύθερο χώρο. Επίσης, ανά τακτά χρονικά διαστήματα λαμβάνει ένα κατάλογο των αιτήσεων που δέχθηκε ο εξυπηρετητής αιτήσεων για να μπορεί να αποφασίζει για την αξία του κάθε αρχείου στην cache.
- Η εύρεση των **πολυτιμότερων** αρχείων της cache και η αποστολή του καταλόγου τους στον συντονιστή δικτύου.



Σχήμα 4.3: Κλήσεις συναρτήσεων του διαχειριστή της cache.

4.2.1 Λεπτομέρειες υλοποίησης

Στην έναρξη της λειτουργίας του δημιουργεί δυο δίαυλους επικοινωνίας με τον εξυπηρετητή αιτήσεων, τον SPACE_SOCK για να λαμβάνει αιτήσεις για την δέσμευση χώρου στην cache και τον HISTORY_SOCK για να λαμβάνει τις αιτήσεις που δέχθηκε ο εξυπηρετητής αιτήσεων. Στην συνέχεια, βρίσκεται σε ατέρμονα βρόγχο εξετάζοντας αν υπάρχει κάποιο μήνυμα στους δίαυλους επικοινωνίας.

Αν το SPACE_SOCK είναι έτοιμο για ανάγνωση, καλείται η *read_data* και κατόπιν η *service_space_req* για να διαβάσουν και να επεξεργαστούν αντίστοιχα, την αίτηση. Αν δεν υπάρχει ο απαιτούμενος ελεύθερος χώρος στην cache, καλείται η συνάρτηση *select_victim* για να αποκτήσει τον χώρο αυτό διαγράφοντας κάποια αρχεία από την cache.

Αν το HISTORY_SOCK είναι έτοιμο για ανάγνωση, καλείται η *read_data* και κατόπιν η *service_history_req* για την επεξεργασία του καταλόγου που διαβάζεται.

Τέλος υπάρχει η συνάρτηση *connect_npc* για να στείλει τον κατάλογο των πολυτιμότερων αρχείων στον συντονιστή δικτύου.

4.3 Υλοποίηση του συντονιστή δικτύου

Σκοπός του συντονιστή δικτύου είναι η διευκόλυνση της συνεργασίας των proxy servers. Στην έναρξη της λειτουργίας του δημιουργεί έναν δίαυλο επικοινωνίας όπου εκεί λαμβάνει τα μηνύματα των άλλων συντονιστών δικτύου ή του proxy server. Στην συνέχεια βρίσκεται σε ατέρμονα βρόγχο εξετάζοντας για την ύπαρξη μηνυμάτων. Ειδικά για τα μηνύματα, υπάρχουν τρεις κατηγορίες συναρτήσεων:

- Αυτές που καλεί για να στείλει αιτήσεις.
- Αυτές που καλεί για να λάβει απάντηση στις αιτήσεις που στέλνει.

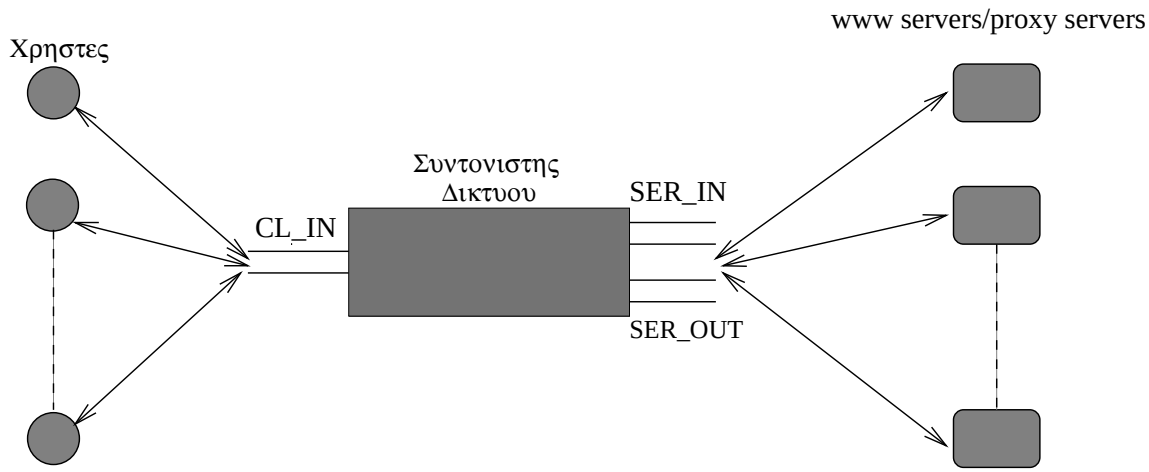
- Αυτές που καλεί για την εξυπηρέτηση αιτήσεων.

Μήνυμα	Αποστολή Μηνύματος	Λήψη απάντησης	Εξυπηρέτηση μηνύματος
ALIVE	send_alive_req_msg	serv_alive_reply	serv_alive_req
ADDME	send_coop_req_msg	serv_coop_reply	serv_coop_req
LISTCACHES	send_listcaches_req_msg	serv_listcaches_reply	serv_listcaches_req
LISTREMDOCS	send_listremdocs_req_msg	-	serv_listremdocs_req_msg
STOP	send_stop_req_msg	-	serv_stop_req
EXIT_COOP	send_exit_coop_req_msg	-	serv_exit_coop_req
PING	send_pind_req_msg	serv_ping_reply	serv_ping_req
URL	-	-	serv_ask_url_req
LISTLOCALDOCS	-	-	serv_listlocaldocs_req
LISTDOCS	send_listdocs_req_msg	-	-
DOWN	send_down_req_msg	-	-
STOP_NPC	-	-	serv_stop_req

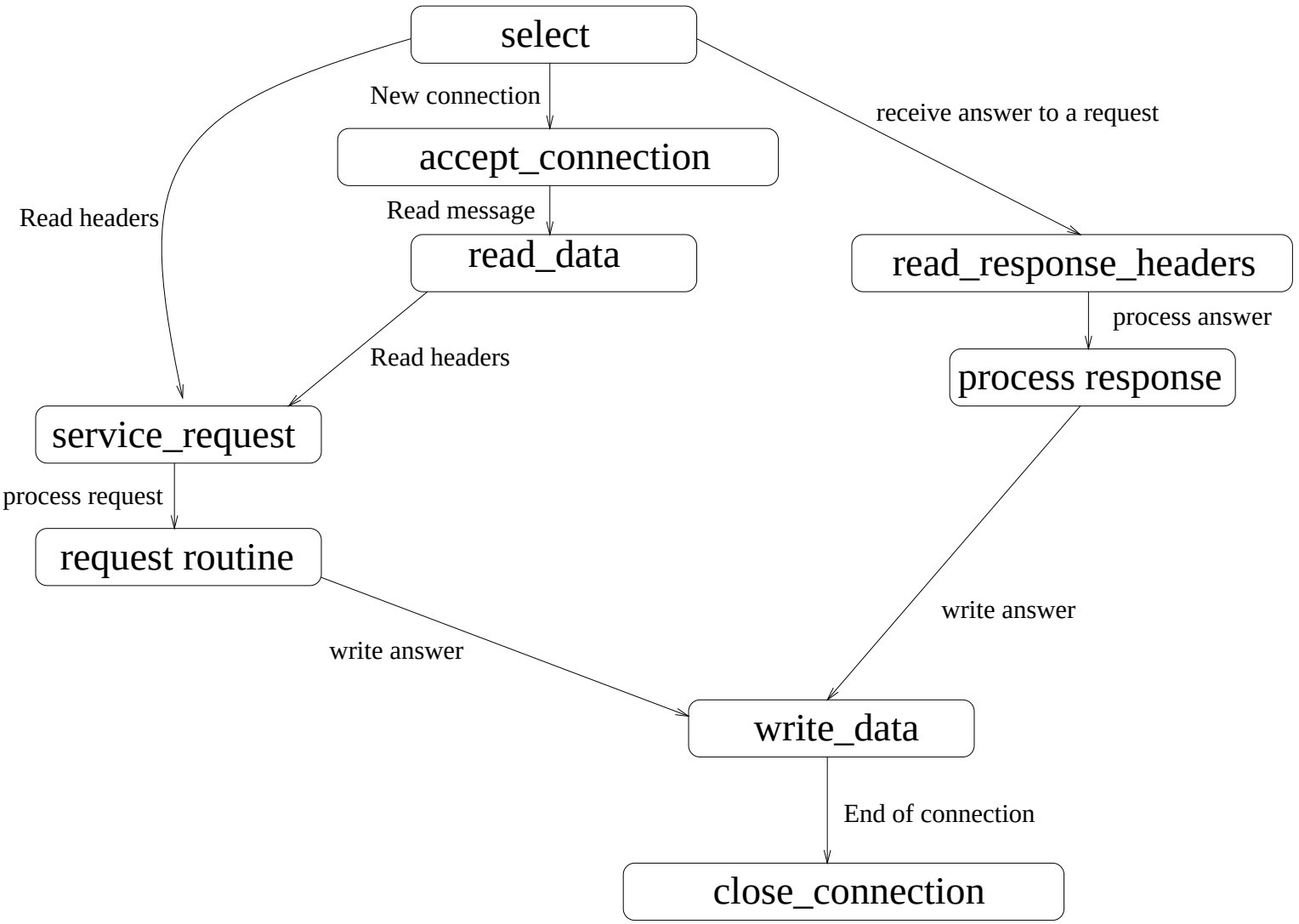
Πίνακας 4.1: Λίστα συναρτήσεων για την διαχείριση των μηνυμάτων του συντονιστή δικτύου.

Κάθε φορά που λαμβάνεται κάποια καινούρια αίτηση, καλείται η συνάρτηση *accept_connection* η οποία αρχικοποιεί τις δομές της σύνδεσης. Κατόπιν καλείται η *read_data* και στην συνέχεια η *service_request* η οποία διαβάζει την επικεφαλίδα της αίτησης (την πρώτη της γραμμή) και καλεί την αντίστοιχη συνάρτηση. Αν η αποστολή κάποιας αίτησης απαιτεί την λήψη κάποιας απάντησης, τότε καλείται η συνάρτηση *read_response_headers* όταν είναι έτοιμη η απάντηση αυτή. Το έργο της είναι να διαβάσει την επικεφαλίδα της απάντησης και να καλέσει την αντίστοιχη συνάρτηση η οποία θα την επεξεργαστεί. Υπάρχουν τρεις δομές (σχήμα 4.4):

- Η *CL_IN* όπου εκεί αποθηκεύονται οι αιτήσεις που λαμβάνονται.
- Η *SER_OUT* η οποία χρησιμοποιείται για την αποστολή αιτήσεων.
- Η *SER_IN* η οποία χρησιμοποιείται για την λήψη απάντησης σε αίτηση.



Σχήμα 4.4: Δομές του συντονιστή δικτύου.



Σχήμα 4.5: Κλήσεις συναρτήσεων του συντονιστή δικτύου

Κεφάλαιο 5

Μελέτη της κίνησης του WWW

Για την σχεδίαση αλγορίθμων οι οποίοι κάνουν αποδοτικότερη διαχείριση της cache, πρέπει να μελετηθούν τα ιδιαίτερα χαρακτηριστικά της κίνησης του WWW. Έχοντας κάποια πειραματικά αποτελέσματα, στην συνέχεια προχωράμε στην αποτίμηση του, και στην σχεδίαση των αλγορίθμων για την διαχείριση της χωρητικότητας της cache.

Αρκετό ενδιαφέρον παρουσιάζει η μελέτη των ενδιαφερόντων των χρηστών στο WWW. Δηλαδή, ποιες είναι οι προτιμήσεις τους. Ποιους τύπους αρχείων ζητάνε συχνότερα; Είναι προφανές ότι ένας αποδοτικός αλγόριθμος διαχείρισης της cache θα πρέπει να λαμβάνει υπόψιν του και τις προτιμήσεις αυτές και να μην στηρίζεται απλά σε ιδέες από τα παραδοσιακά λειτουργικά συστήματα.

Για την μελέτη της κίνησης του WWW συλλέχθηκαν δεδομένα τα οποία απεικονίζουν τις ζητήσεις των χρηστών (πίνακας 5.1). Συλλέξαμε τα παρακάτω WWW traces:

- Traces από τον proxy server του Τμήματος Επιστήμης Υπολογιστών του Παν/μίου Κρήτης. Στην περίπτωση αυτή οι φοιτητές του Τμήματος (που χρησιμοποιούν αυτόν τον proxy server) αποτελούν το σύνολο των χρηστών που εξυπηρετεί ο proxy server και τα ενδιαφέροντα τους είναι ποικίλα.
- Traces από τον proxy server του Ινστιτούτου Πληροφορικής. Προφανώς τα ενδιαφέροντα των χρηστών είναι πιο συγκεκριμένα αφού αποτελούν μέλη μιας ερευνητικής κοινότητας με κοινό επιστημονικό υπόβαθρο.
- Traces από τον proxy server της εταιρείας Forthnet. Η ευρεία διάδοση των υπηρεσιών του Forthnet στην Ελλάδα διαμορφώνει ένα πολυμορφικό σύνολο χρηστών.
- Traces από τον proxy server του πολυτεχνείου Αθηνών (NTUA).
- Traces από τον proxy server του Ελληνικού Δικτύυ (GRNET).

TRACE	Συν. Αιτήσεις	Επαναζητήσεις (Hits)	KBytes	Latency (min)	Διάρκεια
CSD	185764	77401	2053179	21412	4 μήνες
FORTH	44430	15854	475656	21931	2 μήνες
FORTHNET	203936	93269	2715731	84128	1 εβδομάδα
GRNET	120000	11044	1894583	13256	<εβδομάδα
NTUA	120000	43766	1471099	15611	< εβδομάδα

Πίνακας 5.1: Χαρακτηριστικά των WWW Traces

5.1 Χαρακτηριστικά του WWW

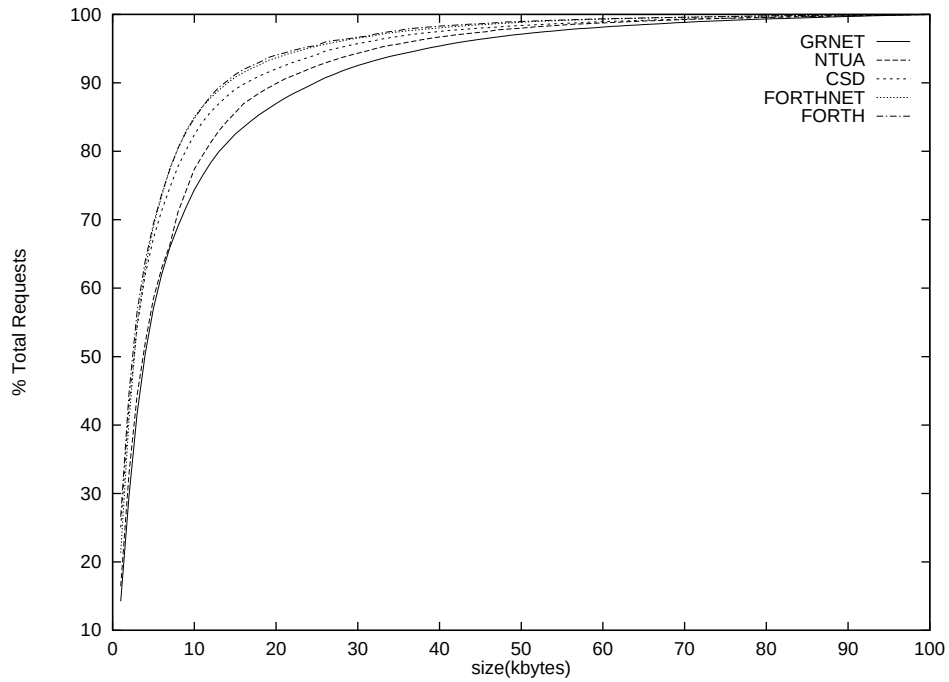
Στην παράγραφο αυτή εξετάζουμε τα κυριότερα χαρακτηριστικά του WWW. Δίνεται ιδιαίτερη έμφαση στην εύρεση κάποιας σχέσης η οποία συνδέει τις προτιμήσεις στο WWW με το μέγεθος (SIZE) και τον χρόνο απόκτησης (LATENCY) των αρχείων που διακινούνται σ' αυτό. Αυτό γίνεται για δύο κυρίως λόγους:

- Η γνώση της σχέσης αυτής για το μέγεθος των αρχείων που ζητάνε οι χρήστες συμβάλλει στην αποδοτικότερη σχεδίαση αλγορίθμων διαχείρισης της cache γιατί η χωρητικότητα της είναι το αγαθό που διαχειρίζεται. Έτσι, αν για παράδειγμα βρεθεί ότι τα αρχεία με μεγάλος μέγεθος συγκεντρώνουν ένα μικρό ποσοστό των προτιμήσεων, τότε, η αξία τους θα είναι περιορισμένη.
- Η βελτίωση του μέσου χρόνου απόκρισης που αντιμετωπίζουν οι χρήστες στην εξυπηρέτηση των αιτήσεων τους καθορίζει την επιτυχία της διαχείρισης της cache. Αν γνωρίζουμε, πχ ότι ζητούνται περισσότερο τα αρχεία με μεγάλο LATENCY, τότε θα δοθεί μεγαλύτερη βαρύτητα στα αρχεία αυτά.

Ορίζουμε τις παρακάτω έννοιες για την μελέτη των προαναφερθέντων σχέσεων:

- **Latency Saved Ratio (LSR)**: το ποσοστό της βελτίωσης του μέσου χρόνου απόκρισης στις αιτήσεις των χρηστών.
- **Size Saved Ratio (SSR)**: το ποσοστό της πληροφορίας που ζητάνε οι χρήστες και εξυπηρετούνται από την cache.
- **Hit Ratio (HR)**: το ποσοστό των αιτήσεων που εξυπηρετούνται από την cache.

Πολλές ιδιότητες στο caching δίσκου σε μνήμη δεν ισχύουν πια εδώ, λόγω των διαφορών που παρουσιάζει με το caching στο WWW. Μια σημαντική παρατήρηση είναι ότι μόνο το HR δεν μπορεί να περιγράψει την επιτυχία της διαχείρισης της cache αλλά πρέπει να ληφθεί ιδιαίτερα υπόψιν και η βελτίωση του μέσου χρόνου εξυπηρέτησης των αιτήσεων των χρηστών δηλαδή το LSR. Στο WWW, τα τρία παραπάνω μετρικά δεν ταυτίζονται σε αντίθεση



Σχήμα 5.1: Γραφική παράσταση όπου απεικονίζεται η σχέση του μέγεθους των αρχείων με την ζήτηση τους. Τα μικρά σε μέγεθος αρχεία παρουσιάζουν μεγαλύτερη ζήτηση

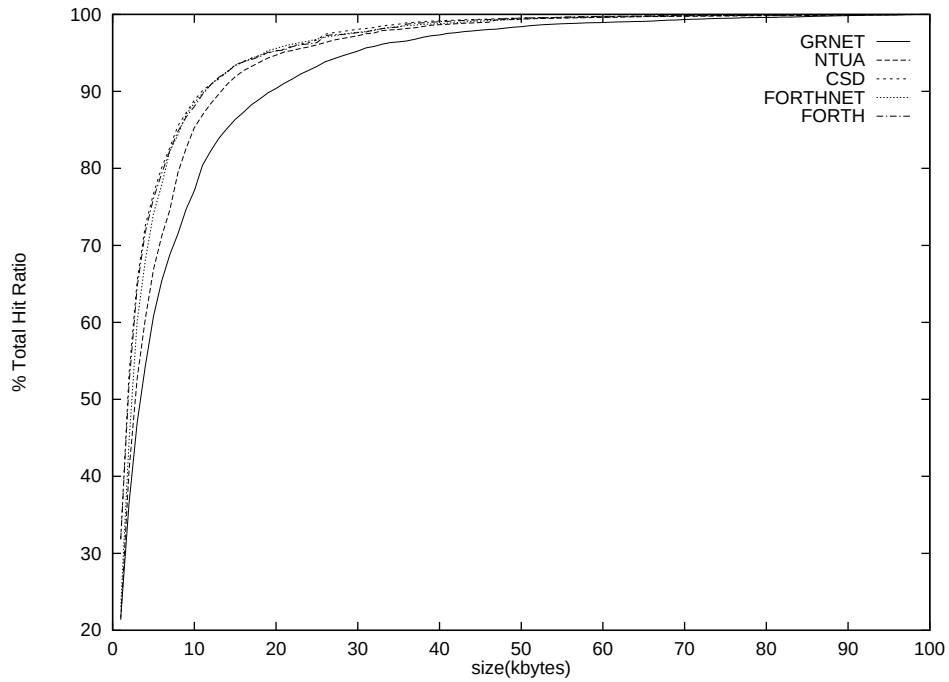
με το caching δίσκου στη μνήμη όπου συντρέχουν. Για την μελέτη καθενός από των δυο χαρακτηριστικών (SIZE και Latency), θα δίνεται η αντίστοιχη γραφική παράσταση του με τα προηγούμενα μεγέθη. Για τον σκοπό αυτό, θα θεωρούμε ότι έχουμε μια cache απείρου μεγέθους και έτσι δεν θα γίνονται ποτέ διαγραφές αρχείων από αυτήν.

5.1.1 Μελέτη του size

Ένα από τα κυριότερα χαρακτηριστικά του WWW είναι ότι οι προτιμήσεις των χρηστών στρέφονται περισσότερο προς αρχεία τα οποία είναι μικρά σε μέγεθος. Παρατηρείται ότι όσο μεγαλώνει το μέγεθος των αρχείων τόσο ελαττώνονται οι αιτήσεις για αυτά. Στο σχήμα 5.1 φαίνεται καθαρά η τάση αυτή προς τα μικρά αρχεία. Ένα άλλο ενδιαφέρον στοιχείο είναι η σχέση του HR με το μέγεθος των αρχείων. Στο σχήμα 5.2 φαίνεται ότι υπάρχει μια αντιστρόφως ανάλογη σχέση. Όσο αυξάνει το μέγεθος των αρχείων τόσο μειώνεται το HR. Τα αποτελέσματα αυτά συμφωνούν με αυτά του [CBC95] όπου αναφέρεται ότι η προτίμηση αυτή για τα μικρά σε μέγεθος αρχεία είναι πιο ισχυρή και από ότι στο UNIX. Οι μετρήσεις γίνανε έχοντας μια cache άπειρης χωρητικότητας, οπότε δεν γίνονται αντικαταστάσεις αρχείων.

5.1.2 Μελέτη του Latency

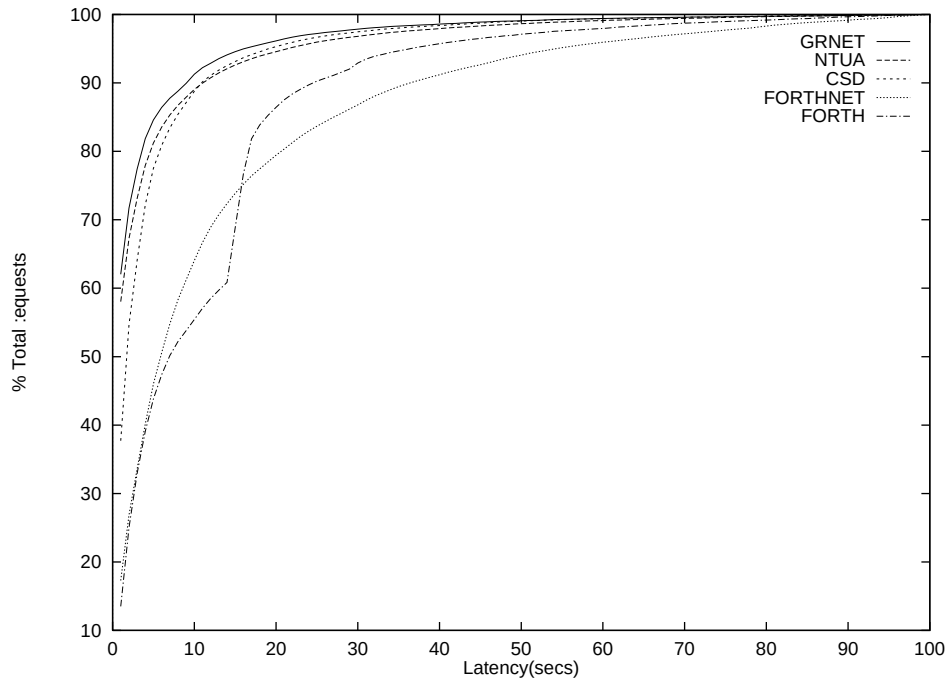
Ένα κύριο χαρακτηριστικό για την σχεδίαση αποδοτικών αλγορίθμων είναι ο χρόνος που απαιτείται για να αποκτηθεί κάποιο αρχείο από τον WWW server. Παρατηρούμε ότι τα αρχεία με μικρό χρόνο απόκτησης έχουν περισσότερες ζητήσεις (σχήμα 5.3). Η μεγάλη



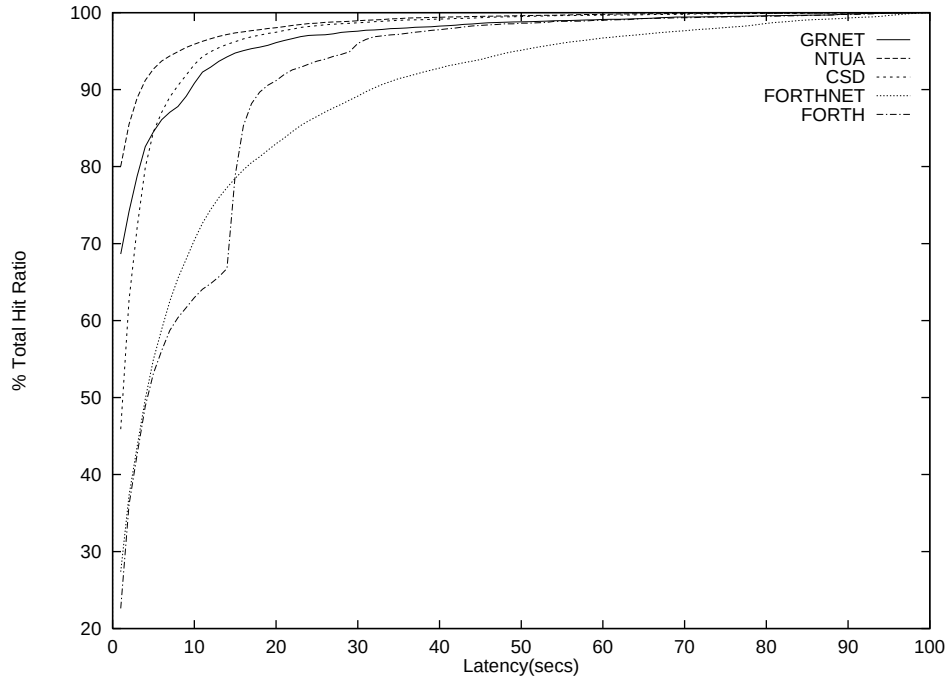
Σχήμα 5.2: Γραφική παράσταση όπου απεικονίζεται η σχέση του μέγεθους των αρχείων με το HR. Η αποθήκευση μικρών σε μέγεθος αρχείων βελτιώνει το Hit Ratio

ζήτηση των αρχείων με μικρό χρόνο απόκτησης φανερώνει ότι οι χρήστες στρέφονται προς αρχεία για τα οποία δεν πρέπει να περιμένουν αρκετή ώρα μέχρι να τα κατεβάσουν από τον WWW server. Επίσης η αποθήκευση αρχείων με μικρό χρόνο απόκτησης είναι αποδοτικότερη για την βελτίωση και του HR. (σχήμα 5.4). Οι μετρήσεις γίνανε έχοντας μια cache άπειρης χωρητικότητας, οπότε δεν γίνονται αντικαταστάσεις αρχείων.

Στο παράρτημα Β παραθέτονται κάποια επιπλέον στοιχεία από τις μετρήσεις.



Σχήμα 5.3: Γραφική παράσταση όπου απεικονίζεται η σχέση του Latency των αρχείων με την ζήτηση τους. Το μεγαλύτερο ποσοστό των αιτήσεων των χρηστών στρέφεται σε αρχεία με μικρό χρόνο απόκτησης



Σχήμα 5.4: Γραφική παράσταση όπου απεικονίζεται η σχέση του Latency των αρχείων με το HR. Η αποθήκευση αρχείων με μικρό χρόνο απόκτησης σταδιακά βελτιώνει το Hit Ratio

Κεφάλαιο 6

Αλγόριθμοι διαχείρισης της cache

6.1 Εισαγωγή

Στο κεφάλαιο αυτό παρουσιάζουμε διάφορους αλγορίθμους διαχείρισης της cache. Αρχικά, περιγράφονται μερικοί γνωστοί αλγόριθμοι οι οποίοι χρησιμοποποιήθηκαν ευρύτατα σε προγενέστερα συστήματα (όπως στα λειτουργικά συστήματα). Ακολούθως, παρουσιάζονται αλγόριθμοι από την βιβλιογραφία οι οποίοι σχεδιάστηκαν ειδικά για την διαχείριση της cache στο WWW. Κατόπιν, δίνεται η περιγραφή των προτεινόμενων αλγορίθμων. Στο τέλος γίνεται η συγκριτική παρουσίαση τους.

Παρόλο που συνεχώς μπορούμε να έχουμε μεγαλύτερη cache, κάτι τέτοιο δεν είναι εφικτό διότι η αύξηση της χρήσης του WWW είναι τεράστια. Ετσι, οι αλγόριθμοι διαχείρισης της cache είναι ιδιαίτερα σημαντικοί.

6.2 Καθιερωμένοι αλγόριθμοι από τα λειτουργικά συστήματα

6.2.1 Αλγόριθμοι LRU, LFU

Στην παράγραφο αυτή περιγράφονται κάποιοι κλασικοί αλγόριθμοι για caching (πχ. από τα λειτουργικά συστήματα). Ένας καλός ευριστικός αλγόριθμος βασίζεται στην παρατήρηση ότι τα αρχεία τα οποία ζητούνται αρκετές φορές στις τελευταίες αιτήσεις, θα ζητηθούν και στο άμεσο μέλλον. Όμοια, αρχεία τα οποία δεν έχουν ζητηθεί για μεγάλο χρονικό διάστημα κατά πάσα πιθανότητα δεν θα ζητηθούν στο άμεσο μέλλον. Ο Least Recently Used (LRU) βασίζεται στις παρατηρήσεις αυτές: κάθε φορά που απαιτείται απόκτηση ελεύθερου χώρου στην cache, απομακρύνεται το αρχείο το οποίο δεν έχει ζητηθεί για το μεγαλύτερο χρονικό διάστημα.

Σε παρόμοια ιδέα βασίζεται και ο Least Frequenlty Used (LFU). Κάποιο αρχείο το οποίο ζητήθηκε αρκετές φορές στο παρελθόν, θα ζητηθεί ακόμη περισσότερες στο μέλλον. Στην περίπτωση του αλγορίθμου αυτού, το κριτήριο στο οποίο βασίζεται η διαμόρφωση της

απόφασης για το ποια αρχεία θα απομακρυνθούν από την cache, (όταν απαιτείται ελεύθερος χώρος) είναι το πόσες φορές έχει ζητηθεί ένα αρχείο και όχι το πόσο πρόσφατα ζητήθηκε. Υποψήφιο για αντικατάσταση είναι το αρχείο με την λιγότερη ζήτηση.

6.3 Αλγόριθμοι από το WWW

Στην παράγραφο αυτή περιγράφονται κάποιοι αλγόριθμοι οι οποίοι είτε σχεδιάστηκαν αποκλειστικά για το WWW είτε βρήκαν μεγάλη εφαρμογή σε αυτό. Η βελτίωση του μέσου χρόνου απόκρισης στην εξυπηρέτηση των αιτήσεων και του ποσοστού των αιτήσεων που εξυπηρετούνται από την cache είναι ο κύριος σκοπός τους.

6.3.1 Αλγόριθμοι LRU-THOLD, LRU-MIN

Ο αλγόριθμος LRU-THOLD είναι μια τροποποίηση του LRU χρησιμοποιώντας το μέγεθος των αρχείων ως κριτήριο για το αν θα αποθηκευτούν στην cache [Mar96, ASA+95]. Πιο συγκεκριμένα, δεν αποθηκεύται στην cache κανένα αρχείο που το μέγεθος του ξεπερνά κάποιο όριο (ακόμα και αν υπάρχει ελεύθερος χώρος στην cache). Έτσι, αντιμετωπίζεται η κατάσταση όπου η αποθήκευση ενός μεγάλου αρχείου προκαλεί την διαγραφή πολλών μικρών αρχείων από την cache. Εκμεταλλεύεται το γεγονός ότι το μεγαλύτερο ποσοστό της κίνησης στο WWW συγκεντρώνονται σε μικρά στο μέγεθος αρχεία. Ο υπολογισμός του άνω ορίου είναι συνάρτηση των χαρακτηριστικών της κίνησης (δηλαδή των χαρακτηριστικών των ζητούμενων αρχείων) και της χωρητικότητας της cache.

Ο αλγόριθμος LRU-MIN αποτελεί και αυτός μια τροποποίηση του LRU ο οποίος προσπαθεί να ελαχιστοποιήσει τον αριθμό των αρχείων που διαγράφονται από την cache [ASA+95]. Η περιγραφή του αλγορίθμου είναι η εξής: κάθε φορά που απαιτείται απόκτηση ελεύθερου χώρου στην cache δημιουργείται μια λίστα όλων των αρχείων που έχουν μέγεθος μεγαλύτερο ή ίσο του μεγέθους που απαιτείται να αποκτηθεί. Κατόπιν, διαγράφονται τα αρχεία από την λίστα αυτή σύμφωνα με τον χρόνο της τελευταίας ζήτησης τους (LRU). Αν δεν αποκτηθεί ο απαιτούμενος χώρος, δημιουργείται μια λίστα με όλα τα αρχεία που έχουν μέγεθος το μισό του απαιτούμενου μεγέθους και συνεχίζεται η ίδια διαδικασία.

6.3.2 Αλγόριθμος LNC-R-W3

Ο αλγόριθμος αυτός είναι μια μεταγενέστερη έκδοση του αλγορίθμου που σχεδιάστηκε για το caching αποτελεσμάτων ερωτήσεων σε ένα data warehouse [SsV96a]. Η εφαρμογή του αλγορίθμου αυτού στο WWW είναι το LNC-R-W3 [SSV96b].

Ο αλγόριθμος διατηρεί τα παρακάτω στοιχεία για το κάθε αρχείο F_i που εισέρχεται στην cache:

- R_i: ο μέσος ρυθμός αιτήσεων για το αρχείο F_i.

- S_i: το μέγεθος του αρχείου F_i.
- L_i: ο χρόνος που απαιτείται για να αποκτηθεί το αρχείο F_i.

Τα παραπάνω στατιστικά χρησιμοποιούνται για την εύρεση των αρχείων των οποίων η αποθήκευση τους στην cache αναμένεται να είναι αποδοτική. Σε κάθε αρχείο επισυνάπτεται κάποιο μέγεθος το οποίο περιγράφει και την πολυτιμότητα του σύμφωνα με τον παρακάτω τύπο:

$$V(F_i) = \frac{R_i * L_i}{S_i}$$

Τα αρχεία ταξινομούνται σύμφωνα με το παραπάνω μέγεθος και υποψήφιο για να διαγραφεί από την cache είναι εκείνο που το caching του είναι λιγότερο αποδοτικό. Ο υπολογισμός του ρυθμού ζητήσεων για κάποιο αρχείο γίνεται κρατώντας τις χρονικές στιγμές των K τελευταίων ζητήσεων του και δίνεται από τον τύπο:

$$R_i = \frac{K}{(t - t_k) * S_i^b}$$

όπου t η παρούσα χρονική στιγμή, t_k η χρονική στιγμή που έγινε η K-στη ζήτηση του αρχείου στο παρελθόν και $b = 1.6$. Αν, για κάποιο αρχείο υπάρχουν λιγότερες από K ζητήσεις, χρησιμοποιούνται οι ζητήσεις αυτές. Ο αλγόριθμος διαγράφει πρώτα τα αρχεία με μια ζήτηση, κατόπιν τα αρχεία με δυο ζητήσεις, κοκ. σύμφωνα πάντα με την σειρά πολυτιμότητας τους μέχρι να αποκτηθεί ο απαραίτητος ελεύθερος χώρος στην cache.

6.3.3 Αλγόριθμος PER

Ο αλγόριθμος αυτός [BH96] λαμβάνει υπόψιν του τα παρακάτω στοιχεία για κάθε αρχείο F_i:

- T_i: η χρονική στιγμή της τελευταίας ζήτησης για το αρχείο.
- S_i: το μέγεθος του αρχείου.
- L_i: ο χρόνος απόκτησης του αρχείου.

Σε κάθε αρχείο επισυνάπτεται κάποιο μέγεθος που υποδηλώνει την πολυτιμότητα του σύμφωνα με τον τύπο:

$$V(T_i, L_i, S_i) = W_1 * L_i + W_2 * S_i + \frac{W_3 + W_4 * S_i}{T_i}$$

όπου :

$$W_1 = 5000 Bytes/Sec, W_2 = 1000, W_3 = 10000 BytesSec, W_4 = 10 Sec$$

Δηλαδή, προσπαθεί να δώσει κάποια πολυτιμότητα σε κάθε αποθηκευμένο αρχείο η οποία διαμορφώνεται από τον συνδυασμό των χαρακτηριστικών του. Παρατηρείται ότι αρχεία με μεγάλο χρόνο απόκτησης έχουν μεγαλύτερη πολυτιμότητα.

6.4 Προτεινόμενοι αλγόριθμοι

6.4.1 Προτεινόμενος αλγόριθμος (LAT)

Το κριτήριο που προσπαθεί να βελτιστοποιήσει ο αλγόριθμος LAT είναι η βελτίωση του μέσου χρόνου απόκρισης που αντιμετωπίζουν οι χρήστες στην εξυπηρέτηση των αιτήσεων τους. Είναι σχεδιασμένος για κόμβους όπου το Bandwidth/Latency είναι σημαντικός παράγοντας απόδοσης (δηλαδή για αργό δίκτυο, γρήγορη CPU). Βασίζεται σε ιδέες οι οποίες εξήχθησαν από την στατιστική μελέτη του WWW και κάποιες ιδέες από τα λειτουργικά συστήματα:

- Αρχεία με μικρό μέγεθος παρουσιάζουν μεγαλύτερη ζήτηση. Τα αρχεία αυτά δεν ζητούνται απλώς μια φορά αλλά επαναζητούνται και άλλες φορές (όπως φαίνεται και στο σχήμα 5.2). Η αποθήκευση των αρχείων αυτών εκτός του ότι βελτιώνει το HR, βελτιώνει και το LSR εξαιτίας της αυξημένης ζήτησης τους.
- Αρχεία με μικρό χρόνο απόκτησης (Latency) παρουσιάζουν μεγαλύτερη ζήτηση. Προφανώς, οι χρήστες του WWW ενοχλούνται κάθε φορά που περιμένουν αρκετή ώρα για να προσπελάσουν κάποιο αρχείο και έτσι δεν το προτιμούν.
- Αρχεία που παρουσιάζουν αρκετή ζήτηση πιθανόν θα ζητηθούν και στο άμεσο μέλλον (ιδέα LFU). Η σημαντικότατη αυτή παρατήρηση δεν μπορεί να λείψει από την σχεδίαση ενός αποδοτικού αλγορίθμου διαχείρισης της χωρητικότητας της cache.
- Αρχεία τα οποία ζητήθηκαν πιο πρόσφατα, θα ζητηθούν πιθανότατα στο άμεσο μέλλον (ιδέα LRU).

Έστω F το σύνολο των αρχείων της cache. Σε κάθε αρχείο f_i αντιστοιχεί κάποια τιμή V_i η οποία υποδηλώνει την πολυτιμότητα του. Τα αρχεία ταξινομούνται σύμφωνα με την πολυτιμότητα τους και το υποψήφιο για την επόμενη αντικατάσταση αρχείο είναι το λιγότερο πολύτιμο.

Για κάθε αρχείο f_i, ορίζουμε τα εξής:

- S_i: το μέγεθος του αρχείου (σε bytes).
- T_i: η χρονική στιγμή στην οποία έγινε η τελευταία ζήτηση για το αρχείο.

- U_i: πόσες φορές ζητήθηκε το αρχείο στο παρελθόν όσο είναι αποθηκευμένο στην cache.
- L_i: ο χρόνος που απαιτείται για την μεταφορά του αρχείου από τον WWW server ή κάποιον άλλον συνεργαζόμενο proxy server στην cache.

Ο προτεινόμενος αλγόριθμος θα λαμβάνει υπόψιν του τα στοιχεία αυτά. Επιπλέον, στηρίζεται και σε μια καινούρια ιδέα η οποία προσπαθεί να προσομοιώσει την "γήρανση" των αρχείων που βρίσκονται στην cache. Σύμφωνα με την ιδέα αυτή, κάθε χρονική στιγμή η κατάσταση της cache περιγράφεται από έναν αριθμό που ονομάζεται **βήμα**. Αρχικά, η cache βρίσκεται στο πρώτο βήμα. Ο αριθμός αυτός αυξάνει κάθε φορά που διαγράφονται από την cache αρχεία των οποίων το συνολικό άθροισμα των μεγεθών τους ισούται με το μέγεθος της cache. Σε κάθε αρχείο ανατίθεται κάποιο βήμα P_i που είναι το βήμα που βρίσκεται η cache την χρονική στιγμή που εισάγεται σε αυτήν. Γίνεται, επιπλέον, διάκριση μεταξύ των αρχείων που βρίσκονται στο ίδιο βήμα.

Κατά την διάρκεια των πειραμάτων αυτών καταλήξαμε στα παρακάτω συμπεράσματα:

- Η χρησιμοποίηση μόνο του χρόνου απόκτησης ενός αρχείου (Latency) για την διαμόρφωση της απόφασης αντικατάστασης αρχείων δεν είναι αποδοτική.
- Στο WWW, παρουσιάζεται σαν ιδιαίτερο χαρακτηριστικό γνώρισμα η επαναζήτηση αρχείων. Για τον λόγο αυτό υψώνεται στο τετράγωνο ο αριθμός των ζητήσεων του αρχείου. Με τον τρόπο αυτό γίνεται και κάποια επιπλέον διαφοροποίηση μεταξύ των αρχείων που ζητήθηκαν μια φορά ($U^2 = 1$) και των αρχείων που ζητήθηκαν περισσότερες φορές ($U^2 > 1$).
- Τέλος σημαντικό ρόλο παίζει και ο χρόνος της τελευταίας ζήτησης για κάποιο αρχείο που βρίσκεται αποθηκευμένο στην cache.

Ο αλγόριθμος ο οποίος στηρίζεται στις προηγούμενες ιδέες, προσάπτει πολυτιμότητα σε κάθε αρχείο που εισέρχεται στην cache, σύμφωνα με τον τύπο:

$$\mathbf{V_i(L_i, U_i, T_i, S_i, P_i)} = \frac{\mathbf{a} * \mathbf{T_i^b} * \mathbf{U_i^2} * \mathbf{P_i} * \mathbf{L_i}}{\mathbf{S_i^c}}$$

όπου a, b, c μεταβλητές του προβλήματος. Μετά από πειράματα δοκιμάζοντας διάφορους συνδυασμούς τιμών τους καταλήξαμε στις τιμές $a = 1.5, b = 1.4, c = 1.8$.

Η τελική διαμόρφωση του τύπου ήτανε μια αρκετά δύσκολη υπόθεση και προέκυψε από την μελέτη αρκετών εναλλακτικών αλγορίθμων (πίνακας 6.1). Τα αποτελέσματα των αλγορίθμων αυτών φαίνονται στο σχήμα 6.1.

6.4.2 Προτεινόμενος αλγόριθμος HIT

Το κριτήριο που προσπαθεί να βελτιστοποιήσει ο αλγόριθμος HIT είναι η αύξηση του ποσοστού των αιτήσεων που εξυπηρετούνται από την cache. Για τον λόγο αυτό λαμβάνει

Αλγ. 1	$\frac{4*U^3*P*L}{S}$
Αλγ. 2	$\frac{2*U^3*P*L}{S}$
Αλγ. 3	$\frac{U^2*P*L}{S}$
Αλγ. 4	$\frac{U*P*L}{S}$
Αλγ. 5	$\frac{P*L}{S}$
Αλγ. 6	$\frac{L}{S}$
Αλγ. 7	$\frac{U^3*L}{S}$
Αλγ. 8	$\frac{U^2*L}{S}$
Αλγ. 9	$\frac{U*L}{S}$
Αλγ. 10	$U * L$
Αλγ. 11	$\frac{4*T*U^3*L}{S}$
Αλγ. 12	$\frac{T*U^3*L}{S}$
Αλγ. 13	$\frac{T*U^2*L}{S}$
Αλγ. 14	$\frac{T*U*L}{S}$
Αλγ. 15	$\frac{4*T*U^3*L*P}{S}$
Αλγ. 16	$\frac{T*U^2*P*L}{S}$
Αλγ. 17	$\frac{2*T*U^2*P*L}{S}$
Αλγ. 18	$\frac{1.5*T*U^2*P*L}{S}$
Αλγ. 19	$\frac{4*T^{1.4}*U^2*L*P}{S^{1.8}}$
Αλγ. 20	L

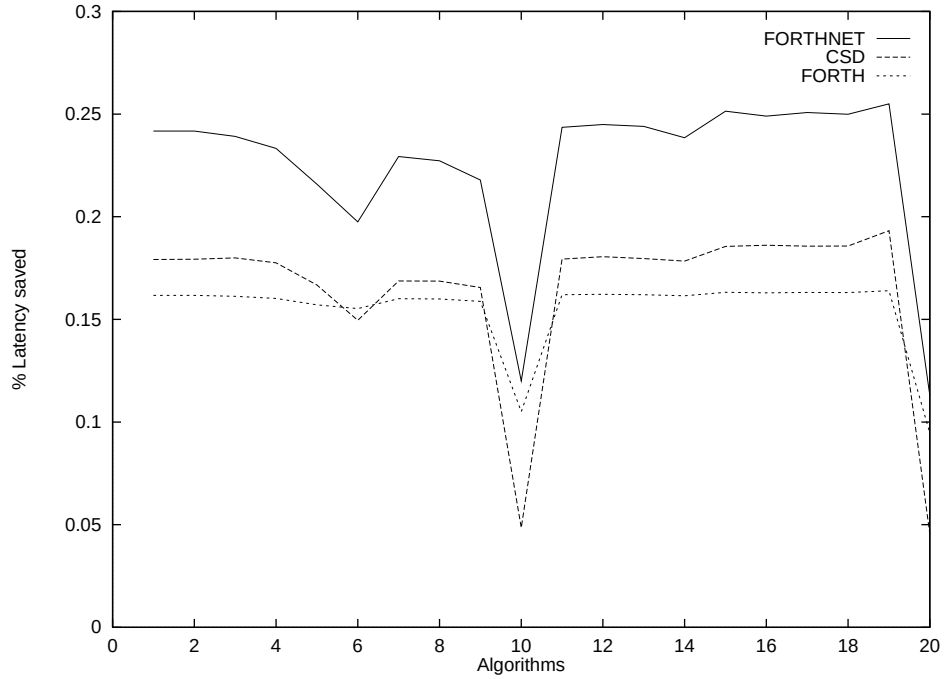
Πίνακας 6.1: Λίστα εναλλακτικών αλγορίθμων για την βελτίωση του μέσου χρόνου απόκρισης στην εξυπηρέτηση των αιτήσεων των χρηστών

υπόψιν του ιδιότητες που χαρακτηρίζουν αρχεία με πολλές επαναζητήσεις. Καταλήξαμε στο συμπέρασμα ότι τα κυριότερα χαρακτηριστικά είναι το μέγεθος του αρχείου (αρχεία με μικρό μέγεθος έχουν περισσότερες πιθανότητες να ζητηθούν και στο μέλλον), ο αριθμός των ζητήσεων για το κάθε αρχείο (LFU) καθώς και η χρονική στιγμή της τελευταίας ζήτησης (LRU).

Ο αλγόριθμος αυτός προσάπτει πολυτιμότητα σε κάθε αρχείο που εισέρχεται στην cache σύμφωνα με τον τύπο:

$$\mathbf{V_i}(\mathbf{U_i}, \mathbf{T_i}, \mathbf{S_i}) = -\frac{\mathbf{S_i}}{\mathbf{U_i} * \mathbf{T_i}}$$

Δηλαδή, προσάπτει αρνητική αξία σε κάθε αρχείο και το πολυτιμότερο αρχείο είναι πάλι αυτό με την μεγαλύτερη αξία.



Σχήμα 6.1: Γραφική παράσταση της απόδοσης των εναλλακτικών αλγορίθμων που δοκιμάστηκαν για την βελτίωση του μέσου χρόνου εξυπηρέτησης των αιτήσεων των χρηστών. Το μέγεθος της cache είναι 131072 Kbytes στα CSD και FORTHNET και 65536 στο FORTH

6.4.3 Επιπλέον αλγόριθμοι που δοκιμάστηκαν

Εκτός από τους παραπάνω προτεινόμενους αλγορίθμους διαχείρισης της cache, δοκιμάστηκαν και άλλοι αλγόριθμοι οι οποίοι δεν είχαν τα επιθυμητά αποτελέσματα. Μπορούμε να τους κατατάξουμε σε 2 κατηγορίες:

- Αυτούς που προσπαθούν να κάνουν κάποια πρόβλεψη της χρονικής στιγμής που θα ζητηθεί κάποιο αρχείο στο μέλλον [PR94]. Σύμφωνα με τους αλγορίθμους αυτούς, πολυτιμότερα είναι τα αρχεία που θα ζητηθούν πιο σύντομα. Για τον υπολογισμό των χρονικών αυτών στιγμών χρησιμοποιήθηκε είτε η μέθοδος των ελαχίστων τετραγώνων είτε η μέθοδος της πολυωνυμικής παρεμβολής λαμβάνοντας υπόψιν τις χρονικές στιγμές που ζητήθηκε κάποιο αρχείο στο παρελθόν. Τα αποτελέσματα δεν ήταν ενθαρρυντικά προφανώς γιατί ο υπολογισμός αυτός δεν ήτανε σωστός. Το συμπέρασμα είναι ότι δεν μπορούν να προβλεφθούν με ικανοποιητική ακρίβεια οι χρονικές στιγμές που θα ζητηθεί κάποιο αρχείο στο μέλλον γιατί αφενός μεν κάτι τέτοιο έγκειται στις ιδιαιτερότητες του αρχείου (πχ. η WWW σελίδα μιας εφημερίδας παρουσιάζει μεγάλη ζήτηση στην αρχή και στην συνέχεια πέφτει απότομα) όσο και στις προτιμήσεις των χρηστών.
- Αυτούς που κατατάσσουν τα αρχεία ανά κατηγορίες σύμφωνα με κάποια χαρακτηριστικά όπως πχ. ο τύπος των αρχείων (gif, ps, bin, txt...) ή το μέγεθος των αρχείων. Κινηθήκαμε προς αυτήν την κατεύθυνση διότι από τα πειράματα μας

συμπεράναμε ότι οι χρήστες προτιμούν ιδιαίτερα κάποιες κατηγορίες αρχείων (εικόνες, κείμενο) από κάποιες άλλες (εκτελέσιμα αρχεία ή postscript) Στο [Sed94] αναφέρεται η προτίμηση αυτή. Σίγουρα οι αλγόριθμοι της κατηγορίας αυτής επιδέχονται επιπλέον βελτίωση.

6.5 Σύγκριση των αλγορίθμων

Στην παράγραφο αυτή δίνονται κάποια πειραματικά αποτελέσματα τα οποία και αποδεικνύουν την καλύτερη απόδοση των προτεινόμενων αλγορίθμων. Για την δοκιμή των αλγορίθμων, ο προσομοιωτής εξομοίωσε την λειτουργία του proxy server για διάφορα μεγέθη της cache (σε δυνάμεις του 2 από 8 ώς και 512 MB) και για όλα τα προαναφερθέντα traces. Η προσομοίωση για μεγαλύτερου μεγέθους cache φέρνει την απόδοση των αλγορίθμων σε κοντινά επίπεδα ειδικά όταν η cache είναι τόσο μεγάλη που δεν γίνονται αντικαταστάσεις αρχείων (δηλαδή επαρκεί η χωρητικότητα της cache). Επιπλέον, η κίνηση αυξάνει με ρυθμούς που ποτέ δεν επιτρέπει να έχουμε πρακτικά άπειρη cache. Στον πίνακα 1.1 φαίνεται η τεράστια αύξηση της χρήσης του WWW. Δίνονται δυο ειδών γραφικές παραστάσεις: αυτές που παρουσιάζουν την βελτίωση που επιφέρει ο κάθε αλγόριθμος στον χρόνο εξυπηρέτησης των αιτήσεων καθώς και αυτές που παριστάνουν την αντίστοιχη βελτίωση του ποσοστού των αιτήσεων που εξυπηρετούνται από την cache.

Για τον υπολογισμό του βέλτιστου ορίου (THRESHOLD) στον αλγόριθμο LRU-THOLD κάναμε την προσομοίωση του για κάποια από τα διαθέσιμα traces και για διαφορετικές τιμές του THRESHOLD. Για αυτά τα traces αποκτήθηκαν δύο τιμές: αυτή που βελτιστοποιεί το HR και αυτή που βελτιστοποιεί το LSR (σχήματα 6.2, 6.3)

Στους πίνακες 6.2 και 6.3 γίνεται κάποια συγκριτική παρουσίαση των αλγορίθμων. Η σειρά με την οποία παραθέτονται οι αλγόριθμοι δεν είναι τυχαία. Στην αρχή υπάρχουν οι αλγόριθμοι που χρησιμοποιούν μόνο την ζήτηση των αρχείων και ακολουθούν αυτοί που χρησιμοποιούν επιπλέον το μέγεθος και κατόπιν το χρόνο απόκτησης των αρχείων. Στο τέλος κάθε πίνακα είναι ο προτεινόμενος αλγόριθμος.

Στον πίνακα 6.2 φαίνεται ότι ο LAT μπορεί να έχει καλύτερα αποτελέσματα συγκρινόμενος με τους άλλους αλγορίθμους από 1% - 18%. Ειδικά με τους αλγορίθμους LRU και LFU, υπερτερεί σταθερά σε ποσοστό 9-12%. Παρατηρείται ότι η απόδοση των αλγορίθμων βελτιώνεται όσο αυξάνονται τα κριτήρια που χρησιμοποιούν. Για παράδειγμα, αν ο αλγόριθμος LRU χρησιμοποιεί και το μέγεθος των αρχείων για την διαμόρφωση των αποφάσεων του (LRU-MIN, LRU-THOLD), τότε βελτιώνεται η απόδοση του. Οι αλγόριθμοι LRU-MIN και HIT φαίνεται να έχουν αρκετά καλή απόδοση.

Όμοια, στον πίνακα 6.3 παρατηρείται ότι η απόδοση του αλγορίθμου HIT είναι σε αρκετά υψηλά επίπεδα. Η βελτίωση που επιφέρει εν συγκρίσει με τους άλλους αλγορίθμους κυμαίνεται από 1-20% ενώ οι αλγόριθμοι LRU-MIN και LAT έχουν την αμέσως καλύτερη απόδοση.

Στις γραφικές παραστάσεις 6.4, 6.5, 6.6, 6.7, 6.8 φαίνεται η βελτίωση του μέσου χρόνου

Αλγόριθμος	CSD	FORTH	FORTHNET	GRNET	NTUA
LFU	14.70	11.80	19.21	5.71	7.15
LRU	14.12	11.68	18.21	5.56	6.75
LRU-MIN	15.87	13.28	19.56	6.02	7.88
LRU-THOLD	15.25	12.91	19.05	5.85	7.43
HIT	15.58	13.36	19.39	6.06	8.36
PER	14.37	12.87	16.70	5.83	7.31
LNC-R-W3	14.72	12.52	19.31	5.94	8.17
LAT	16.18	13.63	20.31	6.23	8.11

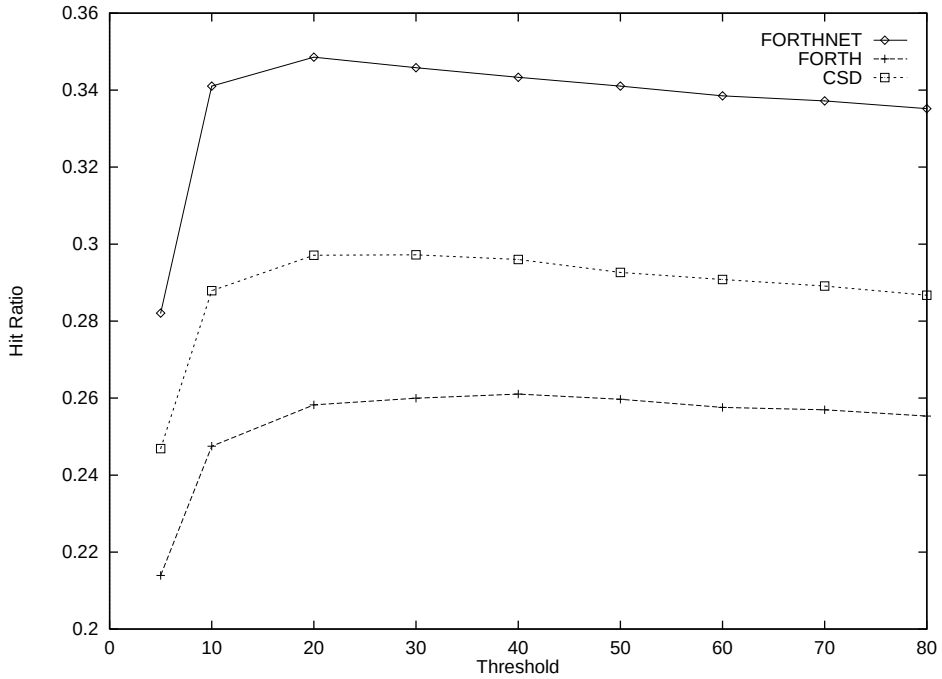
Πίνακας 6.2: Αποτελέσματα των αλγορίθμων για μέγεθος της cache 131072 Kbytes στα CSD, FORTHNET, GRNET, NTUA και 32768 στο FORTH. Δίνεται η βελτίωση του μέσου χρόνου εξυπηρέτησης των αιτήσεων

Αλγόριθμος	CSD	FORTH	FORTHNET	GRNET	NTUA
LFU	27.44	23.62	33.79	3.6	23.7
LRU	25.86	22.28	31.18	3.5	21.7
LRU-MIN	31.47	27.36	36.94	4.9	27.2
LRU-THOLD	29.72	26.10	34.85	4.3	25.1
PER	30.39	26.63	35.68	5.4	27.6
LNC-R-W3	29.12	25.53	35.73	4.7	27
LAT	31.39	27.32	36.35	4.9	26.7
HIT	31.73	27.98	37.95	5.3	28.6

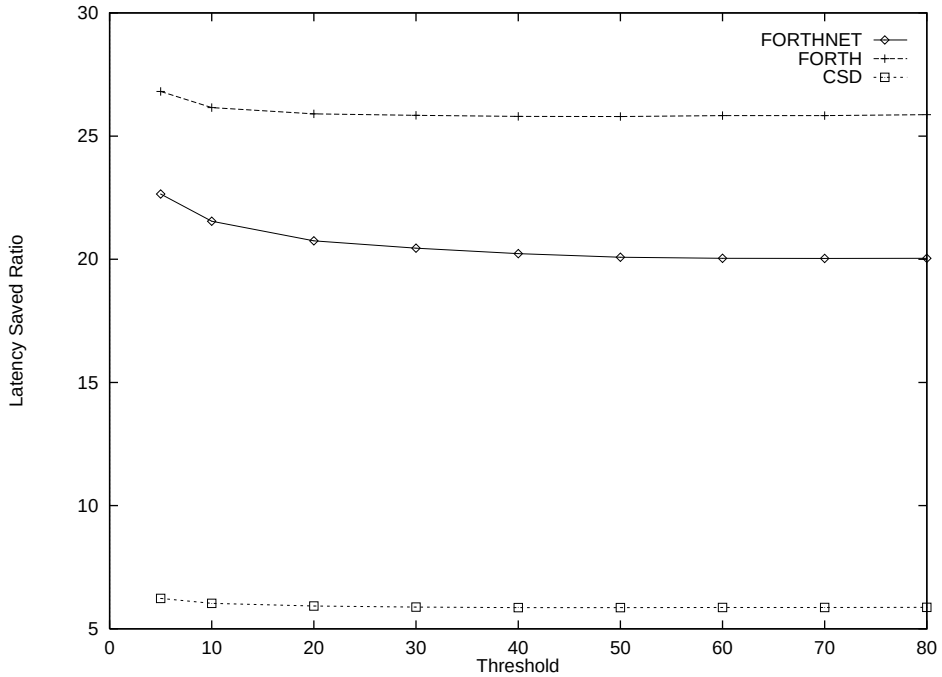
Πίνακας 6.3: Αποτελέσματα των αλγορίθμων για μέγεθος της cache 131072 Kbytes στα CSD, FORTHNET, GRNET, NTUA και 32768 στο FORTH. Δίνεται το ποσοστό των αιτήσεων που εξυπηρετούνται από την cache

εξυπηρέτησης των αιτήσεων σε σχέση με το μέγεθος της cache. Παρατηρείται καλύτερη απόδοση του αλγορίθμου LAT εν συγκρίσει με τους άλλους αλγορίθμους σε όλα σχεδόν τα μεγέθη.

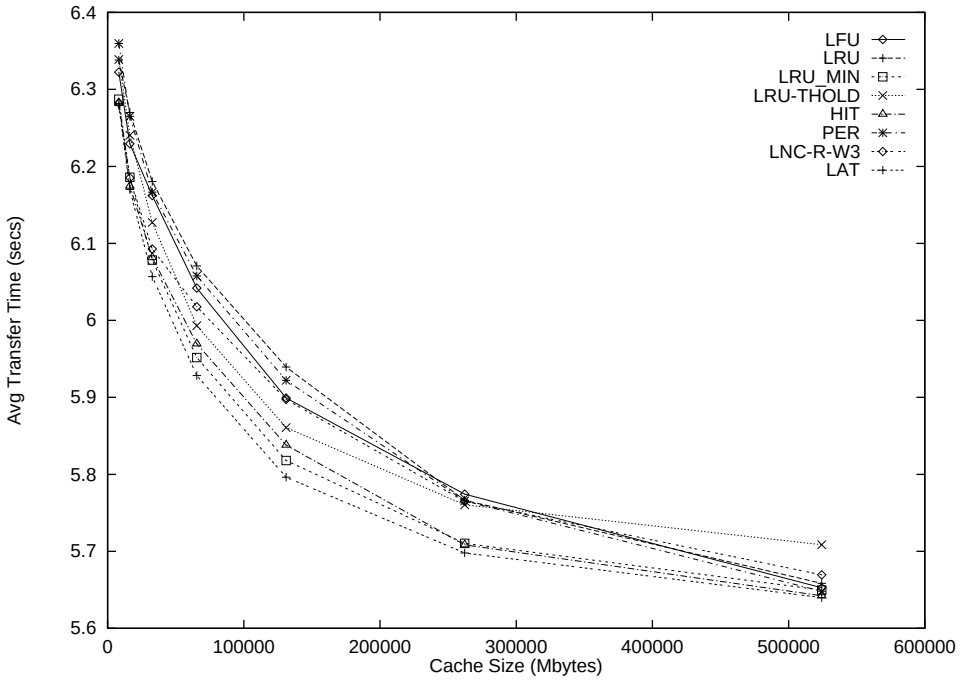
Στις γραφικές παραστάσεις 6.9, 6.10, 6.11, 6.12, 6.13 φαίνεται το αντίστοιχο Hit Ratio των αλγορίθμων. Και εδώ παρατηρείται η καθολική υπεροχή του αλγορίθμου HIT (εκτός από την περίπτωση των GRNET traces όπου ο PER παρουσιάζει την καλύτερη απόδοση από ένα σημείο και μετά).



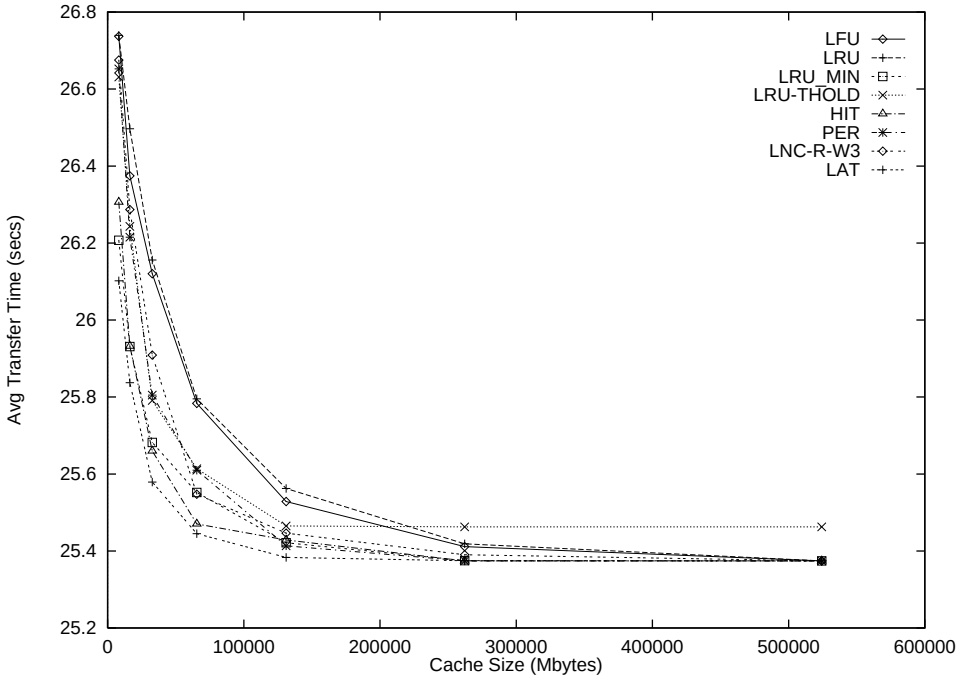
Σχήμα 6.2: Το Hit Ratio που έχει ο αλγόριθμος LRU-THOLD για διάφορες τιμές του THRESHOLD. Το μέγεθος της cache είναι 132 MB για το CSD και το FORTHNET και 32 MB για το FORTH



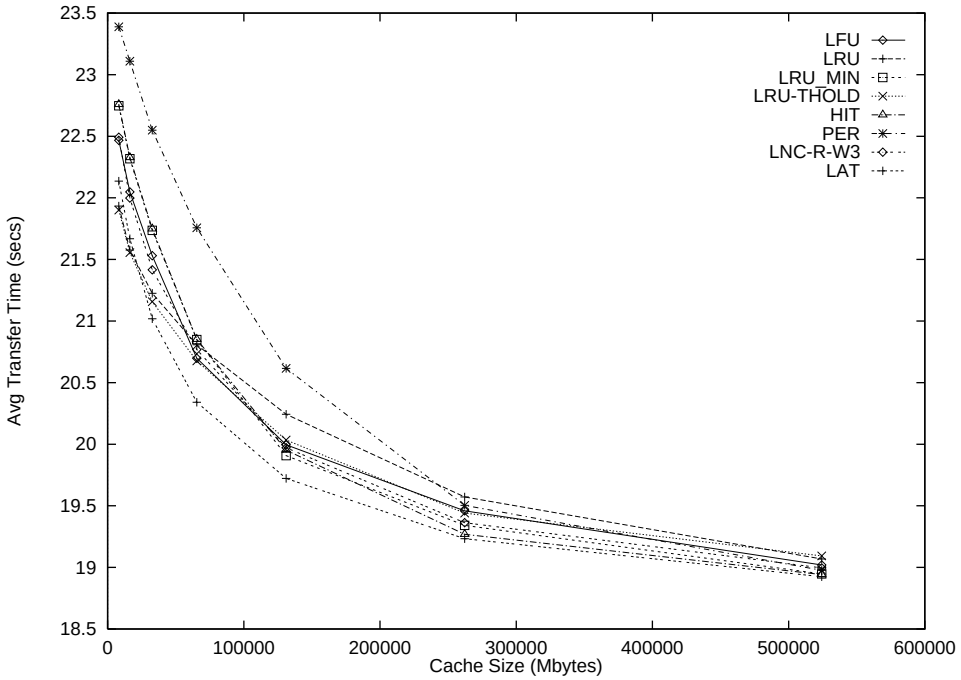
Σχήμα 6.3: Το Latency Saved Ratio που έχει ο αλγόριθμος LRU-THOLD για διάφορες τιμές του THRESHOLD. Το μέγεθος της cache είναι 132 MB για το CSD και το FORTHNET και 32 MB για το FORTH



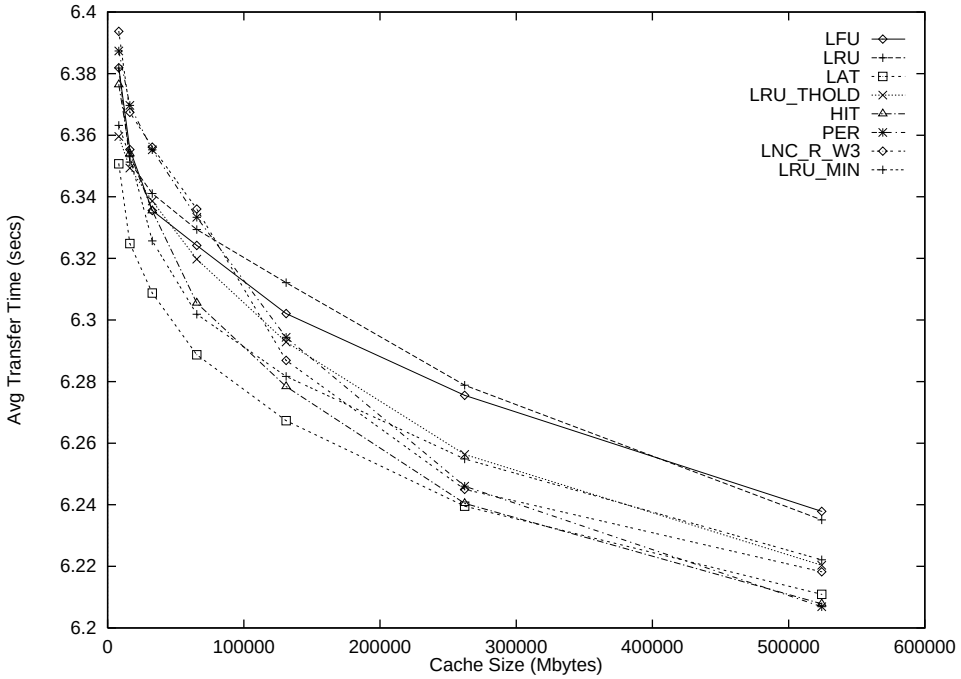
Σχήμα 6.4: Ο μέσος χρόνος εξυπηρέτησης των αιτήσεων στα CSD traces. Ο LAT παρουσιάζει την καλύτερη απόδοση ενώ πολύ καλή απόδοση έχουν και οι LRU-MIN, HIT



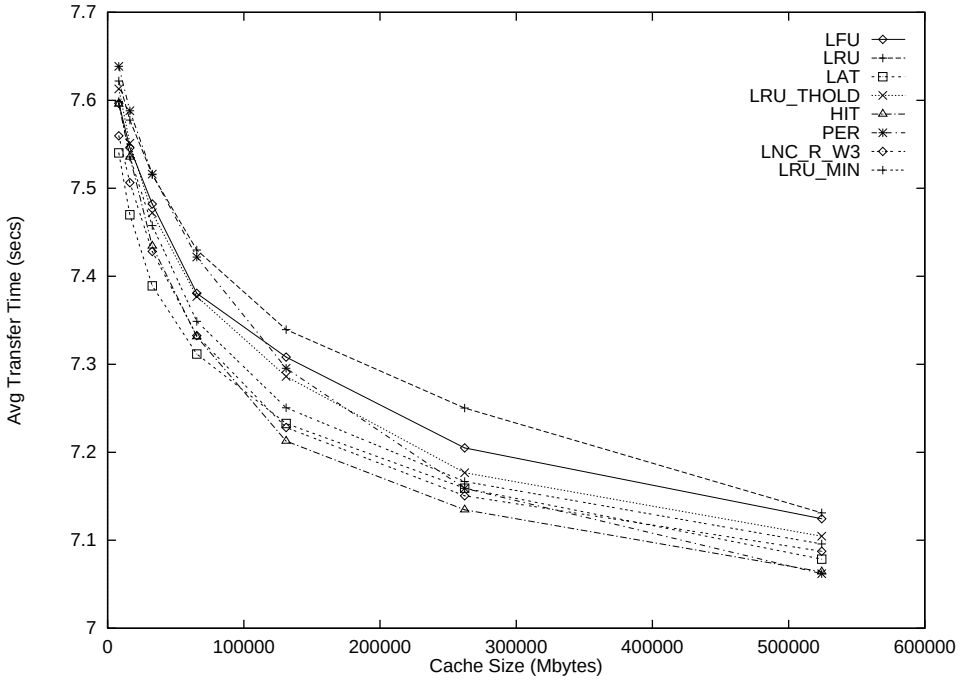
Σχήμα 6.5: Ο μέσος χρόνος εξυπηρέτησης των αιτήσεων στα FORTH traces. Ο LAT παρουσιάζει την καλύτερη απόδοση ενώ πολύ καλή απόδοση έχουν και οι LRU-MIN, HIT



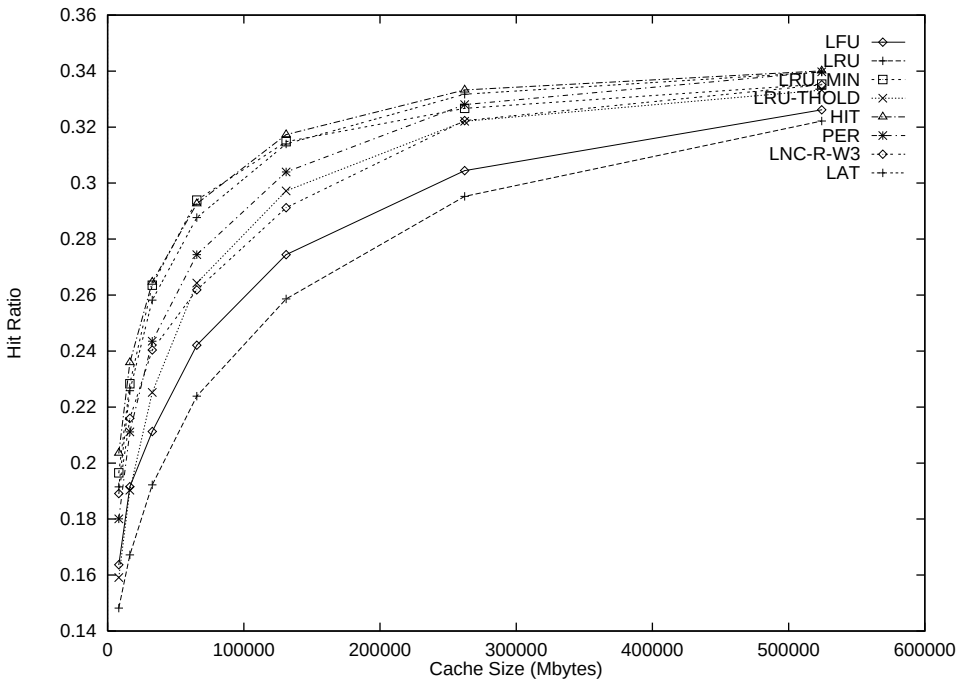
Σχήμα 6.6: Ο μέσος χρόνος εξυπηρέτησης των αιτήσεων στα FORTHNET traces. Ο LAT παρουσιάζει την καλύτερη απόδοση ενώ πολύ καλή απόδοση έχει και ο LRU-MIN



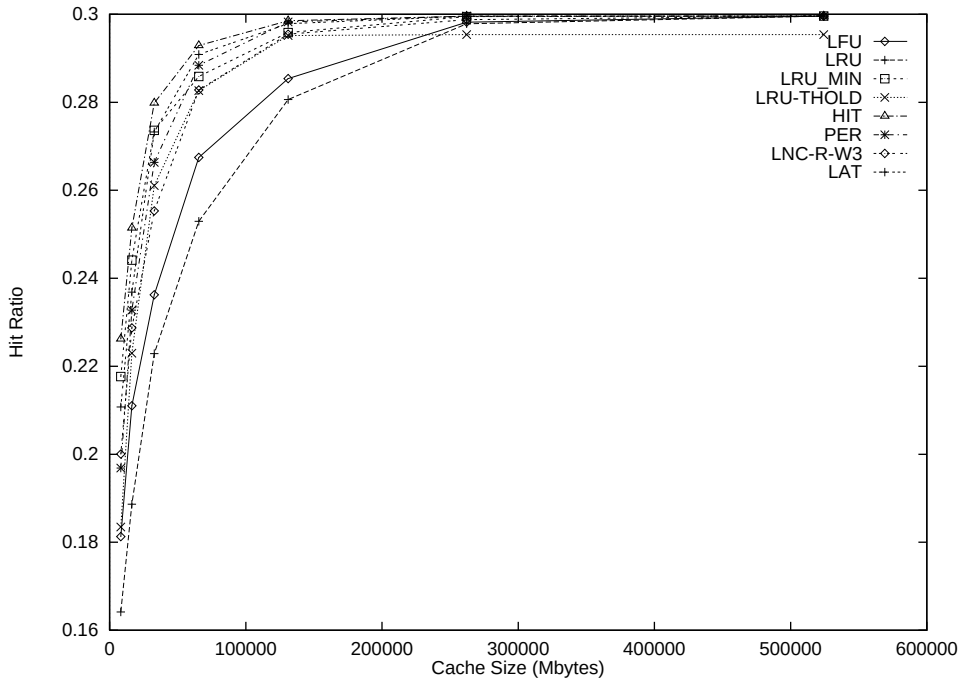
Σχήμα 6.7: Ο μέσος χρόνος εξυπηρέτησης των αιτήσεων στα GRNET traces. Ο LAT παρουσιάζει αρκετά καλή απόδοση



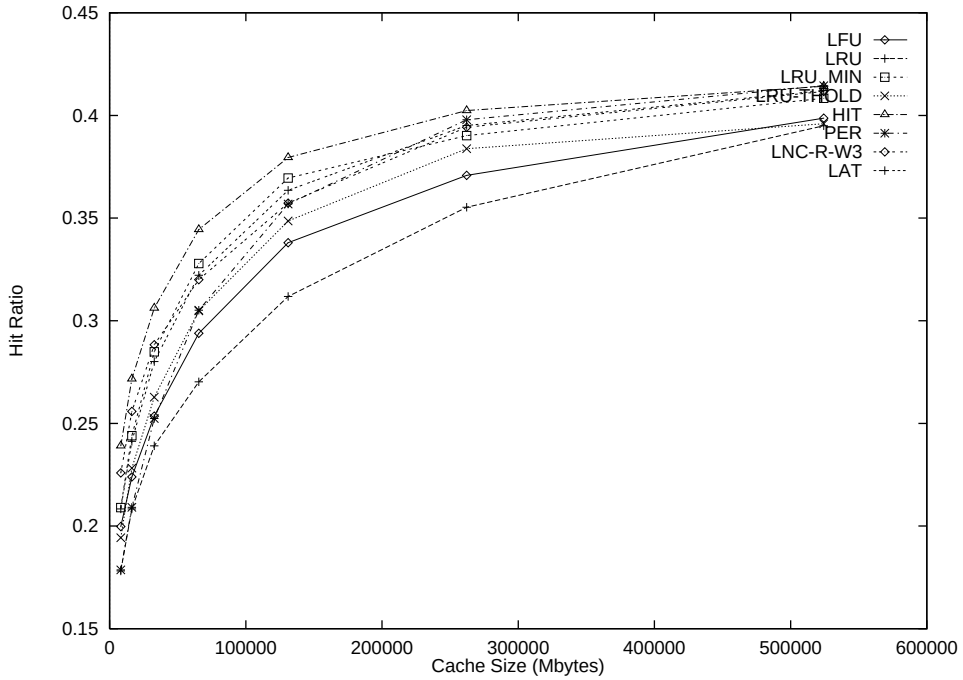
Σχήμα 6.8: Ο μέσος χρόνος εξυπηρέτησης των αιτήσεων στα NTUA traces. Ο LAT παρουσιάζει αρκετά καλή απόδοση



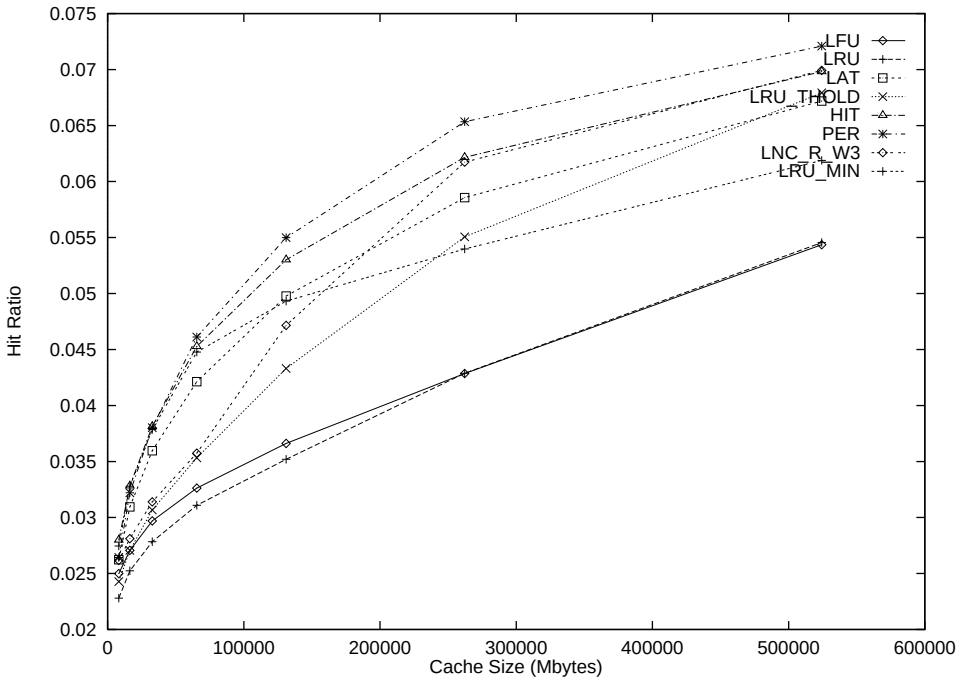
Σχήμα 6.9: Το Hit Ratio στα CSD traces. Παρατηρείται η ανωτερότητα των HIT και LRU-MIN



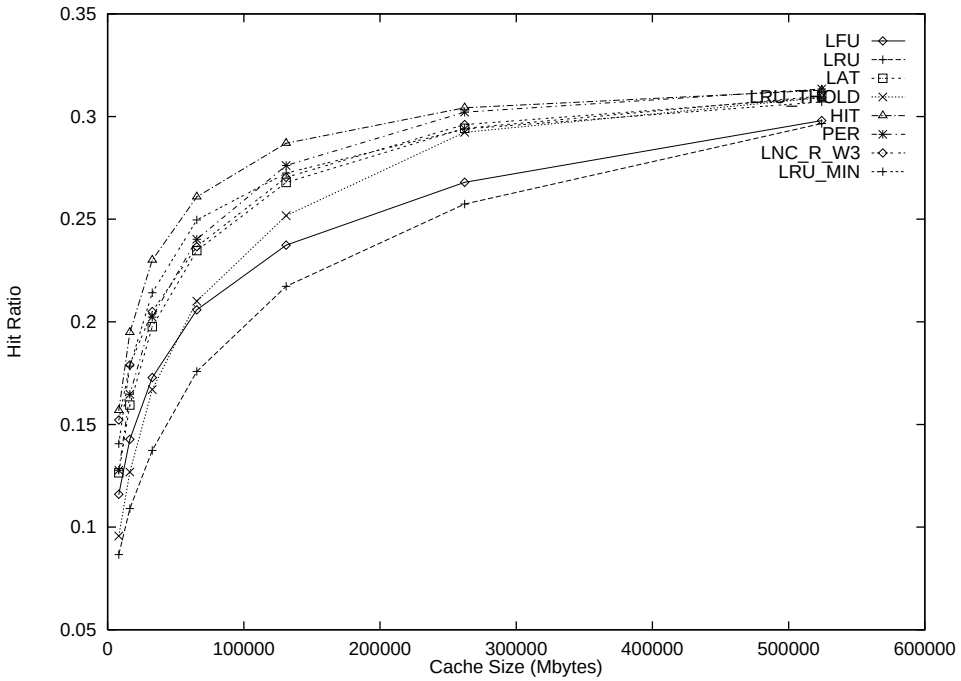
Σχήμα 6.10: Το Hit Ratio στα FORTH traces. Ο HIT παρουσιάζει την καλύτερη απόδοση



Σχήμα 6.11: Το Hit Ratio στα FORTHNET traces. Ο HIT παρουσιάζει την καλύτερη απόδοση ενώ πολύ καλή απόδοση έχει και ο LRU-MIN



Σχήμα 6.12: Το Hit Ratio στα GRNET traces. Ο HIT παρουσιάζει αρκετά καλή απόδοση



Σχήμα 6.13: Το Hit Ratio στα NTUA traces. Ο HIT παρουσιάζει αρκετά καλή απόδοση

Ώρα	LSR
0	36.85%
2	36.73%
2.5	36.12%
4	38.68%
5	36.62%
6	38.15%
24	35.91%
26	35.33%
27	35.43%

Πίνακας 6.4: Η μεταβολή του LSR με την έλευση του χρόνου. Στην πρώτη στήλη δίνεται η έλευση του χρόνου (σε ώρες) από τη πρώτη μέτρηση ενώ στην δεύτερη το ποσοστό της βελτίωσης του μέσου χρόνου εξυπηρέτησης των αιτήσεων των χρηστών

6.6 Αξιοπιστία του κριτηρίου LSR

Ένα ερώτημα το οποίο ανακύπτει είναι ότι αφού για τον υπολογισμό του LSR (παράγραφος 5.1) χρησιμοποιούνται οι χρόνοι που βρίσκονται στα traces, τότε δεν μπορούμε να γνωρίζουμε με ασφάλεια κατά πόσο βελτιώνεται πραγματικά ο μέσος χρόνος εξυπηρέτησης των αιτήσεων. Για παράδειγμα, αν έχουμε μια cache άπειρης χωρητικότητας, τότε φέρνουμε μόνο μια φορά κάθε αρχείο από τον WWW server και οι μελλοντικές αιτήσεις για κάθε αποθηκευμένο αρχείο εξυπηρετούνται από την cache. Για τον υπολογισμό του LSR χρησιμοποιείται ο αρχικός χρόνος απόκτησης του κάθε αρχείου. Το ερώτημα που προκύπτει είναι αν έχει αλλάξει ο χρόνος αυτός και κατά πόσο έχει αλλάξει (διαφορετικές καταστάσεις δικτύου).

Για να μελετήσουμε την αξιοπιστία του κριτηρίου αυτού, στείλαμε 2000 αιτήσεις σε ένα proxy server ανά τακτά χρονικά διαστήματα και υπολογίσαμε το LSR για κάθε περίπτωση. Η χωρητικότητα της cache είναι 3 MB. Ο proxy server και ο πελάτης που στέλνει τις αιτήσεις βρίσκονται στο ίδιο τοπικό δίκτυο. Στην αρχή της κάθε μέτρησης η cache είναι άδεια. Το Hit Ratio για το σύνολο των αιτήσεων είναι περίπου 30%.

Στον πίνακα 6.4 φαίνεται ότι δεν μεταβάλλεται σημαντικά το LSR στις μετρήσεις που κάναμε. Το μεγαλύτερο ποσοστό μεταβολής είναι περίπου 3% και συμβαίνει αρκετά σπάνια.

6.7 Σύγκριση με το Harvest Cache

Στην παράγραφο αυτή δίνεται η συγκριτική παρουσίαση του συστήματος μας caching με το Harvest Cache (που περιγράφεται στην παράγραφο 2.1). Επιλέχθηκε αυτό το σύστημα γιατί αποτελεί ένα από τα καλύτερα συστήματα caching. Για τον σκοπό αυτό, μετρήσαμε τις παρακάτω παραμέτρους:

Κριτήριο	Harvest	Το δικό μας σύστημα
Εξυπηρέτηση αιτήσεων από το δίκτυο	400 msecs/req	600 msecs/req
Εξυπηρέτηση αιτήσεων από την cache	30-35 msecs/req	200-220 msecs/req
LSR	21%	23.8%
Γλώσσα υλοποίησης	C	Perl
Μοντέλο λειτουργίας	Non Blocking I/O	Non Blocking I/O
Συνεργασία proxy servers	Ιεραρχική	Πρωτόκολλο (κατάλογοι)

Πίνακας 6.5: Σύγκριση με το Harvest Cache

- Τον ρυθμό εξυπηρέτησης αιτήσεων που δεν βρίσκονται στην cache (δηλαδή από το δίκτυο) για κάθε σύστημα. Για τον υπολογισμό του, στείλαμε 1000 αιτήσεις σε κάθε σύστημα το οποίο ξεκίνησε την λειτουργία του με άδεια cache. Η cache δεν γεμίζει. Οι αιτήσεις αυτές προσπελαύνουν αρχεία εκτός Ελλάδας. Ο proxy server και οι πελάτες (clients) που στέλνουν τις αιτήσεις βρίσκονται στο ίδιο Domain (ίδιο τοπικό δίκτυο). Υπάρχουν 4 πελάτες οι οποίοι στέλνουν από 250 αιτήσεις σε καθέναν από τους proxy servers.

- Τον ρυθμό εξυπηρέτησης αιτήσεων από την cache. Για τον υπολογισμό του, στείλαμε τις ίδιες 1000 αιτήσεις σε κάθε σύστημα οι οποίες μπορούν να εξυπηρετηθούν από την cache. Υπάρχουν 4 πελάτες οι οποίοι στέλνουν από 250 αιτήσεις και σε αυτήν την περίπτωση.

- Την απόδοση της cache (καθυστέρηση στην εξυπηρέτηση των αιτήσεων). Ο υπολογισμός του έγινε για 45000 αιτήσεις και cache 32 MB.

- Θέματα υλοποίησης των δύο συστημάτων.

Από τον πίνακα 6.5 φαίνεται ότι το Harvest Cache εξυπηρετεί τις αιτήσεις με γρηγορότερο ρυθμό από το δικό μας σύστημα Τα αποτελέσματα αυτά είναι σε συμφωνία με αυτά που αναφέρονται στο [CDCNW95]. Επειδή και στα δύο συστήματα το μοντέλο λειτουργίας είναι το ίδιο, η διαφορά στον ρυθμό εξυπηρέτησης αιτήσεων οφείλεται περισσότερο στην γλώσσα υλοποίησης. Επιπλέον, ο δικός μας proxy server αποτελεί μια πιλοτική υλοποίηση η οποία μας βοήθησε για την εφαρμογή και δοκιμή διαφόρων πολιτικών. Τέλος, το Lattency Saved Ratio (LSR) είναι υψηλότερο στον δικό μας proxy server κατά ένα ποσοστό 10% από ότι στο Harvest Cache.

Αν θέλουμε να βελτιστοποιήσουμε το HR (Hit Ratio) τότε ο αλγόριθμος HIT έχει 32.7% Hit Ratio ενώ το Harvest Cache έχει 26% Hit Ratio.

6.8 Σύγκριση συνεργασίας proxy servers

Στην παράγραφο αυτή δίνεται η συγκριτική παρουσίαση των δυο εναλλακτικών τρόπων συνεργασίας των proxy servers:

- Με ανταλλαγή καταλόγων οι οποίοι περιέχουν τα πολυτιμότερα αρχεία που υπάρχουν στις caches τους.
- Με συνεχείς ερωτήσεις για τα αρχεία που δεν υπάρχουν στις caches τους. Έτσι, στην περίπτωση αυτή, κάθε φορά που κάποιος proxy server δέχεται μια αίτηση για αρχείο το οποίο δεν υπάρχει στην cache του, στέλνει την αίτηση σε όλους τους συνεργαζόμενους proxy servers (δηλ. τους ρωτάει).

Αρχικά, συγκρίναμε την απόδοση του δικού μας πρωτοκόλλου συνεργασίας των proxy servers με την απόδοση των συνεργαζόμεων Harvest proxy servers. Το δίκτυο αποτελείται από 4 proxy servers, τρεις από τους οποίους βρίσκονται στο ίδιο δίκτυο (domain) ενώ ο άλλος σε διαφορετικό. Υπάρχουν 4 πελάτες που στέλνουν από 250 αιτήσεις σε αυτούς τους proxy servers. Οι proxy servers ξεκίνησαν με άδεια cache. Στην συνέχεια, οι πελάτες στέλνουν τις αιτήσεις τους σε διαφορετικούς proxy servers από ότι στην αρχή ούτως ώστε η εξυπηρέτηση τους να γίνεται μέσω ερωτήσεων των συνεργαζομένων proxy servers. Έτσι, αν για παράδειγμα, ο πελάτης C_y στείλει αρχικά τις αιτήσεις του στον P_j proxy server, κατόπιν, τις στέλνει στον P_i. Ο P_i θα προωθήσει τις αιτήσεις αυτές στον P_j.

Και στις δυο περιπτώσεις δεν παρατηρείται μεγάλη αύξηση του μέσου χρόνου εξυπηρέτησης των αιτήσεων. Στο Harvest, ο χρόνος αυτός αυξάνει κατά 10-15% (περίπου 430-450 msecs για την εξυπηρέτηση αιτήσεων από το δίκτυο και 35-40 msecs για την εξυπηρέτηση αιτήσεων από την cache). Στον δικό μας proxy server, ο επιπλέον φόρτος (δηλαδή η επιπλέον αύξηση του μέσου χρόνου εξυπηρέτησης των αιτήσεων) είναι κατά 20-25% (περίπου 700 msecs για την εξυπηρέτηση αιτήσεων από το δίκτυο και 240-260 msecs για την εξυπηρέτηση αιτήσεων από την cache). Ειδικά στην δική μας περίπτωση, η συνεργασία των proxy servers, δηλαδή η ανταλλαγή καταλόγων, είναι σαν μια επιπλέον αίτηση (μιας και οι συνεργαζόμενοι proxy servers λαμβάνουν έναν κατάλογο ο οποίος δεν διαφέρει σε τίποτα από μια HTTP αίτηση).

6.9 Αλγόριθμοι δικτύου

Στην παράγραφο αυτή αυτό παρουσιάζουμε έναν αλγόριθμο δικτύου ο οποίος χρησιμοποιείται σε κατανεμημένο επίπεδο, στην συνεργασία των proxy servers. Οι proxy servers διαδραματίζουν τον ρόλο του χρήστη γιατί ζητάνε αρχεία από άλλους proxy servers αλλά και τον ρόλο του εξυπηρετητή γιατί παρέχουν αρχεία σε άλλους proxy servers. Το εύρος ζώνης του δικτύου που τους ενώνει κατέχει σημαντικό ρόλο στις συναλλαγές τους. Για παράδειγμα, αν κάποιο ζητούμενο αρχείο υπάρχει σε δυο γειτονικούς proxy servers, ενδεχομένως να είναι προτιμότερο να ληφθεί από εκείνον που απαιτεί λιγότερο χρόνο.

6.9.1 Προτεινόμενος αλγόριθμος δικτύου

Κύριος σκοπός του προτεινόμενου αλγορίθμου είναι να βελτιώσει τον μέσο χρόνο εξυπηρέτησης των αιτήσεων. Για τον σκοπό αυτό, κάθε φορά που ο proxy server λαμβάνει αίτηση για κάποιο αρχείο που δεν υπάρχει στην cache του, αναζητά το αρχείο αυτό σε κάποιον συνεργαζόμενο proxy server ή στον WWW server. Για να επέλθει βελτίωση του μέσου χρόνου εξυπηρέτησης των αιτήσεων, θα πρέπει να λάβει το αρχείο από την κοντινότερη πηγή.

Σύμφωνα με το πρωτόκολλο συνεργασίας των proxy servers, κάθε proxy server στέλνει έναν κατάλογο με τα σημαντικότερα αρχεία της cache του, στους συνεργαζόμενους proxy servers. Τα αρχεία που βρίσκονται σε υψηλότερη θέση στον κατάλογο έχουν μικρότερη πιθανότητα να διαγραφούν από την cache στο άμεσο χρονικό διάστημα. Η πιθανότητα αυτή υπολογίζεται με στατιστικές μεθόδους. Έτσι, οι proxy servers γνωρίζουν εκ των προτέρων που πρέπει να αναζητήσουν τα αρχεία που δεν υπάρχουν στις caches τους.

Για την τελική διαμόρφωση της απόφασης, θα πρέπει να εκτιμηθεί ο αναμενόμενος χρόνος που απαιτείται για την αναζήτηση του ζητούμενου αρχείου στους συνεργαζόμενους proxy servers και στον WWW server. Για τον υπολογισμό αυτό, χρησιμοποιούνται στατιστικές μέθοδοι.

Θεωρούμε την περίπτωση όπου ο proxy server P_j πρέπει να αποφασίσει σε ποιον συνεργαζόμενο proxy server θα αναζητήσει κάποιο ζητούμενο αρχείο F το οποίο δεν υπάρχει στην cache του. Έστω C το σύνολο των proxy servers που έχουν το αρχείο στην cache τους (n proxy servers). Το σύνολο αυτό ο P_j μπορεί να το δημιουργήσει λαμβάνοντας υπόψιν τους καταλόγους που διατηρεί για τους άλλους proxy servers. Για κάθε proxy server P_i του συνόλου C αντιστοιχεί κάποιος αναμενόμενος χρόνος AT_i. Η αίτηση θα επιλεγεί να δρομολογηθεί σε εκείνον τον proxy server που έχει ελάχιστο AT. Θεωρούμε ό,τι οι proxy servers του συνόλου C είναι ταξινομημένοι ως προς AT.

$$\text{A}T_i = P_n * T_n + (1 - P_n) * (P_{n-1} * T_{n-1} + (1 - P_{n-1}) * \ldots * (P_1 * T_1 + (1 - P_1) * T_0))$$

όπου

- T_i : ο μέσος χρόνος που απαιτείται για να ληφθεί το αρχείο από τον P_i proxy server. Μπορεί να υπολογιστεί κρατώντας στατιστικά του χρόνου επικοινωνίας με τους συνεργαζόμενους proxy servers.
- T_0 : ο μέσος χρόνος που απαιτείται για να ληφθεί το αρχείο από τον WWW server. Και στην περίπτωση αυτή, ο χρόνος αυτός υπολογίζεται με στατιστική μέθοδο.
- P_i : η πιθανότητα να υπάρχει το αρχείο F στην cache του P_i και να μην έχει διαγραφεί από αυτή. Μπορεί να εκτιμηθεί από την θέση του αρχείου στον κατάλογο που αντιστοιχεί στον P_i proxy server.

Δηλαδή, ο αναμενόμενος χρόνος είναι ο συνολικός χρόνος που εκτιμάται ότι απαιτείται

για να αποκτηθεί το αρχείο. Αν το αρχείο αναζητηθεί στον P_i proxy server και δεν βρεθεί εκεί, κατόπιν θα αναζητηθεί στον P_{i-1} proxy server και ούτως καθεξής μέχρι να βρεθεί.

Η εύρεση του παραπάνω αναμενόμενου χρόνου είναι ισοδύναμη με τη εύρεση του ελάχιστου:

$$min(ET_i) = min(P_i * T_i + (1 - P_i) * T_0)$$

Ο αλγόριθμος υπολογίζει τα ET_i καθώς και τον χρόνο T_0 (ενδεχομένως να είναι προτιμότερο να σταλεί η αίτηση στον WWW server) και δρομολογεί την εξυπηρέτηση της στον proxy server με τον μικρότερο εκτιμώμενο χρόνο ET. Για την διαχείριση της χωρητικότητας της cache, χρησιμοποιείται ο αλγόριθμος **LAT**.

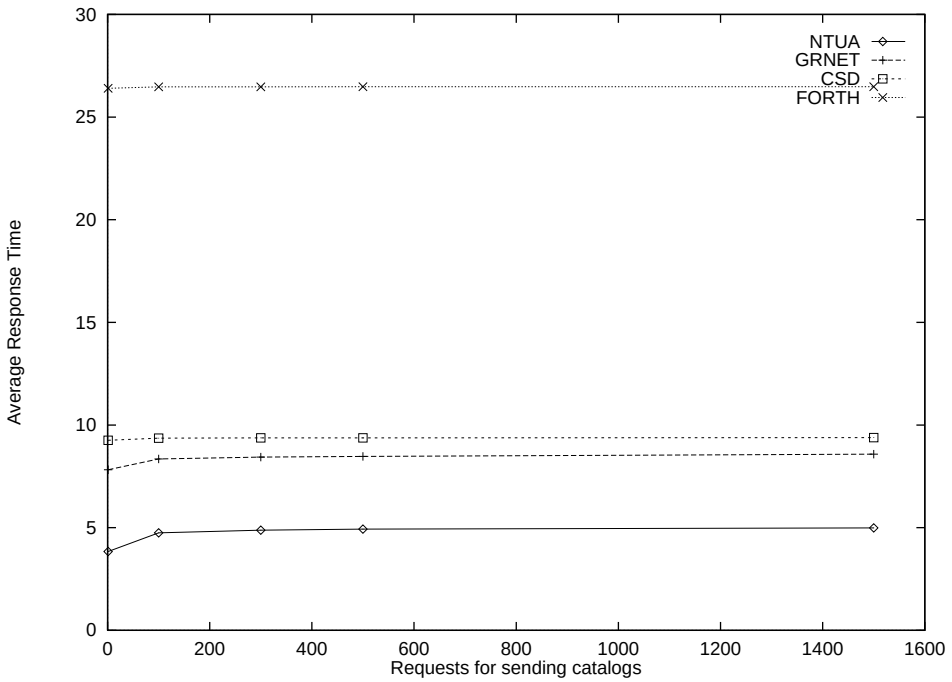
6.9.2 Πειραματικά αποτελέσματα

Η σύγκριση του προτεινόμενου αλγορίθμου δικτύου έγινε με κάποιον αλγόριθμο που τυχαία αποφασίζει σε ποιον συνεργαζόμενο proxy server θα δρομολογήσει την αίτηση. Ο αλγόριθμος αυτός βασίζεται στον **LAT** για την διαχείριση της χωρητικότητας της cache. Έτσι, μπορούμε να ελέγξουμε την αποδοτικότητα των αποφάσεων που λαμβάνει ο προτεινόμενος αλγόριθμος δικτύου.

Η σύγκριση έγινε προσομοιώνοντας την λειτουργία 4 proxy servers για το χρονικό διάστημα Φεβρουαρίου 1997. Χρησιμοποιήθηκαν traces από το Τμήμα Επιστήμης Υπολογιστών (CSD), το Ινστιτούτο Πληροφορικής (FORTH), το Πολυτεχνείο Αθηνών (NTUA), και το Ελληνικό δίκτυο (GRNET).

Τα αποτελέσματα της συγκρίσεως δείχνουν ό,τι οι αποφάσεις που λαμβάνει ο προτεινόμενος αλγόριθμος δικτύου βελτιώνουν τον μέσο χρόνο εξυπηρέτησης των αιτήσεων κατά επιπλέον 1%. Αυτό είναι ιδιαίτερα σημαντικό αν ληφθεί υπόψιν ό,τι τις περισσότερες φορές δεν υπάρχουν πολλές δυνατές επιλογές για να αναζητηθεί το αρχείο σε περισσότερους του ενός συνεργαζόμενους proxy servers. Έτσι, ο προτεινόμενος αλγόριθμος δικτύου λαμβάνει την ίδια απόφαση με τον τυχαίο αλγόριθμο δικτύου (εξ'ανάγκης).

Για την τεκμηρίωση του προηγούμενου ποσοστού βελτίωσης, συγκρίναμε την απόδοση του προτεινόμενου αλγόριθμου δικτύου με έναν αλγόριθμο που θεωρητικά θα μπορούσε να χαρακτηριστεί βέλτιστος. Ο αλγόριθμος αυτός υποθέτει ό,τι γνωρίζει ανά πάσα χρονική στιγμή τα αρχεία που βρίσκονται στις caches των συνεργαζομένων proxy servers (χωρίς αποστολή καταλόγου) και δρομολογεί την αίτηση στον πιο κοντινό συνεργαζόμενο proxy server. Για να λάβει την απόφαση αυτή δεν χρειάζεται να υπολογίσει με ποια πιθανότητα μπορεί να βρει το ζητούμενο αρχείο στις caches των άλλων proxy servers αφού είναι 1. Η σύγκριση έγινε για το ίδιο μέγεθος της cache και για τους δυο αλγορίθμους αλλά μεταβάλλοντας την χρονική περίοδο που κάθε proxy server στέλνει καταλόγους των πολυτιμότερων αρχείων της cache του στους άλλους proxy servers. Η χρονική αυτή περίοδος καθορίζεται από τον αριθμό των αιτήσεων που λαμβάνει κάθε proxy server (πχ κάθε 300, 500, ... αιτήσεις)



Σχήμα 6.14: Ο μέσος χρόνος εξυπηρέτησης των αιτήσεων όταν συνεργάζονται οι proxy servers. Παριστάνεται γραφικά συναρτήσει του χρονικού διαστήματος που στέλνονται οι κατάλογοι των πολυτιμότερων αρχείων. Στην θέση 1, παριστάνεται η απόδοση του θεωρητικά βέλτιστου αλγορίθμου. Η βελτίωση που παρατηρείται είναι αρκετά σημαντική

Από την γραφική παράσταση 6.14 φαίνεται ό,τι ο προτεινόμενος αλγόριθμος δικτύου παρουσιάζει σχεδόν την ίδια απόδοση με τον υποτιθέμενο βέλτιστο αλγόριθμο απόφασης δικτύου. Επίσης, φαίνεται ό,τι ακόμα και αν οι κατάλογοι στέλνονται κάθε 1500 αιτήσεις, η βελτίωση είναι αρκετά σημαντική.

Κεφάλαιο 7

Συμπεράσματα

Στην εργασία αυτή εξετάζουμε κάποιες μεθόδους αντιμετώπισης της συμφόρησης στο WWW. Από τις δυο πιο διαδεδομένες τεχνικές, το **caching** και το **replication**, προτιμήσαμε την πρώτη επειδή η λειτουργία του είναι είναι αυτόματη και οι αποφάσεις του λαμβάνονται ανάλογα με τις προτιμήσεις των χρηστών. Για την εφαρμογή των νέων μεθόδων, σχεδιάσαμε και υλοποιήσαμε έναν proxy server. Επίσης υλοποιήσαμε ένα πρωτόκολλο επικοινωνίας που διέπει τις συναλλαγές τους. Στην εργασία αυτή:

- Περιγράφουμε την σχεδίαση και υλοποίηση ενός proxy server ο οποίος ασκεί μεσολαβητικό ρόλο στην εξυπηρέτηση των αιτήσεων των WWW χρηστών. Επικοινωνεί με τον συντονιστή δικτύου. Ρωτώντας τον συντονιστή δικτύου μπορεί να γνωρίζει τα πολυτιμότερα αρχεία που βρίσκονται στις caches των συνεργαζομένων proxy servers. Έτσι, μπορεί να αναζητήσει τα αρχεία που δεν υπάρχουν στην δική του cache, στις caches των άλλων proxy servers. Έγινε μια πιλοτική υλοποίηση του proxy server για να εφαρμόσουμε τους προτεινόμενους αλγορίθμους διαχείρισης της cache και το πρωτόκολλο συνεργασίας τους. Η τροποποίηση ενός από τους ήδη υπάρχοντες proxy servers θα ήτανε ιδιαίτερα δύσκολη αφού θα έπρεπε να αλλάξει εντελώς ο τρόπος επικοινωνίας τους.
- Περιγράφουμε την σχεδίαση και υλοποίηση του συντονιστή δικτύου ο οποίος αναλαμβάνει την συνεργασία των proxy servers. Αναλύθηκε το πρωτόκολλο που διέπει τις συναλλαγές τους. Ανταλλάσσουν μεταξύ τους λίστες με τα πολυτιμότερα αρχεία των cache τους.
- Προτείνουμε αλγορίθμους διαχείρισης της cache συγκρίνοντας την απόδοση τους με άλλους καθιερωμένους αλγορίθμους (όπως LRU, LFU, FIFO) σε έναν προσομοιωτή για να δοκιμάσουμε διάφορα σενάρια χωρίς ιδιαίτερο υπολογιστικό κόστος.

Βασισμένοι στην υλοποίηση μας και στα πειραματικά αποτελέσματα, προέκυψαν τα εξής συμπεράσματα:

- Η διαχείριση της cache του proxy server είναι πιο περίπλοκη από ότι στα προγενέστερα συστήματα (όπως σε ένα λειτουργικό σύστημα). Για την αποδοτικότερη λειτουργία

της πρέπει να ληφθούν υπόψιν αρκετοί παράγοντες, όπως:

(1) Η καθυστέρηση στην απόκτηση των αρχείων: είναι πιο σημαντικό ένα αρχείο ενός μακρυνού WWW server παρά ενός κοντινού. Οι χρήστες ενοχλούνται κάθε φορά που πρέπει να περιμένουν αρκετή ώρα για να λάβουν ένα αρχείο.

(2) Το μέγεθος των αρχείων: αρχεία με μικρό μέγεθος, αφενός μεν παρουσιάζουν περισσότερες αιτήσεις (από στατιστικά αποτελέσματα) αφετέρου δε έχουν μικρότερες απαιτήσεις για αποθήκευση στον δίσκο της cache καταλαμβάνοντας μικρότερο χώρο.

(3) Ο αριθμός των ζητήσεων για το κάθε αρχείο καθώς και η χρονική στιγμή της τελευταίας ζήτησης του. Οι ιδέες των αλγορίθμων LFU και LRU αποδεικνύονται σημαντικές στην περίπτωση μας.

(4) Η γήρανση των αρχείων που είναι αποθηκευμένα στην cache σύμφωνα με την ιδέα του *βήματος* της cache. Έχει αποδειχθεί ό,τι μπορεί να συμβάλλει στην σχεδίαση αποδοτικότερων αλγορίθμων διαχείρισης της cache.

- Οι proxy servers δεν πρέπει να αναζητούν αρχεία που δεν υπάρχουν στην cache τους στις caches άλλων proxy servers παρά μόνο όταν γνωρίζουν ότι κάτι τέτοιο έχει μεγάλη πιθανότητα να πετύχει. Σύμφωνα με την δική μας προσέγγιση, η επιτυχής αναζήτηση αρχείων κυμαίνεται σε ποσοστά 98-99%. Διαφορετικά, επιβαρύνουν το δίκτυο με επιπλέον μηνύματα.

7.1 Μελλοντική εργασία

- Το περιεχόμενο των καταλόγων που ανταλλάσσουν οι συνεργαζόμενοι proxy servers μπορεί να αποτελέσει πεδίο μελλοντικής εργασίας. Έτσι, οι proxy servers θα μπορούσαν να στέλνουν κάθε φορά όχι ολόκληρο τον κατάλογο των πολυτιμότερων αρχείων της cache τους, αλλά μόνο τα αρχεία που έχουν εισαχθεί ή διαγραφεί από αυτή εν σχέσει με την προηγούμενη αποστολή καταλόγου τους. Αυτό θα μπορούσε να συμβάλλει στην συχνότερη αποστολή καταλόγων αφού δεν θα στέλνονται ολόκληροι κατάλογοι αλλά μόνο τροποποιήσεις τους. Η εφαρμογή της μεθόδου αυτής δεν είναι ιδιαίτερα εύκολη και απαιτεί την αντιμετώπιση ορισμένων προβλημάτων. Το πρώτο πρόβλημα είναι ότι αυξάνεται ο φόρτος των servers μιας και πρέπει να επεξεργαστούν τις τροποποιήσεις των καταλόγων που λαμβάνουν. Ένα επιπλέον πρόβλημα είναι η αντιμετώπιση περιπτώσεων που κάποιος συντονιστής δικτύου τεθεί εκτός λειτουργίας όποτε δεν θα διατηρεί τους σωστούς καταλόγους. Προς την αποφυγή φόρτωσης των servers μπορεί να βοηθήσει το γεγονός ό,τι ο συντονιστής δικτύου είναι υπεύθυνος για την αποστολή και λήψη των καταλόγων. Απαιτείται επέκταση του πρωτοκόλλου επικοινωνίας των συντονιστών δικτύου για την εφαρμογή της μεθόδου αυτής.

- Σχεδίαση αλγορίθμου διαχείρισης της cache ο οποίος εκτιμά την χρονική στιγμή που θα ζητηθεί κάποιο αποθηκευμένο αρχείο της cache [PR94]. Αυτό μπορεί να γίνει είτε με την μέθοδο της πολυωνυμικής παρεμβολής είτε με την μέθοδο των ελαχίστων τετραγώνων χρησιμοποιώντας σαν παραμέτρους τις χρονικές στιγμές των προηγούμενων ζητήσεων

για κάποιο αρχείο. Πολυτιμότερο θα είναι εκείνο το αρχείο το οποίο εκτιμάται ό,τι θα ζητηθεί πιο κοντά στην παρούσα χρονική στιγμή. Οι αλγόριθμοι που δοκιμάστηκαν προς αυτήν την κατεύθυνση δεν είχαν τα επιθυμητά αποτελέσματα αλλά σίγουρα επιδέχονται επιπλέον βελτίωση.

Παράρτημα Α

Το πρωτόκολλο HTTP

Το WWW στηρίζεται στην αρχιτεκτονική πελάτη-εξυπηρετητή (client-server). Ένας WWW server είναι ένα πρόγραμμα του οποίου ο μοναδικός σκοπός του είναι να παρέχει αρχεία (γενικότερα πόρους) σε άλλα προγράμματα κάθε φορά που του ζητείται. Ένας WWW client είναι ένα πρόγραμμα το οποίο αλληλεπιδρά με τον χρήστη και μεταφέρει τις αιτήσεις του για αρχεία σε WWW servers. Το πρωτόκολλο που χρησιμοποιούν οι WWW clients και servers για την ολοκλήρωση των συναλλαγών τους είναι το **HTTP** (Hypertext Transfer Protocol).

Α.1 Γενική περιγραφή αίτησης

Το HTTP πρωτόκολλο [FGM+96, BLFF96, PJM95] χρησιμοποιήθηκε στο WWW αρχικά το 1990. Βασίζεται στο μοντέλο αίτησης/απάντησης. Ο πελάτης αρχικοποιεί μια σύνδεση με τον εξυπηρετητή στέλνοντας του την αίτηση. Η αίτηση αποτελείται από τρία μέρη:

- Την επικεφαλίδα της αίτησης (δηλαδή η πρώτη γραμμή της) η οποία δηλώνει το αρχείο που ζητάει ο πελάτης, την μέθοδο που θα εφαρμοστεί στο ζητούμενο αρχείο και την έκδοση του πρωτοκόλλου.
- Πληροφορία του πελάτη η οποία περιγράφει τις δυνατότητες του και τις προτιμήσεις του.
- Ένα προαιρετικό τμήμα δεδομένων.

Α.1.1 Επικεφαλίδα της αίτησης

Όπως αναφέρθηκε, η επικεφαλίδα της αίτησης περιγράφει το ζητούμενο αρχείο καθώς και την μέθοδο η οποία θα εφαρμοστεί σε αυτό. Το συντακτικό της επικεφαλίδας είναι:

Request-Line = Method Request-URI HTTP-Version

Η μέθοδος συνήθως είναι ένα από τα επόμενα:

- **GET**: χρησιμοποιείται για την απόκτηση (retrieve) του αρχείου που περιγράφεται από το `Request-URI`.
- **HEAD**: είναι όμοια με την προηγούμενη μέθοδο με την διαφορά ότι ο server που λαμβάνει την αίτηση επιστρέφει μόνο μεταπληροφορία που σχετίζεται με το ζητούμενο αρχείο (πχ. πότε τροποποιήθηκε τελευταία φορά, ποιο είναι το μέγεθος του) και όχι το ζητούμενο αρχείο.
- **POST**: χρησιμοποιείται για να πληροφορήσει τον server ότι το τμήμα δεδομένων που εσωκλείεται στην αίτηση πρέπει να ληφθεί σαν είσοδος στο ζητούμενο αρχείο.

A.1.2 Πληροφορία του πελάτη

Δίνεται η δυνατότητα στον πελάτη να στείλει στον server επιπλέον πληροφορία που πρέπει να λάβει υπόψιν του για την εξυπηρέτηση της αίτησης. Τα κυριότερα πεδία, είναι τα εξής:

- **Authorization**: όταν απαιτείται κάποια πιστοποίηση του πελάτη που ζητά το συγκεκριμένο αρχείο.
- **If-Modified-Since**: χρησιμοποιείται για την υλοποίηση ενός *υπό συνθήκην* GET. Αν το ζητούμενο αρχείο του server δεν έχει τροποποιηθεί μετά από την χρονική στιγμή που περιγράφεται στο πεδίο αυτό, ο server επιστρέφει μόνο μια ειδοποίηση με την πληροφορία αυτή. Διαφορετικά, επιστρέφει το τροποποιημένο ζητούμενο αρχείο. Ουσιαστικά, η πρόσθεση αυτού του πεδίου έγινε για λόγους βελτίωσης της απόδοσης του λειτουργίας του πρωτοκόλλου. Χωρίς το πεδίο αυτό, πρέπει να σταλλούν δυο αιτήσεις, για την υλοποίηση του *υπό συνθήκην* GET. Μια με την μέθοδο HEAD για να αποκτήσουμε μεταπληροφορία για το αρχείο (και έτσι να διαπιστώσουμε τον χρόνο της τελευταίας τροποποίησης του) και σε περίπτωση που το αρχείο έχει τροποποιηθεί, ακόμα μια αίτηση με την μέθοδο GET για να αποκτηθεί το ίδιο το αρχείο.
- **User-Agent**: παρέχει πληροφορία για το λογισμικό (WWW browser) που χρησιμοποιεί ο πελάτης για να στείλει την αίτηση και να λάβει την απάντηση.

A.2 Γενική περιγραφή απάντησης

Αφού ο WWW server λάβει και επεξεργαστεί την αίτηση, απαντά στον πελάτη με την μορφή μιας HTTP απάντησης:

Απάντηση	=	Απλή απάντηση	|	Σύνθετη απάντηση

Απλή απάντηση = [Ζητούμενο αρχείο]
Σύνθετη απάντηση = Επικεφαλίδα απάντησης
Πεδία απάντησης
[Ζητούμενο αρχείο]

Η επικεφαλίδα της απάντησης αποτελείται από την έκδοση του HTTP πρωτόκολλου, ακολουθούμενη από έναν κωδικό αριθμό απάντησης και μια επεξήγηση του προηγούμενου αριθμού :

Επικεφαλίδα = HTTP-Version Code Phrase

Ο κωδικός αριθμός απάντησης είναι ένας τριψήφιος ακέραιος αριθμός ο οποίος δείχνει το αποτέλεσμα της επεξεργασίας της αίτησης του πελάτη από τον εξυπηρετητή. Το πρώτο ψηφίο καθορίζει και το είδος της απάντησης:

- **1***XX*: Δεν χρησιμοποιείται ακόμη,θα χρησιμοποιηθεί μελλοντικά.
- **2***XX*: Η αίτηση λήφθηκε και επεξεργάστηκε επιτυχώς από τον εξυπηρετητή.
- **3***XX*: Ανακατεύθυνση. Το ζητούμενο αρχείο βρίσκεται σε κάποιον άλλον WWW server.
- **4***XX*: Λάθος αίτησης. Η αίτηση του πελάτη πιθανόν να περιέχει συντακτικά λάθη και δεν μπορεί να εξυπηρετηθεί από τον WWW server.
- **5***XX*: Ο server απέτυχε να εξυπηρετήσει την αίτηση εξαιτίας ενός δικού του σφάλματος.

A.2.1 Πεδία απάντησης

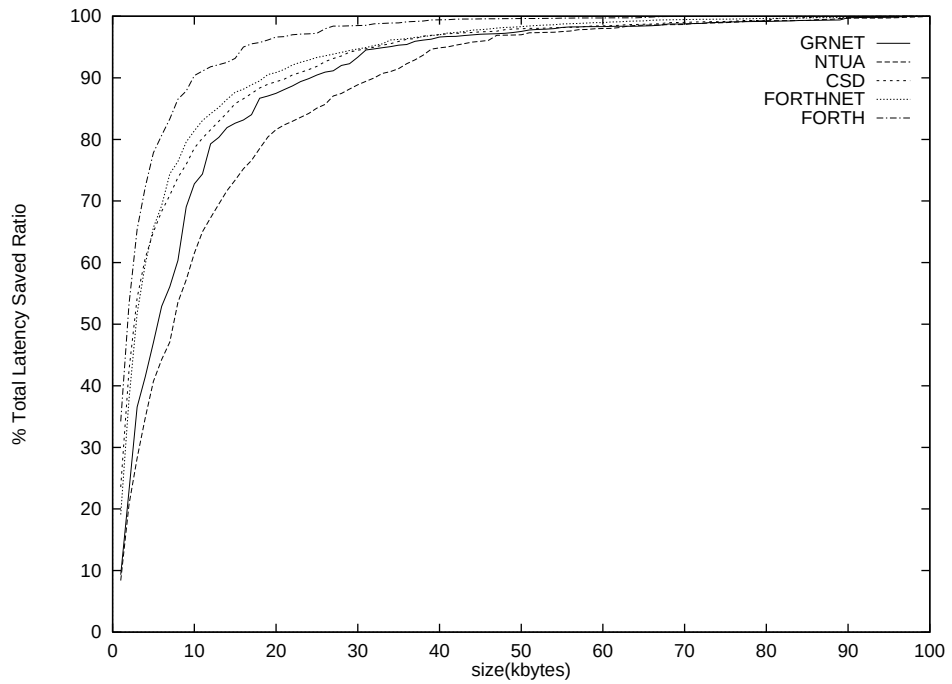
Χωρίζονται σε δυο κατηγορίες:

- Αυτά που περιέχουν επιπλέον πληροφορία σχετικά με την απάντηση και δεν μπορούν να τοποθετηθούν στην επικεφαλίδα της. Τα κυριότερα είναι:
  - **server**: παρέχει πληροφορία για το λογισμικό που χρησιμοποιήθηκε από τον server για να εξυπηρετήσει την αίτηση.
  - **WWW-Authenticate**: για να πληροφορήσει τον πελάτη ότι απαιτείται κάποια πιστοποίηση για να προσπελάσει το αρχείο.
- Αυτά που περιέχουν πληροφορία σχετικά με το ζητούμενο αρχείο που επιστρέφει ο server στον πελάτη. Τα κυριότερα στην κατηγορία αυτή είναι:
  - **Content-Encoding**: κωδικοποίηση που έχει υποστεί το αρχείο.

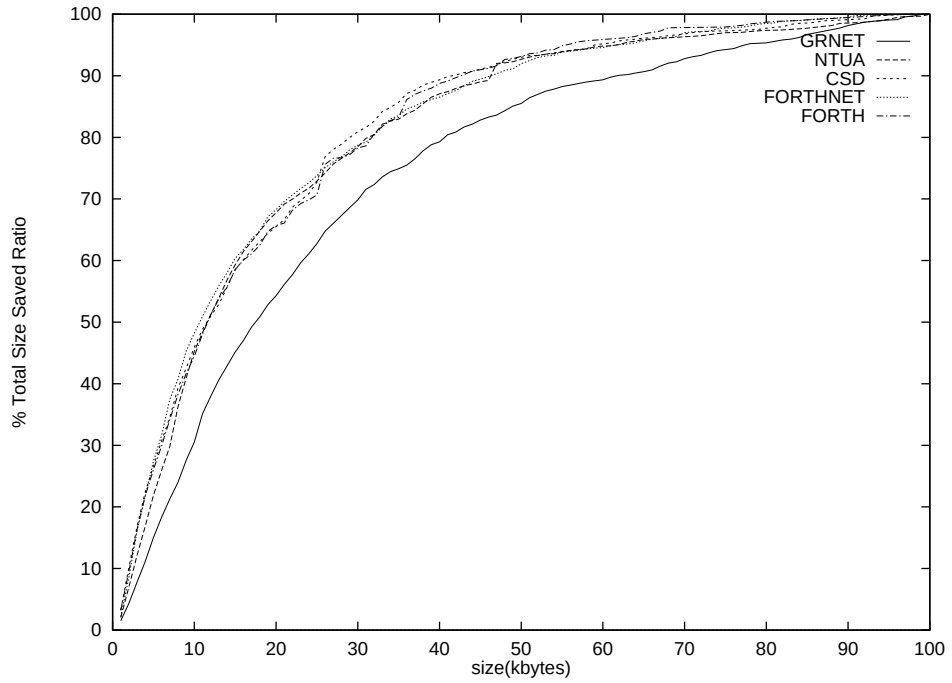
-- **Content-Length**: το μέγεθος της απάντησης (δηλαδή του ζητούμενου αρχείου).
-- **Expires**: ημερομηνία μέχρι την οποία δεν θα τροποποιηθεί το αρχείο.
-- **Last-Modified**: ημερομηνία τελευταίας τροποποίησης του αρχείου.

Παράρτημα Β

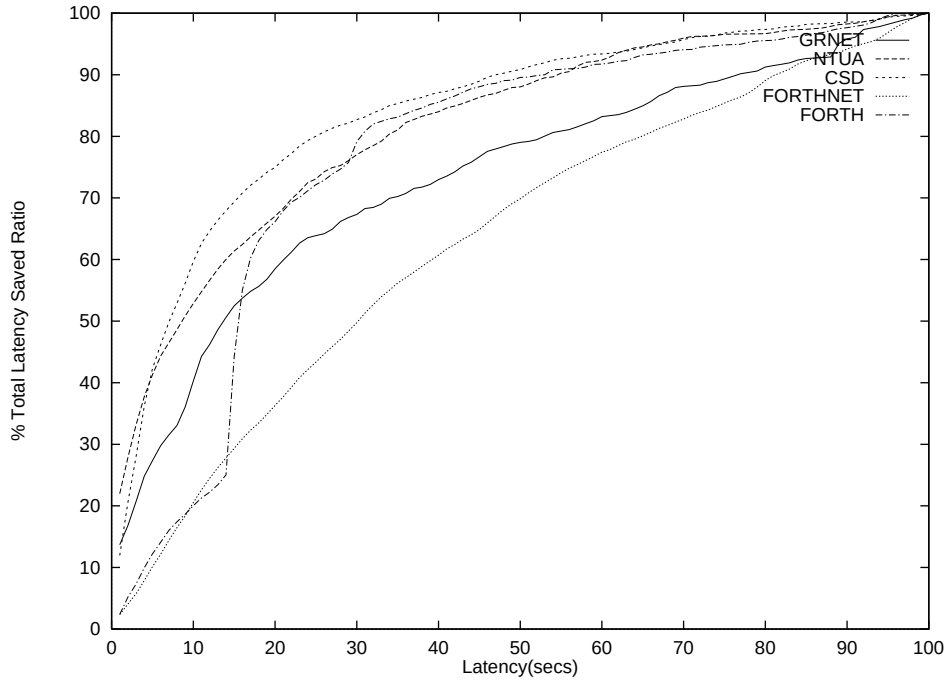
Επιπλέον μετρήσεις στο WWW



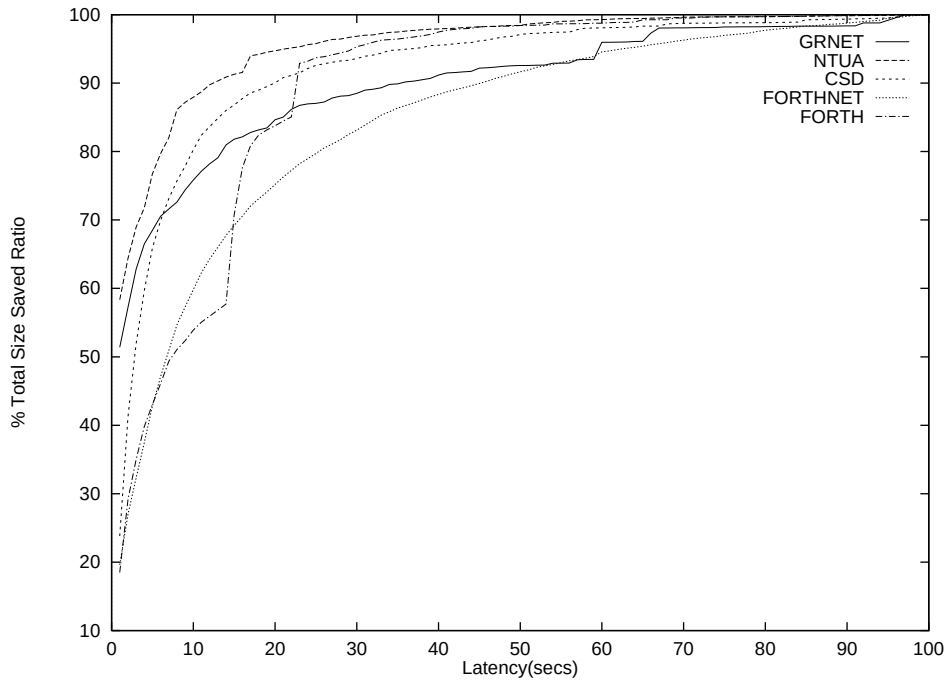
Σχήμα Β.1: Γραφική παράσταση όπου απεικονίζεται η σχέση του μέγεθους των αρχείων με το Latency Saved Ratio. Τα μικρά σε μέγεθος αρχεία βελτιώνουν σταθερά το LSR ενώ δεν παρατηρείται η ίδια σταθερότητα όσο μεγαλώνει το μέγεθος



Σχήμα Β.2: Γραφική παράσταση όπου απεικονίζεται η σχέση του μέγεθους των αρχείων με το Size Saved Ratio. Και στην περίπτωση αυτή, τα μικρά σε μέγεθος αρχεία βελτιώνουν το SSR αλλά με αργό ρυθμό



Σχήμα Β.3: Γραφική παράσταση όπου απεικονίζεται η σχέση του Latency των αρχείων με το Latency Saved Ratio. Τα αρχεία με μικρό χρόνο απόκτησης είναι πολύτιμα για το LSR αλλά δεν αρκούν από μόνα τους



Σχήμα Β.4: Γραφική παράσταση όπου απεικονίζεται η σχέση του Latency των αρχείων με το Size Saved Ratio. Τα αρχεία με μικρό χρόνο απόκτησης συγκεντρώνουν το μεγαλύτερο ποσοστό του SSR

Παράρτημα C

Ορολογία

Caching
Τεχνική η οποία στηρίζεται στην σχετικότητα και την τοπικότητα των αιτήσεων. Τα πολυτιμότερα αρχεία (πχ. τα αρχεία που ζητούνται συχνότερα) αποθηκεύονται στον δίσκο της cache. Η εξυπηρέτηση αιτήσεων για αρχεία που βρίσκονται στην cache γίνεται κατευθείαν από αυτήν.

FTP
Συντομογραφία του File Transfer Protocol. Ενα πρωτόκολλο το οποίο χρησιμοποιείται για την μεταφορά αρχείων στο Internet.

HIT
Ο αλγόριθμος που προσπαθεί να αυξήσει το ποσοστό των αιτήσεων που εξυπηρετούνται από την cache του proxy server (δηλαδή το HR).

HR
Συντομογραφία του Hit Ratio. Είναι το ποσοστό των αιτήσεων που εξυπηρετούνται από την cache του proxy server.

HTTP protocol
Το πρωτόκολλο που χρησιμοποιείται για τις συναλλαγές των WWW servers και clients.

HTTP request
Η αίτηση που στέλνει ο WWW client σε κάποιον WWW server (ή proxy server) προκειμένου να ζητήσει κάποιο αρχείο.

HTTP response
Η απάντηση που λαμβάνει ο WWW client από τον WWW server ή κάποιον proxy server.

LAT
Ο αλγόριθμος που προσπαθεί να βελτιώσει τον μέσο χρόνο εξυπηρέτησης των αιτήσεων από τον proxy server (δηλαδή το LSR).

LFU
Ο αλγόριθμος Least Frequently Used για την αντικατάσταση αρχείων στην cache, κάθε

φορά που απαιτείται ελεύθερος χώρος. Πολυτιμότερο είναι το αρχείο με την μεγαλύτερη ζήτηση.

LNC-R-W3
Ο αλγόριθμος διαχείρισης της cache για την βελτιστοποίηση του μέσου χρόνου εξυπηρέτησης των αιτήσεων.

LRU
Ο αλγόριθμος Least-Recently Used για την αντικατάσταση αρχείων στην cache, κάθε φορά που απαιτείται ελεύθερος χώρος. Πολυτιμότερο είναι το αρχείο το οποίο ζητήθηκε πιο πρόσφατα.

LRU-MIN
Τροποποίηση του αλγορίθμου LRU για την διαχείριση της cache.

LRU-THOLD
Τροποποίηση του αλγορίθμου LRU για την διαχείριση της cache.

LSR
Συντομογραφία του Latency Saved Ratio. Είναι η βελτίωση του μέσου χρόνου εξυπηρέτησης των αιτήσεων των χρηστών.

Network Proxy Coordinator
Ενα πρόγραμμα το οποίο αναλαμβάνει την συνεργασία των proxy servers.

PER
Ο αλγόριθμος διαχείρισης της cache που αποτιμά τα χαρακτηριστικά των αρχείων.

Proxy Server
Ενα πρόγραμμα το οποίο ασκεί μεσολαβητικό ρόλο στις συναλλαγές των WWW servers και clients. Οι αιτήσεις των clients λαμβάνονται από τον proxy server ο οποίος πορσπαθεί αρχικά να τις εξυπηρετήσει από την cache του. Διαφορετικά, αναζητά το αρχείο στον WWW server και το επιστρέφει πίσω στον client.

Replication
Τεχνική η οποία αντιγράφει τα αρχεία των servers (πχ. FTP, WWW servers) σε άλλους servers που βρίσκονται πιο κοντά στους χρήστες.

SSR
Συντομογραφία του Size Saved Ratio. Είναι η ποσότητα της πληροφορίας που ζητείται από τον proxy server και εξυπηρετείται από την cache του.

URL
Ενας συντακτικός τρόπος για να εντοπίσουμε κάποιο αντικείμενο (συνήθως αρχείο) στο WWW. Συντάσσεται ως:

`http`: //*host*[: *port*] [*abs_path*]

όπου http το πρωτόκολλο που χρησιμοποιείται, host και port είναι η διεύθυνση του WWW server και abs_path το όνομα του ζητούμενου αντικειμένου (συνήθως κάποιο όνομα αρχείου).

WWW

Συντομογραφία του World-Wide-Web (Παγκόσμιος Δικτυακός Ιστός). Συγχωνευεί όλα τα συστήματα απόκτησης πληροφορίας του Internet με ένα ενιαίο τρόπο.

WWW client

Ο τελικός χρήστης του WWW.

WWW server

Ενα πρόγραμμα το οποίο δέχεται και εξυπηρετεί τις αιτήσεις των WWW clients για αντικείμενα (συνήθως αρχεία).

Βιβλιογραφία

[ASA+95] Marc Abrams, Charles R. Standridge, Ghaleb Abdulla, Stepphe Williams, and Edward A. Fox. Caching proxies: limitations and potentials. In 4th International World-Wide-Web Conference, Boston, December 1995.

[BBM+] M. Baentcsh, L. Baum, G. Molter, S. Rothkugel, and P. Sturm. Caching and replication in the world wide web or a european perspective on www connectivity. *AG Systemsoftware, University of Kaiserslautern.*

[BBMSRK] Michael Baentsch, Lotar Baum, Georg Molter, and Peter Sturm Steffen Roth Kugel. Enhancing the web's infrastructure-from caching to replication. *AG Systemsoftware, University of Kaiserslautern.*

[BC95] Hars-Werner Braun and Kimberly C. Claffy. Web traffic characterization: an assessment of the impact of caching documents from ncsa's web server. *Computer Networks and ISDN systems*, pages 37--51, 1995.

[BCG+95] Azer Bestavros, Robert L. Carter, Mark E. Grovella, Carlos R. Cunha, Abdelsalam Heddaye, and Sulaiman A. Mirdad. Application level document caching in the internet. In Proceedings of the Second International Workshop on Services in Distributed and Networked Environments (SDNE'95), June 1995.

[Bes95] Azer Bestavros. Using speculation to reduce server load and service time on the WWW. In Proceedings of CIKM'95: The Fourth ACM International Conference on Information and Knowledge Management, November 1995.

[Bes96] Azer Bestavros. Speculative data dissemination and service to reduce server load, network traffic and service time. In Proceedings of ICDE'96: The 1996 International Conference on Data Engineering, March 1996.

[BH96] Jean-Chrysostome Bolot and Philipp Hoschka. Performance engineering of the World Wide Web: application to dimensioning and cache design. In Proceedings of fifth international world wide web conference, 1996.

[BLFF96] T. Berners-Lee, R. Fielding, and H. Frystyk. *Hypertext Transfer protocol HTTP/1.0.* HTTP Working group, February 1996.

[BLM+97] M. Baentcsh, A. Laufer, G. Molter, S. Rothkugel, and P. Sturm. Quantifying the overall impact of caching and replication in the web. *Computer Science Department, University of Kaiserslautern*, February 1997.

[Bra] Tim Bray. Measuring the web. *Computer Networks and ISDN systems*, 28.

[CBC95] Carlos R. Cunha, Azer Bestavros, and Mark E. Crovella. Characteristics of www client based traces. Tech report BU-CS-95-010, Boston university, Computer Science Department, 1995.

[CDCNW95] Anawat Chankhunhod, Peter B. Danzig, Michael I. Schwartz Chuck Neerdaels, and Kurt J. Worrel. A hierarchical internet object cache. Tech report CU-CS-766-95, Department of computer science, University of Colorado-Boulder, March 1995.

[Cor96] Andrew Cormack. Web caching. *University of Wales, Cardiff*, September 1996.

[DP96] Adam Dingle and Tomas Partl. Web cache coherence. Fifth International World Wide Web Conference, May 1996.

[FGM+96] R. Fielding, J. Gettys, J.C Mogul, H. Frystyk, and T. Berners-Lee. *Hypertext Transfer protocol HTTP/1.1.* HTTP Working group, 1996.

[Fla] A.J. Flavell. WWW cache briefing. *Glasgow University, PPE Group.*

[Gen] Genesys. World Wide Web caching -- the application level view of the internet. *Compute Science Department, University of Kaiserslautern.*

[Gra96] Matthew Gray. Internet statistics: Web growth, internet growth, 1996. http://www.mit.edu/people/mkgray/net/web-growth-summary.html.

[Hug93] Kevin Hughes. Entering the world wide web: a guide to cyberspace. *Honolulu Community College*, September 1993.

[KBM] Eric Dean Katz, Michelle Buttler, and Robert McGrath. A scalable http server. the ncsa prototype. Technical report.

[KsA] A.M Kermarrec, M. Van steen, and A.Tanenbaum. A flexible alternative to web caching. *Dept. of Math & Computer Science, Vrije University.*

[LA94] Ari Luotonen and Keven Altis. World wide web proxies. *Computer networks and ISDN systems*, 27(2), 1994.

[LBL] Ari Luotonen and T. Berners-Lee. *CERN httpd reference manual. A guide to a World Wide Web hypertext daemon.* CERN.

[Mar96] E. Markatos. Main memory caching of web documents. In Proceedings of fifth international world wide web conference, 1996.

[MC96] Evangelos Markatos and Catherine Chronaki. A top-10 approach to prefetching on the web. Technical Report FORTH-ICS/TR-173, August 1996.

[MLB95] Radhika Malpani, Jacob Lorch, and David Berger. Making world wide web cachign servers cooperate. In proceedings of 4th Iternational World-Wide-Web Conference, Boston, December 1995.

[PJM95] Venkata N. Padmanabhan, Jeffrey, and C. Mogul. Improving http lattency. Technical Report 95-875, University Of California at Berkeley/CSD, 1995.

[PR94] James E. Pitkow and Margaret M. Recker. A simple yet robust caching algorithm based on dynamic access patterns. In Electronic Proc. of the 2nd WWW Conference, 1994.

[Sed94] Jeff Sedayao. Mosaic will kill my network. studying network traffic patterns of mosaic use. In Electronic Proceedings of the Second World Wide Web Conference, Chicago, Illinois, October 1994.

[Smi94] Neil Smith. What can archives offer the world wide web? *The University of Kent at Canterbury*, April 1994.

[Smi95] Neil G. Smith. Guidelines for the installation of a World Wide Web proxy cache. *The University of Kent at Canterbury*, December 1995.

[SsV96a] Peter Scheuermann, Junho shim, and Redek Vigralek. A data warehouse intelligent cache manager. In *Proceedings of the 22nd VLDB Conference, Bombay*, 1996.

[SSV96b] Peter Scheuermann, Junho Shim, and Radek Vingralek. A case for delay conscious caching of web documents. *Fifth International World Wide Web Conference*, 1996.

[WA96] Roland P. Wooster and Marc Abrams. Proxy caching that estimates page load delays. 1996.

[WAS+96] Stephen Williams, Marc Abrams, Charles R. Standridge, Ghaleb Abdulla, and Edward Fox. Removal policies in network caches for world wide web documents. ACM SIGCOMM Conference, Stanford University, August 1996.

[Wes95] Duane Wessels. Intelligent caching for world wide web objects. INET 95 Proceedings, University of Colorado, 1995.