ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μια Ομότιμη Αρχιτεκτονική για Ανταλλαγή Περιεχομένου σε Υπερκείμενα Δίκτυα

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Αθανάσιος-Δημήτριος Κ. Σωτηρίου

Αθήνα, Ιούνιος 2008

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μια Ομότιμη Αρχιτεκτονική για Ανταλλαγή Περιεχομένου σε Υπερκείμενα Δίκτυα

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Αθανάσιος-Δημήτριος Κ. Σωτηρίου

Συμβουλευτική Επιτροπή: Νικόλαος Μήτρου, επιβλέπων
Ευστάθιος Συκάς
Μιχαήλ Θεολόγου

Εγκρίθηκε από την επταμελή εξεταστική επιτροπή την 19η Ιουνίου 2008

.....	.....	.....
Ν. Μήτρου	Ε. Συκάς	Μ. Θεολόγου
Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ
.....		.....
Γ. Στασινόπουλος		Ι. Βασιλείου
Καθηγητής ΕΜΠ		Καθηγητής ΕΜΠ
.....		.....
Β. Λούμος		Π.Δεμέστιχας
Καθηγητής ΕΜΠ		Αναπ. Καθηγητής. Πανεπ. Πειραιώς

Αθήνα, Ιούνιος 2008

.....

Αθανάσιος-Δημήτριος Κ. Σωτηρίου
Διδάκτωρ Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Στην διατριβή αυτή παρουσιάζεται μια πρωτότυπη ομότιμη αρχιτεκτονική, η οποία παρέχει στους χρήστες τη δυνατότητα ανταλλαγής περιεχομένου. Μέσα από μια αναλυτική θεωρητική και βιβλιογραφική αναφορά στα ομότιμα συστήματα και πιο συγκεκριμένα σε αρχιτεκτονικές συστημάτων που βασίζονται σε κατανεμημένους πίνακες κατατεμαχισμού (Distributed Hash Tables - DHT) πάνω σε υπερκείμενα δίκτυα (Overlay Networks), αναπτύσσεται αρχικά το τεχνολογικό υπόβαθρο της εργασίας αυτής. Το σύστημα ακολουθεί τις βασικές αρχές των συστημάτων DHT και πιο συγκεκριμένα ακολουθεί τις βασικές αρχές που εισήγαγε το δίκτυο Kademlia διαφοροποιώντας όμως ως προς τη χρήση K-δέντρων για την δημιουργία ενός καλά οργανωμένου δικτύου με σταθερό πίνακα γειτόνων, το οποίο έχει δυνατότητα δρομολόγησης σε λογαριθμικό αριθμό βημάτων. Η χρήση των K-δέντρων παρέχει στο σύστημα τη δυνατότητα γρήγορης δρομολόγησης αλλά και ανεκτικότητα ως προς την παρουσίαση σφαλμάτων στο δίκτυο (λόγω αποχώρησης κόμβων ή διακοπών στην επικοινωνία).

Η αρχιτεκτονική που παρουσιάζεται βασίζεται στην χρήση ενός πίνακα γειτόνων που επιτρέπει τόσο την οριζόντια όσο και την κάθετη μετακίνηση στην δενδρική δομή. Μέσα από τη διατριβή αυτή αναλύεται η δομή της αρχικής αρχιτεκτονικής και παρουσιάζονται αναλυτικά οι βασικές διαδικασίες που παρέχει το σύστημα, αυτές της εισαγωγής, δημοσίευσης, αναζήτησης, εξόδου και επιδιόρθωσης. Στην συνέχεια, μέσα από την παρουσίαση των αποτελεσμάτων μιας σειράς εξομοιώσεων που έγιναν, αποδεικνύεται η ορθότητα της προτεινόμενης σχεδίασης. Πιο συγκεκριμένα, επεκτείνοντας τον εξομοιωτή neurogrid με βάση το αναλυτικό μοντέλο, αποδεικνύεται αρχικά ότι το σύστημα είναι σε θέση να δρομολογεί σε λογαριθμικό αριθμό βημάτων ενώ παράλληλα διατηρεί τον πίνακα γειτόνων σταθερό και τον αριθμό των συνολικών μηνυμάτων που ανταλλάσσονται στο δίκτυο πολύ χαμηλό.

Εν συνεχεία παρουσιάζεται μια σειρά από επεκτάσεις στους βασικούς αλγορίθμους που αυξάνουν την ανεκτικότητα σε σφάλματα του δικτύου και βελτιώνουν την απόδοσή του. Η επίπτωση αυτών των αλλαγών γίνεται εμφανή μέσα από τα αποτελέσματα μιας ακόμα σειράς εξομοιώσεων, όπου παρουσιάζεται επίσης η λειτουργία του συστήματος σε περιβάλλον μαζικών αποχωρήσεων κόμβων (δίχως τη χρήση μηχανισμού επιδιόρθωσης).

Για να βελτιωθεί ακόμα περισσότερο η συμπεριφορά του δικτύου, εισάγεται αρχικά ένας αλγόριθμο επιδιόρθωσης, ο οποίος επιτυγχάνει την αλλαγή της συμπεριφοράς του δικτύου κατά την μαζική αποχώρηση κόμβων,

καθώς πλέον η μείωση των επιτυχών αναζητήσεων από λογαριθμική γίνεται γραμμική ως προς το ποσοστό σφαλμάτων. Η βελτίωση αυτή γίνεται ακόμα πιο αισθητή κατά την εισαγωγή δύο επεκτάσεων, αυτών των σχημάτων αντιγράφων και των εικονικών δικτύων. Η πρώτη επέκταση επιτρέπει τη δημοσίευση και αναζήτηση περιεχομένου με τη χρήση επιπλέον συναρτήσεων κατατεμαχισμού, ενώ η δεύτερη εισάγει την έννοια των εικονικών δικτύων, όπου κάθε κόμβος ανήκει σε περισσότερα από ένα υπερκείμενα δίκτυα. Η χρήση αυτών των επεκτάσεων σε συνδυασμό με τον αλγόριθμο επιδιόρθωσης επιτρέπει στο δίκτυο να δρομολογεί επιτυχώς ακόμα και όταν το ποσοστό κόμβων που σφάλουν φτάνει το 80% του συνολικού πληθυσμού.

Αφού αναλυθεί διεξοδικά η επίπτωση αυτών των αλλαγών, παρουσιάζεται μια πιο γενική αρχιτεκτονική, που δίνει τη δυνατότητα παραμετροποίησης με βάση την απόδοση και την κατανομή του φόρτου κίνησης. Στην αρχιτεκτονική αυτή δίδεται η δυνατότητα αλλαγής τόσο του μεγέθους του K-δέντρου όσο και του τρόπου δρομολόγησης (από καθαρά κατανεμημένο σε υβριδικό ή πλήρως ιεραρχικό). Η αλλαγή αυτή δίνει τη δυνατότητα χρήσης του δικτύου σε διαφορετικές εφαρμογές, ανάλογα με τις απαιτήσεις όσο αφορά την ομοιομορφία στην κατανομή του φόρτου κίνησης (σε όλα τα επίπεδα του K-δέντρου) και τη βέλτιστη δρομολόγηση. Αφού αρχικά παρουσιαστεί αναλυτικά το νέο σύστημα καθώς και όλες οι βασικές λειτουργίες, μέσα από μια σειρά εξομοιώσεων παρατίθεντο η απόδοσή του καθώς και η επιβεβαίωση των δυνατοτήτων παραμετροποίησης που παρέχει. Τέλος, αφού έχει αναλυθεί το νέο αυτό σύστημα, η διατριβή καταλήγει σε μια σειρά συμπερασμάτων και μελλοντικών ερευνητικών ζητημάτων.

Λέξεις Κλειδιά

Ομότιμη αρχιτεκτονική, ανταλλαγή περιεχομένου, κατανεμημένος πίνακας κατατεμαχισμού, υπερκείμενα δίκτυα, Κ-δέντρα, αλγόριθμος δρομολόγησης, ανεκτικότητα σε σφάλματα, σχήματα αντιγράφων, εικονικά δίκτυα, παραμετροποίηση

Overview

In this thesis a novel peer-to-peer architecture is presented, which provides users the capability to exchange content. Through an analytical theoretical and bibliographical study in peer-to-peer networks, and more specifically in architectures for systems based on Distributed Hash Tables (DHT) on top of Overlay Networks, it firstly describes the technological background of the study. The system follows the basic principals of DHT systems and more specifically those introduced by the Kademlia network, differentiating itself through the use of a K-ary tree structure for the production of a well defined and organized network with constrained number of neighboring nodes, which allows routing in logarithmic steps. The use of K-ary trees provides the system with the capability of both efficient routing and tolerance to network errors (due to node departure or communication failure).

The architecture presented is based on the use of a neighborhood table which allows both the horizontal and vertical movement in the tree structure. Through this study the structure of the original architecture is analyzed and analytically presented for all of the basic functions that the system provides, namely those of insertion, publication, searching, departure and repair. It is then proven, through the results of a number of simulations that have been performed, the validity of the proposed design. More specifically, by extending the neurogrid simulator with the analytic model, it is initially proven that the system is able to route in logarithmic steps while retaining both a constant number of neighbors and a low total of network traffic (messages).

A number of extensions is then present in addition to the basic algorithms that increases the system's tolerance to errors and its efficiency. The effect of these changes becomes even more evident through the results of another series of simulations, where the functionality of the system in environments with massive node failures (without the use of a repair mechanism) is presented.

In order to further better the performance of the network, a repair algorithm is introduced, which is able to change the system's behavior during massive node failures and reduce the drop of successful inquires from logarithmic to linear, with respect to node failure percentage. This improvement becomes even more apparent with the inclusion of two additional extensions, those of replication schemas and virtual networks. The first one allows the publication and retrieval of content through the use of additional hash functions, while the second one introduces the concept of

virtual networks, where each node belongs to more than one overlay network. The use of these two extension combined with the repair algorithm allows the network to successfully route even when the percentage of failing nodes surpasses that of 80% of the total node population.

Once the effect of all these changes has been analyzed, a more general architecture is introduced, which provides the capability of varying the network's structure and behavior according to its performance and distribution of network load. In this architecture the capability of changing both the size of the K-ary tree and the routing behavior (from purely distributed to hybrid or pure hierarchical) is provided. This change offers the capability of using the network for different application domains, depending on the requirements with regards to work load distribution (across all levels of the K-ary tree) and optimal routing. Once the new system and all of its basic functionalities are presented in full detail, the results of a number of simulations are provided as means of assessing its performance and validating the variability it provides. Finally, once the new system has been thoroughly analyzed, a number of conclusions and open research issues are provided.

Keywords

Peer-to-peer architecture, content exchange, distributed hash table, overlay networks, K-ary trees, routing algorithm, tolerance to errors, replication schemas, virtual networks, parameterization

Στους γονείς μου και στον αδερφό μου

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΠΙΝΑΚΑΣ ΣΧΗΜΑΤΩΝ

ΠΙΝΑΚΑΣ ΠΙΝΑΚΩΝ

1 ΕΙΣΑΓΩΓΗ

Τις τελευταίες δεκαετίες η σημαντική βελτίωση των τηλεπικοινωνιακών υποδομών οδήγησε στην διεύρυνση του Διαδικτύου και στην καθιέρωσή του ως περιβάλλοντος οικονομικής και κοινωνικής δραστηριότητας. Καθώς η πρόσβαση σε αυτό έχει εξελιχθεί από απλές τηλεφωνικές συνδέσεις, στις αρχές της προηγούμενης δεκαετίας, σε ευρυζωνικές συνδέσεις υψηλής ταχύτητας και μόνιμης σύνδεσης, παράλληλα έχουν εξελιχθεί και οι εφαρμογές που υποστηρίζονται από αυτό. Η απλή πρόσβαση σε ιστοσελίδες θεωρείται πλέον ένα μικρό μέρος των δυνατοτήτων που παρέχει το Διαδίκτυο. Οι χρήστες έχουν πλέον τη δυνατότητα να χρησιμοποιούν το Διαδίκτυο ως μέσο εμπορικών συναλλαγών, πνευματικών αναζητήσεων, κοινωνικών συναναστροφών αλλά και κέντρο ψυχαγωγίας.

Η χρήση του Διαδικτύου ως ψυχαγωγικού μέσου έχει αναπτυχθεί σημαντικά τα τελευταία χρόνια, καθώς οι ταχύτητες πρόσβασης σε αυτό παρέχουν στον χρήστη τη δυνατότητα ανταλλαγής οπτικοακουστικού περιεχομένου (πολυμεσικού), είτε σε στατική (αποθηκευμένη) είτε σε δυναμική (ζωντανή μετάδοση) μορφή. Ακόμα και η πηγή παροχής τέτοιου περιεχομένου έχει αλλάξει. Ενώ στο παρελθόν ο χρήστης βασιζόταν σε κάποιο πάροχο περιεχομένου για την απόκτηση πληροφοριών και πολυμεσικού περιεχομένου, σήμερα έχει τη δυνατότητα να παρέχει ο ίδιος περιεχόμενο στον υπόλοιπο κόσμο. Η αρχή έγινε με το Napster [1], όπου η συλλογή του περιεχομένου μετατέθηκε στους ίδιους τους χρήστες, με αποτέλεσμα τη δημιουργία της μεγαλύτερης βάσης μουσικού περιεχομένου που υπήρξε ποτέ και συνεχίζεται με την εξέλιξη του Διαδικτύου με επίκεντρο πλέον τη διαδραστικότητα και τη συμμετοχή των χρηστών ως ενεργούς δημιουργούς πληροφορίας, γνωστό ως Web2.0 [2].

1.1. Το πρόβλημα

Καθώς οι δυνατότητες των τερματικών συσκευών των χρηστών του Διαδικτύου αυξάνονται διαρκώς, τόσο σε επεξεργαστική ισχύ όσο και σε αποθηκευτικό χώρο και σε εύρος ζώνης, όλο και περισσότερα προγράμματα άρχισαν να εκμεταλλεύονται τις δυνατότητες αυτές ώστε να διανείμουν τις λειτουργίες τους. Η ιδέα αυτή οδήγησε στην ανάπτυξη των ομότιμων

συστημάτων (Peer-to-Peer, P2P [3]), όπου οι χρήστες αποτελούν ταυτόχρονα το ίδιο το σύστημα. Τα συστήματα αυτά έχουν αναπτυχθεί ιδιαίτερα τα τελευταία χρόνια και ένας από τους βασικότερους τομείς δραστηριότητάς τους είναι αυτός της ανταλλαγής περιεχομένου [4].

Οι αυξημένες δυνατότητες των χρηστών έχουν οδηγήσει αυτά τα συστήματα στην ανταλλαγή από απλά μουσικά αρχεία μέχρι ολόκληρες κινηματογραφικές ταινίες, προγράμματα και γενικότερα περιεχόμενο με διαρκώς αυξανόμενο όγκο. Το γεγονός αυτό έχει δημιουργήσει την ανάγκη σχεδίασης συστημάτων που να αντέχουν τον αυξημένο αριθμό χρηστών (που πλέον υπολογίζεται σε τάξη εκατομμυρίων) και τον ακόμη μεγαλύτερο όγκο περιεχομένου. Είναι λοιπόν απαραίτητη η ανάπτυξη μιας αρχιτεκτονικής που θα οργανώνει κατάλληλα τους χρήστες και θα παρέχει τη δυνατότητα δημοσίευσης αλλά και εύρεσης περιεχομένου με επιτυχία και δίχως να υπερφορτώνει το ίδιο το δίκτυο.

Τη λύση σε αυτό το πρόβλημα προσπαθούν να δώσουν τα συστήματα που χρησιμοποιούν αλγορίθμους κατανεμημένων πινάκων κατατεμαχισμού (Distributed Hash-Tables, DHT [5]) βασισμένα σε υπερκείμενα δίκτυα. Οι χρήστες των συστημάτων αυτών δημιουργούν ένα δικό τους δίκτυο πάνω από το επίπεδο εφαρμογής (υπερκείμενο) το οποίο οργανώνεται με βάση κάποιο αναγνωριστικό, που ορίζεται σε κάθε χρήστη. Το αναγνωριστικό, τόσο για το χρήστη όσο και για το περιεχόμενό του, παράγεται από μια συνάρτηση κατατεμαχισμού και στην συνέχεια κάθε χρήστης είναι υπεύθυνος για μια σειρά από περιεχόμενα, με βάση κάποιον χώρο αναγνωριστικών. Μια σειρά από τέτοια συστήματα έχουν προταθεί, χωρίς όμως κάποιο να έχει εδραιωθεί, καθώς υπάρχει ακόμη αρκετή ερευνητική δραστηριότητα.

1.2. Αντικείμενο της διατριβής

Το αντικείμενο της παρούσας διατριβής είναι η πρόταση μιας νέας αρχιτεκτονικής που θα παρέχει τη δυνατότητα ανταλλαγής περιεχομένου στους χρήστες, και η οποία βασίζεται στην δημιουργία ενός υπερκείμενου δικτύου με βάση μια συνάρτηση κατατεμαχισμού. Η αρχιτεκτονική που θα παρουσιάσουμε έχει τη δυνατότητα να οργανώνει πλήρως τους χρήστες χωρίς να επιβαρύνει το σύστημα με σημαντική επιπρόσθετη κίνηση, και να παρέχει τη δυνατότητα στους χρήστες να δημοσιεύουν και να αναζητούν περιεχόμενο. Στην διάρκεια της διατριβής εξετάσαμε και θα παρουσιάσουμε στην συνέχεια τόσο τη θεωρία των ομότιμων συστημάτων όσο και την τρέχουσα ερευνητική δραστηριότητα στο συγκεκριμένο χώρο. Θα αναλύσουμε τα κριτήρια και τα

συμπεράσματα που μας οδήγησαν στην επιλογή της συγκεκριμένης αρχιτεκτονικής, την οποία στην συνέχεια θα παρουσιάσουμε αναλυτικά.

Εν συνεχεία θα παρουσιάσουμε τα αποτελέσματα της εξομοίωσης που αναπτύξαμε για την επιβεβαίωση της ορθότητας της αρχιτεκτονικής μας αλλά και τον έλεγχο της απόδοσής της. Μέσα από τη διατριβή αυτή θα παρουσιάσουμε πλήρως την προτεινόμενη αρχιτεκτονική και θα αποδείξουμε μέσα από θεωρητική και πειραματική μελέτη τη λειτουργικότητά της και τις δυνατότητες που παρέχει.

Κατά την ανάπτυξη της αρχιτεκτονικής μας εντοπίσαμε επίσης διάφορα χαρακτηριστικά και τεχνικές που θα μπορούσαν να βελτιώσουν περεταίρω το σύστημά μας. Αυτές οι διαπιστώσεις οδήγησαν στην ανάπτυξη μια σειράς βελτιώσεων, τις οποίες αναλύσαμε τόσο θεωρητικά όσο και πρακτικά μέσω του εξομοιωτή μας. Οι επεκτάσεις αυτές δίνουν τη δυνατότητα στο σύστημά μας να λειτουργεί αποδοτικότερα και κυρίως κάτω από πιο αντίξοες συνθήκες. Επιπλέον, η αναλυτική επεξεργασία των αποτελεσμάτων των εξομοιώσεων μας οδήγησε στον εντοπισμό διαφόρων σημείων και χαρακτηριστικών της αρχιτεκτονικής που θα μπορούσαν να βελτιωθούν. Ως αποτέλεσμα, κατά το δεύτερο μισό αυτής της διατριβής θα παρουσιάσουμε μια βελτιωμένη αρχιτεκτονική, η οποία βασίζεται μεν στην αρχική σχεδίαση αλλά έχει αρκετές διαφοροποιήσεις και επεκτάσεις ώστε να επιλύει τα προβλήματα που εντοπίστηκαν στην πορεία.

Η νέα αυτή αρχιτεκτονική παρουσιάζεται και αναλύεται διεξοδικά, τόσο θεωρητικά όσο και πρακτικά μέσω μια νέας σειράς αποτελεσμάτων εξομοίωσης. Η νέα αυτή αρχιτεκτονική εισάγει επίσης την έννοια της παραμετροποίησης στην προηγούμενη αρχιτεκτονική, γενικεύοντας έτσι τα προηγούμενα αποτελέσματα και δίνει τη δυνατότητα εξεύρεσης της βέλτιστης λειτουργίας, ανάλογα με τις απαιτήσεις των εφαρμογών που θα βασίζονται σε αυτήν. Στο δεύτερο μισό της διατριβής θα εξετάσουμε επίσης όλες τις παραμέτρους της νέας αυτής αρχιτεκτονικής και θα τεκμηριώσουμε τις αρχικές μας υποθέσεις, πως δηλαδή αυτές οι αλλαγές βελτιώνουν την απόδοση του συστήματος και επιλύουν τις όποιες ελλείψεις εντοπίσαμε κατά την ανάλυση του αρχικού συστήματος.

1.3. Η δομή της εργασίας

Η εργασία που παρουσιάζουμε είναι δομημένη σε 11 κεφάλαια, το πρώτο από τα οποία αποτελεί η παρούσα γενική εισαγωγή στο θέμα της εργασίας. Στο δεύτερο κεφάλαιο θα αναπτύξουμε τα σημαντικότερα θέματα που αφορούν τα ομότιμα συστήματα, πώς αυτά ορίζονται, ποια τα

χαρακτηριστικά τους και σε ποιες κατηγορίες μπορούν να διαχωριστούν. Επίσης θα εξετάσουμε μερικά βασικά ζητήματα, που αφορούν στη σωστή σχεδίαση ομότιμων συστημάτων και τα οποία εξασφαλίζουν την επιτυχία τους.

Στο τρίτο κεφάλαιο θα επικεντρωθούμε στα ομότιμα συστήματα που προσφέρουν δυνατότητα ανταλλαγής περιεχομένου και θα παρουσιάσουμε τη σημαντικότερη ερευνητική δραστηριότητα στον τομέα αυτό. Εν συνεχεία θα αναλύσουμε μερικά βασικά θέματα, που πρέπει να επιλύσουν τα συστήματα αυτά και τα οποία έπρεπε να εξετάσουμε κατά την σχεδίαση της αρχιτεκτονικής μας.

Στο τέταρτο κεφάλαιο γίνεται η βασική παρουσίαση της προτεινόμενης αρχιτεκτονικής. Παραθέτουμε αρχικά το στόχο αλλά και τους βασικούς λόγους επιλογής της συγκεκριμένης σχεδίασης και εν συνεχεία παρουσιάζουμε αναλυτικά την αρχιτεκτονική μας. Στο κεφάλαιο αυτό δίνεται ο τρόπος δημιουργίας του συστήματος από τους χρήστες και οι βασικές λειτουργίες που παρέχονται σε αυτούς. Παράλληλα με την περιγραφή δίνονται και οι αλγόριθμοι των βασικών λειτουργιών αλλά και οι πίνακες που πρέπει να διατηρούν οι χρήστες.

Στο πέμπτο κεφάλαιο παρουσιάζονται τα αρχικά πειράματα που διεξήγαμε για τον έλεγχο της αρχιτεκτονικής αλλά και της απόδοσής της. Αρχικά παρουσιάζουμε τον εξομοιωτή που αναπτύξαμε για τις ανάγκες των πειραμάτων μας και στην συνέχεια παραθέτουμε τα αποτελέσματα μιας σειράς εξομοιώσεων, τα οποία επιβεβαιώνουν όλες τις λειτουργίες που περιγράφονται στο τέταρτο κεφάλαιο.

Στο έκτο κεφάλαιο αναλύουμε κάποιους επιπρόσθετους μηχανισμούς της αρχιτεκτονικής μας για την αντιμετώπιση προβλημάτων κατά τη λειτουργία του συστήματος. Παρουσιάζουμε τις αλλαγές που απαιτούνται στους αλγορίθμους των βασικών λειτουργιών ώστε το σύστημα να είναι ανεκτικό σε αποτυχίες κόμβων και στην συνέχεια παραθέτουμε και σχολιάζουμε τα αποτελέσματα μιας σειράς εξομοιώσεων με βάση τις παραπάνω τροποποιήσεις.

Στο έβδομο κεφάλαιο παρουσιάζουμε τον αλγόριθμο επιδιόρθωσης, που επιτρέπει στην αρχιτεκτονική μας να επιδιορθώνει απότομες αποχωρήσεις κόμβων (σφάλματα κόμβων) ώστε να διατηρείται ομαλή η λειτουργία ακόμα και κάτω από αντίξοες συνθήκες. Παράλληλα με τον αλγόριθμο, παρουσιάζουμε και τα αποτελέσματα μιας σειράς εξομοιώσεων που επιβεβαιώνουν την σωστή λειτουργία του αλγορίθμου επιδιόρθωσης και τα πλεονεκτήματα που επιφέρει.

Στο όγδοο κεφάλαιο εισάγουμε δύο επεκτάσεις στην αρχιτεκτονική μας, αυτήν των σχημάτων αντιγράφων και των εικονικών δικτύων. Οι δύο αυτές επεκτάσεις επιτρέπουν τη βελτίωση της απόδοσης του συστήματός μας και την αύξηση της ανεκτικότητάς του σε σφάλματα. Όλα τα παραπάνω επιβεβαιώνονται μέσα από μια επιπλέον σειρά εξομοιώσεων.

Στο ένατο κεφάλαιο δίνεται αναλυτική περιγραφή του συστήματος K-Umbrella, μια επέκταση του αρχικού συστήματος με βάση τις παρατηρήσεις και τα μειονεκτήματα που εντοπίσαμε στην αρχική μας αρχιτεκτονική. Η νέα αυτή αρχιτεκτονική μας επιτρέπει να παραμετροποιούμε κάποια χαρακτηριστικά του συστήματος και προσφέρει βέλτιστη απόδοση με βάση τις ανάγκες των υπερκείμενων εφαρμογών. Στο κεφάλαιο αυτό δίνονται αναλυτικά η δομή της αρχιτεκτονικής και όλες οι βασικές λειτουργίες.

Στο δέκατο κεφάλαιο παρουσιάζουμε τα αποτελέσματα των εξομοιώσεων που εκτελέσαμε με βάση την καινούργια αρχιτεκτονική K-Umbrella. Τα αποτελέσματα αυτά επιβεβαιώνουν τη θεωρητική ανάλυση που έγινε στο προηγούμενο κεφάλαιο και παρουσιάζουν την επίδραση των δύο βασικών παραμέτρων στην απόδοση και ανεκτικότητα του δικτύου.

Τέλος, στο ενδέκατο κεφάλαιο δίνονται μια σειρά από συμπεράσματα για την αρχιτεκτονική μας, με βάση τη μελέτη που παρουσιάστηκε κατά τη διάρκεια της διατριβής αυτής και παραθέτουμε κάποια ανοιχτά ερευνητικά θέματα που μπορούν να μελετηθούν περαιτέρω.

1.4. Αναφορές

[1] Napster, www.napster.com

[2] O'Reilly T., “What Is Web 2.0 Design Patterns and Business Models for the Next Generation of Software,” 09/30/2005

[3] Wikipedia, http://en.wikipedia.org/wiki/Peer-to-peer

[4] Antoniadis, P., Courcoubetis, C. , “Market Models for P2P Content Distribution,” AP2PC'02, Bologna, Italy, July, 2002.

[5] Ratnaswamy, S., Shenker, S., Stoica, I., “Routing Algorithms for DHTs: Some Open Questions," in First International Workshop on Peer-to-Peer Systems, March, 2002.

2 ΟΜΟΤΙΜΑ ΣΥΣΤΗΜΑΤΑ

Με την εμφάνιση εφαρμογών που βασίζονταν σε ομότιμα συστήματα για την ανταλλαγή αρχείων η χρήση τέτοιων συστημάτων διαδόθηκε ταχύτατα και υπήρξε ερευνητική άνθηση στον τομέα αυτό. Εφαρμογές όπως το Napster [1], που επέτρεπε τη διάδοση μουσικών κυρίως αρχείων , έφεραν τα ομότιμα συστήματα στο επίκεντρο της επιστημονικής κοινότητας αλλά και ολόκληρου του υπόλοιπου κόσμου καθώς πέρα από την τεχνολογική επιτυχία του συστήματος, αυτό που κυρίως κατάφερε ήταν να αλλάξει τη νοοτροπία όσον αφορά τη χρήση του Διαδικτύου ως ψυχαγωγικού μέσου. Μέσω της δωρεάν ανταλλαγής μουσικών αρχείων ,οι χρήστες είχαν τη δυνατότητα να παρακάμψουν τις πολυεθνικές δισκογραφικές εταιρείες και τη νομοθεσία περί πνευματικών δικαιωμάτων και να έχουν πρόσβαση σε έναν τεράστιο κατάλογο από μουσικές επιλογές, το μέγεθος του οποίου εξαρτιόταν από τους ίδιους τους χρήστες.

2.1. Ιστορική αναδρομή

Η επιτυχία του Napster βασίστηκε στο γεγονός ότι μετέθεσε τη διατήρηση των δεδομένων (μουσικών αρχείων) από έναν κεντρικό διαχειριστή στους ίδιους τους χρήστες και απλά το σύστημα τους καθοδηγούσε στην εύρεση του χρήστη που κατείχε τα δεδομένα που αναζητούσαν. Μειονέκτημά του αποτέλεσε αυτή ακριβώς η ανάγκη λειτουργίας ενός κεντρικού διαχειριστή για την εύρεση και καθοδήγηση των χρηστών. Στο γεγονός αυτό βασίστηκαν οι δισκογραφικές εταιρείες και τελικά οδήγησαν και στον τερματισμό της εφαρμογής μέσω νομικών διώξεων [2], καθώς ο νομικός κάτοχος του κεντρικού διαχειριστή (δηλαδή η εταιρεία που λειτουργούσε το Napster) θεωρήθηκε υπεύθυνος.

Αυτό που οδήγησε το Napster να γίνει μια από τις πιο πετυχημένες εφαρμογές όλων των εποχών δεν ήταν ότι βασίστηκε σε ομότιμη αρχιτεκτονική για να προωθήσει μια καινοτομική τεχνολογία αλλά ότι χρησιμοποίησε την ομότιμη τεχνολογία ως εργαλείο για να πετύχει αυτό που ζητούσαν οι χρήστες. Ήταν δηλαδή μια εφαρμογή που αναπτύχθηκε με επίκεντρο τις ανάγκες των χρηστών και για αυτό κατάφερε να επιτύχει. Το γεγονός αυτό το τονίζουμε διότι είναι σημαντικό να καθορίζονται πρώτα οι

ανάγκες ενός συστήματος και στην συνέχεια να καθορίζονται οι τεχνολογίες που θα χρησιμοποιηθούν. Για παράδειγμα, συστήματα μηχανών αναζήτησης ,όπως το Google [3] ή το Lycos [4], λειτουργούν σαφώς αποδοτικότερα μέσω της κλασσικής τεχνολογίας πελάτη-διαχειριστή.

Επιπλέον, το Napster ήταν από τις πρώτες εφαρμογές που επωφελήθηκαν από τη διαγραφόμενη (πλέον δεδομένη) άνοδο των πολυμεσικών εφαρμογών και τη χρήση του υπολογιστή ως ψυχαγωγικού μέσου. Η εμφάνιση της κωδικοποίησης MPEG-1 Audio Layer 3 (γνωστή ως mp3) [5] έδωσε τη δυνατότητα στους χρήστες να αρχειοθετούν στους υπολογιστές τους ένα μεγάλο αριθμό μουσικών κομματιών και το Napster εκμεταλλεύτηκε αυτό το γεγονός για τη δημιουργία της μεγαλύτερης βάσης πολυμεσικών δεδομένων στην εποχή του.

Σε αντίθεση με το Napster, μεταγενέστερα συστήματα όπως το Gnutella [6], μετέφεραν όλες τις λειτουργίες στον χρήστη, με αποτέλεσμα να μην υπάρχει καμία κεντρική οντότητα που να μπορεί να θεωρηθεί υπεύθυνη. Το Gnutella επιπλέον έδινε τη δυνατότητα ανταλλαγής κάθε είδους περιεχομένου, από μουσικά αρχεία έως ταινίες και προγράμματα, με αποτέλεσμα να διαδοθεί ακόμη περισσότερο.

Με την εμφάνιση του Gnutella,το όποιο δεν αποτελεί εφαρμογή αλλά ένα πρωτόκολλο που χαρακτηρίζει όλες τις εφαρμογές που βασίζονται σε αυτό, έχουμε τη δημιουργία ενός νέου Διαδικτύου βασιζόμενου στο υπάρχον και γνωστού ως υπερκείμενου δικτύου. Σε αυτό το δίκτυο, κάθε χρήστης επικοινωνεί με τους υπόλοιπους μέσω μηνυμάτων (ping και pong) και είναι ο ίδιος υπεύθυνος τόσο για την εύρεση όσο και την παροχή αρχείων και δεδομένων. Με άλλα λόγια, ο κάθε κόμβος στο δίκτυο του Gnutella παίζει τόσο το ρόλο του χρήστη όσο και του διακομιστή.

Τα ομότιμα όμως συστήματα προϋπήρχαν του Napster, του Gnutella και των υπόλοιπων παρόμοιων εφαρμογών και μπορεί κανείς να υποστηρίξει ότι αποτελούσαν το Διαδίκτυο στην πρωταρχική του μορφή, προτού διαδοθεί η αρχιτεκτονική πελάτη-εξυπηρετητή. Στο αρχικό Διαδίκτυο, ο κάθε υπολογιστής επικοινωνούσε μέσω τηλεφωνικής γραμμής απευθείας με έναν άλλο υπολογιστή και αντάλλασσαν δεδομένα με ομότιμο τρόπο. Ακόμα και βασικές εφαρμογές όπως το telnet και το ftp, αρχικά λειτουργούσαν με μια ομότιμη μορφή καθώς ο κάθε υπολογιστής άλλοτε λειτουργούσε ως πελάτης και άλλοτε ως εξυπηρετητής . Ο τρόπος αυτός επικοινωνίας ήταν γνωστός ως ένας – προς – έναν (one-to-one).

Μια από τις πρώτες εφαρμογές που βασίστηκε σε ομότιμη αρχιτεκτονική ήταν το Usenet, όπου οι υπολογιστές επικοινωνούσαν μεταξύ τους και αντάλλασσαν τις καινούργιες ανακοινώσεις μέσω του πρωτοκόλλου

UUCP (Unix-to-Unix-Copy-Protocol) [7]. Ακόμα και το Domain Name System (DNS) [8] βασίζεται σε μια μορφή ομότιμης τεχνολογίας, καθώς κάθε κόμβος μπορεί είτε να αναζητά κάποιον άλλον είτε να προσφέρει πληροφορίες, δρώντας έτσι τόσο ως πελάτης όσο και ως εξυπηρετητής.

Με την ξαφνική έκρηξη όμως του Διαδικτύου το 1994 και μετά ,χάρις κυρίως στην εμπορική επιτυχία του παγκόσμιου ιστού και του ηλεκτρονικού ταχυδρομείου, οι αυξημένες ανάγκες και απαιτήσεις των χρηστών οδήγησαν στην εντατική και σχεδόν αποκλειστική χρήση συστημάτων πελάτη / εξυπηρετητή. Αυτά τα συστήματα μπορούσαν να διαχειριστούν καλύτερα τις χιλιάδες αιτήσεις που γίνονταν από τους χρήστες και με τη χρήση ακριβών μηχανημάτων-εξυπηρετητών να αναβαθμίζουν διαρκώς την ποιότητα υπηρεσίας τους. Εξάλλου, οι περισσότερες εφαρμογές εκείνης της περιόδου, κυρίως η περιήγηση σε ιστοσελίδες, ήταν από τη φύση τους σχεδιασμένες να εξυπηρετούν πολλούς χρήστες από μια κεντρική μονάδα διαχείρισης και παροχής πληροφοριών.

Αυτό που οδήγησε στην χρήση και πάλι των ομότιμων συστημάτων και κατά κάποιο τρόπο την επιστροφή στα αρχικά συστήματα του Διαδικτύου ήταν η αλλαγή των αναγκών και απαιτήσεων των χρηστών. Από απλοί δέκτες πληροφοριών, οι σημερινοί χρήστες απαιτούν να μπορούν να συμμετέχουν στην διάδοσή τους και να συνεισφέρουν συνειδητά στην εξάπλωσή τους. Επιπλέον, η διάδοση των πολυμεσικών εφαρμογών και η χρήση των ηλεκτρονικών υπολογιστών ως ψυχαγωγικών μέσων, σε συνδυασμό με καινούργιες τεχνικές κωδικοποίησης που έχουν κάνει την ψηφιακή μετάδοση και διανομή ψυχαγωγικού υλικού δυνατή ακόμα και σε πραγματικό χρόνο με αρκετά ικανοποιητικά επίπεδα, ανάγκασαν στον επαναπροσδιορισμό των εφαρμογών και τον τρόπο διάδοσης της πληροφορίας.

Σε αυτό συνέβαλλε η τεχνολογική ανάπτυξη των υπολογιστών τόσο σε επίπεδο υλικού όσο και σε επίπεδο δικτυακής υποδομής. Οι σημερινοί υπολογιστές προσφέρουν όχι απλά μεγάλη επεξεργαστική ισχύ και ανεπτυγμένες δυνατότητες παρουσίασης αλλά και πολυμορφικότητα και κινητικότητα. Οι φορητοί υπολογιστές έχουν πλέον κατακτήσει σημαντικό μερίδιο αγοράς ενώ παράλληλα έχουμε την διάδοση ακόμα πιο φορητών συσκευών όπως τα table-pc, τα PDA αλλά ακόμα και έξυπνα τηλέφωνα που έχουν τη δυνατότητα αναπαραγωγής πολυμεσικού περιεχομένου. Αν αυτό συνδυαστεί με την άνθηση των ασυρμάτων δικτύων (WLAN, WMAX, 2.5G , 3G) ,τη διαγραφόμενη ενοποίησή τους στα δίκτυα τέταρτης γενιάς (4G) και την καθιέρωση των γρήγορων ασύμμετρων γραμμών (ADSL) για οικιακή χρήση, είναι σαφές ότι πλέον οι απαιτήσεις των χρηστών έχουν αλλάξει

πλήρως και απαιτούνται καινούργια συστήματα για να καλύψουν τόσο τεχνολογικά όσο και εμπορικά τις νέες αυτές απαιτήσεις.

Είναι λοιπόν σαφές ότι η χρήση των ομότιμων συστημάτων δεν αποτελεί κάποια καινοτομική ιδέα αλλά απλώς επαναπροσδιορίζει μια παλιά τεχνολογία με σύγχρονο τρόπο. Λόγω όμως της εμπορικότητας των ομότιμων εφαρμογών, πολλά σύγχρονα συστήματα αποκαλούνται και προσπαθούν να εμφανίζονται ως ομότιμα χωρίς στην ουσία να εκμεταλλεύονται αυτή την τεχνολογία. Είναι επομένως αναγκαίο να ορίσουμε με σαφήνεια τα ομότιμα συστήματα και δίκτυα τόσο μέσω κάποιον ορισμών όσο , και κυρίως , μέσω των στόχων που θέτουν και των χαρακτηριστικών που τα προσδιορίζουν.

2.2. Ορισμός

Ο πρώτος ορισμός που δόθηκε για τα ομότιμα συστήματα ήταν αρκετά γενικός και προσδιόριζε ένα σύστημα που δεν ήταν πλήρως ομότιμο αλλά εισήγαγε για πρώτη φορά τον όρο που είναι σήμερα ευρέως διαδεδομένος. Η IBM ανέπτυξε ως μέρος του CS/AIX [9] συστήματος μια αρχιτεκτονική που αποκαλούσε Systems Network Architecture (SNA) [10]. Προέκταση αυτής της αρχιτεκτονικής αποτέλεσε το σύστημα Advanced Peer-to-Peer Networking (APPN) [11],[12] , όπου έχουμε και την πρώτη χρήση του όρου. Ο ορισμός που δόθηκε σε αυτό το σύστημα ήταν «μια αρχιτεκτονική δικτύου που υποστηρίζει κατανεμημένη διαχείριση δικτύου, επιτρέποντας την εύκολη ρύθμιση και χρήση του δικτύου, προσφέροντας κεντρική διαχείριση και ευέλικτη διασύνδεση». Ο ορισμός αυτός προσδιορίζει μεν κάποια χαρακτηριστικά των ομότιμων δικτύων αλλά περιέχει και στοιχεία που εντοπίζουμε και σε άλλες αρχιτεκτονικές.

Στην συνέχεια οι Wray , Glauert και Hopper όρισαν το ομότιμο σύστημα στο [13] ως «μια συλλογή από κατανεμημένες πηγές που συνδέονται μέσω του Διαδικτύου», παρέχοντας έναν γενικότερο ορισμό που όμως έπρεπε να προσδιοριστεί σαφέστερα. Ο Singh στο [14] και οι Thomas, Suchter και Rifkin στο [15] προσδιορίζουν τοπολογικά τα ομότιμα συστήματα προσθέτοντας στον παραπάνω ορισμό ότι τα συστήματα αυτά «βρίσκονται στις άκρες του Διαδικτύου».

Ένας σαφέστερος και εκτενέστερος ορισμός δίνεται από τον Kellerer στο [16] όπου ορίζεται το ομότιμο δίκτυο ως «κάθε κατανεμημένη αρχιτεκτονική δικτύου, όπου οι συμμετέχοντες μοιράζονται ένα μέρος των υλικών πηγών τους (επεξεργαστική ισχύ, αποθηκευτικό χώρο, χωρητικότητα δικτυακού διαύλου, εκτυπωτές κ.α.). Αυτές οι κοινές πηγές είναι απαραίτητες ώστε να προσφερθούν ως υπηρεσίες και περιεχόμενο από το δίκτυο (π.χ.

ανταλλαγή αρχείων ή κοινοί χώροι εργασίας). Οι συμμετέχοντες σε ένα τέτοιο δίκτυο είναι επομένως τόσο πάροχοι πηγών (υπηρεσιών και περιεχομένου) όσο και αιτούντες πηγών (υπηρεσιών και περιεχομένου)».

Έχοντας τα παραπάνω υπόψη μπορούμε να ορίσουμε με συνοπτικό τρόπο την ομότιμη αρχιτεκτονική ως εξής :

«Ένα σύστημα μπορεί να χαρακτηριστεί ως ομότιμο όταν η αρχιτεκτονική του εκμεταλλεύεται τις πηγές των χρηστών και κατανέμει τις λειτουργίες στο σύνολο του πληθυσμού του, ώστε να παρέχει υπηρεσίες και περιεχόμενο στους ίδιους».

2.3. Στόχοι

Είναι σημαντικό να δικαιολογήσουμε την επιλογή μιας ομότιμης αρχιτεκτονικής στο προτεινόμενο σύστημά μας. Πολλές είναι οι περιπτώσεις όπου σύγχρονα συστήματα χρησιμοποιούν τέτοιου είδους αρχιτεκτονική λόγω της δημοτικότητάς της και όχι λόγω λειτουργικότητας και σωστής σχεδίασης. Ένα σύστημα δικαιολογεί τη χρήση ομότιμης αρχιτεκτονικής όταν απαιτεί να εκπληρώσει στόχους που επιτυγχάνονται αποκλειστικά ή σαφώς αποδοτικότερα μέσω αυτής της τεχνολογίας. Στη συνέχεια παραθέτουμε τους βασικούς στόχους που θα πρέπει να έχει ένα σύστημα ώστε να προτιμήσει τη χρήση ομότιμης αρχιτεκτονικής.

- **Μείωση ή διανομή κόστους**

  Στις κλασσικές αρχιτεκτονικές πελάτη / εξυπηρετητή το κόστος βαραίνει αποκλειστικά την πλευρά του εξυπηρετητή και μπορεί πολλές φορές να είναι υπερβολικό. Κυρίως για εφαρμογές διανομής πολυμεσικής πληροφορίας, το κόστος αυτό είναι ακόμη μεγαλύτερο καθώς εκτός της αυξημένης ζήτησης για εύρος ζώνης και επεξεργαστική ισχύ υπάρχει πολλές φορές και το κόστος πνευματικών δικαιωμάτων. Μέσω των ομότιμων εφαρμογών το κόστος αυτό μπορεί να μειωθεί δραματικά καθώς επιμερίζεται στους χρήστες του συστήματος. Στην περίπτωση του Napster το κόστος μειώθηκε αρκετά καθώς ο κεντρικός εξυπηρετητής δεν αποθήκευε τα δεδομένα αλλά παρείχε απλώς τη δυνατότητα αναζήτησης. Ακόμα και αυτό το κόστος επιμερίστηκε στην περίπτωση του Gnutella, όπου την αναζήτηση την κάνουν οι ίδιοι οι χρήστες. Στην περίπτωση του SETI@home [17], το κόστος της επεξεργαστικής ισχύος που απαιτούσαν οι πολύπλοκοι υπολογισμοί διανεμήθηκε στους χρήστες του συστήματος. Είναι σημαντικό όμως η διανομή του κόστους να γίνεται με αμεροληψία ώστε το σύστημα να χαρακτηρίζεται από δικαιοσύνη.

- **Βελτιωμένη αξιοπιστία**

  Η έλλειψη κεντρικής διαχείρισης μπορεί εκ πρώτης όψεως να κλονίζει την αξιοπιστία του συστήματος όμως, με τη σωστή χρήση νέων αλγορίθμων, μπορεί να ενισχύσει αυτό ακριβώς το χαρακτηριστικό. Ένα κεντρικό σύστημα περιέχει αρκετά πιθανά σημεία αποτυχίας καθώς όλες οι διεργασίες βασίζονται σε ελάχιστες, σε σχέση με τον αριθμό των χρηστών, κεντρικές μονάδες. Αντίθετα, στα ομότιμα συστήματα υπάρχουν πάντα εναλλακτικές λύσεις για την παροχή αξιοπιστίας, καθώς οι διεργασίες κατανέμονται στο σύνολο των χρηστών (ή τουλάχιστον σε ένα αρκετά μεγάλο ποσοστό).

- **Συνάθροιση πηγών**

  Η συνάθροιση πηγών αποτελεί φυσικό χαρακτηριστικό των ομότιμων συστημάτων καθώς ακολουθούν μια κατανεμημένη αρχιτεκτονική. Κάθε κόμβος παρέχει τις πηγές του στο σύστημα για τη δημιουργία ενός ενιαίου χώρου. Εφαρμογές όπως το SETI@home εκμεταλλεύονται την επεξεργαστική ισχύ των χρηστών για την επεξεργασία δεδομένων. Συστήματα όπως το Napster ,το Gnutella και το Bittorent [18] συναθροίζουν τον αποθηκευτικό χώρο των χρηστών και το διαθέσιμο εύρος ζώνης τους για την αποδοτικότερη διανομή δεδομένων.

- **Αυξημένη αυτονομία**

  Τα ομότιμα συστήματα βασίζονται στους χρήστες για τη σωστή λειτουργία τους με αποτέλεσμα να επιδεικνύουν αυξημένη αυτονομία. Κάθε χρήστης είναι σε θέση να λειτουργεί αυτόνομα και να μην βασίζεται σε έναν κεντρικό εξυπηρετητή. Επιπλέον, ο κάθε χρήστης έχει τη δυνατότητα να διαχειρίζεται το περιεχόμενο των δεδομένων που θα διανέμει ή τις υπηρεσίες που θα προσφέρει καθώς παράλληλα με την ιδιότητα του χρήστη έχει και αυτήν του εξυπηρετητή.

- **Ανωνυμία / απόρρητο**

  Η ανωνυμία και το απόρρητο αποτελούν πολλές φορές ένα από τα σημαντικότερα ζητήματα σε ένα ομότιμο σύστημα. Καθώς οι χρήστες έχουν τη δυνατότητα ανταλλαγής πληροφοριών και δεδομένων είναι απαραίτητο να διατηρείται η ανωνυμία τους. Στα κεντρικοποιημένα συστήματα, η ανωνυμία και το απόρρητο του χρήστη βασίζεται πάντα στην προθυμία και την αξιοπιστία του εξυπηρετητή. Αντίθετα, στα ομότιμα συστήματα όλες οι διεργασίες πραγματοποιούνται τοπικά και ο χρήστης μπορεί να ελέγξει το είδος και το περιεχόμενο των πληροφοριών που παρέχει. Στο FreeNet [19] για παράδειγμα, η ανωνυμία αποτελεί λειτουργικό συστατικό του συστήματος καθώς όλες

οι αιτήσεις και οι απαντήσεις των χρηστών προωθούνται με βάση αλγορίθμους πιθανοτήτων, ώστε ο αποστολέας και ο παραλήπτης να παραμένουν ανώνυμοι.

- **Δυναμικότητα**

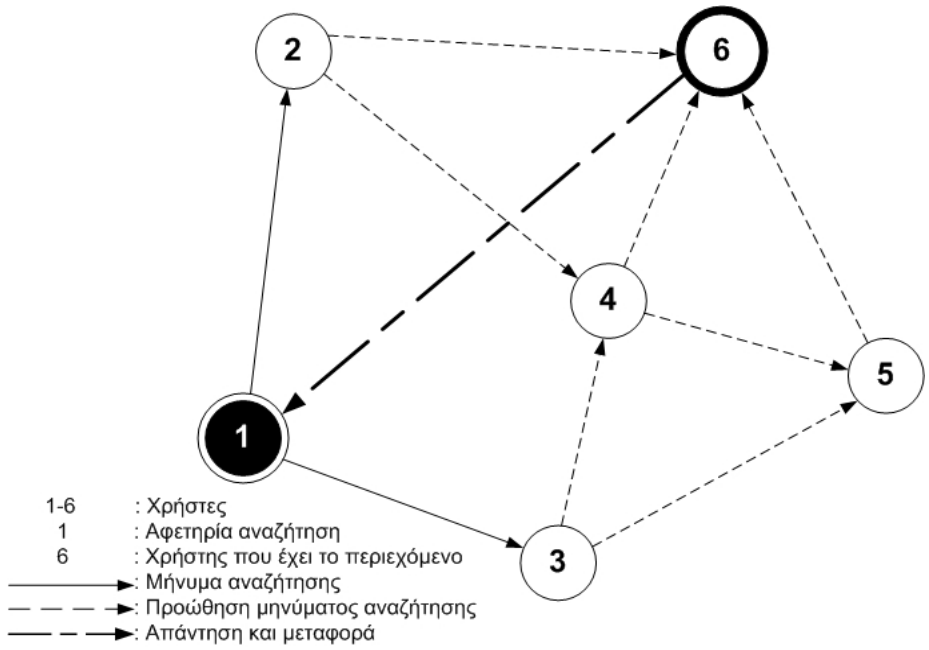
  Στα ομότιμα συστήματα το περιβάλλον είναι ιδιαίτερα δυναμικό. Οι χρήστες και κατ' επέκταση οι πηγές που αυτοί διαθέτουν στο σύστημα μπορούν να εισέρχονται ή να εξέρχονται τυχαία. Τα ομότιμα συστήματα βασίζονται στο σύνολο των χρηστών και όχι σε κάθε έναν ξεχωριστά, με αποτέλεσμα να μπορούν να ανταποκρίνονται δυναμικά στις αλλαγές του συστήματος. Στο Jabber [20] οι χρήστες μπορούν να γνωρίζουν ανά πάσα στιγμή ποιοι χρήστες είναι συνδεδεμένοι και να επικοινωνούν μαζί τους χωρίς να απαιτείται περιοδικός έλεγχος. Στο SETI@home ,το σύστημα έχει την ικανότητα να ελέγχει το σύνολο των χρηστών και να επαναπροσδιορίζει τις επεξεργαστικές λειτουργίες όταν ένας συμμετέχων εγκαταλείψει το δίκτυο χωρίς να έχει εκπληρώσει τη διεργασία του.

2.4. Ομότιμα μοντέλα

Τα συστήματα υπολογιστών μπορούν να διαχωριστούν σε δύο βασικές κατηγορίες, στα συγκεντρωτικά και στα κατανεμημένα. Τα κατανεμημένα διαχωρίζονται περαιτέρω σε συστήματα πελάτη / εξυπηρετητή και στα ομότιμα. Τα πρώτα μπορεί να είναι είτε επίπεδα, όπου έχουμε έναν εξυπηρετητή (και ίσως κάποιους εφεδρικούς), είτε ιεραρχικά, για βελτιωμένη κλιμάκωση. Τα ομότιμα συστήματα μπορούν να διαχωριστούν σε δυο βασικές κατηγορίες ανάλογα με την αρχιτεκτονική τους, σε «καθαρά» και σε «υβριδικά». Η αναφερόμενη ταξινομία παρουσιάζεται στο Σχήμα 1.

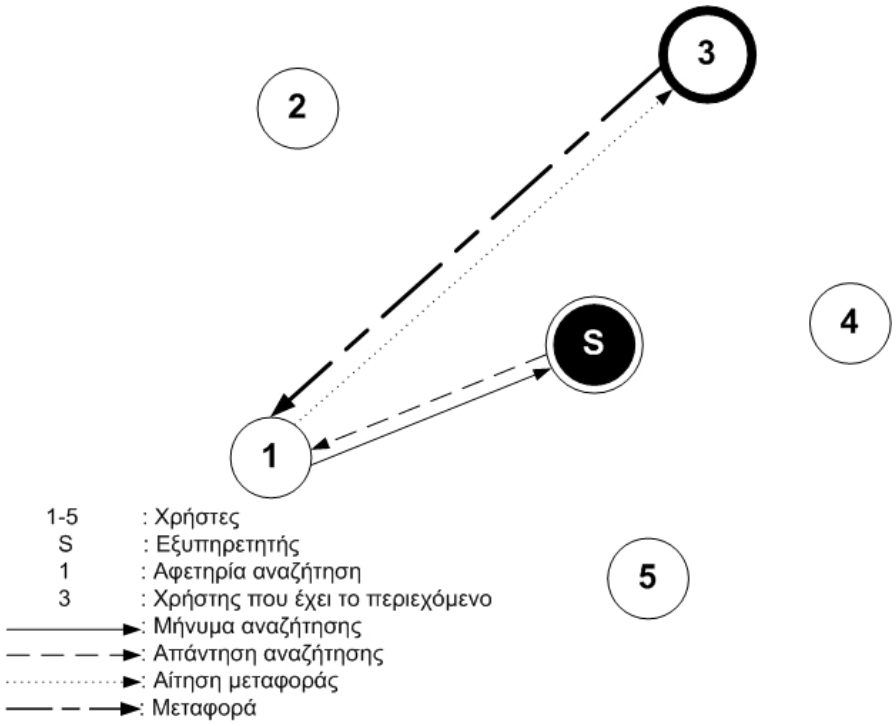
Σχήμα 1. Ταξινομία των συστημάτων υπολογιστών

Αν χρησιμοποιήσουμε τον ορισμό που δίνεται από τον Schollmeier στο [21], μπορούμε αρχικά να ορίσουμε ως «καθαρό» ομότιμο σύστημα «**κάθε ομότιμο σύστημα στο οποίο αν αναιρεθεί οποιαδήποτε μία, αυθαίρετα επιλεγμένη, τερματική οντότητα από το δίκτυο τότε το δίκτυο δεν υφίσταται καμία απώλεια υπηρεσίας**». Η αρχιτεκτονική των «καθαρών» ομότιμων συστημάτων απεικονίζεται στο Σχήμα 2 και προϋποθέτει ότι δεν υπάρχει κάποιος κεντρικός εξυπηρετητής ,που να παρέχει οποιαδήποτε επιπρόσθετη υπηρεσία σε σχέση με τους υπόλοιπους χρήστες. Τέτοιες εφαρμογές αποτελούν το Gnutella και το FreeNet.



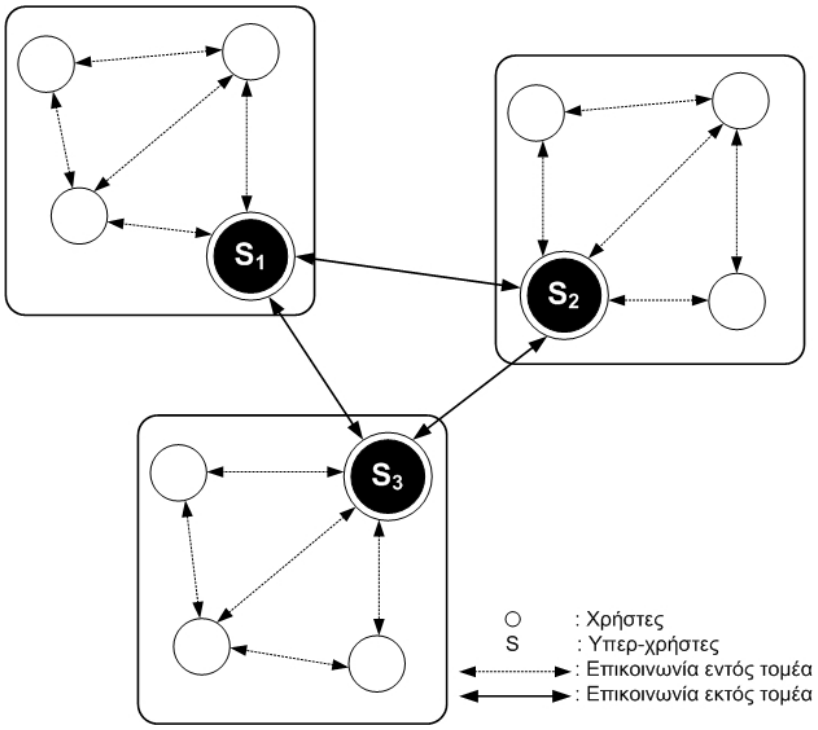
Σχήμα 2. «Καθαρό» ομότιμο μοντέλο

Αντίθετα τα «υβριδικά» ομότιμα συστήματα μπορούν να οριστούν ως « **κάθε ομότιμο σύστημα στο οποίο απαιτείται μια κεντρική οντότητα για την παροχή κάποιων από τις προσφερόμενες υπηρεσίες του δικτύου** ». Στα «υβριδικά» συστήματα έχουμε ένα συνδυασμό των ομότιμων και των πελάτη / εξυπηρετητή συστημάτων ,όπως φαίνεται στο Σχήμα 3, με την αναγκαία ύπαρξη ενός εξυπηρετητή. Κλασσικές εφαρμογές βασισμένες σε «υβριδική» αρχιτεκτονική αποτελούν τα Napster, Groove [22] και Aimster [23].



Σχήμα 3. «Υβριδικό» ομότιμο μοντέλο

Ενδιάμεσες λύσεις έχουν προταθεί τα τελευταία χρόνια ,όπως η χρήση SuperPeers [24]. Αυτή η αρχιτεκτονική βασίζεται στα «καθαρά» συστήματα με τη μετατροπή κάποιων χρηστών σε υπερ-χρήστες, με βάση κάποια κριτήρια (επεξεργαστική ισχύ, αποθηκευτικό χώρο, εύρος ζώνης, αξιοπιστία και διαθεσιμότητα), ώστε να δημιουργείται μια ιεραρχική δομή. Η λύση αυτή, που φαίνεται στο Σχήμα 4, δανείζεται αρκετά χαρακτηριστικά από το DNS και καταφέρνει να περιορίσει τα προβλήματα κλιμάκωσης.



Σχήμα 4. Ιεραρχική δομή με υπερ-χρήστες

2.5. Ταξινόμηση Εφαρμογών

Αν αναλύσουμε περαιτέρω τα παραπάνω μοντέλα μπορούμε να ταξινομήσουμε τις ομότιμες εφαρμογές ανάλογα με το είδος των υπηρεσιών που προσφέρουν. Αρχικά θα διαχωρίσουμε τι εφαρμογές σε τρεις βασικές κατηγορίες σύμφωνα με το [25] και στη συνέχεια θα δούμε πως μπορεί μια ομότιμη εφαρμογή να χαρτογραφηθεί στις πέντε διαστάσεις που προτείνουν οι Kant, Iyer και Tewari στο [26].

Σχήμα 5. Ταξινόμηση ομότιμων εφαρμογών

Μια εφαρμογή μπορεί να ανήκει σε μια από τις τρεις βασικές κατηγορίες , που απεικονίζονται στο Σχήμα 5 : κοινών πόρων , διαχείρισης περιεχομένου , ή συνεργατική.

- **Εφαρμογές Κοινών Πόρων**

  Είναι εφαρμογές που χρησιμοποιούν την επεξεργαστική ισχύ και γενικότερα τους υλικούς πόρους των χρηστών (συνήθως όταν αυτοί είναι ανενεργοί) για την επεξεργασία δεδομένων ή την περάτωση κάποιας λειτουργίας. Η κατηγορία αυτή μπορεί να διαχωριστεί σε δυο υποκατηγορίες, στα συστήματα που απαιτούν έντονη επεξεργασία και σε αυτά που βασίζονται στον πλουραλισμό των μονάδων.

  Στην πρώτη κατηγορία χρησιμοποιούνται οι πηγές των χρηστών για την επίλυση ενός προβλήματος που απαιτεί την επαναληπτική επεξεργασία δεδομένων. Συνήθως οι αλγόριθμοι για την επίλυση του προβλήματος διαχωρίζουν την επίλυση σε ανεξάρτητα τμήματα που ο κάθε χρήστης μπορεί να αναλάβει την επίλυσή τους και στη συνέχεια μια κεντρική πλατφόρμα ενοποιεί τα αποτελέσματα. Η λειτουργία αυτή έχει χαρακτηριστεί ως «ντροπιασμένα παράλληλη» (embarrassingly parallel) [27]. Παραδείγματα τέτοιων εφαρμογών αποτελούν το SETI@home , το Climateprediction.net [28] (που μοντελοποιεί τις αλλαγές στο κλίμα της γης και προσπαθεί να κάνει προβλέψεις) , το fightAIDS@home [29] (που μοντελοποιεί την εξέλιξη της αντίστασης των φαρμάκων για το AIDS και αξιολογεί ενδεχόμενους υποψήφιους για την ανακάλυψη νέων φαρμάκων) και η πλατφόρμα Berkeley Open Infrastructure for Network Computing (BOINC) [30](που αποτελεί μια ανοιχτή πλατφόρμα για τη δημιουργία παρόμοιων ομότιμων εφαρμογών).

  Στην δεύτερη κατηγορία έχουμε τη δημιουργία μιας εικονικής επεξεργαστικής πλατφόρμας ,όπου ο κάθε χρήστης αντί να επαναλαμβάνει τις ίδιες διεργασίες με τους υπόλοιπους χρήστες αλλά με διαφορετικά δεδομένα, όπως γίνεται συνήθως στην πρώτη κατηγορία, αναλαμβάνει μια διαφορετική λειτουργία ανάλογα με το είδος των πηγών που μπορεί να προσφέρει. Παραδείγματα τέτοιων εφαρμογών αποτελούν το JavaBeans [31] και γενικότερα τα Web Services.

- **Εφαρμογές Διαχείρισης Περιεχομένου**

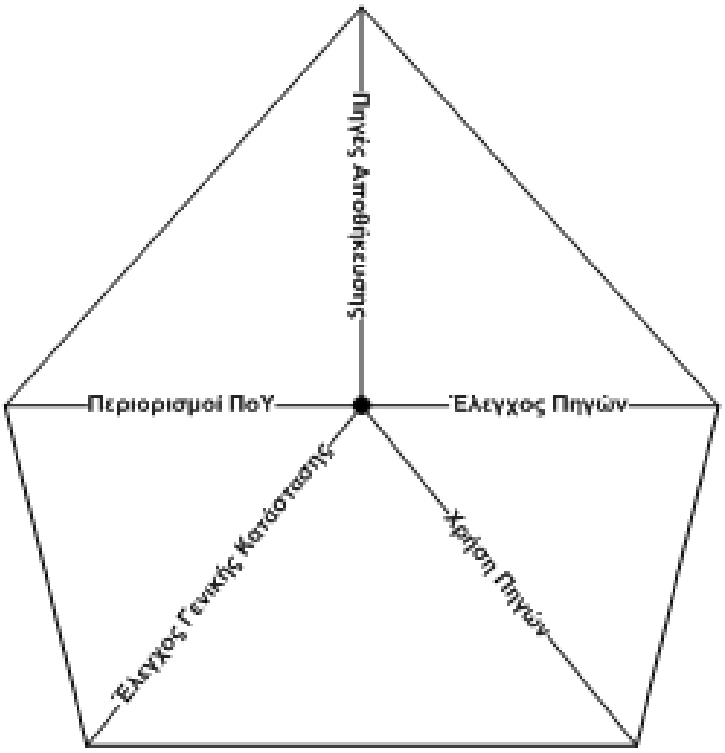
  Οι εφαρμογές αυτής της κατηγορίας επικεντρώνονται στην ανταλλαγή και διάδοση πληροφοριών και αρχείων. Το βασικό μοντέλο είναι αυτό της ανταλλαγής περιεχομένου ,όπως στο Napster και στο Gnutella, το οποίο και έκανε γνωστά τα ομότιμα συστήματα. Ο τρόπος εύρεσης των κατάλληλων δεδομένων διαφέρει ανά εφαρμογή και διάφορα μοντέλα έχουν προταθεί [32],[33],[34],[35],[36]. Μια άλλη κατηγορία αποτελεί αυτή των συστημάτων αρχείων ,όπως στο OceanStore [37] , στο CAN

[38] ή στο GLS [39] ,όπου τα ομότιμα συστήματα χρησιμοποιούνται για την αξιόπιστη αποθήκευση δεδομένων. Μια τελευταία κατηγορία είναι αυτή των μηχανών αναζήτησης και φιλτραρίσματος ,όπως στο OpenCOLA [40] ή στο JXTASearch [41],όπου αναπτύσσονται τεχνικές συλλογικού φιλτραρίσματος των μεταδεδομένων σε ένα ομότιμο δίκτυο.

- **Συνεργατικές Εφαρμογές**

  Η κατηγορία αυτή αναφέρεται σε εφαρμογές που προάγουν την επικοινωνία μεταξύ των χρηστών δίχως την ύπαρξη μιας κεντρικής μονάδας συντονισμού. Μια πρώτη κατηγορία αποτελούν οι εφαρμογές άμεσων μηνυμάτων όπως το Yahoo! [42] και το Jabber [43] ,που επιτρέπουν στους χρήστες να δημιουργούν λίστες από φιλικά τους άτομα, να είναι ενήμεροι για το αν είναι συνδεδεμένοι και να επικοινωνούν μεταξύ τους [44]. Μια άλλη κατηγορία αποτελούν προγράμματα όπως το Groove ή το Buzzpad [45] , που παρέχουν έναν κοινό χώρο εργασίας σε άτομα απομακρυσμένα μεταξύ τους. Σε αυτούς τους χώρους εκτός της δυνατότητας ανταλλαγής μηνυμάτων οι χρήστες έχουν κοινή πρόσβαση σε αρχεία ή πίνακες ώστε να μπορούν να συνεργάζονται. Μια τελευταία κατηγορία αποτελούν τα ομότιμα παιχνίδια όπως η πλατφόρμα NetZ [46] (που επιτρέπει στους δημιουργούς παιχνιδιών να αναπτύξουν δικτυακά παιχνίδια δίχως την επιβάρυνση της διατήρησης ενός κεντρικού εξυπηρετητή) ή η αρχιτεκτονική που προτείνεται από τους Knutsson ,Lu ,Xu και Hopkins για την υποστήριξη Massively Multiplayer Games (MMG) μέσω ομότιμης τεχνολογίας [47].

Εκτός όμως από αυτές τις κατηγορίες, ένα ομότιμο σύστημα μπορεί ,όπως προαναφέραμε, να χαρτογραφηθεί ανάλογα με τα χαρακτηριστικά που παρουσιάζει. Οι πέντε διαστάσεις που χαρακτηρίζουν ένα ομότιμο σύστημα απεικονίζονται στο Σχήμα 6 και αναλύονται περαιτέρω στην συνέχεια. Σε κάθε διάσταση αναφέρουμε τις δύο ακραίες καταστάσεις ενώ , προφανώς, κάθε σύστημα μπορεί να βρίσκεται κάπου ενδιάμεσα.



Σχήμα 6. Οι πέντε διαστάσεις των ομότιμων συστημάτων

- **Πηγές Αποθήκευσης**

  Αυτή η διάσταση αναφέρεται στον τρόπο με τον οποίο αποθηκεύονται τα δεδομένα στο σύστημα. Ένα σύστημα μπορεί να αποθηκεύει τα δεδομένα είτε σε λίγους γνωστούς κόμβους ,οπότε έχουμε μια αρχιτεκτονική παρόμοια με αυτή των εξυπηρετητών στις κλασσικές βάσεις δεδομένων, είτε στο άλλο άκρο να απαιτεί από όλους τους χρήστες να συνεισφέρουν αποθηκευτικό χώρο. Την πρώτη περίπτωση θα την ονομάσουμε «κεντρική» ενώ τη δεύτερη «κατανεμημένη». Στην δεύτερη περίπτωση απαιτείται κάποιος μηχανισμός για την εύρεση της δικτυακής τοποθεσίας στην οποία βρίσκονται τα δεδομένα, συνήθως μέσω της δημιουργίας μεταδεδομένων, καθώς αυτά μπορεί να έχουν αποθηκευτεί σε ένα εύρος κόμβων.

- **Έλεγχος Πηγών**

  Ο άξονας αυτός αποτελεί συνέχεια του προηγούμενου και αναφέρεται στον τρόπο με τον οποίο γίνεται η πρόσβαση στα μεταδεδομένα, που επιτρέπουν στον χρήστη να αναζητήσει τα δεδομένα. Μπορεί πάλι να είναι είτε τελείως «κεντρικά», οπότε υπάρχουν κάποιοι γνωστοί κεντρικοί εξυπηρετητές για την εύρεση ,είτε «κατανεμημένα» ,οπότε κάθε χρήστης προωθεί την αναζήτησή τους στους υπόλοιπους χρήστες εωσότου λάβει απάντηση. Μια μορφή «κεντρικού» ελέγχου πηγών αποτελεί και η ιεραρχική δόμηση των χρηστών. Σε τέτοιου είδους

συστήματα, οι χρήστες ομαδοποιούνται σε τομείς με αποτέλεσμα κάθε φορά η αναζήτηση να γίνεται πρώτα εντός του τομέα και σε περίπτωση μη εύρεσης να προωθείται στον ανώτερο τομέα μέσω ενός διαχειριστή. Με αυτόν τον τρόπο δημιουργείται ένα δέντρο για την αποθήκευση πληροφοριών ελέγχου.

- **Χρήση Πηγών**

  Αφού έχει γίνει η εύρεση των δεδομένων και έχουν αναγνωριστεί οι πηγές στις οποίες έχουν αποθηκευτεί ξεκινάει η διαδικασία ανάκτησής τους. Ο συγκεκριμένος άξονας προσδιορίζει τον τρόπο με τον οποίο γίνεται αυτή η διαδικασία και μπορεί να είναι είτε «απομονωμένα» είτε «συνεργατικά». Στην πρώτη περίπτωση ο χρήστης ανακτά απευθείας τα δεδομένα από μια πηγή , εφόσον πρώτα έχει επικοινωνήσει μαζί της και έχει δημιουργήσει μια σύνοδο με κατάλληλα χαρακτηριστικά. Στην δεύτερη περίπτωση η διαδικασία είναι πιο περίπλοκη καθώς η πηγή μπορεί να συνεργάζεται και με άλλες πηγές για την αποδοτικότερη αποστολή των δεδομένων και να απαιτούνται επιπλέον συντονιστικές ενέργειες, όπως στην περίπτωση της πολυεκπομπής [48] ή στην κωδικοποίηση πολλαπλών στρωμάτων [49].

- **Έλεγχος Γενικής Κατάστασης**

  Η διάσταση αυτή αναφέρεται στις απαιτήσεις των εφαρμογών για έλεγχο της γενικής κατάστασης του δικτύου. Εφαρμογές όπως το Gnutella ή το Freenet δεν απαιτούν τέτοιες γνώσεις καθώς οι χρήστες μπορούν να προσφέρουν ή να αναιρούν πηγές διαρκώς και η προσπάθεια διατήρησης μιας γενικής εικόνας ή συγχρονισμού των πηγών θα ήταν χρονοβόρα και σε πολλές περιπτώσεις αδύνατη. Αντίθετα, εφαρμογές όπως η συνεργατική επεξεργασία απαιτούν μια πιο συντονισμένη προσέγγιση, ώστε να διαδίδονται οι αλλαγές σε όλους τους χρήστες και να υπάρχει ένα υψηλό επίπεδο συνέπειας. Γενικότερα πάντως στις ομότιμες εφαρμογές απαιτείται ο συγχρονισμός και η συνέπεια ,τόσο στα δεδομένα όσο και στα μεταδεδομένα που τα περιγράφουν, να είναι αρκετά πιο ελαστική από ότι στις κλασσικές αρχιτεκτονικές. Αυτό συνήθως παρακάμπτεται μέσω της χρήσης κατάλληλων αλγορίθμων, όπως για παράδειγμα με τη χρήση πολλαπλών αντιγράφων.

- **Περιορισμοί Ποιότητας Υπηρεσίας (ΠοΥ)**

  Ο χρόνος απόκρισης κατά τη φάση της χρήσης των πηγών (ανάκτηση δεδομένων) και γενικότερα η Ποιότητα Υπηρεσίας (ΠοΥ) καθορίζει

πολλές φορές και κατά πόσο ένα ομότιμο σύστημα είναι σε θέση να ικανοποιήσει τις απαιτήσεις της εφαρμογής για την οποία σχεδιάστηκε. Μπορούμε επομένως να διαχωρίσουμε σε αυτόν τον άξονα τις εφαρμογές ανάλογα με τις απαιτήσεις που έχουν σε σχέση με την παρερχομένη ΠοΥ. Στο ένα άκρο έχουμε εφαρμογές όπως το SETI@home όπου δεν έχουμε πρακτικά καθόλου απαιτήσεις και απλά λειτουργεί με βάση την υπηρεσία βέλτιστης προσπάθειας. Στην ενδιάμεση κατάσταση έχουμε εφαρμογές της μορφής αίτησης και απάντησης, όπου απαιτείται μια λογική ΠοΥ και ένας ικανοποιητικά χαμηλός ρυθμός σφαλμάτων / αποτυχιών. Στο άλλο άκρο υπάρχουν οι εφαρμογές με υψηλές απαιτήσεις ΠοΥ, όπως τα συστήματα ρευμάτωσης πολυμεσικής πληροφορίας, όπου απαιτείται γρήγορη απόκριση και ελάχιστη καθυστέρηση και απώλειες.

2.6. Βασικά Ζητήματα και Προβλήματα

Στην συνέχεια θα εξετάσουμε μερικά από τα βασικά ζητήματα που διέπουν τα ομότιμα συστήματα και η επίλυση των οποίων προσδιορίζει σε πολλές περιπτώσεις την επιτυχία του συστήματος και την ορθότητα της επιλεγμένης αρχιτεκτονικής.

- **Αποκέντρωση**

  Ένα από τα σημαντικότερα ζητήματα, που οδήγησαν στην ομότιμη αρχιτεκτονική, αποτελεί η αποκέντρωση. Μπορεί οι κλασσικές εφαρμογές πελάτη / εξυπηρετητή να αποδίδουν αρκετά ικανοποιητικά σε περιπτώσεις όπου απαιτείται κεντρική διαχείριση για την παροχή ασφάλειας και επίβλεψη των συναλλαγών, το κόστος όμως τέτοιων συστημάτων και η μη βέλτιστη αξιοποίηση των πηγών οδηγούν πολλές φορές σε αρνητικές καταστάσεις.

  Τα ομότιμα συστήματα παρέχουν τη δυνατότητα αποκέντρωσης των αρμοδιοτήτων και πλήρη εκμετάλλευση των πηγών. Η πλήρης όμως αποκέντρωση οδηγεί σε άλλα προβλήματα, όπως έλλειψη ολικής γνώσης του συστήματος. Σε υβριδικά συστήματα όπως το Napster, η χρήση ενός κεντρικού καταλόγου επιλύει αυτό το πρόβλημα και περιορίζει την αποκέντρωση στον τρόπο ανάκτησης του περιεχομένου. Αντίθετα, σε καθαρά ομότιμα συστήματα όπως το Gnutella, η αποκέντρωση είναι πλήρης και κανείς δεν έχει ολική γνώση του συστήματος. Ακόμα και για να συνδεθεί κάποιος σε ένα τέτοιο δίκτυο απαιτείται η γνώση κάποιου άλλου χρήστη ή μιας λίστας χρηστών (στατική), μέσω των οποίων θα μπορέσει να καταστήσει γνωστή την

παρουσία του και να ανακτήσει και να αποθηκεύσει σταδιακά στην λανθάνουσα μνήμη του τις IP διευθύνσεις των υπολοίπων χρηστών.

- **Κλιμακωσιμότητα**

  Ένα άμεσο όφελος της αποκέντρωσης αποτελεί η βελτιωμένη κλιμακωσιμότητα, η οποία περιορίζεται από παράγοντες όπως η ανάγκη κεντρικών λειτουργιών (πχ. συγχρονισμό και συντονισμό) και ο όγκος των πληροφοριών που απαιτείται να διατηρείται ενήμερος.

  Το Napster κατάφερε να επιλύσει το πρόβλημα της κλιμακωσιμότητας εναποθέτοντας τη διάδοση των αρχείων μεταξύ των χρηστών, με αποτέλεσμα να υποστηρίξει έως και 6 εκατομμύρια χρήστες σε περιόδους αιχμής. Πιο συγκεκριμένα, στο Napster ένας κεντρικός εξυπηρετητής διατηρούσε καταχωρήσεις για όλα τα αρχεία (ποιός χρήστης έχει κάθε αρχείο) αλλά τα αρχεία παρέμεναν στους ίδιους τους χρήστες και εναπόκειτο σε αυτούς (και όχι στον εξυπηρετητή) να τα ανακτήσουν από τον αντίστοιχο χρήστη απευθείας. Αντίθετα, το SETI@home βασίστηκε στην φύση του προβλήματος που προσπαθούσε να επιλύσει και ενσωμάτωσε την παράλληλη επεξεργασία στην ίδια την αρχιτεκτονική του. Το σύστημα παραλληλοποιούσε τις διεργασίες και κάθε χρήστης παραλάμβανε ένα ανεξάρτητο υποπρόβλημα προς επίλυση. Όποτε ο χρήστης τελείωνε την επεξεργασία απέστελλε τα αποτελέσματα πίσω στο σύστημα (ασύγχρονα). Με τον τρόπο αυτό το SETI@home έχει υποστηρίξει έως και 3.5 εκατομμύρια χρήστες.

  Απαιτείται όμως προσοχή, ώστε η μεγάλη αυτή επέκταση και υποστήριξη χρηστών να μη γίνεται εις βάρος της απόδοσης του συστήματος. Για το λόγο αυτό αναπτύχθηκαν αρκετές υβριδικές λύσεις για ομότιμα συστήματα, όπως αυτή του Napster, που διατηρούν κάποιες λειτουργίες και αρχεία κεντρικοποιημένες.

  Μερικά από τα πρώτα ομότιμα συστήματα όπως το Gnutella και το Freenet βασίστηκαν στην ad-hoc λειτουργία τους με αποτέλεσμα να έχουν αρκετά προβλήματα κλιμακωσιμότητας. Κάθε χρήστης είναι αναγκασμένος να στέλνει τις αιτήσεις του σε πολλαπλούς χρήστες, οι οποίοι διαδίδουν επίσης σε πολλαπλούς χρήστες, με αποτέλεσμα το δίκτυο να υπερφορτώνεται όταν ο αριθμός αυτών είναι μεγάλος. Επιπλέον, πολλές αναζητήσεις επιστρέφουν ως ανεπιτυχείς ακόμα και όταν υπάρχει στο σύστημα η πληροφορία που ζητηθεί, καθιστώντας το σύστημα μη ντετερμινιστικό.

  Για την επίλυση αυτών των προβλημάτων, αναπτύχθηκαν νέα ομότιμα συστήματα όπως το CAN, το Chord και άλλα που θα παρουσιάσουμε

λεπτομερώς στο επόμενο κεφάλαιο, τα οποία διασκορπίζουν τα κλειδιά (που χαρακτηρίζουν και επιτρέπουν την αναζήτηση ενός αρχείου/περιεχόμενου) σε κόμβους μέσω μιας συνάρτησης κατατεμαχισμού. Με αυτόν τον τρόπο, κάθε αντικείμενο μπορεί να ευρεθεί εφόσον είναι εφικτή η επαφή με τον αντίστοιχο κόμβο. Όλοι οι κόμβοι μαζί δημιουργούν ένα υπερκείμενο δίκτυο, στο οποίο διατηρούν πληροφορίες για έναν περιορισμένο αριθμό άλλων κόμβων του συστήματος, μειώνοντας έτσι τον όγκο της πληροφορίας που πρέπει να διατηρείται επίκαιρη και αυξάνοντας επομένως την κλιμακωσιμότητα του συστήματος. Αυτά τα συστήματα έχουν την ικανότητα να υποστηρίξουν δισεκατομμύρια χρήστες, εκατομμύρια εξυπηρετητές και πάνω από 10^{14} αρχεία.

- **Αυτοργάνωση**

  Στα ομότιμα συστήματα, η αυτοργάνωση γίνεται επιτακτική λόγω της κλιμακωσιμότητας, της ανεκτικότητας σε αποτυχίες, τη διακοπτόμενη επικοινωνία των πηγών και του κόστους ιδιοκτησίας. Τα ομότιμα συστήματα υποστηρίζουν ένα απροσδιόριστα μεγάλο αριθμό χρηστών και κίνησης, με αποτέλεσμα να απαιτείται διαρκής ρύθμιση και έλεγχος του συστήματος, ώστε να διατηρείται η ομαλή λειτουργία του. Η συνεχής είσοδος και έξοδος χρηστών και πηγών στο σύστημα απαιτεί επίσης διαρκή αναπροσαρμογή και έλεγχο του συστήματος. Τέλος, καθώς μια συνεχής επίβλεψη και αναπροσαρμογή ενός τόσο μεγάλου συστήματος είναι και δύσκολη και πολυδάπανη, η λειτουργία της διανέμεται στους ίδιους τους χρήστες.

  Πολλά συστήματα έχουν προτείνει λύσεις στο πρόβλημα της αυτοργάνωσης. Το OceanStore εφαρμόζει την αυτοργάνωση στην υποδομή τοποθεσίας και δρομολόγησή του. Όλη η υποδομή του OceanStore προσαρμόζεται διαρκώς καθώς ο πληθυσμός μεταβάλλεται. Στο Pastry, η αυτοργάνωση ενσωματώνεται στα πρωτόκολλα εισόδου και εξόδου των κόμβων με βάση ένα ανεκτικό σε αποτυχίες υπερκείμενο δίκτυο. Τέλος, στο FastTrack [50] το σύστημα ορίζει αυτόματα υπερκόμβους (SuperNodes), οι οποίοι αποτελούν υπολογιστές με μεγάλη επεξεργαστική ισχύ και διαθέσιμο εύρος ζώνης, και λειτουργούν ως πλήμνη αναζήτησης (hub).

- **Κόστος ιδιοκτησίας**

  Ένα από τα πλεονεκτήματα των ομότιμων συστημάτων αποτελεί η κοινή ιδιοκτησία. Η κοινή ιδιοκτησία μειώνει το κόστος δημιουργίας και συντήρησης του συστήματος και απόκτησης του περιεχομένου. Αυτό είναι εφαρμόσιμο σε όλα τα είδη των ομότιμων συστημάτων και

ακόμη περισσότερο στα συστήματα κοινών πόρων. Το SETI@home για παράδειγμα είναι ταχύτερο από τον πιο ισχυρό υπερ-υπολογιστή του κόσμου με μόλις 1% του κόστους του.

Η φιλοσοφία του Napster βασίστηκε στην συμβολή περιεχομένου από κάθε χρήστη, με αποτέλεσμα να δημιουργηθεί μια τεράστια πηγή μουσικών αρχείων. Σε παρόμοια αρχή βασίζονται όλα τα συστήματα διαχείρισης περιεχομένου, όπως στο OceanStore.

- **Ανωνυμία**

Λόγω των κοινωνικών δραστηριοτήτων που προωθούν τα ομότιμα συστήματα (διανομή περιεχομένου και ελεύθερη πρόσβαση στην πληροφορία), η ανωνυμία αποτελεί βασικό ζήτημα.

Η ανωνυμία μπορεί να διαχωριστεί σε διάφορες μορφές σύμφωνα με τον δημιουργό του Free Haven [51]. Έτσι έχουμε τις εξής ανωνυμίες:

- Συγγραφέα : Ο συγγραφέας ή δημιουργός ενός εγγράφου δε μπορεί να αναγνωριστεί
- Εκδότη : Ο χρήστης που εξέδωσε το έγγραφο στο σύστημα δε μπορεί να αναγνωριστεί
- Αναγνώστη : Οι χρήστες που διαβάζουν ή έχουν πρόσβαση στο έγγραφο δε μπορούν να αναγνωριστούν
- Εξυπηρετητή : Οι εξυπηρετητές που κατέχουν το έγγραφο δε μπορούν να εντοπιστούν
- Εγγράφου : Οι εξυπηρετητές δε γνωρίζουν ποια έγγραφα κατέχουν
- Αναζήτησης : Ο εξυπηρετητής δε γνωρίζει ποιο έγγραφο χρησιμοποιεί για να απαντήσει στην ερώτηση αναζήτησης ενός χρήστη

Ανεξάρτητα από τη μορφή της ανωνυμίας, η εφαρμογή της συνήθως απαιτεί την υλοποίηση ενός τύπου ανώνυμης επικοινωνίας μεταξύ χρηστών : την *ανωνυμία αποστολέα*, που αποκρύπτει την ταυτότητα του αποστολέα, την *ανωνυμία δέκτη*, που αποκρύπτει την ταυτότητα του δέκτη και την *κοινή ανωνυμία*, που αποκρύπτει την ταυτότητα του δέκτη και του αποστολέα από τους ίδιους ή άλλους χρήστες [52].

Εκτός όμως από τους τύπους της ανωνυμίας, που υποστηρίζουν τα διάφορα ομότιμα συστήματα, θα πρέπει να γνωρίζουμε και το βαθμό της ανωνυμίας που προσφέρουν. Οι Reiter και Rubin [53] όρισαν τέσσερις βαθμούς ανωνυμίας, που περιλαμβάνουν την *απόλυτη*

ανωνυμία (absolute privacy), την ανωνυμία *πέραν υποψίας (beyond suspicion)*, την ανωνυμία *πιθανής αθωότητας (probable innocence)* και την ανωνυμία της *πιθανής αποκάλυψης (provably exposed)*. Η ανωνυμία *πέραν υποψίας* για παράδειγμα περιγράφει την περίπτωση όπου ένας εισβολέας μπορεί να αναγνωρίσει σημάδια ενός σταλμένου μηνύματος, αλλά ο αποστολέας εμφανίζεται με τις ίδιες πιθανότητες όσο όλοι οι υπόλοιποι χρήστες του συστήματος.

Για την επίτευξη της ανωνυμίας, έχουν αναπτυχθεί 6 κυρίως τεχνικές. Αυτές είναι επιγραμματικά οι εξής :

- Πολυεκπομπή : Δημιουργούνται ομάδες χρηστών με αποτέλεσμα όποτε γίνεται αίτηση για κάποιο έγγραφο, αυτό αποστέλλεται σε όλους τους χρήστες της ομάδας
- Παραπλάνηση διεύθυνσης αποστολέα : Η διεύθυνση του αποστολέα αλλάζει ώστε να μη μπορεί να ανακαλυφθεί. Το πρόβλημα με αυτή την τεχνική είναι ότι πολλά πρωτόκολλα δεν επιτρέπουν τη μετάδοση μηνυμάτων από άγνωστες ή λανθασμένες διευθύνσεις
- Παραπλάνηση ταυτότητας : Όταν κάποιος προωθεί μια αναζήτηση αλλάζει την ταυτότητα του αποστολέα και τοποθετεί τη δικιά του [54]
- Συγκαλυμμένα μονοπάτια : Τυχαίοι κόμβοι τοποθετούνται μεταξύ του αποστολέα και του παραλήπτη ώστε να χαθεί το πραγματικό μονοπάτι του μηνύματος [55]
- Δυσεπίλυτα ψευδώνυμα : Ένας εξυπηρετητής αναλαμβάνει να δίνει ψευδώνυμα στους χρήστες ώστε να παραμένει κρυφή η ταυτότητά τους. Μειονέκτημα σε αυτήν την τεχνική αποτελεί η ανάγκη εμπιστοσύνης ως προς τον εξυπηρετητή [56]
- Εξαναγκαστική τοποθέτηση : Τα αρχεία τοποθετούνται τυχαία σε χρήστες μέσω μιας συνάρτησης κατατεμαχισμού οπότε ο χρήστης δε γνωρίζει και δεν είναι υπεύθυνος για τα αρχεία που έχει αποθηκευμένα

Στον Πίνακας 1 φαίνονται μερικά ομότιμα συστήματα και οι τεχνικές ανωνυμίας που προσφέρουν όσον αφορά τις βασικές μορφές ανωνυμίας που αναφέραμε.

Σύστημα	Μορφές και τεχνικές Ανωνυμίας			
	Εκδότη	**Αναγνώστη**	**Εξυπηρετητή**	**Εγγράφου**
Gnutella	πολυεκπομπής, συγκαλυμμένα μονοπάτια	---	---	---
Freenet	συγκαλυμμένα μονοπάτια, παραπλάνηση ταυτότητας	συγκαλυμμένα μονοπάτια	εξαναγκαστική τοποθέτηση	κωδικοποίηση
FreeHeaven	συγκαλυμμένα μονοπάτια	συγκαλυμμένα μονοπάτια	πολυεκπομπής	κωδικοποίηση, τεμαχισμός των αρχείων
CAN	---	---	εξαναγκαστική τοποθέτηση	κωδικοποίηση

Πίνακας 1. Μορφές ανωνυμίας και τεχνικές για την ενίσχυσή τους

- **Ασφάλεια**

  Όπως σε όλα τα σύγχρονα συστήματα, η ασφάλεια αποτελεί σημαντικό ζήτημα και για τα ομότιμα συστήματα. Ο δυναμικός πληθυσμός και η ανωνυμία πολλές φορές των χρηστών τους αποτελεί ένα ακόμα σημείο που καθιστά την ασφάλεια ιδιαιτέρως κρίσιμη. Συνοπτικά, μπορούμε να αναγνωρίσουμε πέντε βασικά θέματα ασφαλείας που απασχολούν τα ομότιμα συστήματα.

  - Κωδικοποίηση πολλαπλών κλειδιών : Τα συστήματα διανομής περιεχομένου στην προσπάθεια παροχής ανωνυμίας στους χρήστες αλλά και στα έγγραφα κωδικοποιούν την ταυτότητα αυτών χρησιμοποιώντας ασύμμετρη κωδικοποίηση πολλαπλών κλειδιών [57]. Ανακύπτει όμως πρόβλημα με τον τρόπο διάδοσης αυτών των κλειδιών καθώς μπορούν να αποτελέσουν σημεία επίθεσης.
  - Προστασία από εκτέλεση κώδικα : Πολλά από τα ομότιμα συστήματα εκμεταλλεύονται την επεξεργαστική ισχύ των χρηστών για να επιτύχουν την επίλυση κάποιου προβλήματος. Πρέπει όμως να παρέχεται προστασία στον χρήστη ώστε κατά πρώτον να μη δημιουργεί προβλήματα ο κώδικας που θα εκτελεστεί στον χρήστη και κατά δεύτερον να μη μπορεί κάποιος άλλος χρήστης να εκτελέσει εντολές στον υπολογιστή του, για τις οποίες δεν έχει συμφωνήσει ο χρήστης [58].

- Διαχείριση Ψηφιακών Δικαιωμάτων : Το υλικό που διανέμεται μεταξύ των χρηστών θα πρέπει να ελέγχεται ώστε να διατηρούνται τα πνευματικά δικαιώματα των δημιουργών τους. Το θέμα αυτό έχει αποτελέσει σημείο μεγάλης διαμάχης κυρίως στην μουσική και κινηματογραφική βιομηχανία και δεν είναι λίγες οι περιπτώσεις όπου χρήστες οδηγούνται σε δικαστικές διαμάχες με εταιρείες.
- Φήμη και εμπιστοσύνη : Οι κοινοί πόροι των ομότιμων συστημάτων βασίζονται στην συμμετοχή των χρηστών. Είναι αναγκαίο να ελέγχεται κατά πόσο τα περιεχόμενα που συνεισφέρει ένας χρήστης είναι ορθά και να αποφευχθεί το φαινόμενο του free riding [59], κατά το οποίο ένας χρήστης δεν προσφέρει περιεχόμενο αλλά διαρκώς αποκτάει περιεχόμενο από τους υπόλοιπους χρήστες.
- Τείχη προστασίας (firewalls) : Οι ομότιμες εφαρμογές απαιτούν άμεση επικοινωνία μεταξύ των χρηστών. Αυτό όμως πολλές φορές αναχαιτίζεται από τα τείχη προστασίας των δικτύων για λόγους ασφαλείας καθιστώντας πολλούς χρήστες αποκλεισμένους. Πρέπει λοιπόν να βρεθούν τρόποι διασύνδεσης αυτών των χρηστών (συνήθως μέσω της πόρτας 80, HTTP) χωρίς όμως παράλληλα να αφήνουν κενά ασφαλείας με ενδεχόμενο κίνδυνο επιθέσεων.

- **Απόδοση**

  Τα ομότιμα συστήματα στοχεύουν στην βελτίωση της απόδοσης συναθροίζοντας τον κατανεμημένο αποθηκευτικό χώρο (πχ. Napster, Gnutella) και τους επεξεργαστικούς κύκλους (πχ. SETI@home) συσκευών που είναι διάσπαρτοι σε ένα δίκτυο. Λόγω της αποκεντρωμένης φύσης αυτών των μοντέλων, η απόδοση επηρεάζεται από τρία είδη πηγών, την επεξεργαστική ισχύ, τον αποθηκευτικό χώρο και το διαθέσιμο εύρος ζώνης. Το διαθέσιμο εύρος ζώνης αποτελεί ίσως το σημαντικότερο παράγοντα καθώς ένας μεγάλος αριθμός μηνυμάτων χρησιμοποιείται ώστε να συντονιστεί η μεταφορά αρχείων μεταξύ των χρηστών. Το γεγονός αυτό περιορίζει την κλιμακωσιμότητα του συστήματος. Η απόδοση σε αυτήν την περίπτωση δεν καταγράφεται σε κλίμακα κλάσματος του δευτερολέπτου (όπως συνηθίζεται για τα σύγχρονα δίκτυα) αλλά σε κλίμακα συνολικού χρόνου που απαιτείται για να αποκτηθεί ένα αρχείο και πόσο εύρος ζώνης έχει καταναλωθεί στην αντίστοιχη αναζήτησή του.

Στα συστήματα με κεντρική διαχείριση (πχ. Napster, SETI@home), ο συντονισμός μεταξύ των χρηστών ελέγχεται από έναν κεντρικό διαχειριστή και εν συνεχεία οι χρήστες μπορούν να επικοινωνήσουν άμεσα μεταξύ τους. Αυτό καθιστά τα συστήματα αυτά ευάλωτα σε προβλήματα που αντιμετωπίζουν οι κεντρικοί διαχειριστές. Για να αποφευχθεί αυτό, διάφορες υβριδικές αρχιτεκτονικές έχουν προταθεί [60] ώστε να διανεμηθεί αυτή η λειτουργία σε πολλαπλούς διαχειριστές, που βρίσκονται διάσπαρτοι στο δίκτυο, όπως στην περίπτωση του DNS.

Στα αποκεντρικοποιημένα διαχειριστικά συστήματα, όπως το Gnutella και το Freenet, η διαχείριση αποτελεί αρμοδιότητα του κάθε χρήστη. Το πρόβλημα σε αυτά τα συστήματα είναι ότι οι χρήστες πρέπει να προωθούν μηνύματα αναζήτησης άλλων χρηστών, με αποτέλεσμα τη δημιουργία επιπρόσθετης κίνησης. Η κίνηση αυτή είναι ανάλογη με τον αριθμό των βημάτων που απαιτείται να κάνει μια αναζήτηση και αυξάνει με τον αριθμό των χρηστών.

Για τη βελτίωση της απόδοσης στα ομότιμα συστήματα υπάρχουν τρεις βασικές προσεγγίσεις. Η πρώτη είναι η αναπαραγωγή του περιεχομένου. Με αυτήν την τεχνική, το περιεχόμενο τοποθετείται πλησιέστερα στους χρήστες που το αναζητούν και μειώνονται τα βήματα αναζήτησης. Τα αντίγραφα αυτά του περιεχομένου πρέπει να διατηρούνται ενήμερα καθώς δεν πρέπει να υπάρχουν διαφορές μεταξύ τους. Η τεχνική αυτή βελτιώνει επίσης την ανεκτικότητα του συστήματος σε αποτυχίες, καθώς η αποχώρηση ενός χρήστη από το σύστημα δεν περιορίζει την αναζήτηση και απόκτηση των περιεχομένων που παρείχε.

Η δεύτερη τεχνική είναι αυτή της προσωρινής αποθήκευσης (caching). Οι χρήστες μπορούν να διατηρούν προσωρινά πληροφορίες για αρχεία ή κλειδιά που έχουν αναζητηθεί πρόσφατα, με αποτέλεσμα να μη χρειάζεται να προωθούν διαρκώς τις ίδιες αναζητήσεις. Με αυτό τον τρόπο περιορίζονται τα μηνύματα αναζήτησης. Στο Freenet για παράδειγμα, όποτε επιστρέφεται η απάντηση μιας αναζήτησης, αυτή αποθηκεύεται προσωρινά στους ενδιάμεσους χρήστες ώστε μελλοντικά να μπορούν να απαντήσουν άμεσα.

Η τρίτη τεχνική αφορά την έξυπνη δρομολόγηση και οργάνωση του δικτύου. Για να εφαρμοστεί αυτό θα πρέπει πρώτα να μελετηθεί η κοινωνική φύση των συναλλαγών των χρηστών. Το πιο γνωστό ίσως παράδειγμα τέτοιας μελέτης αποτελεί το «φαινόμενο μικρού κόσμου» [61] του Milgram. Σύμφωνα με αυτό, διαπιστώθηκε πως όλοι οι

άνθρωποι στις Ηνωμένες Πολιτείες συνδέονται μεταξύ τους μέσω έξι γνωριμιών. Ο Ramanathan [62] στη συνέχεια διαχώρισε τους χρήστες με βάση τα ενδιαφέροντα τους και δυναμικά διαχειρίστηκε τις συνδέσεις μεταξύ τους ώστε οι χρήστες με παρόμοια ενδιαφέροντα να τοποθετούνται κοντά στο σύστημα. Με την ίδια λογική μπορούμε να τοποθετήσουμε τους χρήστες στο σύστημα με βάση τη γεωγραφική τους θέση, μειώνοντας έτσι το χρόνο αναζήτησης. Αρκετά από τα συστήματα που έχουμε αναφέρει προσπαθούν να ενσωματώσουν παρόμοιους έξυπνους αλγόριθμους ώστε να μειώσουν τα βήματα αναζήτησης και δρομολόγησης.

- **Ανεκτικότητα σε αποτυχίες**

  Ένας από τους πρωταρχικούς στόχους των ομότιμων συστημάτων αποτέλεσε η αποφυγή των κεντρικών σημείων αποτυχίας. Αν και τα περισσότερα συστήματα το επιτυγχάνουν (καθαρά ομότιμα συστήματα), παραμένουν ανοικτά σε αποτυχίες που σχετίζονται με συστήματα που εκτείνονται σε πολλαπλούς ξενιστές και δίκτυα, όπως διακοπές επικοινωνίας, αδυναμία επικοινωνίας και αποτυχίες κόμβων. Το ζητούμενο αποτελεί η αδιάκοπη λειτουργία των χρηστών που παραμένουν στο σύστημα και η σωστή επανασύνδεση των χρηστών που είχαν εξέλθει.

  Το πρόβλημα της ανεκτικότητας μπορεί να διαχωριστεί σε δύο κατηγορίες. Η πρώτη αφορά την αδυναμία απόκτησης του περιεχομένου των χρηστών που αποχώρησαν. Επίλυση σε αυτό το πρόβλημα μπορεί να δοθεί με τη δημιουργία αντιγράφων στο δίκτυο [63], που εγγυάται τη διάδοση του περιεχομένου μετά την αποχώρηση ενός χρήστη. Η τεχνική αυτή απαιτεί όμως τον έλεγχο και την ανανέωση των αντιγράφων, ώστε να παραμένουν ενήμερα.

  Η δεύτερη κατηγορία αφορά τη δρομολόγηση και την επικοινωνία γενικότερα των χρηστών. Σε πολλά ομότιμα συστήματα, η δρομολόγηση βασίζεται στην προώθηση μηνυμάτων μεταξύ των χρηστών. Το σύστημα πρέπει να εγγυάται την ύπαρξη μονοπατιών αναζήτησης και κατ' επέκταση δρομολόγησης, ανεξάρτητα με τη διάταξη και τον πληθυσμό του συστήματος, καθώς αυτός αλλάζει διαρκώς.

2.7. Σύνοψη - Συμπεράσματα

Στην ενότητα αυτή παρουσιάσαμε μια ιστορική διαδρομή των ομότιμων συστημάτων, από το Usenet και την αρχιτεκτονική SNA έως την εμφάνιση του Napster και του Gnutella. Μέσα από αυτήν την αναδρομή είδαμε και τους πρώτους ορισμούς των ομότιμων συστημάτων και θέσαμε το δικό μας ορισμό που βασίζεται στον τύπο των πηγών, την κατανομή των λειτουργιών και τις παρεχόμενες υπηρεσίες. Εν συνεχεία θέσαμε τους βασικούς στόχους των ομότιμων συστημάτων καθώς και πως αυτά μπορούν να διαχωριστούν σε κατηγορίες ανάλογα με το μοντέλο που ακολουθούν, τις εφαρμογές που υποστηρίζουν και τα χαρακτηριστικά που παρουσιάζουν. Όλα τα παραπάνω μας βοήθησαν να παρουσιάσουμε και να αναλύσουμε κάποια από τα βασικότερα ζητήματα και προβλήματα των ομότιμων συστημάτων, όπως αυτά της αποκέντρωσης, της κλιμακωσιμότητας, της απόδοσης και της ανεκτικότητας σε σφάλματα, τα οποία θα αποτελέσουν σημαντικούς στόχους του συστήματός μας προσφέροντας πρωτότυπες λύσεις και βελτιώσεις.

2.8. Αναφορές

[1] Napster, www.napster.com

[2] United States District Court Northern District of California, , "Memorandum and Order," No. MDL 00-1369 MHPC 99-5183 MHPM, released Feb. 22, 2002.

[3] Google, www.google.com

[4] Lycos, www.lycos.com

[5] Brandenburg, K., Stoll, G., "The ISO/MPEG-1 Audio Codec: A Generic Standard for Coding of High Quality Digital Audio," JAES, October, 1994.

[6] Gnutella, http://gnutella.wego.com

[7] Nowitz, D., "UUCP Implementation Description. UNIX Programmer's Manual," Bell Laboratories, 1978.

[8] Mockapetris, P., "DOMAIN NAMES - CONCEPTS and FACILITIES," RFC882, November, 1983.

[9] Communications Server for AIX, http://www-306.ibm.com/software/network/commserver/aix/library/publications.html

[10] IBM Corporation , "SNA Format and Protocol Reference: Architectural Logic," SC30-3112, 1980.

[11] Sultan, R. A., Kermani, P. , Grover, G. A., Barzilai, T. P., Baratz, A. E., "Implementing System/36 Advanced Peer-to-Peer Networking," IBM Systems Journal 26, No. 4, 1987.

[12]Green, P. E. , Chappuis, R. J., Fisher, J. D., Frosch, P. S., Wood, C. E., "A Perspective on Advanced Peer-to-Peer Networking," IBM Systems Journal 26, No. 4, 1987.

[13]Wray, S., Glauter, T., Hopper, A., "The Medusa applications environment," in Proceeding of the International Conference on Multimedia Computing and Systems, 1994.

[14]Singh. , M. P., "Peering at Peer-to-Peer Computing," IEEE Internet Computing, Vol.5 , No. 1, January/February, 2001.

[15]Thomas, L., Suchter, S., Rifkin, A., "Developing Peer-to-Peer Applications on the Internet: the Distributed Editor, SimulEdit," Dr.Dobb's Journal, January, 1998.

[16]Kellerer, W., "Dienstarchitekturen in der Telekommunikation - Evolution, Methoden und Vergleich," Technical Report TUM- LKN-TR-9801, 1998.

[17]SETI@HOME, http://setiathome.ssl.berkeley.edu

[18]Cohen, B., "Incentives Build Robustness in BitTorrent," presented at the 1st Workshop Economics of Peer-2-Peer Systems, 2003.

[19]Clarke, I., Hong, T.W., Miller, S.G., Sandberg, O., Wile, B., "Protecting Free Expression Online with Freenet," IEEE Internet Computing, 6(1), 2002.

[20]Saint-Andre, P., "Jabber Technology Overview," Jabber Software Foundation, 2001.

[21]Schollmeier, R., "A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications," in Proceedings of the First International Conference on Peer-to-Peer Computing, 2002.

[22]Groove Networks, "Introducing Groove," http://www.groovenetworks.com/ , 2000.

[23]Aimster, www.aimster.com

[24]Nejdl, W., Wolpers, M., Siberski, W., Schmitz, C., Schlosser, M., Brunkhorst, I., Löser, A., "Super-Peer-Based Routing and Clustering Strategies for RDF-Based Peer-to-Peer Networks," 2003.

[25]Milojicic, D., Kalogeraki, V., Lukose, R., Nagaraja, K., Pruyne, J., Richard, B., Rollins, S., Xu, Z., "Peer-to-Peer Computing,", HPL-2002-57R1, HP Labs Technical Report, 2002.

[26]Kant, K., Iyer, R., Tewari, V., "A Framework for Classifying Peer-to-Peer Technologies," in 2nd IEEE International Symposium on Cluster Computing and the Grid ,May , 2002.

[27]Williams R., et al., "Parallel Computing Works," Morgan Kaufmann Publishers.

[28]ClimatePrediction, http://climateprediction.net

[29]FightAids@Home, http://fightaidsathome.scripps.edu/

[30]BOINC, http://boinc.berkeley.edu/
[31]JavaBeans, http://java.sun.com/products/javabeans/docs/index.html
[32]Bolosky, W., Douceur, J., Ely, D., Theimer, M., "Feasibility of a Serverless Distributed File System Deployed on an Existing Set of Desktop PCs," in Proceedings of SIGMETRICS, Satnta Clara, CA, June, 2000.
[33]Rowstron, A., Druschel, P., "Storage Management and Caching in PAST, a Large-Scale, Persistent, Peer-to-Peer Storage Utility," in Proceedings of SOSP, 2001.
[34]Gribble, S., Halevy, A., Ives, Z., Rodrig, M., Suciu, D., "What Can Peer-to-Peer Do for Databases and Vice Versa?," in Proceedings of the WebDB: Workshop on Databases and the Web, Santa Barbara, CA, USA, 2001.
[35]Stoica, I., Morris, R., Karger, D., Kaashoek, F., Balakrishnan, H., "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," in Proceedings of the SIGCOMM, 2001.
[36]Rowston, A., Druschel, P., "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," in Proceedings of International Conference on Distributed Systems Platforms (Middleware), ACM Press, 2001.
[37]Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R., Rhea, R., Weatherspoon, H., Weimer, W., Wells, C., Zhaio, B. "Oceanstore : An Architecture for Global-Scale Persistent Storage," in Proceedings of the Ninth international Conference on Architectural Support for Programming Languages and Operations Systems, November, 2000.
[38]Ratnasamy, S., Francis, O., Handley, M., Karp, R., Shenker, S., "A Scalable Content-Addressable Network," in Proceedings of the SIGCOMM, 2001.
[39]Li, J., Jannotti, J., Couto, D.D., Karger, D., Morris, R., "A Scalable Location Service for Geographic Ad-hoc Routing," in Proceedings of ACM Conference on Mobile Computing and Networking (MOBICOM), Boston, MA, 2000.
[40]OpenCola, www.opencola.com
[41]Waterhouse, S., Doolin, D.M., Kan, G., Faybisheniko, Y., "Distributed Search in P2P Networks," IEEE Internet Computing, January-February, 2002.
[42]Yahoo!, http://messenger.yahoo.com/
[43]Jabber, www.jabber.org/
[44]Storm, D., "Business Embrace Instant Messaging," enterprise.cnet.com/enterprise/ 0-9534-7-4403317.html, January 2001.
[45]BuzzPad, www.buzzpad.com
[46]Quazal, www.quazal.com

[47]Knutsson, B., Lu, H., Xu, W., Hopkins, B., "Peer-to-Peer Support for Massively Multiplayer Games," in IEEE Infocom, March, 2004.
[48]Tran, D., Hua, K., Do, T., "ZIGZAG: An Efficient Peer-toPeer Scheme for Media Streaming," in Proc. of IEEE INFOCOM, 2003.
[49]Rejaie R., Ortega, A., "Pals: peer-to-peer adaptive layered streaming," in Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video, 2003.
[50]FastTrack, http://www.fasttrack.nu/
[51]Dingledine, R., "The Free Haven Project: Design and Deployment of an Anonymous Secure Data Haven," MIT Master's Thesis, June, 2000.
[52]Pfitzmann, A., Waidner, M., "Networks without user observability," Computers & Security, 1987.
[53]Reiter, M.K., "Crowds: Anonymity for web transactions," ACM Transactions on Information and System Security, 1998.
[54]Clark, I., Sandberg, O., Wiley, B., Hong, T.W., "Freenet: a distributed anonymous information storage and retrieval system," International Workshop on Design Issues in Anonymity and Unobservability, LNCS 2009, Springer: New York 2001.
[55]Chaum, D., "Untraceable electronic mail, return addresses and digital pseudonyms," Communication of the ACM, 2(24), February, 1981.
[56]Gabber, E., Gibbons, P., Kristol, D., Matias, Y., Mayer, A., "Consistent, yet anonymous, Web access with LPWA," Communications of the ACM, 1999.
[57]Shamir, A., "How to share a secret," Communications of the Association for Computing Machinery, November, 1979.
[58]Xu, Z., Reps, T., Miller, B., "Typestate checking of machine code," in Proceedings of the European Symposium on Programming, 2001.
[59]Adar, E., Huberman, B., "Free Riding on Gnutella," First Monday, Vol. 5, No. 10, October, 2000.
[60]Yang, B., Molina, H.G., "Comparing Hybrid P2P Systems," in Proc. VLDB '01, 2001.
[61]Milgram, S., "The Small World Problem," Psychology today, 1967.
[62]Ramanathan, M. K., Kalogeraki, V., Pruyne, J., "Finding Good Peers in Peer-to-Peer Networks," IEEE International Parallel and Distributed Processing Symposium, Fort Lauderdale, FL, April, 2002.
[63]Waldman, M., Rubin, A.D., Cranor, L.F., "Publius: A robust, tamperevident, censorship-resistant, web publishing system," in Proceedings of the 9th USENIX Security Symposium, August, 2000.

3 ΣΥΣΤΗΜΑΤΑ ΑΝΤΑΛΛΑΓΗΣ ΠΕΡΙΕΧΟΜΕΝΟΥ

Η υπάρχουσα ερευνητική βιβλιογραφία σε ομότιμα συστήματα χωρίζεται κυρίως σε τέσσερις βασικές κατηγορίες, σε αυτές της αναζήτησης, της ασφάλειας, της αποθήκευσης και των εφαρμογών. Η διατριβή αυτή θα επικεντρωθεί στον τομέα της αναζήτησης, καθώς αυτός ο τομέας παρουσιάζει αρκετό ερευνητικό ενδιαφέρον και αποτελεί το βασικό κορμό, πάνω στον οποίο στηρίζονται οι υπόλοιποι τρεις. Ο τομέας αυτός μπορεί να διαιρεθεί περεταίρω σε δύο κατηγορίες, τους αλγορίθμους που βασίζονται σε εννοιολογικούς δείκτες αναζήτησης (semantic index) και σε αυτούς που είναι μη-εννοιολογικοί (semantic-free). Οι πρώτοι παρέχουν τη δυνατότητα αναζήτησης με βάση μια σειρά κριτηρίων αλλά συνήθως είναι πιο πολύπλοκοι και όχι βέλτιστοι, ενώ οι δεύτεροι παρέχουν τη βέλτιστη απόδοση και κλιμακωσιμότητα, περιορίζοντας όμως τις δυνατότητες εκτέλεσης σύνθετων αναζητήσεων. Η δική μας αρχιτεκτονική βασίζεται σε μη-εννοιολογικές αναζητήσεις.

3.1. Δομημένη και μη-δομημένη δρομολόγηση

Το βασικό στοιχείο όλων των ομότιμων συστημάτων είναι ο αλγόριθμος δρομολόγησης στον οποίο βασίζονται. Οι αλγόριθμοι δρομολόγησης στα ομότιμα συστήματα χωρίζονται αρχικά σε δύο κατηγορίες, σε δομημένους και μη-δομημένους. Τα πρώτα ομότιμα συστήματα, όπως το Gnutella, βασίζονταν σε μη-δομημένους αλγόριθμους, όπου κάθε μήνυμα προωθούταν σε όλους τους γνωστούς κόμβους, «πλημμυρίζοντας» έτσι το δίκτυο. Μεταγενέστεροι αλγόριθμοι όπως οι Plaxton, Pastry, Tapestrym Chord, CAN κ.α. βασίστηκαν σε δομημένους αλγορίθμους, όπου τα μηνύματα προωθούνται με βάση μια σειρά κανόνων, με στόχο τον περιορισμό της κίνησης και τη μείωση των απαιτούμενων βημάτων για μια επιτυχή αναζήτηση. Οι μη-δομημένοι αλγόριθμοι έχουν ακόμη χαρακτηριστεί και ως αλγόριθμοι «πρώτης γενιάς», σε σχέση με τους μεταγενέστερους αλγορίθμους «δεύτερης γενιάς» [1],[2]. Οι δομημένοι αλγόριθμοι «δεύτερης γενιάς» παρέχουν τη δυνατότητα αναζήτησης μέσω λέξεων-κλειδιών σε

εγγυημένο αριθμό βημάτων και χωρίς να αποσυμφορούν το υπερκείμενο δίκτυο [3]. Αντιθέτως, οι μη-δομημένοι αλγόριθμοι προκαλούν αυξημένο όγκο κίνησης και δεν μπορούν να εγγυηθούν σε όλες τις περιπτώσεις επιτυχείς αναζητήσεις [4]. Παρόλα αυτά, αρκετές ερευνητικές ομάδες ακόμη ασχολούνται με μη-δομημένους αλγορίθμους.

Τα δομημένα συστήματα έχουν δύο βασικά ζητήματα για τα οποία δέχονται κριτική. Πρώτον, δεν υποστηρίζουν κόμβους με υψηλά χαρακτηριστικά μεταβατικότητας. Όσον αφορά αυτή την κριτική, πρέπει πρώτον να σημειώσουμε ότι πολλά μη-δομημένα συστήματα αδυνατούν επίσης να υποστηρίξουν τέτοιου είδους κόμβους και δεύτερον, νέα συστήματα δομημένων αλγορίθμων έχουν εμφανισθεί που καταφέρνουν να ξεπεράσουν αυτό το πρόβλημα χωρίς να επιβαρύνουν ιδιαίτερα το σύστημα. Παρόλα αυτά, πολλοί από τους αλγορίθμους δρομολόγησης δομημένων συστημάτων αδυνατούν να παρουσιάσουν την επίπτωση που έχει η συνεχής είσοδος και αποχώρηση κόμβων στα συστήματά τους. Μέσα από αυτήν τη διατριβή θα δείξουμε μεθόδους με τις οποίες μπορεί να ξεπερασθεί αυτό το πρόβλημα στην δική μας αρχιτεκτονική.

Η δεύτερη κύρια κριτική των δομημένων συστημάτων είναι ότι δεν υποστηρίζουν πολύπλοκα ερωτήματα και αναζητήσεις, σε αντίθεση με τα μη-δομημένα συστήματα. Όσον αφορά αυτήν την κριτική πρέπει πρώτα να εξετάσουμε το είδος των αναζητήσεων στα ομότιμα συστήματα. Στις περισσότερες περιπτώσεις, μια απλή αναζήτηση με βάση λέξεις κλειδιά είναι αρκετή για την εύρεση του κατάλληλου αρχείου. Υπάρχουν βέβαια και εφαρμογές που απαιτούν πιο πολύπλοκες αναζητήσεις, όπως για παράδειγμα συστήματα αποθήκευσης που εξομοιώνουν μια κατανεμημένη βάση δεδομένων. Ακόμα όμως και σε αυτήν την περίπτωση, το πρόβλημα δεν έγκειται στο αν ο αλγόριθμος δρομολόγησης είναι δομημένος ή μη-δομημένος αλλά αν υποστηρίζει εννοιολογικές αναζητήσεις.

Τα δομημένα συστήματα χαρακτηρίζονται κυρίως από τη δυνατότητα να παρέχουν επιτυχείς αναζητήσεις ακόμα και για σπάνια δεδομένα. Αντίθετα, τα μη-δομημένα συστήματα παρέχουν καλύτερη απόδοση στην εύρεση διαδεδομένων δεδομένων. Λόγω των παραπάνω παρατηρήσεων, στην συνέχεια αυτού του κεφαλαίου δε θα διαχωρίζουμε τα ομότιμα συστήματα με βάση τη δόμηση τους αλλά με βάση το σύστημα δεικτοδότηση που υλοποιούν. Αυτή η κατηγοριοποίηση θα γίνει για δύο λόγους. Πρώτον, τα περισσότερα ομότιμα συστήματα βασίζονται σε κάποιο είδος δομημένης αρχιτεκτονικής. Τα καθαρά μη-δομημένα συστήματα έχουν σχεδόν εκλείψει. Ακόμα και το Gnutella έχει πλέον επεκταθεί με τη χρήση υπερκόμβων και εμπεριέχει μια μορφή δόμησης (ιεραρχίας). Δεύτερον, η κατηγοριοποίηση με βάση το

σύστημα δεικτοδότησης έχει τα τελευταία χρόνια οριοθετήσει τις δύο βασικές κατευθύνσεις έρευνας στα ομότιμα συστήματα. Η πρώτη κατεύθυνση είναι αυτή των μη-εννοιολογικών συστημάτων, που επικεντρώνονται σε ζητήματα απόδοσης, κλιμακωσιμότητας και ανεκτικότητας σε αποχωρήσεις κόμβων. Η δεύτερη κατεύθυνση είναι αυτή των εννοιολογικών συστημάτων, που επικεντρώνονται στην υποστήριξη πολύπλοκων αναζητήσεων. Η διατριβή αυτή, όπως έχει αναφερθεί ήδη, θα επικεντρωθεί στα συστήματα μη-εννοιολογικών συστημάτων. Για το λόγο αυτό, στην συνέχεια αυτού του κεφαλαίου θα επικεντρωθούμε στην περιγραφή των σημαντικότερων μη-εννοιολογικών συστημάτων που ήδη υπάρχουν, με μια μικρή μόνο αναφορά στα κυριότερα εννοιολογικά συστήματα.

3.2. Δεικτοδότηση και Αναζήτηση

Η έρευνα στον τομέα της δεικτοδότησης αποτελεί ένα από τα βασικά συστατικά των ομότιμων συστημάτων αναζήτησης. Εμπεριέχει έναν ευρύ τομέα ζητημάτων των ομότιμων συστημάτων, όπως αυτό επιδεικνύεται και από το μοντέλο Εύρεσης/Δεικτών (Search/Index Links (SIL) model) [5]. Ο Manber το έθεσε λεπτομερέστερα ως, "το πιο σημαντικό εργαλείο για την ανάκτηση πληροφοριών είναι ο δείκτης- μια συλλογή από όρους με δείκτες στα σημεία όπου βρίσκονται πληροφορίες για το που μπορούν να ανακτηθούν τα δεδομένα" [6]. Οι Sen και Wang αναφέρουν πως τα ομότιμα δίκτυα συνήθως απαρτίζονται από μια σειρά συνδέσεων μεταξύ κόμβων που προσφέρουν σηματοδοσία στο επίπεδο εφαρμογών, σε αντίθεση με απλή μεταφορά δεδομένων [7]. Στην ίδια κατεύθυνση, στην διατριβή αυτή θα επικεντρωθούμε στον τρόπο «σηματοδοσίας» δεικτών και αναζητήσεων, με έμφαση στην εξαρτησιμότητα και στην προσαρμοστικότητα του υπερκείμενου δικτύου. Η στατική εξαρτησιμότητα προσφέρει έναν τρόπο μέτρησης της ικανότητας του δικτύου να δρομολογεί όταν υπάρχουν σφάλματα (απώλεια κόμβων) σε ένα στατικό δίκτυο. Η δυναμική εξαρτησιμότητα επεκτείνεται στην δρομολόγηση σε δίκτυα με σφάλματα όταν τόσο οι χρήστες (κόμβοι) όσο και τα δεδομένα (έγγραφα) είναι δυναμικά, δηλαδή διαρκώς εισέρχονται και εξέρχονται στο ομότιμο δίκτυο. Η προσαρμοστικότητα αναφέρεται στην ικανότητα του δικτύου να προσφέρει υπηρεσίες αναζήτησης δεδομένων καθώς αυτά μεταβάλλονται δυναμικά. Με αυτό τον τρόπο, οι εφαρμογές που βασίζονται στο ομότιμο δίκτυο δε χρειάζεται να προβαίνουν σε επιπρόσθετες λειτουργίες για την εύρεση των δεδομένων, καθώς η αρχιτεκτονική του ομότιμου δικτύου εγγυάται για την εύρεση των δεδομένων, ανεξάρτητα από τις υποκείμενες αλλαγές [3].

3.3. Είδη Δεικτοδότησης

Ένας δείκτης για ομότιμα συστήματα μπορεί να είναι είτε τοπικός, είτε κεντρικός είτε κατανεμημένος. Στα συστήματα τοπικών δεικτών, όπως το Gnutella, κάθε κόμβος αποθηκεύει δείκτες μόνο για τα τοπικά δεδομένα και δε δέχεται πληροφορίες για δεδομένα άλλων κόμβων. Σε συστήματα κεντρικών δεικτών, όπως το Napster, ένας κεντρικός κόμβος είναι υπεύθυνος για την αποθήκευση όλων των δεικτών όλων των κόμβων ή τουλάχιστον μιας σειράς κόμβων. Τέλος, στα συστήματα με κατανεμημένους δείκτες, κάθε κόμβος αποθηκεύει μια σειρά από δείκτες που αναφέρονται και σε άλλους κόμβους, με πολλούς από τους δείκτες να αποθηκεύονται σε παραπάνω από έναν κόμβο (αναπαραγωγή/αντίγραφα). Τα περισσότερα ομότιμα συστήματα βασίζονται σε κατανεμημένους δείκτες καθώς προσφέρουν καλύτερη κλιμακοσιμότητα και αξιοπιστία.

Οι δείκτες στα ομότιμα συστήματα μπορούν επίσης να κατηγοριοποιηθούν σε αυτούς που προωθούν και αυτούς που δεν προωθούν. Αυτοί που βασίζονται στην δεύτερη κατηγορία απαιτούν ένα μόνο βήμα για μια επιτυχή αναζήτηση [8],[9][10]. Στην κατηγορία αυτή μπορούν επίσης να συμπεριληφθούν και κάποιοι αλγόριθμοι που δρομολογούν σε δύο βήματα για την υποστήριξη μεγαλύτερου αριθμού χρηστών [11],[12]. Παρόλα αυτά, η κατηγορία αυτή παρουσιάζει προβλήματα κλιμακοσιμότητας και για το λόγο αυτό τα περισσότερα συστήματα βασίζονται σε δείκτες που προωθούν, όπου η αναζήτηση προωθείται από έναν κόμβο σε άλλον με βάση έναν αλγόριθμο δρομολόγησης.

3.3.1. Τοπικοί δείκτες

Τα ομότιμα συστήματα που βασίζονται σε τοπικούς δείκτες μειώνονται διαρκώς. Σε αυτά τα συστήματα οι κόμβοι πλημμυρίζουν το δίκτυο με ερωτήματα και αποθηκεύουν μόνο τα δικά τους περιεχόμενα. Με αυτόν τον τρόπο παρέχουν τη δυνατότητα εκτέλεσης σύνθετων αναζητήσεων αλλά δεν εγγυώνται ότι μια αναζήτηση θα είναι επιτυχής, ακόμη και αν το αναζητούμενο περιεχόμενο βρίσκεται στο δίκτυο. Επίσης επιφορτίζουν το δίκτυο με αρκετή κίνηση. Στο αρχικό Gnutella δίκτυο για παράδειγμα, όλοι οι χρήστες εξέπεμπαν «ping» μηνύματα στην αρχή προς όλους τους κόμβους, οι οποίοι απαντούσαν με «pong» μηνύματα, ώστε να γνωστοποιήσουν την ύπαρξή τους. Εν συνεχεία, οι αναζητήσεις γίνονταν με την προώθηση μηνυμάτων αναζητήσεων προς όλους τους γνωστούς κόμβους.[13].

Πολλές μετέπειτα προσπάθειες έχουν γίνει ώστε να βελτιωθεί η κλιμακωσιμότητα των συστημάτων με τοπικούς δείκτες. Το αρχικό Gnutella

χρησιμοποιούσε μηνύματα με σταθερή διάρκεια ζωής (time-to-live) (TTL) μεταξύ 7-10 βημάτων [13]. Η χρήση μικρότερων TTL οδηγεί σε μείωση της κίνησης λόγω μηνυμάτων αλλά επίσης μειώνει και την πιθανότητα επιτυχών αναζητήσεων. Οι Lv και Cao μάλιστα αναφέρουν ότι η χρήση σταθερού TTL «δεν είναι αποτελεσματική» [14]. Για να επιλύσουν αυτό το πρόβλημα πρότειναν έναν επεκτάσιμο δακτύλιο, ο οποίος μεγαλώνει εωσότου επιτευχθεί μια επιτυχής αναζήτηση [15]. Αυτή η μέθοδος καταφέρνει να αυξήσει την πιθανότητα επιτυχής αναζήτησης αλλά αυξάνει επίσης και το φόρτο κίνησης στο δίκτυο. Αντιθέτως η χρήση τυχαίων μονοπατιών (random walk) μειώνει την κίνηση αλλά αυξάνει τα απαιτούμενα βήματα για επιτυχή αναζήτηση. Μια άλλη λύση που έχει προταθεί είναι η χρήση των τυχαίων κ-μονοπατιών (random k-walkers). Σε αυτήν τη μέθοδο κάθε αναζήτηση επαναλαμβάνεται k φορές και συνδυάζεται με TTL όρια [14]. Οι Adamic και Lukose πρότειναν αυτή η μέθοδος να προωθεί τις αναζητήσεις σε κόμβους με μεγαλύτερο βαθμό συνδέσεων, δηλαδή σε κόμβους που έχουν περισσότερες συνδέσεις με άλλους κόμβους [16]. Με αυτόν τον τρόπο οι αναζητήσεις δρομολογούνταν σε πιο κεντρικούς κόμβους. Χωρίς όμως τη χρήση κάποιας τεχνικής αποσυμφόρησης, οι κόμβοι αυτοί επιφορτίζονταν με υπερβολική κίνηση και δημιουργούνταν σημεία αυξημένης κίνησης. Δύο άλλες μέθοδοι που προτάθηκαν είναι η μέθοδος της «προτεραιότητας κατευθυνόμενου εύρους» (directed breadth-first) και της «στατιστικής πλημμύρας» (probabilistic flooding). Στην πρώτη, οι αναζητήσεις προωθούνταν στους κόμβους που επιδείκνυαν μεγαλύτερα ποσοστά επιτυχίας με βάση παλαιότερες αναζητήσεις [15]. Στην δεύτερη, γινόταν μοντελοποίηση με βάση τη θεωρία φιλτραρίσματος (percolation theory) [17].

Αρκετές εργασίες έχουν αναλύσει την απόδοση των ομότιμων συστημάτων που βασίζονται σε τοπικούς δείκτες. Ο Jovanovic παρουσίασε την εκθετική απόδοση του Gnutella [18]. Οι Sen και Wang συνέκριναν την απόδοση του Gnutella, του Fasttrack [19] και του Direct Connect [20],[21],[22]. Από αυτά τα τρία συστήματα μόνο το Gnutella βασιζόταν σε τοπικούς δείκτες. Σήμερα, και τα τρία συστήματα χρησιμοποιούν κατανεμημένους δείκτες με ιεραρχία βασιζόμενη είτε σε Ultrapeers (Gnutella), είτε σε Super-Nodes (FastTrack) είτε σε Hubs (Direct Connect). Η έρευνά τους έδειξε πως αυτά τα συστήματα βασίζονταν σε μεγάλο βαθμό σε έναν μικρό αριθμό κόμβων. Μια έρευνα του πανεπιστημίου της Ουάσιγκτον έδειξε πως το 43% της κίνησης στο πανεπιστήμιο οφείλετο στο Gnutella και στο Kazaa [23]. Επίσης αναφέρουν πως το 26% της κίνησης παραγόταν από 600 εξωτερικούς κόμβους (από σύνολο 281,026). Όλες αυτές οι μελέτες επέδειξαν τόσο τη δημοτικότητα των εφαρμογών που βασίζονται σε ομότιμα δίκτυα όσο και την ανάγκη για καλύτερες αρχιτεκτονικές που θα μπορούν να εξομαλύνουν την

κίνηση ανάμεσα στους κόμβους. Το βασικό στοιχείο που ανέδειξαν ήταν ότι τα ομότιμα συστήματα που βασίζονταν σε τοπικούς δείκτες βασίζονταν σε ένα μικρό αριθμό κόμβων, οι οποίοι επιφορτίζονταν το μεγαλύτερο ποσοστό της συνολικής κίνησης. Αυτή η διαπίστωση ώθησε τους Massoulie και Kermarrec στην δημιουργία του Scamp [24]. Το Scamp υποστηρίζει τόσο τις μεθόδους τυχαίου μονοπατιού όσο και αυτές της πλημμύρας, θεωρώντας όμως ότι κάθε κόμβος έχει τον ίδιο βαθμό σύνδεσης, είναι δηλαδή υπεύθυνος για τον ίδιο όγκο φορτίου. Η ανεκτικότητα όμως του Scamp σε σφάλματα (αποχώρηση κόμβων) δεν εξερευνήθηκε ποτέ.

3.3.2. Κεντρικοί Δείκτες

Οι κεντρικοί δείκτες όπως το Napster θεωρούνται ιστορικά σημαντικοί καθώς ήταν οι πρώτοι που επέδειξαν την κλιμακοσιμότητα των ομότιμων δικτύων μέσα από τη διαφοροποίηση των δεικτών και των ίδιων των δεδομένων. Η αποτυχία του Napster δεν οφείλεται σε τεχνικούς λόγους αλλά στην αδυναμία της κεντρικής διαχείρισης να ξεπεράσει τα νομικά προβλήματα με τις δισκογραφικές εταιρείες [25]. Η έρευνα σε συστήματα με κεντρικούς δείκτες, ή υβριδικά συστήματα όπως αποκαλούνται λόγω της διαφοροποίησης μεταξύ της κεντρικής εύρεσης και της κατανεμημένης ανάκτησης, δεν έχει εξελιχθεί ιδιαίτερα. Οι Yang και Garcia-Molina διαχώρισαν τα συστήματα αυτά σε τέσσερις κατηγορίες [26], στους μη-αλυσωτούς εξυπηρετητές, όπου οι χρήστες αποθηκεύουν τους δείκτες σε έναν εξυπηρετητή και δεν επικοινωνούν με άλλους, στους αλυσωτούς εξυπηρετητές, όπου ο εξυπηρετητής προωθεί μια ανεπιτυχή αναζήτηση σε άλλους εξυπηρετητές, στους πλήρους αντιγραφής, όπου όλοι οι εξυπηρετητές διατηρούν μια πλήρη λίστα από όλους τους δείκτες, και στους κατατεμαχισμένους, όπου οι λέξεις κλειδιά κατατεμαχίζονται και οι εξυπηρετητές διατηρούν τη λίστα αντιστροφής τους.

Το Napster ανήκει στην κατηγορία των μη-αλυσωτών εξυπηρετητών, η οποία έχει το μειονέκτημα ότι οι χρήστες δεν μπορούν να έχουν πρόσβαση στην πλήρη λίστα δεικτών του συστήματος. Οι άλλες τρεις από τις κατηγορίες που προαναφέρθηκαν δεν ανήκουν αυστηρά στην δομή των κεντρικών δεικτών, καθώς εμπεριέχουν μεθόδους κατανεμημένων δεικτών. Η μέθοδος των αλυσωτών εξυπηρετητών θεωρήθηκε ως η πιο κατάλληλη για εφαρμογές ανταλλαγής μουσικών αρχείων και χρησιμοποίησε διάφορες μεθόδους για την καλύτερη κατανομή του φόρτου κίνησης και επεξεργασίας για τους εξυπηρετητές. Κάποιες έρευνες που έγιναν για το Napster και για το OpenNap [27] έδειξαν ότι ένα μεγάλο ποσοστό χρηστών ήταν απρόθυμοι να μοιραστούν περιεχόμενα με άλλους χρήστες (20-40% των χρηστών δεν

προσέφεραν αρχεία) και δήλωναν ψευδή στοιχεία για το εύρος ζώνης που είχαν διαθέσιμο ώστε να απωθήσουν άλλους χρήστες [28].

Η αποτυχία του Napster απέτρεψε ένα μεγάλο μέρος της έρευνας στα ομότιμα δίκτυα από το να αναπτύξει περεταίρω τις αρχιτεκτονικές με κεντρικούς δείκτες. Οι Chawathe και Ratnasamy αντιθέτως παραθέτουν την επιτυχία του Google και του Yahoo ως παράδειγμα χρήσης κεντρικών δεικτών, επισημαίνοντας ότι «τα πραγματικά εμπόδια σε αρχιτεκτονικές τύπου Napster δεν είναι τεχνικά αλλά νομικά και οικονομικά» [29]. Ακόμη όμως και αυτή η δήλωση δεν είναι πλήρως αληθής καθώς υποδηλώνει ότι τα συστήματα με κεντρικούς δείκτες προϋποθέτουν την ύπαρξη αρχικού κεφαλαίου και ύπαρξη κατάλληλου νομικού πλαισίου. Παρόλα αυτά, πολλές αρχιτεκτονικές που δηλώνονται ως κατανεμημένες (όπως αυτή του Google) στην ουσία βασίζονται σε κεντρικούς δείκτες [30].

3.3.3. Κατανεμημένοι Δείκτες

Μια από τις πρώτες προτάσεις για κατανεμημένους δείκτες σε ομότιμα συστήματα ήταν το Freenet [31],[32],[33]. Αν και το Freenet επικεντρώθηκε στην παροχή ανωνυμίας σε ομότιμα συστήματα, εισήγαγε έναν καινούργιο τρόπο δεικτοδότησης. Τα αρχεία στο Freenet χαρακτηρίζονται από δύο είδη κλειδιών, ένα κλειδί με βάση το περιεχόμενο και ένα κλειδί με βάση έναν ασφαλή υποχώρο. Με αυτόν τον τρόπο, όλοι οι χρήστες έχουν τη δυνατότητα πρόσβασης σε ένα αρχείο αλλά μόνο ο εκδότης του αρχείου έχει τη δυνατότητα τροποποίησής του. Η αναζήτηση στο Freenet γινόταν με βάση το δείκτη του περιεχομένου. Κάθε κόμβος δρομολογούσε τα μηνύματα αναζήτησης προς το γειτονικό κόμβο που περιείχε κλειδιά που ο δείκτης τους ήταν παρόμοιος με αυτόν του αρχείου προς αναζήτηση. Η προώθηση συνεχιζόταν μέχρι την εύρεση του αρχείου ή μέχρι κάποιο αριθμό βημάτων με βάση ένα όριο TTL. Όταν μια αναζήτηση ήταν επιτυχής, τότε το μήνυμα επιτυχίας επιστρεφόταν στον αρχικό κόμβο μέσω της αντίστροφης διαδρομής, ενημερώνοντας όλους τους κόμβους για το δείκτη που ευρέθη κατά τη διαδρομή. Η ίδια μέθοδος χρησιμοποιούταν και για τη δημοσίευση των περιεχομένων ενός κόμβου.

Μια πρώτη ανάλυση του Freenet έδειξε πως σε ένα δίκτυο 1000 κόμβων, το μέσο μονοπάτι επιτυχούς αναζήτησης απαιτούσε περίπου 20 βήματα όταν το 30% των κόμβων παρουσίαζαν σφάλμα (δεν ανταποκρίνονταν). Οι σχεδιαστές του Freenet θεώρησαν αυτήν τη συμπεριφορά του δικτύου ως ανεκτικότητα σε σφάλματα [32]. Με βάση όμως μεταγενέστερους αλγορίθμους και αρχιτεκτονικές, η παραπάνω απόδοση θεωρείται αρκετά προβληματική καθώς οι 1000 κόμβοι θεωρούνται πολύ

μικρός πληθυσμός και τα 20 βήματα υπερβολικά πολλά. Η χρήση κατανεμημένου πίνακα κατατεμαχισμού (DHT) έδωσε λύση σε αυτό το πρόβλημα, όπως θα παρουσιάσουμε σε επόμενο υποκεφάλαιο. Παρόλα αυτά, το Freenet έχει αρκετές ομοιότητες με του αλγορίθμους DHT καθώς χρησιμοποιούσε μια συνάρτηση κατατεμαχισμού για τη δημιουργία των κλειδιών. Μια ουσιώδης διαφορά όμως του Freenet με τους DHT αλγορίθμους έγκειται στον τρόπο δημιουργίας του δικτύου. Το Freenet απαιτούσε αρκετό χρόνο για τη δημιουργία του αρχικού πίνακα δρομολόγησης για κάθε κόμβο που εισέρχονταν στο δίκτυο [34], σε αντίθεση με τα συστήματα που βασίζονται σε DHT. Επίσης το Freenet δε μπορούσε να δρομολογήσει με βάση την τοπογραφική τοπολογία των κόμβων ή τα χαρακτηριστικά που επεδείκνυαν (όπως εύρος ζώνης ή καθυστερήσεις). Μετέπειτα βελτιώσεις προτάθηκαν για την επίλυση αυτών των προβλημάτων [35],[36] αλλά δεν είναι αρκετές για να ανταγωνιστούν την απόδοση των συστημάτων που βασίζονται σε DHT. Η βελτιωμένη απόδοση, κλιμακωσιμότητα και ανεκτικότητα σε σφάλματα που παρουσιάζουν τα ομότιμα συστήματα βασισμένα σε DHT είναι ο λόγος που οδήγησε στην επιλογή σχεδίασης της αρχιτεκτονικής μας με την χρήση DHT.

Από την βιβλιογραφική έρευνα των τριών αυτών ειδών δεικτοδότησης γίνεται κατανοητό ότι τα ομότιμα συστήματα που βασίζονται σε κατανεμημένους δείκτες παρουσιάζουν το μεγαλύτερο ενδιαφέρον. Αυτός ήταν ένας από τους βασικούς λόγους που επιλέξαμε τη χρήση DHT για την αρχιτεκτονική μας. Στην συνέχεια, θα παρουσιάσουμε πολύ συνοπτικά την υπάρχουσα ερευνητική εργασία που έχει γίνει στον τομέα των ομότιμων συστημάτων που χρησιμοποιούν εννοιολογικούς δείκτες, προτού συνεχίσουμε αναλυτικότερα με τα συστήματα που βασίζονται σε μη-εννοιολογικούς κατανεμημένους δείκτες, και κυρίως αυτά που βασίζονται σε DHT.

3.4. Εννοιολογικοί Δείκτες

Οι εννοιολογικοί δείκτες επικεντρώνονται στις συσχετίσεις αντικειμένων. Ενώ οι μέθοδοι μη-εννοιολογικών δεικτών (κυρίως DHT) έχουν μεγαλύτερο θεωρητικό υπόβαθρο και μπορούν να εγγυηθούν επιτυχείς αναζητήσεις όταν υπάρχει ένα κλειδί στο δίκτυο, αδυνατούν να συσχετίσουν το όνομα του εγγράφου με το περιεχόμενό του ή τα μεταδεδομένα του. Αντιθέτως τα συστήματα εννοιολογικών δεικτών έχουν αυτήν τη δυνατότητα αλλά δεν μπορούν να εγγυηθούν απολύτως επιτυχείς αναζητήσεις.

Αυτό που προσφέρουν τα συστήματα εννοιολογικών δεικτών είναι πιο εξελιγμένες δυνατότητες για τις υπερκείμενες εφαρμογές, όπως για

παράδειγμα κατανεμημένες σχεσιακές βάσεις δεδομένων με δυνατούς ελέγχους συνάφειας [37]. Παρέχουν ισχυρή ανεξαρτησία δεδομένων, την υποστήριξη SQL ερωτήσεων και ισχυρές συναλλακτικές σημασιολογίες [38], με αρκετά μειονεκτήματα όμως στην απόδοση και στην κλιμακωσιμότητα. Το Mariposa [39],[40] για παράδειγμα περιορίζεται σε 1000 κόμβους ή απαιτεί μεθόδους απόκρυψης των ερωτήσεων για εξισορρόπηση του φόρτου εργασίας [38].

Μια άλλη προσέγγιση είναι αυτή των κατανεμημένων συστημάτων αρχείων, όπως το Network File System (NFS) ή το Serverless Network File Systems (xFS), που παρέχουν όμως μικρή ανεξαρτησία δεδομένων, διεπαφές αναζήτησης αρχείων χαμηλού επιπέδου και μεταβαλλόμενη απόδοση [37]. Τα συστήματα Content Distribution Networks (CDNs) καταφέρνουν να μειώσουν το φόρτο εργασίας των κεντρικών εξυπηρετητών προωθώντας τις αιτήσεις σε γειτονικά αντίγραφα. Μερικά από αυτά τα συστήματα επεξεργάζονται τις αιτήσεις στο επίπεδο του DNS και χρησιμοποιούν DHT μεθόδους για την εύρεση αντιγράφων [41]. Η χρήση όμως DHT επιφέρει και μείωση των δυνατοτήτων όσον αφορά πολύπλοκες αναζητήσεις.

3.5. Μη-εννοιολογικοί Δείκτες

Ενώ οι εννοιολογικοί δείκτες έχουν τη δυνατότητα να συσχετίζουν λέξεις κλειδιά με έγγραφα, μεταδεδομένα ή ακόμα και με ολόκληρους διαχειριστικούς τομείς, δυστυχώς συνοδεύονται και από αρνητικά στοιχεία όπως δυσκολίες στην δημιουργία αντιγράφων και στην μετακίνηση δεδομένων [42] καθώς και έλλειψη εγγυημένης αναζήτησης (δεν είναι σε θέση να εγγυηθούν ότι όταν ένα περιεχόμενο υπάρχει στο σύστημα η αναζήτηση σίγουρα θα επιστρέψει θετική). Για το λόγο αυτό, οι κατανεμημένοι πίνακες κατατεμαχισμού (DHT) έχουν προταθεί ώστε να προσφέρουν μη-εννοιολογική πρόσβαση σε μεγάλο όγκο δεδομένων και χρηστών. Οι DHT δίνουν τη δυνατότητα γρήγορης και εγγυημένης αναζήτησης σταθερών κλειδιών σε μεγάλο όγκο αρχείων, που μπορούν διαρκώς να αλλάζουν ξενιστές. Τα συστήματα αυτά έχουν σχεδιαστεί να προσφέρουν κυρίως [41]:

1. Χαμηλό βαθμό συνδέσεων με άλλους κόμβους, ώστε να υποστηρίζουν μεγάλο αριθμό αφίξεων και αποχωρήσεων κόμβων.
2. Χαμηλή διάμετρο απαιτούμενων βημάτων για κάθε λειτουργία ώστε να ελαχιστοποιούνται τα υπερκείμενα βήματα.
3. Άπληστη δρομολόγηση ώστε να εντοπίζεται η βέλτιστη διαδρομή με βάση τις τοπικές συνδέσεις κάθε κόμβου.

4. Υψηλή ανεκτικότητα ώστε να εντοπίζεται πάντα διαδρομή ακόμα και όταν κάποιοι κόμβοι ή συνδέσεις αποτυγχάνουν.

Οι σημερινοί αλγόριθμοι DHT βασίζονται σε τρεις κυρίως ερευνητικές εργασίες που έγιναν τη δεκαετία του 1990. Οι πρώτες δύο, τα δέντρα Plaxton [43] και ο συνεπής κατατεμαχισμός (Consistent Hashing) [44] δημοσιεύθηκαν σε διάστημα ενός μηνός. Η τρίτη εργασία ήταν η κλιμακώσιμη κατανεμημένη δομή δεδομένων (Scalable Distributed Data Structure, SDDS) [45], η οποία για αρκετό διάστημα είχε παραλειφθεί, παρότι σχεδιαστικά είχε πολλά κοινά σημεία με τα δομημένα ομότιμα συστήματα [46],[47],[48].

Τα Plaxton δέντρα είναι η πιο σύγχρονη από τις τρεις εργασίες και έχει επηρεάσει σημαντικά τη σχεδίαση πολλών ομότιμων συστημάτων, όπως του Pastry [46], του Tapestry [48], του Chord [47] και του Oceanstore [49] μεταξύ άλλων. Η βασική αρχή των δέντρων Plaxton είναι ότι μπορούν να εντοπίσουν αντικείμενα μέσω πινάκων δρομολόγησης σταθερού μήκους [48]. Τόσο στα αντικείμενα όσο και στους κόμβους αναθέτονται μη-εννοιολογικοί δείκτες, όπως για παράδειγμα ένα κλειδί μήκους 160 δυφίων. Εν συνεχεία, κάθε κόμβος θεωρείται η ρίζα ενός ανοιγμένου δέντρου. Κάθε μήνυμα δρομολογείται κατόπιν προς ένα αντικείμενο στόχο ταιριάζοντας διαδοχικά μεγαλύτερα επιθήματα διευθύνσεων εωσότου εντοπιστεί η ρίζα του κόμβου στόχου ή κάποιος άλλος γειτονικός κόμβος με ένα αντίγραφο του αντικειμένου στόχου. Με βάση αυτήν τη λογική δίδεται η δυνατότητα δρομολόγησης ακόμα και όταν εμφανίζονται σφάλματα ή αποτυχίες κόμβων ταιριάζοντας τη διεύθυνση με αυτήν κάποιου παραπλήσιου επιθήματος. Η σχεδίαση των δέντρων Plaxton είχαν όμως και αρκετά μειονεκτήματα [48]:

- Απαιτούσαν ολική γνώση του δικτύου για την κατασκευή του υπερκείμενου δικτύου.
- Η ρίζα κάθε κόμβου αποτελούσε μοναδικό στοιχείο βλάβης (single point of failure).
- Το σύστημα δεν υποστήριζε εισαγωγή νέων κόμβων ή αποχώρηση υπαρχόντων.
- Δεν υπήρχε κατάλληλος μηχανισμός αναζήτησης ώστε να αποφεύγονται σημεία υψηλής κίνησης.

Οι Karger κ.α εισήγαγαν το συνεπή κατατεμαχισμό ως μέθοδο επίλυσης του προβλήματος λανθάνουσας μνήμης του ιστού (web caching) [44]. Βάση αυτής, οι εξυπηρετητές ιστού μπορούσαν να χρησιμοποιούν μεθόδους κατατεμαχισμού για να τοποθετούν αντικείμενα σε μια σειρά από εξυπηρετητές κρυψίματος και κατόπιν οι χρήστες μπορούσαν μέσω της ίδιας συνάρτησης κατατεμαχισμού να εντοπίσουν τα αντικείμενα στον αρμόδιο

εξυπηρετητή κρυψίματος. Κατά την διάρκεια κανονικού κατατεμαχισμού, οι περισσότεροι δείκτες αντικειμένων μετακινούνταν όταν εξυπηρετητές κρυψίματος προστίθεντο ή διαγράφονταν. Αντίθετα, ο συνεπής κατατεμαχισμός προσέφερε ομαλή λειτουργία, ελαχιστοποιώντας τους δείκτες που μετακινούνταν κατά την αναπροσαρμογή του δικτύου εξυπηρετητών κρυψίματος, προσφέροντας παράλληλα ισοκατανομή φορτίου. Επιπλέον, ο συνεπής κατατεμαχισμός εγγυάτο ότι ο συνολικός αριθμός εξυπηρετητών κρυψίματος που ήταν υπεύθυνοι για ένα συγκεκριμένο αντικείμενο ήταν περιορισμένος. Σε αντίθεση με το γραμμικό κατατεμαχισμό του Litwin (LH*), ο οποίος απαιτούσε τα «δοχεία» να προστίθενται σειριακά, ο συνεπής κατατεμαχισμός επέτρεπε να προστίθεντο με οποιαδήποτε σειρά [44]. Ένα από τα προβλήματα που παρουσίαζε ο συνεπής κατατεμαχισμός ήταν το ποσοστό των δεικτών που μετακινούνταν κατά την είσοδο ενός νέου κόμβου [50]. Ο εκτεταμένος συνεπής κατατεμαχισμός (Extended Consistent Hashing) προτάθηκε πρόσφατα ώστε να τυχαιοποιεί τις αναζητήσεις σε όλο το φάσμα των εξυπηρετητών κρυψίματος με στόχο τη σημαντική μείωση της διασποράς φόρτου [51]. Ενδιαφέρον παρουσιάζει η δήλωση του Karger [44] ότι ο αλγόριθμός τους βασιζόταν σε έναν παλαιότερο DHT αλγόριθμο "χρησιμοποιώντας όμως ένα νέο αλγόριθμο αυτόνομου εντοπισμού ο οποίος μαθαίνει τις τοποθεσίες των ταμιευτήρων αντί να χρησιμοποιεί έναν κεντρικό κατάλογο" [52]. Αντίστοιχα, τη βασική πηγή του Devine αποτέλεσε η ερευνητική εργασία του Litwin στο SDDS και ο LH* αλγόριθμος [53]. Το SDDS ικανοποιούσε τρεις βασικές σχεδιαστικές απαιτήσεις:

- Νέα αρχεία προστίθεντο σε νέους εξυπηρετητές μόνο όταν οι υπάρχοντες είχαν ικανοποιητικό φόρτο.
- Δεν υπήρχε κάποιος κεντρικός κατάλογος.
- Οι βασικές λειτουργίες όπως εισαγωγή, αναζήτηση και διαχωρισμός δεν απαιτούσαν ποτέ ατομικές ανανεώσεις σε πολλαπλούς πελάτες.

Οι Honicky και Miller επισήμαναν πως η πρώτη απαίτηση αποτελούσε περιορισμό καθώς η επέκταση σε νέους εξυπηρετητές δεν απαιτούσε διαχειριστικό έλεγχο [54]. Ο Litwin πρόσφατα παρατήρησε πολλές ομοιότητες αλλά και διαφορές μεταξύ του LH* και του Chord [55]. Ανακάλυψε ότι και οι δύο υλοποιούσαν αναζήτηση κλειδιών. Αν και ο LH* αναφέρεται σε πελάτες και εξυπηρετητές, οι κόμβοι μπορούν να λειτουργούν ως ομότιμοι και στα δύο συστήματα. Το Chord διαχωρίζει τους κόμβους όταν ένας νέος κόμβος εισέρχεται ενώ το LH* προγραμματίζει τους διαχωρισμούς ώστε να αποφευχθεί η υπερφόρτωση. Οι λειτουργίες στο Chord απαιτούν O(logN) βήματα ενώ στο LH* απαιτούνται το μέγιστο δύο βήματα. Στο Chord οι κόμβοι αποθηκεύουν έναν μικρό αριθμό συνδέσεων ενώ στο LH* οι

εξυπηρετητές αποθηκεύουν από N/2 έως N συνδέσεις ενώ οι πελάτες από 1 έως N. Αυτός ο ισολογισμός μεταξύ του αριθμού βημάτων και του αριθμού συνδέσεων επηρεάζει τη δυναμική του συστήματος και παρουσιάζει μεγάλη ομοιότητα με τα σύγχρονα ομότιμα συστήματα ενός ή δύο βημάτων όπως ήδη έχουμε αναφέρει. Η είσοδος και αποχώρηση πελατών στο LH* δεν επηρεάζει του εξυπηρετητές του συστήματος. Με βάση το μέγεθος του δείκτη, η εισαγωγή και αποχώρηση εξυπηρετητών στο LH* ενδέχεται να προκαλέσει μεγαλύτερα προβλήματα στο LH* από ότι στο Chord. Σε αντίθεση με το Chord, το LH* έχει ένα κεντρικό σημείο βλάβης, αυτό του συντονιστή των διαχωρίσεων. Οι παραπάνω διαπιστώσεις προσφέρουν ένα μέτρο σύγκρισης του SDDS με τα ομότιμα συστήματα αλλά δυστυχώς δεν υπάρχει αναλυτική σύγκριση των δύο αυτών συστημάτων με βάση κάποια πειραματικά δεδομένα ή στατιστικά δεδομένα με βάση την απόδοσή τους. Δυστυχώς, το ίδιο πρόβλημα παρουσιάζεται και για πολλά ακόμα ομότιμα συστήματα, τα οποία εισάγουν νέες δομές και αρχιτεκτονικές αλλά δεν παραθέτουν κάποια κοινά κριτήρια για την έγκυρη σύγκρισή τους. Στη συνέχεια θα παραθέσουμε μια σειρά από απόπειρες που έχουν γίνει για τη σύγκριση των ομότιμων συστημάτων. Παρόλα αυτά, το πεδίο των ομότιμων συστημάτων βρίσκεται ακόμα σε ερευνητική εξέλιξη και δεν ενδέχεται στο άμεσο μέλλον να συμφωνηθούν κάποια κοινά αποδεκτά κριτήρια για την άμεση σύγκρισή τους. Οι υπάρχουσες συγκρίσεις συνήθως βασίζονται σε τρεις κατηγορίες κριτηρίων:

- Η απόδοσή τους σε συνθήκες έλλειψης σφαλμάτων.
- Η στατική τους εξαρτησιμότητα ως μέτρο απόδοσης προτού εφαρμοσθούν μέθοδοι επιδιόρθωσης του δικτύου.
- Η δυναμική εξαρτησιμότητα σε δυναμικά δίκτυα μεγάλου αριθμού χρηστών και περιεχομένων, όπου παρουσιάζονται διαρκή σφάλματα κόμβων.

Όσον αφορά την απόδοση σε συνθήκες έλλειψης σφαλμάτων οι Christin και Chuang ανέπτυξαν ένα μοντέλο για λανθάνουσα πιθανότητα, για υπηρεσία, για δρομολόγηση και για κόστος διατήρησης και συνέκριναν τα αποτελέσματα για γράφους De Bruijn, τόρους D-διαστάσεων όπως το CAN, για δέντρα Plaxton και για δακτυλίους Chord [56],[57]. Σε αντίθεση με τη διαδομένη άποψη για τα ομότιμα συστήματα, πρότειναν ότι οι καλά συνδεδεμένοι κεντρικοί ομότιμοι κόμβοι μπορούν να μειώσουν το συνολικό κόστος αυτών των συστημάτων. Παρότι δεν κατάφεραν να εντοπίσουν κάποιες δομημένες ομότιμες γεωμετρίες που να υπερισχύουν καθαρά σε θέμα απόδοσης, πρότειναν μια σειρά από βελτιώσεις ώστε να κάνουν το μοντέλο κόστους τους πιο ρεαλιστικό. Οι Rhea κ.α. παρουσίασαν μια εμπειρική σύγκριση απόδοσης των Tapestry και Chord [58]. Όρισαν μια σειρά από

μέτρα απόδοσης στιβαρότητας, όπως το κόστος διαρκούς εισαγωγής και αποχώρησης κόμβων, καθώς αυτές οι λειτουργίες είναι πιο δύσκολο να μετρηθούν από την απόδοση. Για αρχεία μεγάλου μεγέθους δεν βρήκαν κάποια αισθητή διαφορά στον μηχανισμό αναζήτησης. Για αρχεία μικρού μεγέθους, το Tapestry παρουσίαζε μικρότερη καθυστέρηση αναζήτησης από το Chord και ως εκ τούτου συμπέραναν ότι ο αναδρομικός αλγόριθμος αναζήτησης του Tapestry είναι πιο γρήγορος από τον αντίστοιχο του Chord. Επίσης συμπέραναν ότι ο αλγόριθμος εύρεσης γειτονικών αντιγράφων του Tapestry έχει πλεονέκτημα έναντι αυτού του Chord, αλλά παρόλα αυτά δεν παρείχε ιδανική απόδοση.

Στην κατηγορία της στατικής εξαρτησιμότητας, ο Gummadi συνέκρινε τις γεωμετρίες που βασίζονταν σε δέντρα, υπερκύβους, πεταλούδες, δακτυλίδια, XOR και υβριδικά. Σύμφωνα με τα αποτελέσματα της έρευνάς του παρουσίασε τη γεωμετρία του δακτυλίου ως αυτή που παρέχει την καλύτερη ευελιξία και προσαρμοστικότητα [59]. Οι Loguinov κ.α. συνέκριναν το Chord, το CAN και τους γράφους de Bruijn και πρότειναν ότι οι γράφοι Bruijn προσφέρουν τη βέλτιστη διάμετρο και συνδεσιμότητα [60]. Επίσης συμπέραναν ότι η κλασσική ανάλυση σφαλμάτων, δηλαδή η πιθανότητα ένας συγκεκριμένος κόμβος να μην είναι προσβάσιμος, δεν διαφοροποιεί τις τρεις αυτές δομές ως προς την προσαρμοστικότητά τους. Χρησιμοποιώντας το πλάτος διχοτόμησης (το ελάχιστος μήκος ακμών μεταξύ δύο ίσων τμημάτων) και την επικάλυψη μονοπατιού (την πιθανότητα ότι ένα εφεδρικό μονοπάτι θα εμπεριέχει τα ίδια σφάλματα κόμβων με το κύριο μονοπάτι) πρότειναν τους γράφους de Bruijn ως τους πιο προσαρμοστικούς. Οι Hsiao και King συνέκριναν το Chord, το Tapestry, το Pastry και το Tornado με βάση την ευελιξία τους, δηλαδή τον αριθμό των εναλλακτικών κόμβων που προσφέρουν για το επόμενο βήμα [61].

Οι Li κ.α. σημείωσαν ότι οι συγκρίσεις και αναλύσεις που βασίζονταν σε περιβάλλοντα έλλειψης σφαλμάτων ή στατικής εξαρτησιμότητας ήταν προκατειλημμένες προς όφελος των αλγορίθμων που παρείχαν περισσότερους δείκτες σε κάθε κόμβο [62]. Η δική τους σύγκριση αφορούσε τη δυναμική εξαρτησιμότητα και παρουσίαζε τη λανθάνουσα περίοδο (msec) σε σχέση με το μέσο διαθέσιμο εύρος ζώνης (bytes/κόμβο/sec) για το Chord, το Kelips, το Tapesty και το Kadlemia. Το διαθέσιμο εύρος ζώνης χρησιμοποιήθηκε ως μέτρο σύγκρισης καθώς θεωρήθηκε ότι είναι πιο περιοριστικό από ότι η επεξεργαστική ισχύς ή η διαθέσιμη μνήμη. Επίσης ενσωμάτωνε το κόστος κατά τη διάρκεια μαζικής αποχώρησης χρηστών (churn). Ο ισολογισμός μεταξύ λανθάνουσας περιόδου και χρήσης του διαθέσιμου εύρος ζώνης παρατηρήθηκε ότι επηρεαζόταν άμεσα από τις παραμέτρους σχεδίασης του κάθε αλγορίθμου και για αυτό συνέκριναν τη βέλτιστη καμπύλη για κάθε

αλγόριθμο χρησιμοποιώντας έναν εξομοιωτή πακέτων διακριτών συμβάντων [63]. Ενώ και οι τέσσερεις αλγόριθμοι παρουσίαζαν παρόμοια απόδοση σε συνθήκες υψηλού διαθέσιμου εύρος ζώνης, το Chord παρουσίαζε μικρότερη λανθάνουσα περίοδο σε συνθήκες χαμηλού διαθέσιμου εύρος ζώνης. Ο ρυθμός αναζήτησης και αποχώρησης κόμβων ήταν σχετικά αργός, κατανεμημένος εκθετικά με αντίστοιχες μέσες τιμές δέκα λεπτά και μία ώρα. Μερικοί σχεδιαστές ομότιμων συστημάτων ισχυρίζονται ότι αποδεικνύουν θεωρητικά την ορθότητα των συστημάτων τους [47],[64],[65]. Άλλοι δηλώνουν το προβληματισμό τους ότι «όταν πολλές εισαγωγές και αποχωρήσεις συμβαίνουν ταυτόχρονα, δεν είναι σίγουρο ότι οι πίνακες δρομολόγησης παραμένουν σε ικανοποιητική κατάσταση» [66]. Οι Lynch, Malkhi κ.α. δήλωσαν ότι δε μπορούν μεν να εγγυηθούν πλήρη συνέπεια σε ένα δίκτυο με διαρκή και δυναμικά σφάλματα αλλά παρουσίασαν μια σειρά από βελτιώσεις στον αλγόριθμο του Chord ώστε να εγγυώνται την ατομικότητα [67].

Στην συνέχεια θα παρουσιάσουμε τις βασικές γεωμετρίες που βασίζονται σε DHT και οι οποίες παρέχουν τις βασικές λειτουργίες για αναζήτηση μη-εννοιολογικών δεικτών σε υπερκείμενα ομότιμα δίκτυα. Σε κάθε κατηγορία γεωμετρίας θα παρουσιάσουμε τη βασική αρχή στην οποία στηρίζονται, τις υπάρχουσες ερευνητικές εργασίες και τα βασικά πλεονεκτήματα αλλά και μειονεκτήματά τους. Επίσης, σε κάθε κατηγορία θα παρουσιάσουμε αναλυτικότερα μία ή δύο ενδεικτικές ερευνητικές εργασίες ώστε να αναλύσουμε εις βάθος τόσο τα καινοτομικά στοιχεία που εισήγαγαν όσο και τα κενά που παρουσίαζαν. Η διατριβή αυτή θα παρουσιάσει στην συνέχεια μια καινούργια γεωμετρία για DHT ομότιμα δίκτυα, η οποία προεκτείνει τις υπάρχουσες προσπάθειες στον τομέα των DHT συστημάτων και προσφέρει μια εναλλακτική δομή η οποία είναι σε θέση να λύσει αρκετά από τα προβλήματα που παρουσιάζουν οι υπάρχουσες δομές. Για το λόγο αυτό, στην συνέχεια αυτής της ενότητας, θα παρουσιάσουμε αναλυτικά τις υπάρχουσες DHT δομές και θα επισημάνουμε σε κάθε σημείο τις ομοιότητες αλλά και τις διαφορές των συστημάτων αυτών με τη δική μας αρχιτεκτονική αλλά και τους λόγους που μας οδήγησαν στον σχεδιασμό των αλγορίθμων μας. Οι βασικές δομές (γεωμετρίες) που θα παρουσιασθούν αναλυτικά είναι οι εξής:

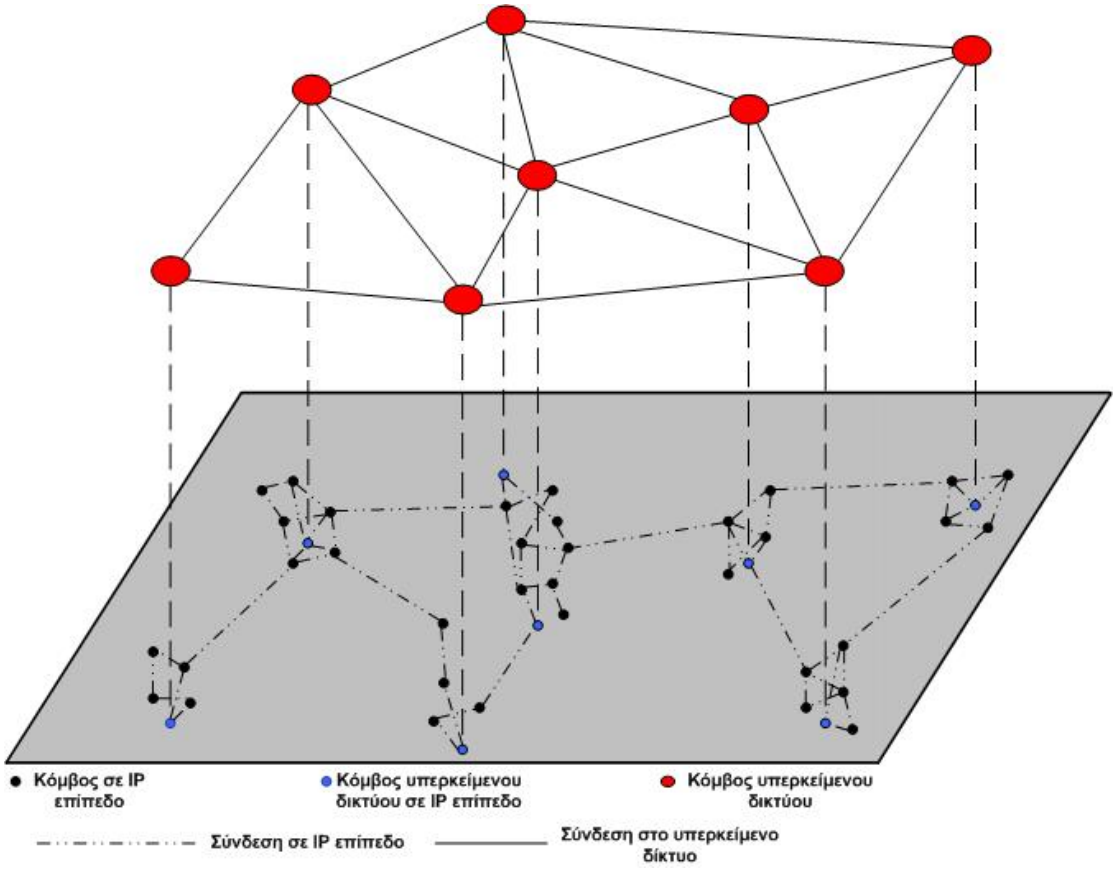
- Plaxton Δέντρα
- Δακτύλιοι
- Τόροι
- Πεταλούδες
- de Bruijn Γράφοι

- Υπερπηδώμενοι Γράφοι

Κάθε ένα από τα προτεινόμενα DHT συστήματα που θα παρουσιάσουμε υλοποιεί διαφορετικό αλγόριθμο δρομολόγησης. Σε όλους αυτούς τους αλγόριθμους υπάρχουν τόσο κοινά σημεία όσο και αρκετές προσεγγιστικές διαφορές. Δεν είναι εύκολο να χαρακτηριστεί κάποιος ως καλύτερος ή χειρότερος καθώς υπάρχουν αρκετές ιδιότητες που χαρακτηρίζουν τα DHT συστήματα, οπότε δε μπορεί να προκύψει ένα απλό μέτρο σύγκρισης. Στην συνέχεια θα εξετάσουμε αρχικά την έννοια των υπερκείμενων δικτύων, που αποτελεί κοινή βάση όλων των DHT συστημάτων, και κατόπιν θα παρουσιάσουμε τα βασικά σημεία του αλγόριθμου αναζήτησης για κάθε μία από τις αναφερόμενες γεωμετρίες DHT συστημάτων, ώστε να έχουμε μια γενική εικόνα. Τέλος θα αναφερθούμε σε διάφορα ζητήματα που είναι σημαντικά για τη δρομολόγηση σε υπερκείμενα δίκτυα, όπως η απόδοση, η ανεκτικότητα σε σφάλματα ,τα σημεία μεγάλης κίνησης, η γεωγραφία και η ετερογένεια.

3.5.1. Υπερκείμενα δίκτυα

Ο όρος υπερκείμενα δίκτυα αναφέρεται σε δίκτυα τα οποία δημιουργούν μια εικονική τοπολογία πάνω από μια φυσική τοπολογία [68]. Αυτό οδηγεί σε συστήματα όπου οι γειτονικοί κόμβοι μπορούν να απέχουν αρκετά βήματα με βάση την πραγματική τοπολογία, όπως φαίνεται στο Σχήμα 7. Τα δίκτυα αυτά δημιουργούνται στο επίπεδο εφαρμογής και εφαρμόζουν δικούς τους αλγορίθμους δρομολόγησης, χρησιμοποιώντας όμως το υπάρχον IP δίκτυο για την πραγματική μεταφορά των μηνυμάτων. Η τοπολογία αυτή δημιουργείται με βάση κάποια χαρακτηριστικά των κόμβων. Το Chord για παράδειγμα οργανώνει το δίκτυο με βάση την IP διεύθυνση των κόμβων, ενώ άλλες αρχιτεκτονικές χρησιμοποιούν τα περιεχόμενα των κόμβων για την οργάνωσή τους.



Σχήμα 7. Αρχιτεκτονική υπερκείμενου δικτύου

Τα υπερκείμενα δίκτυα προσφέρουν τέσσερα βασικά χαρακτηριστικά :

- Εγγυημένη ανάκτηση δεδομένων
- Εγγυημένο αριθμό βημάτων αναζήτησης
- Αυτόματη εξισορρόπηση κίνησης
- Αυτο-οργάνωση

Καθώς τα υπερκείμενα δίκτυα ορίζουν τους γειτονικούς κόμβους με βάση το περιεχόμενό τους, επιτρέπουν τη μετατροπή της λειτουργίας αναζήτησης από ένα πρόβλημα διάσχισης-γράφου σε μια τοπική επαναληπτική διαδικασία. Σε αυτήν τη διαδικασία, κάθε βήμα μέσα στο υπερκείμενο δίκτυο οδηγεί την αναζήτηση ένα βήμα πλησιέστερα στο αναζητούμενο κλειδί, με βάση κάποια μαθηματική συνάρτηση. Με αυτόν τον τρόπο μειώνεται η δικτυακή κίνηση και καθίστανται ντετερμινιστικές οι διαδικασίες αναζήτησης, εισαγωγής και δημοσίευσης.

Τη δομή αυτή των υπερκείμενων δικτύων μπορούμε να τη συγκρίνουμε με αυτήν των κατανεμημένων πινάκων κατατεμαχισμού, που επιτρέπουν την εισαγωγή, αναζήτηση και διαγραφή κλειδιών. Η ομοιότητα

αυτή οδήγησε στον συνδυασμό αυτών των δύο τεχνικών για τη δημιουργία συστημάτων ανταλλαγής περιεχομένων.

3.5.2. Plaxton Δέντρα

Όπως έχουμε αναφέρει, η πρώτη ερευνητική εργασία σε DHT συστήματα ήταν αυτή των Plaxton, Rajaraman και Richa, η οποία θα παρουσιασθεί στην συνέχεια αυτής της ενότητας. Στις αρχές του 2000, οι Zhao κ.α. άρχισαν τη δημιουργία ενός δομημένου συστήματος το οποίο υποστήριζε την παρουσία σφαλμάτων και πρόσφερε δυναμική τοποθέτηση αντικειμένων και δρομολόγηση (Dynamic Object Location and Routing, DOLR) με την ονομασία Tapestry [48],[64]. Οι διεπαφές αυτού του συστήματος επέτρεπαν στις εφαρμογές να ελέγχουν την τοποθέτηση αντικειμένων [2]. Η λογική του Tapestry βασιζόταν στην δρομολόγηση των Plaxton, Rajaraman και Richa (PRR) [43] αλλά επιδιόρθωνε την έλλειψη στιβαρότητας που παρουσίαζε το σύστημά τους. Η αναλυτική δομή του Tapestry θα παρουσιασθεί στην συνέχεια αυτής της ενότητας. Το Tapestry χρησιμοποιεί TCP και UDP μετρητές για να ανιχνεύσει και να ξεπεράσει προβλήματα δρομολόγησης που οφείλονται σε σφάλματα συνδέσεων ή κόμβων. Η ανεκτικότητα του Tapestry έχει δοκιμασθεί μέσω πειραματικής πλατφόρμας που απαρτιζόταν από 100 μηχανήματα και την εξομοίωση 1000 κόμβων. Με βάση αυτήν τη διάταξη μέτρησαν τις επιτυχείς αναζητήσεις και το εύρος ζώνης συντήρησης καθώς δημιουργούσαν μαζικές βλάβες και αποχωρήσεις κόμβων [2]. Ένα άλλο σημαντικό σύστημα, που επίσης βασιζόταν στην εργασία του Plaxton και το οποίο θα παρουσιασθεί αναλυτικά στην συνέχεια, είναι το Pastry [46]. Το Pastry διαφέρει ως προς το Tapestry μόνο στην μέθοδο με την οποία διαχειρίζεται τις τοπικές διευθύνσεις και τα αντίγραφα. Οι Mahajan, Castry κ.α. ανέλυσαν την αξιοπιστία του Pastry με βάση το κόστος διαχείρισης σε σχέση με τις παραμέτρους που παρέχει το Pastry [69] και συμπέραναν ότι οι ανησυχίες σχετικά με το υπερβολικό κόστος διαχείρισης σε περιβάλλον μαζικών σφαλμάτων και αποχωρήσεων ήταν αβάσιμες. Παρόλα αυτά, συνέστησαν μια σειρά βελτιώσεων για βελτίωση της απόδοσης και της αξιοπιστίας του πρωτοκόλλου.

Οι Rhea Geels κ.α. συμπέραναν μετά από μια δικιά τους σειρά αναλύσεων ότι τα υπάρχοντα DHT συστήματα δεν λειτουργούν ομαλά σε συνθήκες μαζικών βλαβών και αποχωρήσεων [70]. Βασιζόμενοι σε μια υλοποίηση του Pastry του πανεπιστημίου Rice ανακάλυψαν ότι οι περισσότερες αναζητήσεις αποτυγχάνουν όταν παρουσιάζονται μαζικά σφάλματα και αποχωρήσεις στο δίκτυο. Με βάση μια σειρά υποθέσεων συμπέραναν ότι οι κόμβοι που δεν παραμένουν στο δίκτυο για μεγάλο

χρονικό διάστημα συχνά αποχωρούν προτού οι αναζητήσεις τους τελειώσουν, χωρίς όμως να παρέχουν διαπιστευτήρια για αυτήν τη θεωρία τους. Παρόλα αυτά, παρουσιάζει αρκετό ενδιαφέρον η κατηγοριοποίηση που κάνανε σχετικά με τη σχεδίαση των DHT αλγορίθμων όσον αφορά τη συμπεριφορά τους σε περιβάλλον μαζικής αποχώρησης και σφαλμάτων. Οι κατηγορίες αυτές ήταν:

- Αντιδραστική συναρτήσει περιοδικής ανάκτησης κόμβων
- Εξωχρονισμός αναζητήσεων
- Επιλογή τοπικών γειτόνων

Στην ανάλυσή τους αναφέρουν ότι η αντιδραστική ανάκτηση κόμβων προσθέτει κίνηση σε ήδη κορεσμένες συνδέσεις και για αυτό χρησιμοποίησαν στον σχεδιασμό τους τη μέθοδο της περιοδικής ανάκτησης. Για τον εξωχρονισμό των αναζητήσεων χρησιμοποίησαν έναν εκθετικά σταθμισμένο μεταβλητό μέσο όρο για τις απαντήσεις κάθε γείτονα, σε αντίθεση με μεθόδους που χρησιμοποιούν είτε σταθερούς εξωχρονισμούς είτε σχήματα με «εικονικές συντεταγμένες». Τέλος, για την επιλογή των τοπικών γειτόνων, πρότειναν τη χρήση «καθολικής δειγματοληψίας» σε αντίθεση με την απλή δειγματοληψία γειτόνων ή τη χρήση «αντίστροφων γειτόνων». Οι Castro, Costa κ.α. διέψευσαν τη δήλωση ότι τα DHT συστήματα δεν λειτουργούν ομαλά σε συνθήκες μαζικών σφαλμάτων και αποχωρήσεων [71] υλοποιώντας μεθόδους για συνεχή αναζήτηση και επιδιόρθωση. Το σύστημά τους MSPastry επετύγχανε μικρότερα μονοπάτια δρομολόγησης και μικρό φόρτο διαχείρισης, το οποίο ήταν λιγότερο από μισό μήνυμα ανά δευτερόλεπτο ανά κόμβο.

Μερικά μεταγενέστερα συστήματα βασισμένα σε Plaxton δέντρα είναι το Kademlia και το Willow. Οι δημιουργοί του Kademlia [65] το περιέγραψαν ως όμοιο του Pastry αλλά με τη διαφορά ότι χρησιμοποιεί την XOR συνάρτηση για τη μέτρηση των αποστάσεων. Λόγω αρκετών ομοιοτήτων που παρουσιάζει το Kademlia με τη δική μας αρχιτεκτονική, ακολουθεί αναλυτική περιγραφή του στην συνέχεια αυτής της ενότητας. Το Willow χρησιμοποιεί επίσης την XOR συνάρτηση [72] για την υλοποίηση ενός πρωτοκόλλου διαχείρισης δέντρων το οποίο συνδέει σπασμένα τμήματα ενός δέντρου. Ενώ οι περισσότεροι DHT αλγόριθμοι προσθέτουν χρήστες σε υπάρχοντες δομές, το Willow ενώνει ή επιδιορθώνει δενδρικές δομές σε O(logN) παράλληλες λειτουργίες.

Στην συνέχεια ακολουθεί η αναλυτική παρουσίαση των πιο γνωστών και σημαντικών ομότιμων συστημάτων που βασίζονται σε Plaxton δέντρα, αυτών των Plaxton, Tapestry, Pastry και Kademlia.

3.5.2.1. Plaxton

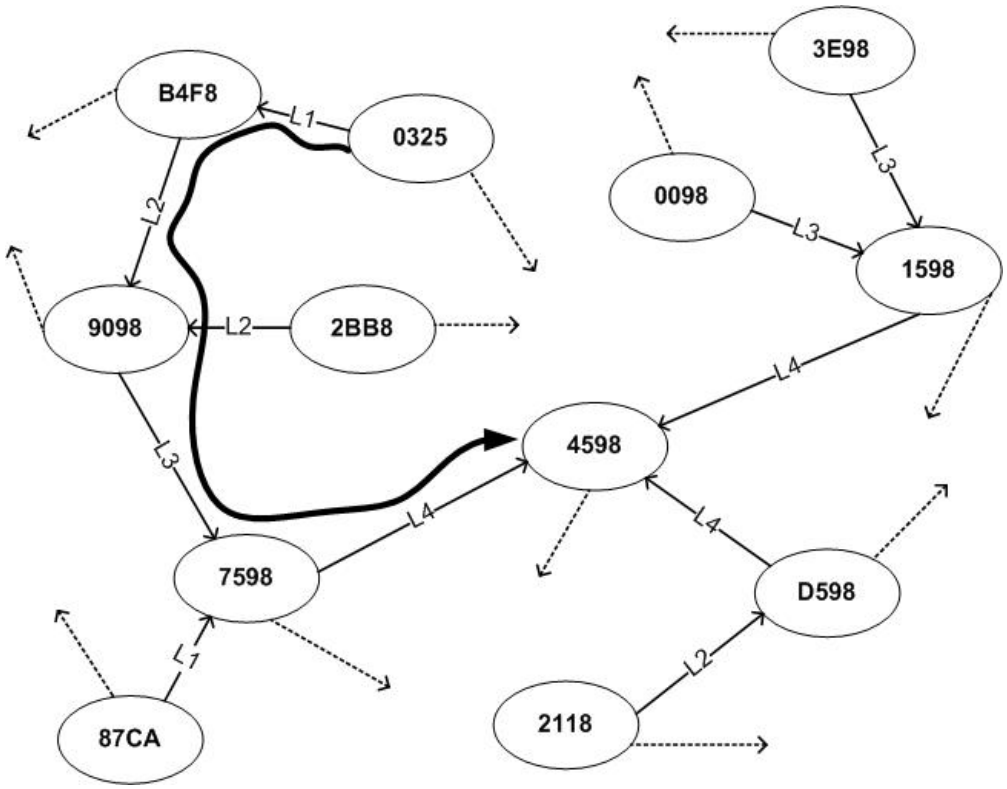
Οι Plaxton, Rajaraman και Richa [43] δημιούργησαν ίσως τον πρώτο αλγόριθμο δρομολόγησης που μπορούσε να εφαρμοστεί στα DHT συστήματα. Αν και δεν αναπτύχθηκε για χρήση σε ομότιμα συστήματα, καθώς θεωρεί ένα στατικό πληθυσμό κόμβων, παρέχει αποδοτική δρομολόγηση όσον αφορά την εύρεση κλειδιών. Ο αλγόριθμος διορθώνει σε κάθε βήμα ένα ακόμα ψηφίο από το αναγνωριστικό του κλειδιού, πχ. αν ο κόμβος με αναγνωριστικό 568932 λάβει μια αίτηση για εύρεση του κλειδιού 568321, το οποίο έχει τα τρία πρώτα ψηφία κοινά, τότε ο αλγόριθμος προωθεί την εύρεση σε έναν κόμβο με τέσσερα κοινά ψηφία (πχ. κόμβο 568398). Για να είναι αυτό εφικτό πρέπει κάθε κόμβος να έχει ως γείτονες κόμβους που διαφέρουν ένα ψηφίο ως προς αυτόν. Για ένα σύστημα με n κόμβους, κάθε κόμβος έχει O(logn) γείτονες. Καθώς ένα ψηφίο διορθώνεται κάθε φορά, κάθε αναζήτηση απαιτεί το πολύ O(logn) υπερκείμενα βήματα.

Ένα επιπλέον χαρακτηριστικό του αρχικού αυτού αλγορίθμου είναι ότι αν οι καθυστερήσεις (ή «αποστάσεις» όπως αποκαλούνται συνήθως) μεταξύ των κόμβων είναι γνωστές, τότε οι πίνακες δρομολόγησης μπορούν να επιλεγούν έτσι ώστε η απόσταση μεταξύ κόμβων στο υπερκείμενο δίκτυο να διαφέρει κατά σταθερό παράγοντα από την πραγματική απόσταση των κόμβων.

3.5.2.2. Tapestry

Ο αλγόριθμος του Tapestry [2] αποτελεί εφαρμογή των ιδεών του Plaxton et al. για ομότιμα συστήματα, με μερικές φυσικά τροποποιήσεις που απαιτεί το δυναμικό περιβάλλον των ομότιμων συστημάτων.

Στο Tapestry ο αλγόριθμος προσπαθεί να ταιριάξει ένα ψηφίο κάθε φορά αρχίζοντας από τα δεξιά και προχωρώντας αριστερά. Κάθε κόμβος N έχει έναν πίνακα από γειτονικούς κόμβους με πολλαπλά επίπεδα, όπου κάθε επίπεδο αναπαριστά ένα κοινό επίθεμα μέχρι κάποιο ψηφίο στο αναγνωριστικό. Σε κάθε επίπεδο περιέχονται πολλοί κόμβοι με κοινό επίθεμα, όπου το στοιχείο i στο επίπεδο j αποτελεί το αναγνωριστικό και την τοποθεσία του πλησιέστερου κόμβου με j-1 τελευταία ψηφία κοινά με το αναγνωριστικό του κόμβου, και το ψηφίο στην θέση j από το τέλος να είναι i. Για παράδειγμα, το 5ο στοιχείο στο 4ο επίπεδο του κόμβου 67FE23 είναι ο πλησιέστερος κόμβος που το αναγνωριστικό του τελειώνει σε 5E23.



Σχήμα 8. Αλγόριθμος δρομολόγησης από τον κόμβο 0325 προς τον κόμβο 4598

Ο αλγόριθμος αυτός βασίζεται στην εύρεση του επόμενου κοινού ψηφίου με αποτέλεσμα τα μέγιστα βήματα εύρεσης για κάθε κλειδί να είναι σταθερά. Αν οι πίνακες δρομολόγησης του κάθε κόμβου είναι συνεπείς (το οποίο βέβαια δεν θεωρείται πάντα δεδομένο) τότε σε ένα σύστημα όπου τα αναγνωριστικά των κόμβων και των κλειδιών προέρχονται από έναν κοινό χώρο και αποτελούνται από N στοιχεία με βάση β, τότε απαιτούνται το μέγιστο $\log_\beta N$ βήματα για την εύρεση κάθε κλειδιού. Καθώς κάθε κόμβος θεωρεί το προηγούμενο βήμα σωστό, απαιτείται η γνώση β κόμβων σε κάθε επίπεδο, με αποτέλεσμα το σύνολο των γειτονικών κόμβων που βρίσκονται στην μνήμη να είναι $\beta * \log_\beta N$.

Στο Σχήμα 8 μπορούμε να παρατηρήσουμε τη διαδρομή που ακολουθεί ένα μήνυμα για να δρομολογηθεί σωστά στον προορισμό του, ενώ στον Πίνακας 2 απεικονίζεται ένας τυπικός πίνακας δρομολόγησης για έναν κόμβο του αλγορίθμου.

0642	x042	xx02	xxx0
1642	x142	xx12	xxx1
2642	x242	xx22	xxx2
3642	x342	xx32	xxx3
4642	x442	xx42	xxx4
5642	x542	xx52	xxx5
6642	x642	xx62	xxx6
7642	x7142	xx72	xxx7

Πίνακας 2. Πίνακας δρομολόγησης για τον κόμβο 0642

3.5.2.3. Pastry

Στο Pastry [1] κάθε κόμβος έχει ένα αναγνωριστικό από 128-bits, το οποίο χρησιμοποιείται για να τοποθετηθεί ο κόμβος σε έναν κυκλικό χώρο, που οριοθετείται μεταξύ 0 και 2^{128-1}. Το αναγνωριστικό του κόμβου δίνεται αυτόματα όταν ο κόμβος εισέρχεται στο σύστημα μέσω κάποιας συνάρτησης κατατεμαχισμού, ώστε τα αναγνωριστικά να είναι ομοιόμορφα κατανεμημένα και ανεξάρτητα από τη γεωγραφική τοποθεσία ή το περιεχόμενο που διαθέτουν.

Όπως και στο Tapestry, κάθε αναζήτηση προχωράει προωθώντας το μήνυμα αναζήτησης στον επόμενο κόμβο που έχει τουλάχιστον ένα ψηφίο του αναγνωριστικού του πλησιέστερο σε σχέση με το κλειδί προς αναζήτηση. Έτσι, αν υπάρχουν N κόμβοι στο σύστημα, τότε μια αναζήτηση χρειάζεται το μέγιστο $\log_{2^b} N$ βήματα, όπου b είναι μια παράμετρος (με τυπική τιμή 4). Επίσης η αναζήτηση μπορεί να είναι επιτυχής ακόμα και όταν κόμβοι εγκαταλείπουν το σύστημα απότομα εφόσον ο αριθμός αυτών δεν ξεπερνά το όριο των L/2 γειτονικών κόμβων ταυτόχρονα, όπου L είναι επίσης μια παράμετρος με τυπική τιμή 16 ή 32.

Κάθε κόμβος του Pastry διατηρεί έναν πίνακα δρομολόγησης, ένα σετ γειτόνων και ένα σετ φύλλου. Ο πίνακας δρομολόγησης κάθε κόμβου περιέχει $\log_{2^b} N$ σειρές με 2^b-1 στοιχεία η καθεμία. Τα 2^b-1 στοιχεία στην σειρά *n* του πίνακα δρομολόγησης αναφέρονται σε κόμβους που το αναγνωριστικό τους έχει τα πρώτα *n* ψηφία κοινά με τον κόμβο αλλά το *n+1* ψηφίο έχει ένα από τα 2^b-1 πιθανά ψηφία εκτός από το *n+1* ψηφίο του παρόντα κόμβου. Η παράμετρος b καθορίζει τον αριθμό των κόμβων που αποθηκεύονται στον πίνακα δρομολόγησης και τον αριθμό των βημάτων που απαιτούνται για να

γίνει μια επιτυχής αναζήτηση. Όσο αυξάνει ο πίνακας δρομολόγησης τόσο μειώνονται τα βήματα που απαιτούνται και το αντίστροφο.

Κόμβος 10233102

Σετ φύλλου			
Μικρότερα		**Μεγαλύτερα**	
10233033	10233021	10233120	10233122
10233001	10233000	10233230	10233232
Πίνακας δρομολόγησης			
-0-2212102	**1**	-2-2301203	-3-1203203
0	1-1-301233	1-2-230203	1-3-021022
10-0-31203	10-1-32102	**2**	10-3-23302
102-0-0230	102-1-1302	102-2-2302	**3**
1023-0-322	1023-1-000	1023-2-121	**3**
10233-0-01	**1**	10233-2-32	
0		102331-2-0	
		2	
Σετ γειτόνων			
13021022	10200230	11301233	31301233
02212102	22301203	31203203	33213321

Πίνακας 3. Τυπική δομή των αποθηκευμένων πληροφοριών σε ένα κόμβο Pastry

Το σετ γειτόνων περιέχει τα αναγνωριστικά των M κόμβων που βρίσκονται πλησιέστερα στον κόμβο, με βάση μια συνάρτηση συσχέτισης που συγκρίνει τα αναγνωριστικά τους. Το σετ αυτό δεν χρησιμοποιείται για τη δρομολόγηση αλλά για λόγους τοπικής συσχέτισης, στο οποίο θα αναφερθούμε στο τέλος του κεφαλαίου αυτού. Το σετ φύλλου τέλος περιέχει τους L/2 κόμβους με αναγνωριστικά πλησιέστερα στον κόμβο με μεγαλύτερη αριθμητική τιμή και τους L/2 κόμβους με αναγνωριστικά πλησιέστερα στον κόμβο με μικρότερη αριθμητική τιμή. Οι κόμβοι αυτοί χρησιμοποιούνται στην δρομολόγηση των αναζητήσεων. Τυπικές τιμές για το L και το M είναι 2^b ή $2*2^b$. Στον Πίνακας 3 φαίνεται η τυπική δομή των πληροφοριών που αποθηκεύονται σε έναν κόμβο του συστήματος Pastry. Ο πίνακας αναφέρεται σε έναν κόμβο με αναγνωριστικό 10233102, με παραμέτρους b = 2 και L = 8. Όλοι οι αριθμοί είναι με βάση το 4. Στον πίνακα δρομολόγησης τα

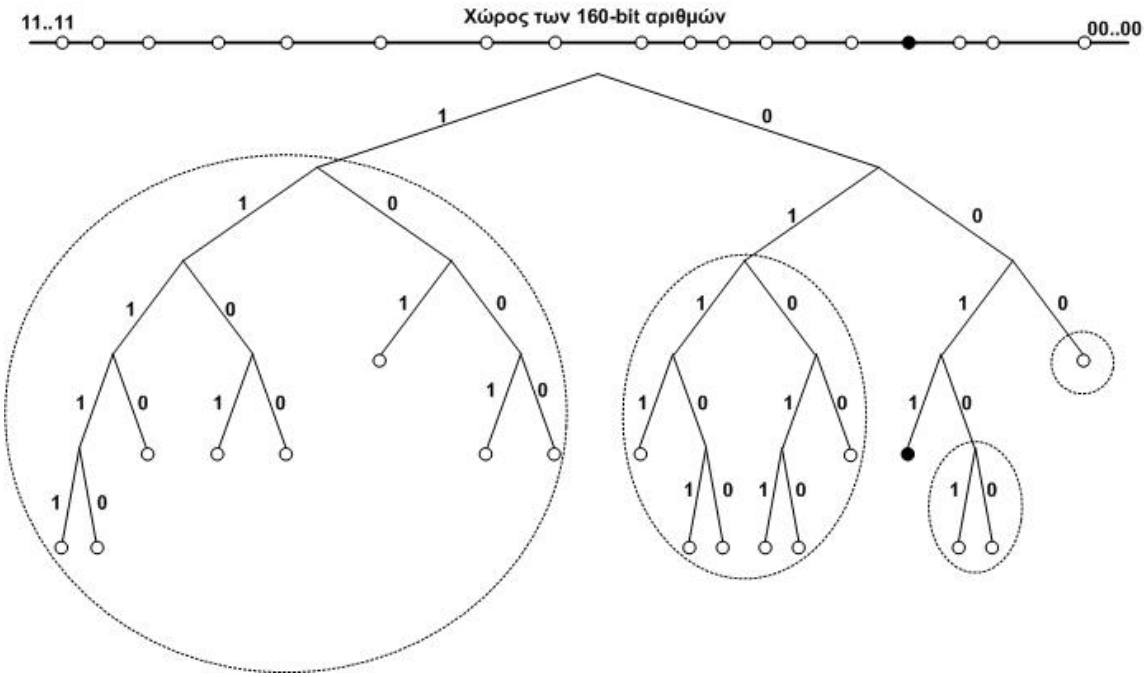
σκιαγραφημένα νούμερα αναφέρονται στο ψηφίο του κόμβου και τα αναγνωριστικά των άλλων κόμβων εμφανίζονται με τη δομή «κοινό πρόθεμα - επόμενο ψηφίο – υπόλοιπο αναγνωριστικό».

Η διαδικασία δρομολόγησης στο Pastry είναι ως εξής. Όταν καταφθάνει μια αναζήτηση πρώτα ελέγχεται αν το κλειδί ανήκει σε έναν από τους κόμβους στο σετ φύλλου και προωθείται σε αυτόν με το πλησιέστερο προς το κλειδί αναγνωριστικό. Εάν δεν ανήκει στην αρμοδιότητα κανενός από αυτούς τους κόμβους τότε προωθείται στον κόμβο που βρίσκεται στον πίνακα δρομολόγησης και έχει τουλάχιστον ένα περισσότερο ψηφίο κοινό με το κλειδί από ότι ο ίδιος ο κόμβος. Αν δεν υπάρχει τέτοιο ψηφίο ή δεν είναι δυνατή η επικοινωνία με αυτόν τον κόμβο, τότε προωθείται στον κόμβο που βρίσκεται στον πίνακα δρομολόγησης και έχει τον ίδιο αριθμό ψηφίων κοινά με τον παρόντα κόμβο αλλά είναι αριθμητικά πλησιέστερος προς το κλειδί.

3.5.2.4. Kademlia

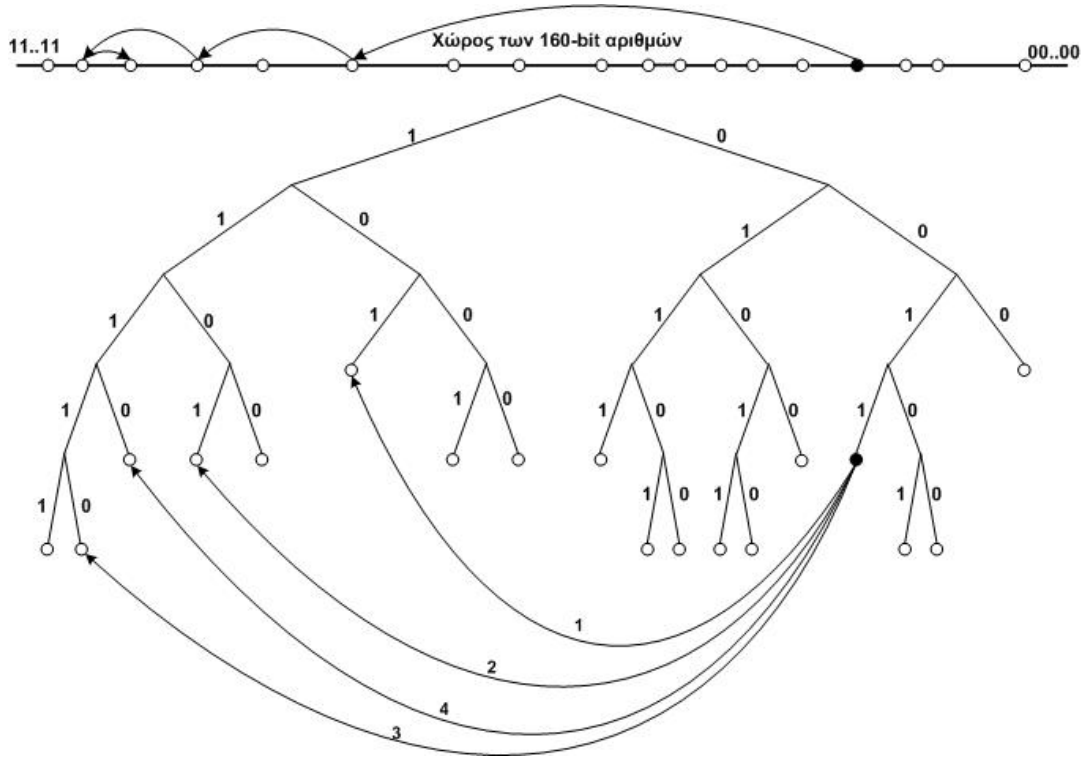
Στο σύστημα Kademlia [65] χρησιμοποιείται πάλι συνάρτηση κατατεμαχισμού 160-bit για τη δημιουργία των αναγνωριστικών των κόμβων και των κλειδιών. Η διαφορά στον αλγόριθμό του έγκειται σε δύο σημεία. Πρώτον, διαχειρίζεται τους κόμβους ως φύλλα σε δυαδικό δέντρο, και δεύτερον, χρησιμοποιεί την *XOR* συνάρτηση για να συγκρίνει την απόσταση μεταξύ των αναγνωριστικών κόμβων ή κλειδιών.

Στο Σχήμα 9 φαίνεται ένα παράδειγμα δέντρου Kademlia. Η θέση κάθε κόμβου ορίζεται από το ελάχιστο μοναδικό πρόθεμα του αναγνωριστικού του. Για κάθε κόμβο, το δέντρο χωρίζεται σε μια σειρά από διαδοχικά μικρότερα υποδέντρα, που δεν τον περιέχουν. Το μεγαλύτερο από αυτά περιέχει το μισό δέντρο. Το επόμενο υποδέντρο περιέχει το μισό από το υπόλοιπο υποδέντρο και ούτω καθεξής. Στο Σχήμα 9 τα υποδέντρα για τον κόμβο 0011 υποδεικνύονται μέσα στους κύκλους.



Σχήμα 9. Δυαδικό δέντρο Kademlia και τα υποδέντρα του κόμβου 0011

Ο αλγόριθμος του Kademlia προϋποθέτει ότι κάθε κόμβος γνωρίζει τουλάχιστον έναν κόμβο από κάθε υποδέντρο του, εφόσον υπάρχει τέτοιος κόμβος. Στο Σχήμα 10 φαίνεται πώς με αυτήν την προϋπόθεση γίνεται η αναζήτηση. Ο κόμβος ρωτώντας διαρκώς κοντινότερους κόμβους από τα υποδέντρα του (μέσω των υποδέντρων και αυτών) καταφέρνει να πλησιάζει όλο και περισσότερο στον σωστό κόμβο.



Σχήμα 10. Αναζήτηση στο δέντρο Kademlia

Για να μπορέσει να δρομολογήσει τις αναζητήσεις ο κάθε κόμβος διατηρεί έναν πίνακα διευθύνσεων από κόμβους με αναγνωριστικά που απέχουν, με βάση τη συνάρτηση απόστασης *XOR*, 2^i έως 2^{i+1}. Αυτοί οι πίνακες ονομάζονται *k-buckets* και μπορούν να διατηρούν έως *k* στοιχεία (τυπική τιμή *k=20*). Οι πίνακες αυτοί ανανεώνονται όποτε λαμβάνουν μήνυμα από ένα νέο κόμβο και χρησιμοποιείται η πολιτική της διαγραφής του πιο παλιού κόμβου με τον οποίο υπήρχε επικοινωνία, χωρίς όμως να διαγράφονται ενεργοί κόμβοι.

3.5.3. Δακτύλιοι

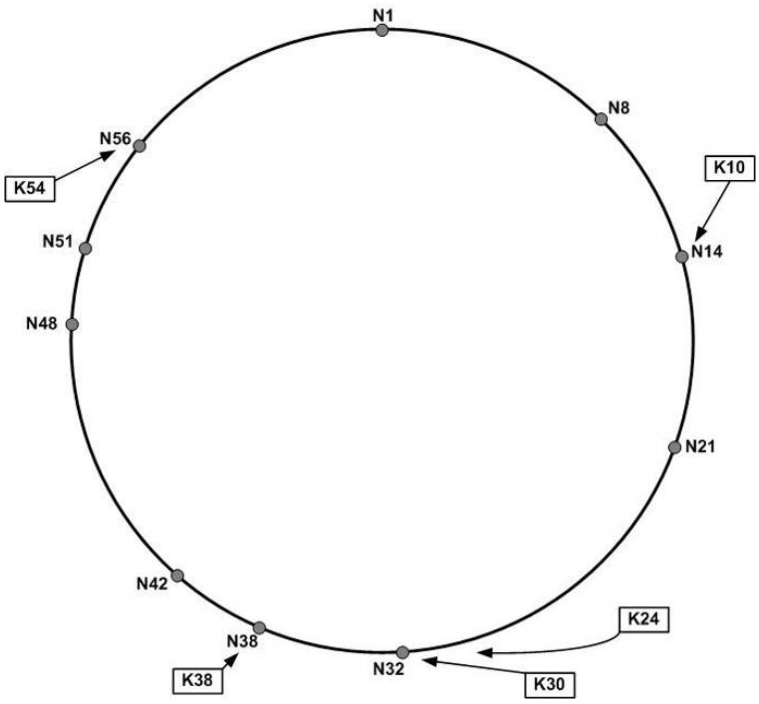
Το Chord αποτελεί το πιο γνωστό DHT σύστημα που βασίζεται σε τοπολογία δακτυλίου και για το λόγο αυτό θα παρουσιασθεί αναλυτικά στην συνέχεια αυτής της ενότητας. Το Chord υποστηρίζει μια βασική λειτουργία, αυτή της εύρεσης ενός κόμβου με βάσει ένα κλειδί και χρησιμοποιεί συνεπή κατατεμαχισμό για τη διανομή των κλειδιών κατά την είσοδο και αποχώρηση των κόμβων στο δακτύλιο. Στην αρχική παρουσίαση του Chord, οι δημιουργοί του είχαν αμφιβολίες για την ανεκτικότητά του σε περιβάλλον υψηλών σφαλμάτων και αποχωρήσεων [47], επισημαίνοντας ότι η ανάλυση του δικτύου τους κατά την αποχώρηση κόμβων δεν αποδεικνύει την ομαλή λειτουργία του και σε πιο ακραίες συνθήκες. Μια άλλη εργασία έθεσε το ζήτημα του ρυθμού εκτέλεσης της μεθόδους ανάκτησης γειτόνων κατά την αποχώρηση χρηστών σε ακραίες συνθήκες [73]. Οι Stoica, Morris κ.α. μοντελοποίησαν μια σειρά από ρυθμούς εισόδου και αποχώρησης από το δίκτυο για διαφορετικούς ρυθμούς εκτέλεσης του μηχανισμού ανάκτησης και επιδιόρθωσης για ένα Chord δίκτυο αποτελούμενο από 1000 κόμβους. Κατά τη μελέτη τους μέτρησαν τον αριθμό των εξωχρονισμών καθώς και τη διακύμανση των αναζητήσεων με βάση τη βέλτιστη διαδρομή (σε σχέση με την πραγματοποιηθέντα). Μέσα από την εργασία τους απέδειξαν το πλεονέκτημα χρήσης αναδρομικών αναζητήσεων σε σχέση με τις επαναληπτικές, επισημαίνοντας όμως ότι υπάρχουν ακόμα περιθώρια βελτίωσης.

Οι Liben-Nowell, Balakrishnan κ.α. πρότειναν την βελτιστοποίηση του ρυθμού επανάκτησης γειτόνων με βάση τη χρήση μετρήσεων της συμπεριφοράς των κόμβων [73] χωρίς όμως να υλοποιήσουν την πρότασή τους. Οι Alima, El-Ansary κ.α. θεώρησαν το κόστος επικοινωνίας του μηχανισμού επανάκτησης του Chord υπερβολικό [74]. Δύο επιπλέον ζητήματα στιβαρότητας τους οδήγησαν στην σχεδίαση της Κατανεμημένης Κ Αναζήτησης (Distributed K-ary Search, DKS), την οποία θα παρουσιάσουμε αναλυτικά στην συνέχεια. Το πρώτο ζήτημα αφορούσε την εξέλιξη του

συστήματος ώστε να προσφέρει τη βέλτιστη ισορροπία μεταξύ του αριθμού των κόμβων, τον αριθμό των απαιτούμενων βημάτων για επιτυχή αναζήτηση και το μέγεθος του πίνακα δρομολόγησης. Το δεύτερο ζήτημα αφορούσε την αξιοπιστία των αναζητήσεων, καθώς θεωρούσαν ότι κάθε ομότιμο δίκτυο θα πρέπει να εγγυάται την εύρεση ενός κλειδιού που έχει δημοσιευθεί. Οι Alima, El-Ansary κ.α. διατύπωσαν ότι τα ομότιμα συστήματα που χρησιμοποιούν μηχανισμούς ενεργής διόρθωσης, όπως το Chord, το Pastry και το Tapesty, δεν είναι σε θέση να εγγυηθούν επιτυχείς αναζητήσεις. Η αρχιτεκτονική που σχεδίασαν επιτύγχανε να εγγυηθεί επιτυχείς αναζητήσεις κατά την ταυτόχρονη είσοδο ή αποχώρηση έως f κόμβων, όπου το f ήταν διαρθρώσιμο.

3.5.3.1. Chord

Στον αλγόριθμο που παρουσιάζεται στο Chord [47], οι κόμβοι και τα κλειδιά αναπαρίστανται όπως και στους υπόλοιπους αλγόριθμους μέσω αναγνωριστικών που ανατίθενται από μια συνάρτηση κατατεμαχισμού (την SHA-1 [75] στην συγκεκριμένη περίπτωση, με είσοδο το IP του κόμβου και το κλειδί του περιεχομένου). Το Chord τοποθετεί τους κόμβους σε έναν κύκλο αναγνωριστικών με μέγεθος 2^m. Κάθε κλειδί k αποθηκεύεται στον κόμβο που έχει το ίδιο αναγνωριστικό ή τον ακριβώς επόμενο (εφόσον δεν υπάρχει ταύτιση). Αυτό ο κόμβος αποκαλείται ο διάδοχος του κλειδιού k και αναπαρίσταται ως successor(k). Καθώς η διάταξη είναι κυκλική, ο successor(k) αποτελεί τον κόμβο που βρίσκεται δεξιόστροφα από τη θέση k στον Chord δακτύλιο (όπως αποκαλείται από τους συγγραφείς).

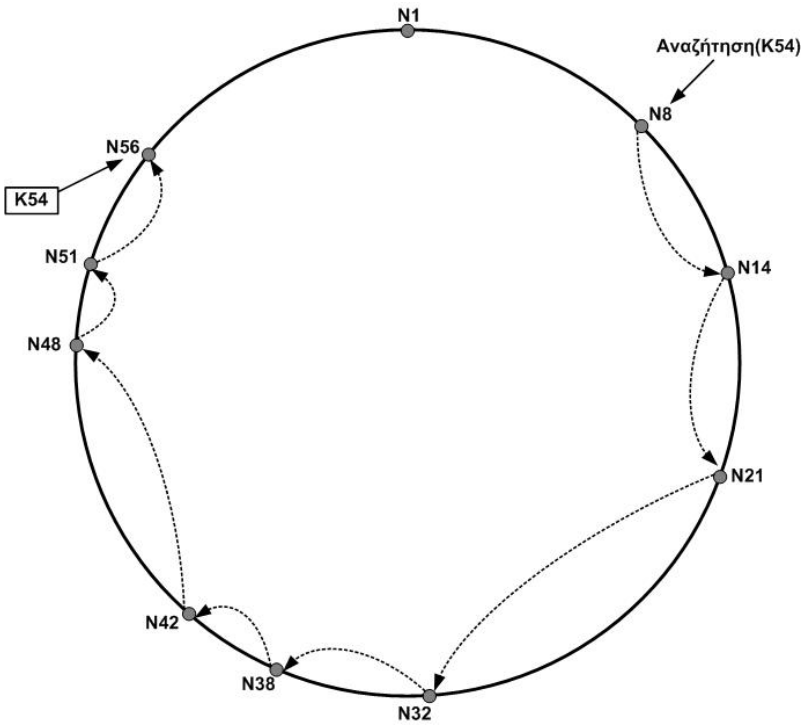


Σχήμα 11. Ένας τυπικός δακτύλιος με 10 κόμβους και 5 κλειδιά

Στο Σχήμα 11 φαίνεται ένας Chord δακτύλιος για $m = 6$. Ο δακτύλιος έχει 10 κόμβους και πέντε αποθηκευμένα κλειδιά. Ο διάδοχος του αναγνωριστικού 10 είναι ο κόμβος 14, οπότε το κλειδί 10 θα αποθηκευτεί στον κόμβο 14. Ομοίως τα κλειδιά 24 και 30 θα αποθηκευτούν στον κόμβο 32, το κλειδί 38 στον κόμβο 38 και το κλειδί 54 στον κόμβο 56.

Για να διατηρηθεί η σωστή δομή του δακτυλίου, όποτε ένας νέος κόμβος *n* εισέρχεται στο σύστημα (και αφού τοποθετηθεί στην σωστή θέση και μάθει τον προηγούμενο και τον επόμενο κόμβο στην διάταξη του δακτυλίου) μερικά κλειδιά που προηγουμένως ήταν αποθηκευμένα στον διάδοχο του *n* μεταφέρονται στον *n*. Όταν ο κόμβος *n* εξέλθει από το σύστημα, όλα τα κλειδιά του μεταφέρονται στον διάδοχό του. Στο Σχήμα 11 αν ένας κόμβος με αναγνωριστικό 26 εισέλθει στο σύστημα θα αποκτήσει το κλειδί 24 από το διάδοχό του (τον κόμβο 32).

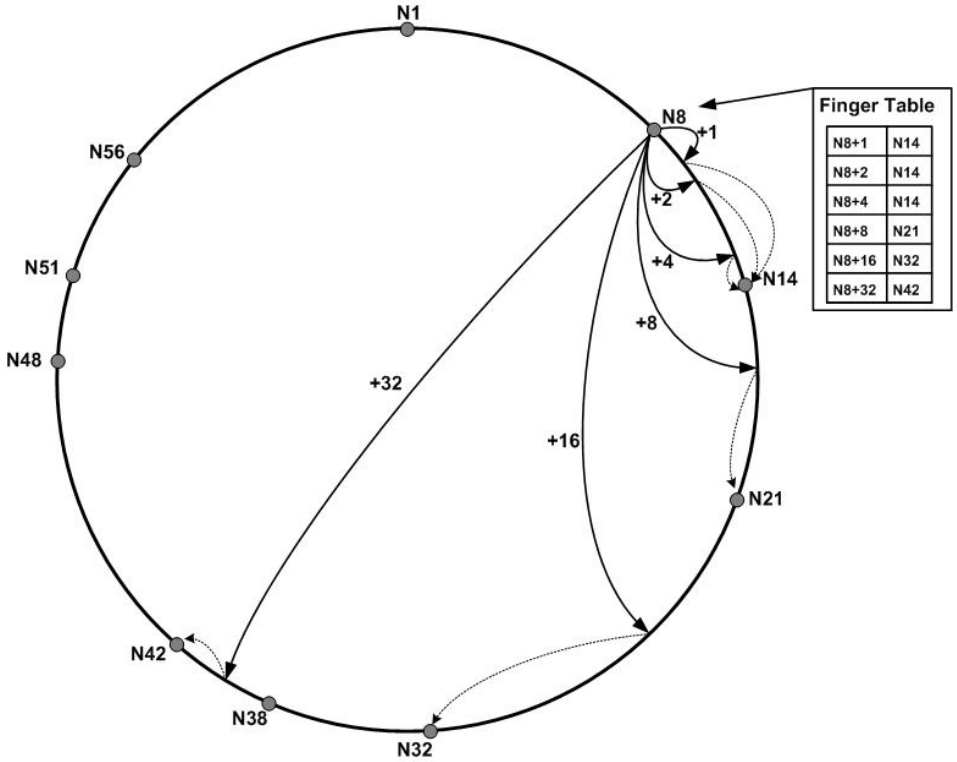
Η αναζήτηση στο Chord γίνεται με δύο τρόπους. Ο απλός τρόπος απαιτεί μόνο τη γνώση του διαδόχου κάθε κόμβου. Όποτε γίνεται μια αναζήτηση, αν το κλειδί είναι μεγαλύτερο από το αναγνωριστικό του κόμβου (και δεν περιέχεται στα αποθηκευμένα κλειδιά του κόμβου) τότε προωθείται στον διάδοχο του κόμβου. Μια τέτοια απλή αναζήτηση φαίνεται στο Σχήμα 12 όπου ο κόμβος 8 αναζητά το κλειδί 54. Με τη διαδοχική αναζήτηση μέσω των διαδόχων τελικά το κλειδί βρίσκεται αποθηκευμένο στον κόμβο 56.



Σχήμα 12. Απλή αναζήτηση μέσω της χρήσης απλώς των διαδόχων

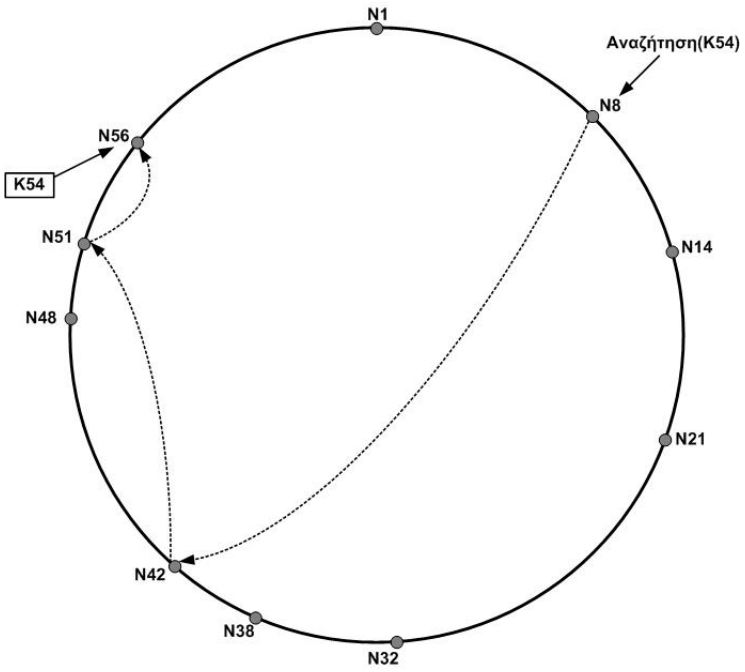
Ο δεύτερος τρόπος απαιτεί τη χρήση ενός πίνακα αποθήκευσης άλλων κόμβων από κάθε κόμβο, που αποκαλείται *finger table.* Αν *m* είναι ο αριθμός

των bits στο αναγνωριστικό των κόμβων και των κλειδιών τότε κάθε κόμβος διατηρεί στην μνήμη του τη γνώση άλλων *m* το πολύ κόμβων. Το στοιχείο στην θέση *i* στον *finger table* περιέχει τον πρώτο κόμβο *s* που διαδέχεται τον *n* κατά τουλάχιστον 2^{i-1} στον δακτύλιο, δηλ. $s = successor(n+2^{i-1})$. Στο Σχήμα 13 φαίνεται ο finger table του κόμβου 8.



Σχήμα 13. Ο *finger table* του κόμβου 8

Όταν γίνεται αναζήτηση μέσω του finger table αναζητείται απευθείας από τον κόμβο το αναγνωριστικό του κόμβου που διαδέχεται το κλειδί (με αποτέλεσμα να μειώνονται τα απαιτούμενα βήματα). Στο Σχήμα 14 φαίνεται η αναζήτηση του κλειδιού 54 από τον κόμβο 8. Η αναζήτηση μέσω του πίνακα προχωράει κατευθείαν στον κόμβο 42 και από εκεί στον 51 (μέσω πάλι του πίνακα) και εν συνεχεία βρίσκεται ο κόμβος απευθείας από το διάδοχο κόμβο.



Σχήμα 14. Αναζήτηση με χρήση του finger table

3.5.3.2. DKS

Το σύστημα DKS σχεδιάστηκε από τους Alima κ.α. [74] ως ένα πλήρως κατανεμημένο υπερκείμενο δίκτυο που χαρακτηρίζεται από τρεις παραμέτρους: το μέγιστο αριθμό N κόμβων που μπορεί να υποστηρίξει το δίκτυο, το βαθμό αναζήτησης k και το βαθμό f ανεκτικότητας σε σφάλματα. Εφόσον οι παραπάνω παράμετροι έχουν καθοριστεί, το σύστημα επιδεικνύει έξι βασικές ιδιότητες:

1. Δεν απαιτεί καμία επιπλέον διαδικασία για τη διατήρηση των πινάκων δρομολόγησης, καθώς όποιες καταχωρήσεις είναι εσφαλμένες διορθώνονται από τον αλγόριθμο δρομολόγησης
2. Κάθε αναζήτηση απαιτεί το μέγιστο $\log_k N$ βήματα
3. Κάθε κόμβος διατηρεί στον πίνακα δρομολόγησης $(k-1)*\log_k N+1$ διευθύνσεις άλλων κόμβων
4. Νέοι κόμβοι μπορούν να εισέρχονται ή να αποχωρούν από το δίκτυο με ελάχιστη επιβάρυνση στο δίκτυο, διατηρώντας τη δυνατότητα επιτυχών αναζητήσεων σε $\log_k N$ βήματα
5. Η πιθανότητα ανεπιτυχούς αναζήτησης είναι αμελητέα
6. Ακόμα και εάν f διαδοχικοί κόμβοι αποτύχουν ταυτόχρονα, το σύστημα εγγυάται επιτυχείς αναζητήσεις

Ο σχεδιασμός του DKS βασίστηκε σε δύο κεντρικές ιδέες. Αρχικά χρησιμοποιούν μια μορφή δρομολόγησης διαστημάτων, που την αποκαλούν κατανεμημένη αναζήτηση k. Κατά δεύτερον, χρησιμοποιούν μια διαφορετική

τεχνική, που αποκαλούν επιδιόρθωση κατά χρήση, για τη διατήρηση των πινάκων δρομολόγησης.

Η βασική αρχή της κατανεμημένης k αναζήτησης έχει ως εξής: Δεδομένου ενός αναγνωριστικού t, για την εύρεση του αντίστοιχου κλειδιού t ο αλγόριθμος απαιτεί $\log_k N$ βήματα. Στην αρχή ο χώρος αναζήτησης ισούται με ολόκληρο το χώρο των αναγνωριστικών. Σε κάθε βήμα της αναζήτησης, ο χώρος χωρίζεται σε k ίσα διαστήματα. Κάθε ένα από αυτά τα διαστήματα βρίσκεται υπό την αρμοδιότητα ενός συγκεκριμένου κόμβου. Αυτός ο τεμαχισμός του χώρου αναζήτησης επαναλαμβάνεται εωσότου τα k ίσα διαστήματα περιλαμβάνουν ένα μόνο στοιχείο. Σε αυτό το σημείο, το διάστημα που περιέχει το στοιχείο t είναι αυτό που επιστρέφει και το κλειδί t. Με δεδομένη τη χρήση χώρου αναγνωριστικών που είναι δύναμη του k, εύκολα συνεπάγεται ότι για μέγεθος N αναγνωριστικού χώρου απαιτούνται το πολύ $\log_k N$ βήματα για μια επιτυχή αναζήτηση. Επιπλέον, κάθε κόμβος απαιτεί πίνακα δρομολόγησης μεγέθους $(k-1)*\log_k N$.

Κάθε DKS(N, k, f) δίκτυο δημιουργείται κατά τέτοιο τρόπο ώστε να επιτρέπει στην κατανεμημένη k αναζήτηση να επιτύχει σε $\log_k N$ το πολύ βήματα. Για να το επιτύχει αυτό, κάθε κόμβος έχει $\log_k N$ επίπεδα αριθμημένα από 1 έως L, όπου $L = \log_k N$ και Λ το διάστημα {1, 2, · · ·,L}. Σε κάθε επίπεδο $l \in \Lambda$, κάθε κόμβος n έχει μια κάτοψη V^l του αναγνωριστικού χώρου. Αυτή η κάτοψη αποτελείται από k ίσια μέρη, που υποδηλώνονται ως I_i^l, $0 \le i \le k+1$ και ορίζονται ως εξής για κάθε επίπεδο:

- Στο επίπεδο 1:

$$V^1 = I_0^1 \cup I_1^1 \cup I_2^1 \cup \cdots \cup I_{k-1}^1, \text{όπου}$$
$$\mathrm{I}_0^1 = [x_0^1, x_1^1], \mathrm{I}_1^1 = [x_1^1, x_2^1], \cdots, \mathrm{I}_{k-1}^1 = [x_{k-1}^1, x_0^1]$$
$$x_i^1 = n \oplus i\frac{N}{k}, for \;\; 0 \le i \le k-1$$

- Στο επίπεδο $2 \le l \le \Lambda$:

$$V^l = I_0^l \cup I_1^l \cup I_2^l \cup \cdots \cup I_{k-1}^l, \text{όπου}$$
$$\mathrm{I}_0^l = [x_0^l, x_1^l], \mathrm{I}_1^l = [x_1^l, x_2^l], \cdots, \mathrm{I}_{k-1}^l = [x_{k-1}^l, x_1^{l-1}]$$
$$x_i^l = n \oplus i\frac{N}{k}, for \;\; 0 \le i \le k-1$$

Με βάση τα παραπάνω, κάθε κόμβος διατηρεί έναν πίνακα δρομολόγησης της μορφής που παρουσιάζεται στον Πίνακας 4. Κάθε κόμβος n έχει k-1 καταχωρήσεις σε κάθε επίπεδο, δηλαδή $(k-1)*\log_k N$ συνολικά καταχωρήσεις. Επιπλέον, κάθε κόμβος διατηρεί ένα δείκτη p προς τον προγενέστερό του κόμβο στον δακτύλιο κινούμενοι αριστερόστροφα.

Επίπεδο	Διαστήματα	Υπεύθυνος
1	I_0^1	n
	I_1^1	$S(x_1^1)$
	I_2^1	$S(x_2^1)$
	•••	•••
	I_{k-1}^1	$S(x_{k-1}^1)$
	•••	
***L*-1**	I_0^{L-1}	n
	I_1^{L-1}	$S(x_1^{L-1})$
	I_2^{L-1}	$S(x_2^{L-1})$
	•••	•••
	I_{k-1}^{L-1}	$S(x_{k-1}^{L-1})$
L	I_0^L	n
	I_1^L	$S(x_1^L)$
	I_2^L	$S(x_2^L)$
	•••	•••
	I_{k-1}^L	$S(x_{k-1}^L)$

Πίνακας 4. Πίνακας δρομολόγησης του κόμβου n στο DKS

Για την επιδιόρθωση σφαλμάτων με βάση την τεχνική επιδιόρθωσης κατά τη χρήση οι δημιουργοί του DKS ενσωμάτωσαν τεχνικές πληροφορίες στα μηνύματα ώστε όταν ένας κόμβος n′ δέχεται ένα μήνυμα MSG από έναν άλλο κόμβο n, ο κόμβος n′ να μπορεί να καθορίσει εάν οι πληροφορίες δρομολόγησης που χρησιμοποίησε ο κόμβος n στέλνοντας το μήνυμα MSG ήταν σωστές ή όχι. Εάν ο κόμβος n′ αντιληφθεί ότι οι πληροφορίες δρομολόγησης του n είναι λανθασμένες τότε ενημερώνει άμεσα τον n και του αποστέλλει έναν υποψήφιο κόμβο ως εναλλακτική καταχώρηση δρομολόγησης. Όταν ο κόμβος n λάβει μια τέτοια ειδοποίηση, ανανεώνει τη λανθασμένη καταχώρησή του και ενημερώνει αντίστοιχα τον κόμβο που προξένησε αυτό το σφάλμα, επιδιορθώνοντας σταδιακά όλη τη λανθασμένη διαδρομή, δίχως τη χρήση περιοδικών ελέγχων. Με αυτόν τον τρόπο λειτουργίας ελαχιστοποιούνται (τουλάχιστον θεωρητικά) τα μηνύματα

επιδιόρθωσης καθώς περιορίζονται σε συνθήκες όπου όντως παρουσιάζεται σφάλμα.

3.5.4. Τόροι

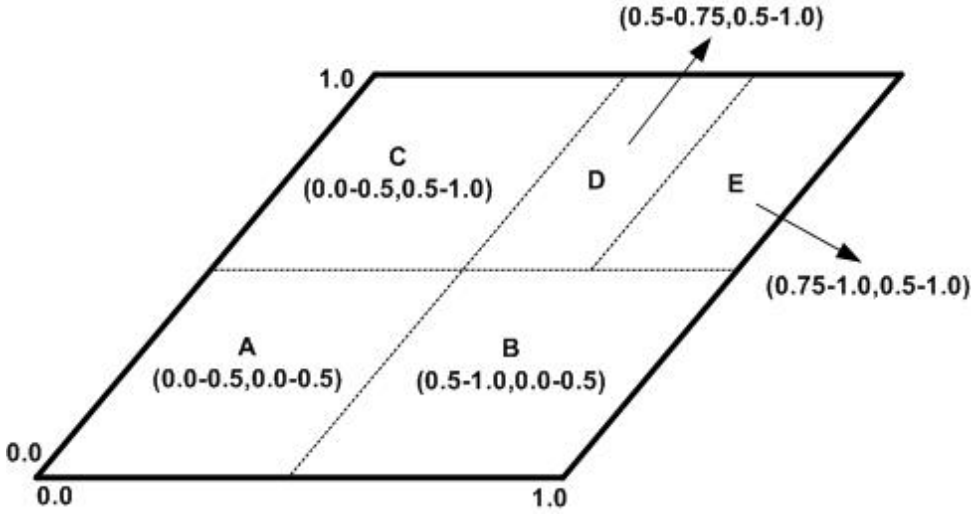
Οι Ratnasamy, Francis κ.α. ανέπτυξαν το Δίκτυο Διευθέτησης με βάση το Περιεχόμενο (Content-Addressable Network, CAN) το οποίο αποτελεί μαζί με το Tapestry, το Pastry και το Chord έναν από τους πιο γνωστούς DHT αλγορίθμους [76]. Η διάταξή του απεικονίζει έναν εικονικό καρτεσιανό χώρο d διαστάσεων με βάση έναν d-τόρο. Η βασική του ιδέα είναι ο διαχωρισμός του χώρου με βάση συντεταγμένες ώστε ο αριθμός των γειτονικών κόμβων να είναι ανεξάρτητος του αριθμού κόμβων. Στην συνέχεια αυτής της ενότητας θα περιγράψουμε αναλυτικά τη δομή του CAN. Οι σχεδιαστές του CAN αρχικά σκέφτηκαν να βασίσουν την αρχιτεκτονική τους στο PRR αλλά τελικά αποφάσισαν να χρησιμοποιήσουν μια απλή δομή με χαμηλό βαθμό συνδέσεων ανά κόμβο. Η βασική τους αιτιολόγηση για τη μη χρήση του PRR ήταν η απόδοσή του σε περιβάλλον μαζικής αποχώρησης και σφαλμάτων, καθώς η εισαγωγή και η αποχώρηση κόμβων θα επηρέαζε ένα λογαριθμικό αριθμό άλλων κόμβων. Κάποιες αρχικές εκτιμήσεις για την ανεκτικότητα του CAN έγιναν με βάση μια σειρά εξομοιώσεων με τη χρήση του Network Simulator (ns) [77], όπου ο πληθυσμός δεν ξεπερνούσε μερικές εκατοντάδες κόμβους. Κατόπιν παρουσιάστηκε ο μέσος αριθμός βημάτων και το ποσοστό επιτυχών αναζητήσεων σε σχέση με το συνολικό αριθμό κόμβων και το ποσοστό σφαλμάτων κόμβων [76]. Παρόλα αυτά, η ίδια η ερευνητική ομάδα του CAN παρουσίασε μια σειρά από ανοικτά ερευνητικά θέματα που αφορούσαν τον αριθμό των βημάτων, την ανεκτικότητα, το φόρτο διαχείρισης και την ετερογένεια των κόμβων [78],[79].

3.5.4.1. CAN

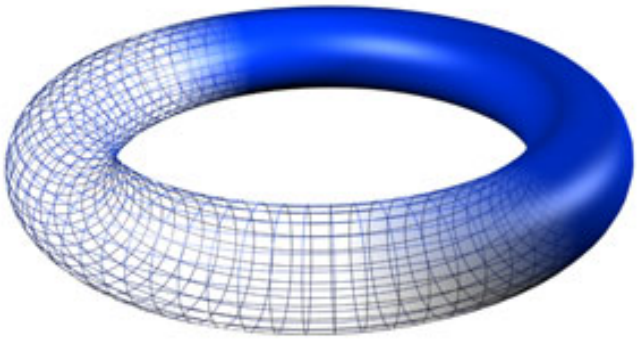
Η αρχιτεκτονική του συστήματος CAN [76] βασίζεται ,όπως και όλοι οι DHT αλγόριθμοι, στην ανάθεση αναγνωριστικών στους κόμβους και στα κλειδιά μέσω μιας συνάρτησης κατατεμαχισμού. Η ιδιαιτερότητα του CAN έγκειται στον τρόπο που αποθηκεύονται τα κλειδιά στους κόμβους.

Το CAN δημιουργεί ένα *d*-διαστάσεων καρτεσιανό χώρο με βάση έναν *d*-τόρο. Ο χώρος αυτός δεν έχει κάποια λογική ερμηνεία και απλά χρησιμοποιείται για να τοποθετήσει τους κόμβους και τα κλειδιά. Κάθε κόμβος, με βάση το αναγνωριστικό του, τοποθετείται σε αυτόν τον *d*-τόρο και είναι υπεύθυνος για τα κλειδιά που έχουν συντεταγμένες εντός της περιοχής του. Ο χώρος αυτός τεμαχίζεται δυναμικά ανάμεσα σε όλους τους κόμβους

που έχουν εισέλθει στο δίκτυο. Στο Σχήμα 15 φαίνεται ένας 2-διαστάσεων [0,1] X [0,1] χώρος όπου έχουν τοποθετηθεί 5 κόμβοι. Πρέπει να σημειώσουμε ότι η διάταξη του τόρου απαιτεί ο χώρος να είναι συνεχείς, όπως αυτός που φαίνεται στο Σχήμα 16 .



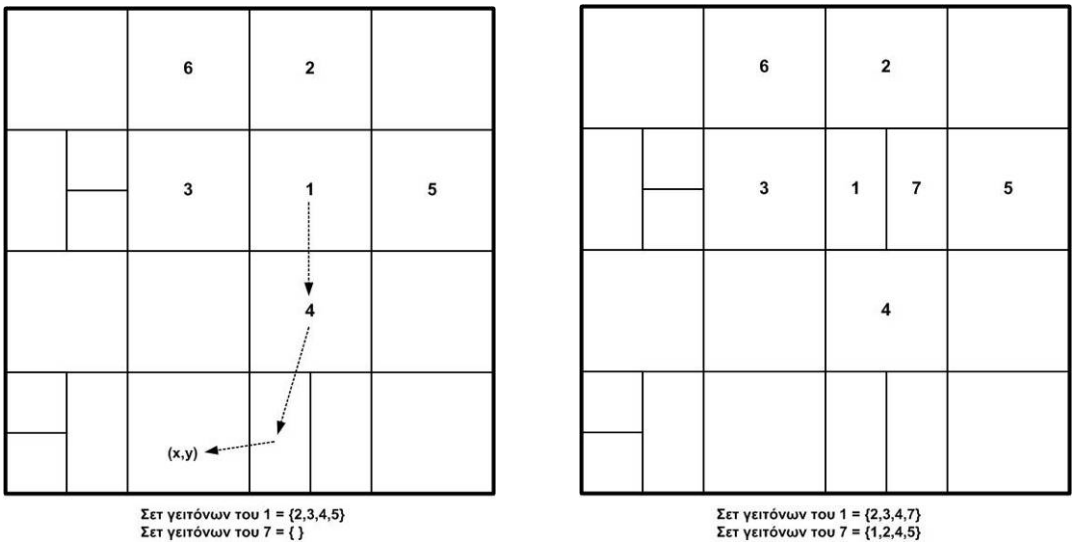
Σχήμα 15. Σύστημα συντεταγμένων 2-διαστάσεων με 5 κόμβους



Σχήμα 16. Σχηματική διάταξη τόρου

Κάθε κόμβος στο CAN διατηρεί έναν πίνακα δρομολόγησης με τις συντεταγμένες των γειτόνων του. Σε ένα *d*-διαστάσεων χώρο, δύο κόμβοι είναι γείτονες αν έχουν *d-1* διαστάσεις κοινές και μια διαφορετική. Στο Σχήμα 17 (α) φαίνεται ένας τέτοιος χώρος και επίσης φαίνεται και ο τρόπος που γίνεται η δρομολόγηση. Όταν γίνεται μια αναζήτηση, απλά ακολουθείται ο δρόμος που ορίζεται από το καρτεσιανό χώρο και μέσω των γειτόνων, το μήνυμα αναζήτησης προωθείται στον κόμβο υπεύθυνο για το κλειδί. Για ένα *d*-διαστάσεων χώρο διαχωρισμένο σε *n* ίσες ζώνες, το μέσο μήκος μονοπατιού είναι $(d/4)(n^{1/d})$ και κάθε κόμβος διατηρεί γνώση $2d$ γειτόνων. Επιπλέον, καθώς ο χώρος αυξάνει με την είσοδο νέων κόμβων δεν απαιτείται επιπλέον αύξηση στην γνώση γειτονικών κόμβων ενώ το μέσο μήκος μονοπατιού αυξάνει κατά $O(n^{1/d})$. Επιπλέον, εάν κάποιοι κόμβοι αποχωρήσουν δε

δημιουργείται πρόβλημα στην αναζήτηση καθώς απλώς ακολουθείται κάποια άλλη διαδρομή από γειτονικούς κόμβους. Η αναζήτηση γίνεται αδύνατη όταν χαθεί η επαφή με όλους τους γειτονικούς κόμβους.



(α) **(β)**

Σχήμα 17. (α) Διάταξη του χώρου πριν την είσοδο του κόμβου 7 και (β) μετά την είσοδο του κόμβου 7

Στο Σχήμα 17 (β) φαίνεται τέλος η λειτουργία του αλγορίθμου CAN κατά την είσοδο νέων κόμβων. Αυτή απαιτεί τρία βήματα. Πρώτον, την εύρεση ενός κόμβου του υπερκείμενου δικτύου. Δεύτερον, την εύρεση της κατάλληλης θέσης όπου θα τοποθετηθεί ο νέος κόμβος και τέλος, τη διάσπαση της περιοχής του παλιού κόμβου σε δυο νέες περιοχές και την ενημέρωση των γειτονικών κόμβων για τη νέα διάταξη του χώρου.

3.5.5. Πεταλούδες

Το πιο γνωστό DHT σύστημα που βασίζεται σε τοπολογία πεταλούδας είναι αυτό του Viceroy [80], το οποίο και θα περιγράψουμε αναλυτικά στην συνέχεια αυτής της ενότητας. Η καινοτομία του Viceroy ήταν ότι βελτίωνε τόσο το σταθερό βαθμό συνδέσεων του CAN όσο και τη λογαριθμική διάμετρο των Chord, Tapesty και Pastry. Οι σχεδιαστές του όμως δεν έθιξαν καθόλου το ζήτημα της στιβαρότητας καθώς θεώρησαν ότι οι κόμβοι δεν αποτυγχάνουν. Επίσης θεώρησαν ότι οι λειτουργίες εισαγωγής και αποχώρησης δεν συμπίπτανε στο σύστημα ώστε να αποφύγουν προβλήματα πολυπλοκότητας, όπως μηχανισμούς κλειδώματος. Οι Kaashoek και Karger ήταν αρκετά επικριτικοί για την πολυπλοκότητα του Viceroy [81]. Επισήμαναν επίσης την έλλειψη κατάλληλου μηχανισμού ώστε το σύστημα να ήταν ανεκτικό σε σφάλματα.

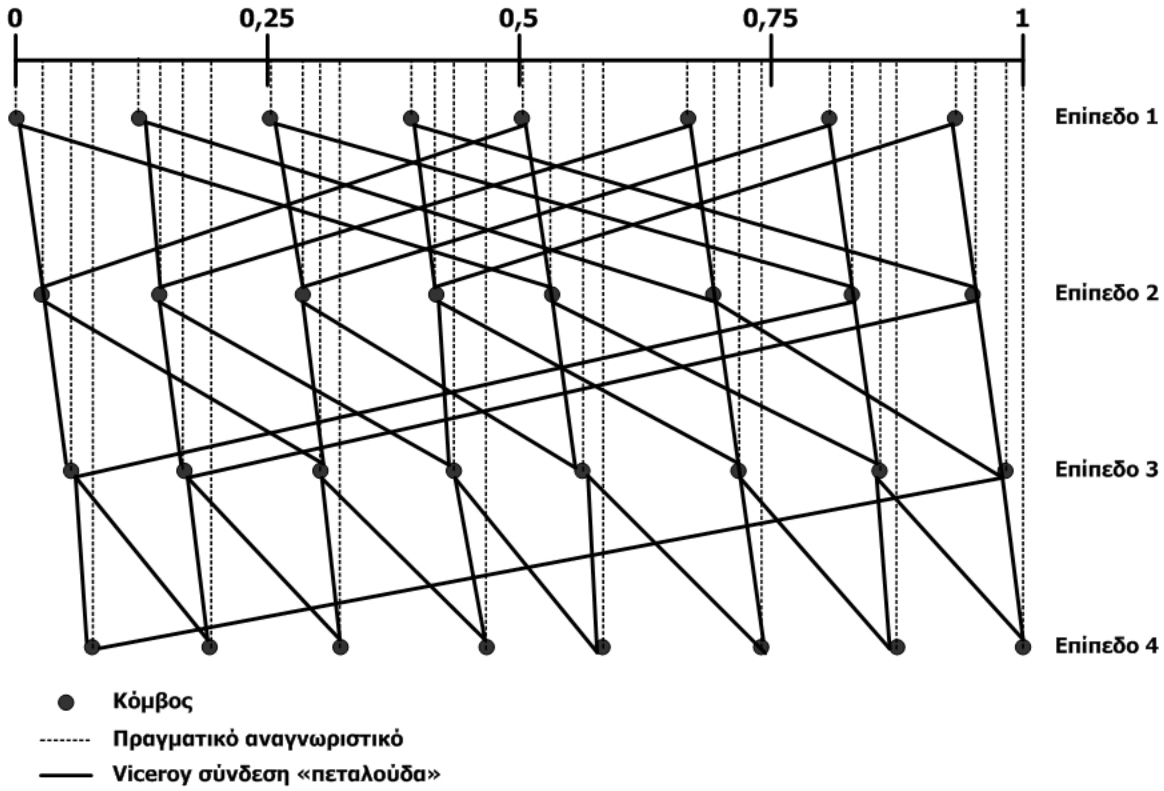
Οι Li και Plaxton συνέστησαν περαιτέρω έρευνα όσον αφορά τη χρήση τέτοιων αλγορίθμων που υποστήριζαν σταθερό βαθμό συνδέσεων [82]. Σύμφωνα με την ανάλυσή τους, τέτοιου είδους αλγόριθμοι προσφέρουν αρκετά πλεονεκτήματα αλλά και μειονεκτήματα. Ο περιορισμένος βαθμός συνδέσεων μπορεί να αυξήσει την πιθανότητα καταμερισμού του δικτύου ή να εμποδίσει τη χρήση τοπικών γειτόνων, καθώς αυτοί είναι λιγότεροι. Προσφέρουν όμως το θετικό ότι τόσο η δρομολόγηση όσο και η λογική τους είναι πολύ πιο απλή και εγγυημένη. Ένας από τους σχεδιαστές του Viceroy πρότεινε στην συνέχεια έναν καινούργιο αλγόριθμο βασισμένο σε σταθερό βαθμό συνδέσεων, ο οποίος συνυπολογίζει τη γεωγραφική θέση των κόμβων [83] και βελτιώνει την προσαρμοστικότητα του δικτύου. Ένας άλλος σχεδιαστής του Viceroy εξερεύνησε έναν εναλλακτικό γράφο περιορισμένου βαθμού για ομότιμα δίκτυα, το γράφο de Bruijn [84], που θα αναλύσουμε στην επόμενη ενότητα.

3.5.5.1. Viceroy

Το δίκτυο Viceroy [80] βασίζεται στην σύνθεση ενός δικτύου που προσομοιώνει τη δομή της πεταλούδας και ενός δακτυλίου με προγόνους και απογόνους. Επιπρόσθετα κάθε εξυπηρετητής διατηρεί και πέντε εξερχόμενες συνδέσεις ώστε να μπορεί να επιλέγει μακρινούς συνδέσμους. Αρχικά, κάθε κόμβος επιλέγει ένα επίπεδο τυχαία έτσι ώστε όταν n εξυπηρετητές είναι σε λειτουργία, ένα από τα logn επίπεδα να επιλέγεται ισοπίθανα. Για έναν κόμβο επιπέδου l δύο ακμές προστίθενται ώστε να συνδέεται με κόμβους στο επίπεδο $l+1$. Πιο συγκεκριμένα, προστίθεται μια «κάτω δεξιά» ακμή σε απόσταση περίπου $1/2^l$ και μια «κάτω αριστερή» ακμή σε κοντινή απόσταση στον δακτύλιο του επιπέδου $l+1$. Επιπλέον, μια «άνω» ακμή προστίθεται σε κοντινή απόσταση από έναν κόμβο σε επίπεδο $l-1$, εφόσον $l>1$. Τέλος, προστίθενται συνδέσεις «επιπέδου-δακτυλίου» προς τον προηγούμενο και επόμενο κόμβο στο ίδιο επίπεδο l. Στο Σχήμα 18 απεικονίζεται μια τυπική δομή του δικτύου Viceroy.

Η δρομολόγηση στο Viceroy γίνεται σε τρία στάδια. Το πρώτο στάδιο αποτελείται από μια άνοδο με τη χρήση συνδέσεων προς τα άνω μέχρι το επίπεδο *level-1*. Στο δεύτερο στάδιο η δρομολόγηση συνεχίζεται προς τα κάτω επίπεδα του δέντρου μέσω των κάτω συνδέσεων. Σε αυτό το στάδιο η κάθοδος γίνεται μέσω της μετακίνησης από το επίπεδο l στο $l+1$ είτε με τη χρήση των κοντινών συνδέσεων ή των απομακρυσμένων, ανάλογα αν το κλειδί x βρίσκεται σε απόσταση μεγαλύτερη του $1/2^l$. Η διαδικασία αυτή συνεχίζεται εωσότου ευρεθεί κόμβος που δεν έχει συνδέσεις προς τα κάτω, που συνεπάγεται ότι βρίσκεται κοντά στο στόχο. Τέλος, εκτελείται μια αναζήτηση

«γειτνίασης» χρησιμοποιώντας τις συνδέσεις του δακτυλίου και τις «επιπέδου-δακτυλίου» συνδέσεις εωσότου ευρεθεί ο στόχος. Η όλη διαδικασία αναζήτησης απαιτεί με μεγάλη πιθανότητα O(logn) βήματα.



Σχήμα 18. Η δομή του δικτύου Viceroy

Κάθε εξυπηρετητής στο Viceroy συνδέεται με δύο τιμές που καθορίζουν τη σύνδεσή του με το υπόλοιπο σύστημα: την ταυτότητά του $s.id \in [0,1)$ και το επίπεδό του l. Η ταυτότητά του παραμένει σταθερή αλλά το επίπεδό του μπορεί να αλλάξει κατά την εξέλιξη του δικτύου. Οι συνδέσεις και η δομή δρομολόγησης ενός δικτύου Viceroy καθορίζεται αποκλειστικά από τις ταυτότητες και τις πληροφορίες επιπέδων των ενεργών εξυπηρετητών και δεν επηρεάζεται από προηγούμενες ρυθμίσεις ή άλλες τυχαίες πηγές.

Το δίκτυο αποτελείται από τρία σετ συνδέσεων:

- Ένα γενικό δακτύλιο, όπου κάθε κόμβος συνδέεται με τον προγενέστερο και μεταγενέστερο κόμβο
- Δακτυλίους επιπέδων, όπου κόμβοι ίδιου επιπέδου συνδέονται μέσω ενός δακτυλίου
- Μια πεταλούδα, όπου κάθε κόμβος επιπέδου l που δεν είναι φυλλικός συνδέεται με δύο «κάτω» κόμβους του επιπέδου $l+1$ και κάθε κόμβος σε επίπεδο $l > 1$ με έναν επιπλέον «άνω» κόμβο σε επίπεδο $l-1$. Η «κάτω αριστερά» σύνδεση επιλέγεται ως ο πρώτος κόμβος επιπέδου $l+1$ δεξιόστροφα προς το $s.id$ ($\text{NLEVEL}_{l+1}(s.id)$) ενώ η «κάτω δεξιά» σύνδεση επιλέγεται ως ο δεύτερος κόμβος επιπέδου $l+1$ δεξιόστροφα του $s.id+1/2^l$

($\text{NLEVEL}_{l+1}(s.id+1/2^l)$). Τέλος η σύνδεση προς τα «άνω» επιλέγεται ως ο κόμβος επιπέδου l-1 δεξιόστροφα του $s.id$ ($\text{NLEVEL}_{l-1}(s.id)$). Η συνδεσμολογία αυτή φαίνεται γραφικά στο Σχήμα 18.

Η αναζήτηση στο Viceroy χωρίζεται σε απλή αναζήτηση και βελτιωμένη. Στην απλή αναζήτηση δεν χρησιμοποιούνται οι συνδέσεις «επιπέδου-δακτυλίου» αλλά μόνο οι συνδέσεις προγενέστερων και μεταγενέστερων κόμβων και οι συνδέσεις πεταλούδα. Σκοπός της αναζήτησης είναι εύρεση του εξυπηρετητή που είναι δεξιόστροφα πλησιέστερος προς μια τιμή κλειδί. Η αναζήτηση χωρίζεται σε τρία στάδια. Στο πρώτο στάδιο, ένας εξυπηρετητής επιπέδου-1 (ρίζα) εντοπίζεται ακολουθώντας τις «άνω» συνδέσεις. Στο δεύτερο στάδιο, οι αναζητήσεις δρομολογούνται προς τα «κάτω» από τη ρίζα μέσω των «κάτω» συνδέσεων. Σε αυτό το στάδιο, όταν μια αναζήτηση βρίσκεται σε έναν κόμβο επιπέδου i τότε βρίσκεται το πολύ σε απόσταση $1/2^{i-1}$από το στόχο. Στην συνέχεια, αν ο στόχος βρίσκεται σε απόσταση $1/2^i$ ή μεγαλύτερη τότε η αναζήτηση συνεχίζεται προς τον κάτω δεξιά σύνδεσμο ειδάλλως προς τον κάτω αριστερά. Αυτό το στάδιο τερματίζεται όταν δεν υπάρχουν περαιτέρω «κάτω» συνδέσεις ή εντοπιστεί ο στόχος. Κατά το τελικό στάδιο, η δρομολόγηση συνεχίζεται στον δακτύλιο είτε δεξιόστροφα είτε αριστερόστροφα. Η απλή αναζήτηση οδηγεί μεν σε επιτυχή αναζήτηση αλλά δεν επιτυγχάνει απόδοση O(logn). Το βασικό πρόβλημα της απλής αναζήτησης εντοπίζεται στο τρίτο στάδιο, όπου η δρομολόγηση κατά μήκος του δακτυλίου μπορεί να επιφέρει $O(\log^2 n)$ βήματα. Η ιδέα της βελτιωμένης αναζήτησης διατηρεί τα πρώτα δύο στάδια αλλά χρησιμοποιεί τις συνδέσεις «επιπέδων-δακτυλίων» για το τρίτο στάδιο με αποτέλεσμα η δρομολόγηση σε έναν πολυ-λογαριθμικό αριθμό κόμβων να απαιτεί λογαριθμικό αριθμό βημάτων.

3.5.6. de Bruijn Γράφοι

Οι γράφοι de Bruijn έχουν αναπτυχθεί και εφαρμοστεί σε διάφορους τομείς από την πρώτη τους εμφάνιση [85],[86]. Ο Schlumberger ήταν ο πρώτος που τους χρησιμοποίησε σε δίκτυα [87]. Δυο διαφορετικές ερευνητικές ομάδες επινόησαν ανεξάρτητα το γενικευμένο γράφο de Bruijn στις αρχές της δεκαετίας του 1980, ο οποίος μπορούσε να εξυπηρετήσει ένα μεταβλητό αριθμό κόμβων σε ένα σύστημα [88],[89]. Οι Rowley και Bose μελέτησαν δακτυλίους ανεκτικούς σε σφάλματα οι οποίοι ήταν υπερκείμενοι σε γράφους de Bruijn [90]. Οι Lee, Liu κ.α. ανέπτυξαν μια ιεραρχία de Bruijn δύο επιπέδων, όπου συστάδες τοπικών κόμβων συνδέονταν μέσω ενός δακτυλίου δευτέρου επιπέδου [91]. Πολλοί από τους αλγορίθμους των προηγούμενων τοπολογιών θεωρούνται άπληστοι αλγόριθμοι καθώς κάθε φορά που μια

αναζήτηση προωθείται, κινείται πιο κοντά προς τον προορισμό. Τέτοιου είδους αλγόριθμοι όμως δεν είναι βέλτιστοι καθώς για ένα συγκεκριμένο βαθμό, η απόσταση δρομολόγησης είναι μεγαλύτερη από την απαραίτητη [92]. Σε αντίθεση με αυτούς τους αλγορίθμους, οι γράφοι de Bruijn βαθμού k επιτυγχάνουν ασυμπτωτικά βέλτιστη διάμετρο $\log_k n$, όπου n είναι ο αριθμός των κόμβων στο δίκτυο και το k μπορεί να μεταβληθεί ώστε να βελτιωθεί η ανεκτικότητα. Αν υπάρχουν O(logn) γείτονες ανά κόμβο, τότε ο απαιτούμενος αριθμός βημάτων για ένα de Bruijn σύστημα είναι O(logn/loglogn). Για παράδειγμα, σε ένα δίκτυο με ένα εκατομμύριο κόμβους και βαθμό συνδέσεων 20, το Chord έχει διάμετρο 20 ενώ ο de Bruijn έχει 5 [60]. Το 2003 υπήρξε μια σειρά από προτάσεις ομότιμων συστημάτων βασισμένα σε γράφους de Bruijn, όπως το D2B [93], το Koorde [81], το Distance Halving [84] και το Optimal Diameter Routing Infrastructre (ODRI) [60]. Στην συνέχεια της ενότητας αυτής θα παρουσιάσουμε αναλυτικά τις δομές των D2B και Distance Halving.

Οι Fraigniaud και Gauron ξεκίνησαν το σχεδιασμό του D2B παραθέτοντας μια σειρά από απαιτήσεις:

- τα κλειδιά πρέπει να κατανέμονται ομοιόμορφα
- η λανθάνουσα καθυστέρηση αναζήτησης πρέπει να είναι μικρή
- η κίνηση φόρτου πρέπει να κατανέμεται ομοιόμορφα
- η ανανέωση των πινάκων δρομολόγησης και η ανακατανομή των κλειδιών πρέπει να γίνεται γρήγορα όταν εισέρχονται ή αποχωρούν κόμβοι από το δίκτυο

Όρισαν επίσης ως μέτρο συμφόρηση ενός κόμβου την πιθανότητα μια αναζήτηση να το διαπεράσει. Ενώ οι σχεδιαστές του D2B παρουσιάζουν αναλυτικά τα πλεονεκτήματα της σχεδίασής τους, παρουσιάζουν ελάχιστα την ανεκτικότητα της δομής τους.

Το Koorde επεκτείνει το Chord ώστε να επιτύχει τον τέλειο ισολογισμό βαθμού και διαμέτρου που χαρακτηρίζει τους γράφους de Bruijn [81]. Σε αντίθεση με το D2B, το Koorde δεν περιορίζει την επιλογή των αναγνωριστικών των κόμβων. Επίσης σε αντίθεση με το D2B, παρέχει μηχανισμούς παράλληλων εισαγωγών, ως επέκταση της λειτουργικότητας του Chord. Οι Kaashoek και Karger μελέτησαν την ανεκτικότητα του Koorde με βάση ένα αρκετά απαιτητικό σενάριο: «για να έχει το δίκτυο την ικανότητα να παραμένει συνδεδεμένο ακόμα και όταν όλοι οι κόμβοι αποτυγχάνουν με πιθανότητα ½, κάποιο κόμβοι πρέπει να έχουν λογαριθμικό βαθμό συνδέσεων (logn)" [81]. Σχεδίασαν ένα μηχανισμό για να αυξήσουν το βαθμό συνδέσεων του Koorde ώστε να ανταποκρίνεται σε αυτήν την αυξημένη

ανεκτικότητα, χάνοντας όμως ταυτόχρονα το πλεονέκτημα των γράφων de Bruijn, αυτό του σταθερού βαθμού συνδέσεων. Παρομοίως, για να επιτύχουν ισόποση κατανομή φόρτου κίνησης, το Koorde θυσίασε το βέλτιστο βαθμό συνδέσεών του. Σύμφωνα με την ανάλυσή τους, η ικανότητα να ανταλλάσσει το βαθμό συνδέσεων, οπότε και τον επίφορτο διαχείρισης, με το μέσο αριθμό απαιτούμενων βημάτων για επιτυχή αναζήτηση ήταν αρκετά σημαντικό για συστήματα όπου παρουσιάζονται πολλά σφάλματα και αποχωρήσεις. Επίσης εντόπισαν ένα άλυτο πρόβλημα: τη σχεδίαση ενός DHT αλγορίθμου που θα έχει ταυτόχρονα βέλτιστο βαθμό συνδέσεων αλλά και ισοκατανομή φόρτου. Οι Datta, Girdzijauskas κ.α. αμφισβητούν την κατανομή των κλειδιών του Koorde καθώς θεωρούν ότι είναι μη ρεαλιστική και υπεραπλουστεύει την ομοιόμορφη κατανομή τους [94]. Απέδειξαν πως για αυθαίρετες κατανομές κλειδιών, οι γράφοι de Bruijn αποτυγχάνουν να ικανοποιήσουν ταυτόχρονα και ισοκατανομή φόρτου και αποδοτική αναζήτηση. Στο τέλος της εργασίας τους έθεσαν το εξής ερώτημα, «υπάρχει DHT αλγόριθμος με πίνακα δρομολόγησης σταθερού μεγέθους ο οποίος να ικανοποιεί ταυτόχρονα και ισοκατανομή φόρτου και αποδοτική αναζήτηση για αυθαίρετες και μεταβαλλόμενες κατανομές κλειδιών;».

Το Distance Halving βασίστηκε επίσης στις ιδέες του de Bruijn [84] και έχει τον ίδιο βέλτιστο βαθμό συνδέσεων. Στον σχεδιασμό τους οι Naor και Wieder χρησιμοποίησαν μια μέθοδο «συνεχούς-διακριτού» χώρου δύο βημάτων. Η σωστή λειτουργία τους συστήματος αποδεικνύεται σε συνεχές χώρο και στην συνέχεια οι αλγόριθμοι εφαρμόζονται σε διακριτό χώρο. Στην εργασία τους, οι αλγόριθμοι εισόδου και αποχώρησης περιγράφονταν μεν αναλυτικά αλλά δεν υπήρχε αναφορά για την απόδοσή τους σε περιβάλλοντα πολλαπλών σφαλμάτων και αποχωρήσεων. Χρησιμοποιώντας τα χαρακτηριστικά του Distance Halving οι σχεδιαστές τους στην συνέχεια όρισαν ένα σχήμα απόκρυψης για την αποσυμφόρηση μεγάλων ομότιμων δικτύων. Τροποποίησαν επίσης τον αλγόριθμο δρομολόγησης ώστε να είναι πιο στιβαρός κατά την εμφάνιση τυχαίων σφαλμάτων [95].

Όπως έχουμε ήδη αναφέρει, οι μελέτες σύγκρισης της ανεκτικότητας των DHT αλγορίθμων είναι σπάνιες. Παρόλα αυτά , οι Loguinov, Kumar κ.α. στην εργασία τους ODRI [60] συνέκριναν τα συστήματα Chord, CAN και τους γράφους de Bruijn σε σχέση με την ικανότητά τους να δρομολογούν αποδοτικά, να επεκτείνουν γράφους και να συσταδοποιούν. Αρχικά πρότειναν ως βέλτιστο αλγόριθμο όσον αφορά τη διάμετρο (η μέγιστη απόσταση μεταξύ οποιοδήποτε δύο κόμβων σε ένα γράφο) και το μέσο μονοπάτι τους γράφους σταθερού βαθμού. Σύμφωνα με τους συγγραφείς, οι γράφοι De Bruijn συγκλίνουν προς το βέλτιστο και υπερέχουν σε απόδοση τόσο του Chord όσο και του CAN και για τα δύο αυτά μέτρα σύγκρισης. Αυτή

η βέλτιστη απόδοση επηρεάζει τόσο την καθυστέρηση όσο και το συνολικό φόρτο αναζήτησης. Στην συνέχεια παρουσίασαν δύο μεθόδους συσταδοποίησης που βελτιώνουν την ανεκτικότητα ενός συστήματος, αυτής της επέκτασης ακμής και αυτής της επέκτασης κόμβου. Δυστυχώς, αν και η έρευνα σε γράφους de Bruijn απαριθμεί αρκετές δεκαετίες, δεν έχει ακόμη βρεθεί υλοποίηση για αυτές τις μεθόδους.

3.5.6.1. D2B

Η τοπολογία του D2B [93] d-διαστάσεων, d≥2, είναι ένας γράφος de Bruijn B(d,k), με k≥1. Αποτελείται από έναν κατευθυνόμενο γράφο του οποίου οι κόμβοι είναι μια ακολουθία χαρακτήρων μήκους k με βάση το αλφάβητο {0,...d-1} και οι ακμές ενώνουν κάθε κόμβο $x_1x_2...x_k$ με τους d κόμβου $x_1x_2...x_k\alpha$, όπου α=0,...,d-1.

Ο de Bruijn γράφος B(d,k) έχει d^k κόμβους, με βαθμό εισόδου και εξόδου d και διάμετρο k. Η δρομολόγηση από τον κόμβο $x_1...x_k$ προς τον κόμβο $y_1...y_k$ επιτυγχάνεται ακολουθώντας τη διαδρομή:

$$x_1...x_k \rightarrow x_2...x_ky_1 \rightarrow . . . \rightarrow x_ky_1...y_{k-1} \rightarrow y_1...y_k$$

Η επίτευξη συντομότερης διαδρομής είναι εφικτή εφόσον ευρεθεί η μεγαλύτερη ακολουθία που να είναι κατάληξη του $x_1...x_k$ και πρόθεμα του $y_1...y_k$. Αν υπάρχει τέτοια ακολουθία όπου $x_i...x_k = y_1...y_{k-i+1}$ τότε η συντομότερη διαδρομή από το $x_1...x_k$ στο $y_1...y_k$ είναι:

$$x_1...x_k \rightarrow x_2...x_ky_{k-i+2} \rightarrow . . . \rightarrow x_{i-1}....x_ky_{k-i+2}...y_{k-1} \rightarrow y_1...y_k$$

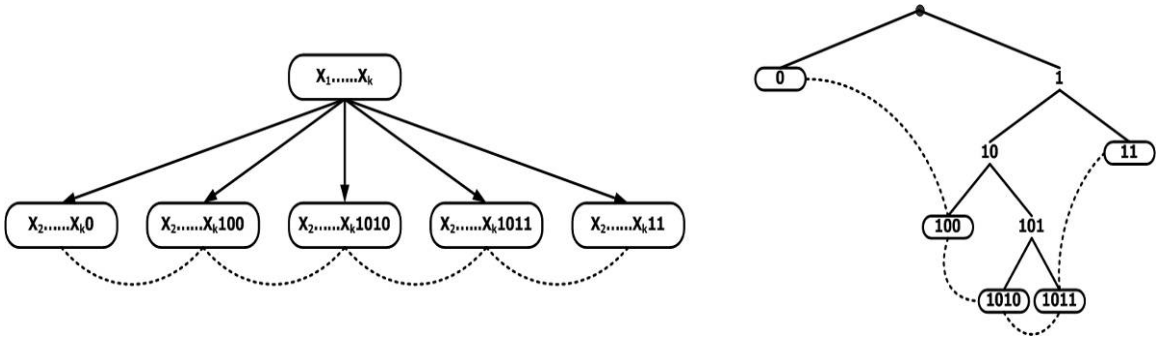
Το D2B 2-διαστάσεων χρησιμοποιεί το σύνολο **K**={0,...,2m-1} ως χώρο κλειδιών, το οποίο αποτελεί το σύνολο των δυαδικών ακολουθιών μήκους m. Όλοι οι συμμετέχοντες στο D2B χρησιμοποιούν μια συνάρτηση h που κατατεμαχίζει τα αναγνωριστικά στο **K**. Οι κόμβοι στο D2B έχουν επίσης αναγνωριστικά που αποτελούνται από δυαδικές ακολουθίες αλλά με μέγιστο μήκος m. Συνεπώς στο D2B μπορούν να συνδεθούν το πολύ 2^m κόμβοι. Η τιμή ενός κόμβους u με αναγνωριστικό $x_1...x_k$, $x_i \in \{0,1\}$ είναι $val(u) = 2^{m-k} \cdot \sum_{i=1}^{k} x_i 2^{k-i}$.

Ένας κόμβος με αναγνωριστικό $x_1...x_k$ είναι υπεύθυνος για όλα τα κλειδιά μεταξύ της τιμής val(u) και $2^{m-k}(1+\sum_{i=1}^{k} x_i 2^{k-i})-1$. Πιο συγκεκριμένα το κλειδί με δυαδικό αναγνωριστικό $k_1...k_m$ διαχειρίζεται από τον κόμβο $x_1...x_k$ που βρίσκεται ήδη στο σύστημα αν και μόνο αν το $x_1...x_k$ είναι πρόθεμα του $k_1...k_m$. Συνεπώς ένας κόμβος με αναγνωριστικό $x_1...x_k$ είναι υπεύθυνος

για 2^{m-k} κλειδιά. Αντιστρόφως, ένας κόμβος που είναι υπεύθυνος για 2^q κλειδιά έχει αναγνωριστικό με m-q δυφία.

Κάθε κόμβος με αναγνωριστικό $x_1...x_k$ έχει είτε ένα μοναδικό εξωτερικό γείτονα της μορφής $x_2...x_j$,όπου $j \leq k$,είτε μια σειρά εξωτερικών γειτόνων της μορφής $x_2...x_k y_1...y_l$, όπου $1 \leq l \leq m-k+1$. Στην δεύτερη περίπτωση το σύνολο των ακολουθιών $y_1...y_l$ σχηματίζει ένα γενικό σύνολο προθεμάτων. Πιο συγκεκριμένα, αν το $x_2...x_k y_1...y_l$ είναι ένας εξωτερικός γείτονας του $x_1...x_k$ τότε κανένα από τα αναγνωριστικά $x_2...x_k y_1...y_i$,, με $i<l$, δεν χρησιμοποιείται στο δίκτυο. Στο D2B ένας εξωτερικός γείτονας του κόμβου u αναφέρεται ως παιδί του u. Στο Σχήμα 19 παρουσιάζονται γραφικά τα παιδιά ενός κόμβου με αναγνωριστικό $x_1...x_k$. Στο παράδειγμα αυτό ο κόμβος $x_1...x_k$ έχει πέντε παιδιά με αναγνωριστικά $x_2...x_k0$, $x_2...x_k100$, $x_2...x_k1010$, $x_2...x_k1011$ και $x_2...x_k11$. Στο δίκτυο αυτό δεν μπορεί να υπάρχει κόμβος με αναγνωριστικό $x_2...x_k1$, $x_2...x_k10$ ή $x_2...x_k101$.

Κατά συμμετρικό τρόπο, κάθε κόμβος με αναγνωριστικό $x_1...x_k$ έχει εισερχόμενους γείτονες είτε της μορφής $\alpha x_1...x_j$ με $\alpha \in \{0,1\}$ και $j \leq k$ είτε της μορφής $\beta x_1...x_k y_1...y_l$, με $\beta \in \{0,1\}$και $1 \leq l \leq m-k-1$. Στην δεύτερη περίπτωση, το σύνολο των ακολουθιών $y_1...y_l$ αποτελεί ένα γενικό σύνολο προθεμάτων.

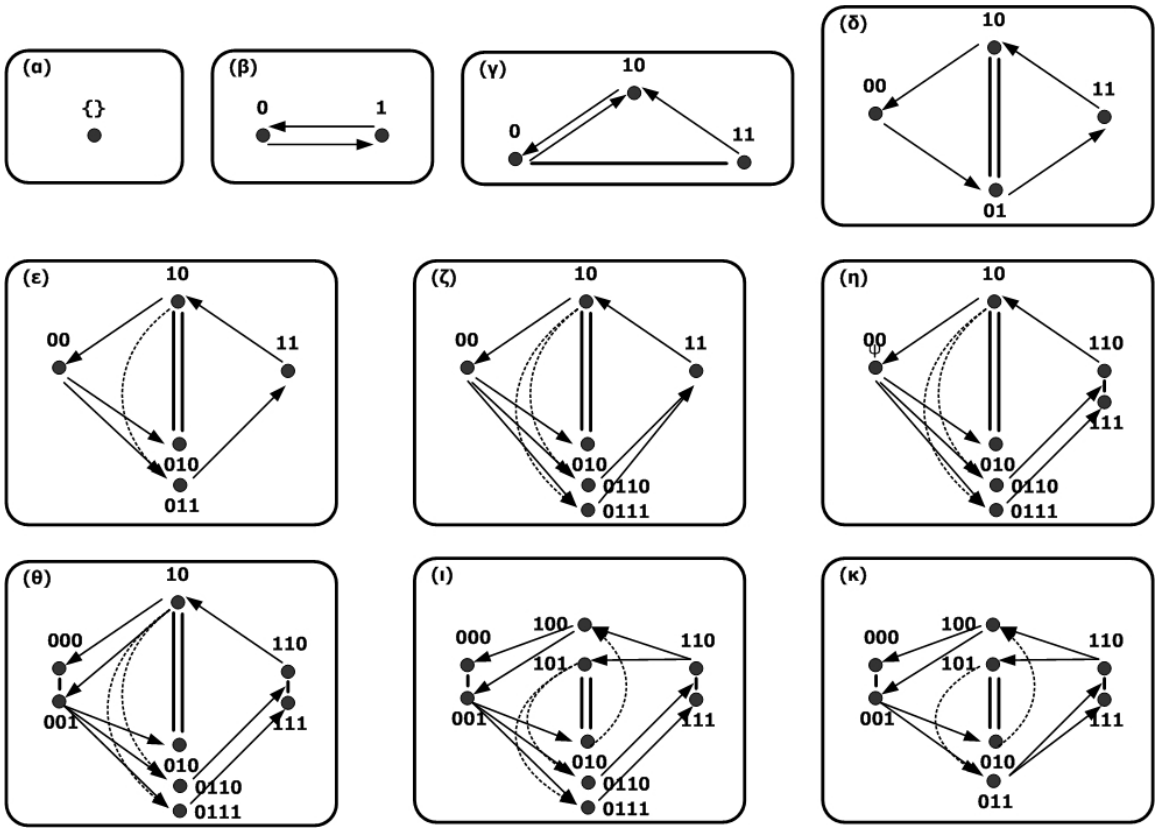


Σχήμα 19. Συνδέσεις παιδιών (απλά βέλη) και αδελφών (διακεκομμένες γραμμές) στο D2B

Εκτός από τις συνδέσεις παιδιών και γονέων, κάθε παιδί ενός κόμβου u συνδέεται επιπλέον και με έναν αδερφό με τον ακόλουθο τρόπο. Εάν ο κόμβος v είναι παιδί του u στο D2B τότε υπάρχει μια σύνδεση άνω-αδελφού από το v στο w, όπου το w είναι το παιδί του u με τη μικρότερη τιμή val(w) που είναι μεγαλύτερη του val(v) (εάν ο κόμβος v έχει τη μεγαλύτερη τιμή από όλα τα παιδιά του u τότε δεν έχει άνω-αδελφό). Παρομοίως, υπάρχει μια σύνδεση κάτω-αδελφού από έναν κόμβο v προς έναν κόμβο w όπου ο w είναι το παιδί του u με τη μεγαλύτερη τιμή που είναι μικρότερη από το val(v) (εάν ο v έχει τη μικρότερη τιμή από όλα τα παιδιά του u τότε ο v δεν έχει κάτω-αδελφό). Οι συνδέσεις αδελφών στο D2B χρησιμοποιούνται για την κατανομή κλειδιών και όχι για δρομολόγηση.

Η δρομολόγηση στο D2B εκτελείται όπως και στους γράφους de Bruijn. Πιο συγκεκριμένα, έστω $x_1...x_k$ το αναγνωριστικό του κόμβου u και ένα $k=k_1...k_m$ τυχαίο κλειδί. Έστω **S** η μεγαλύτερη δυαδική ακολουθία που είναι κατάληξη του $x_1...x_k$ και πρόθεμα του $k_1...k_m$, με την πιθανότητα το **S** να είναι κενό (**S**=∅). Εάν **S**=$x_1...x_k$ τότε το u περιέχει το k. Ειδάλλως, εάν το u έχει ένα μοναδικό παιδί v με αναγνωριστικό $x_2...x_j$, τότε η αναζήτηση για το κλειδί k προωθείται στο παιδί αυτό. Εάν ο u έχει πολλά παιδιά, τότε η αναζήτηση για το κλειδί k προωθείται στο παιδί v με αναγνωριστικό $x_2...x_k y_1...y_l$ τέτοιο ώστε το **S**$y_1...y_l$ να είναι πρόθεμα του $k_1...k_m$.

Στο Σχήμα 20 παρουσιάζεται ο τρόπος λειτουργία του D2B. Ο πρώτος κόμβος που εισέρχεται στο δίκτυο λαμβάνει το ∅ ως αναγνωριστικό (α). Όταν εισέρχεται και ένας δεύτερος κόμβος (β) τότε το αναγνωριστικό επεκτείνεται στο 0 ενώ ο νέος κόμβος λαμβάνει το αναγνωριστικό 1. Στην συνέχεια εισέρχεται ένας τρίτος κόμβος (γ). Θεωρώντας ότι ο νέος κόμβος επιλέγει ένα προσωρινό αναγνωριστικό 1..., ο κόμβος με αναγνωριστικό 1 επεκτείνει το αναγνωριστικό του σε 10 ενώ ο νέος κόμβος λαμβάνει το αναγνωριστικό 11. Εν συνεχεία συνδέεται ένας τέταρτος κόμβος (δ). Θεωρώντας ότι επιλέγει ένα προσωρινό αναγνωριστικό 0..., ο κόμβος 0 επεκτείνει το αναγνωριστικό του σε 00 και ο νέος κόμβος λαμβάνει το αναγνωριστικό 01. Η συγκεκριμένη δομή του δικτύου αποτελεί ένα γράφο B(2,2). Εν συνεχεία προστίθενται διαδοχικά τέσσερις κόμβοι με προσωρινά αναγνωριστικά 01... (ε), 011...(ζ), 11...(η) και 00...(θ) αντίστοιχα. Στο (θ) βλέπουμε ότι ο εξωτερικός βαθμός συνδέσεων του κόμβου 10 είναι 5. Κατόπιν στο (ι) προστίθεται ένας ακόμα κόμβος με προσωρινό αναγνωριστικό 10.... Παρατηρούμε πλέον ότι οι κόμβοι 100 και 101 έχουν περίπου το μισό βαθμό συνδέσεων από ότι ο κόμβος 10 στο (θ). Τέλος, στο (κ) ο κόμβος με αναγνωριστικό 0110 αποχωρεί από το δίκτυο με αποτέλεσμα ο κόμβος 0111 να περιορίσει το αναγνωριστικό του σε 011. Η δομή του δικτύου πλέον αποτελεί ένα γράφο B(2,3). Το γεγονός πως τα βήματα (δ) και (κ) οδήγησαν σε γράφους de Bruijn είναι εντελώς τυχαίο καθώς γενικότερα η τοπολογία του D2B δεν είναι ισομορφική ως προς την τοπολογία του γράφου de Bruijn. Παρόλα αυτά, η «αναμενόμενη τοπολογία» του υπερκείμενου δικτύου D2B είναι αρκετά κοντά σε ένα γράφο de Bruijn για τιμές του n παραπλήσιες σε δύναμη του 2.



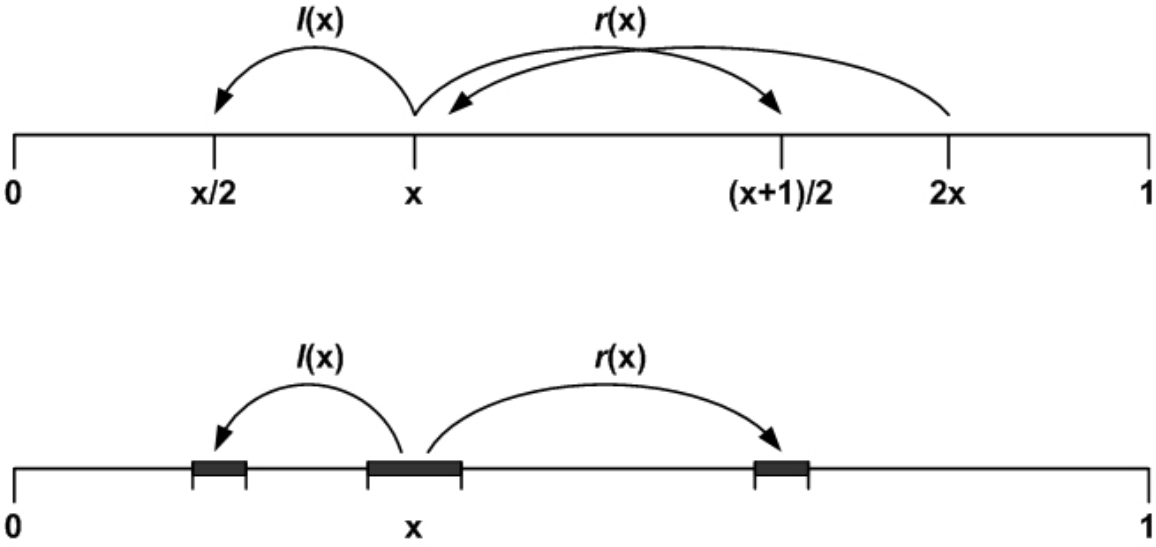
Σχήμα 20. Τυπική συμπεριφορά ενός δικτύου D2B

3.5.6.2. Distance Halving

Το Distance Halving [84] βασίζεται σε θεωρητικό επίπεδο σε ένα συνεχές χώρο, ενώ πρακτικά υλοποιείται σε ένα διακριτό χώρο. Έστω I ένας Ευκλείδειος χώρος και $\mathbf{G_c}$ ένας καθορισμένος συνεχής γράφος του οποίου το σύνολο κορυφών αποτελεί ο I και κάθε σημείο του συνδέεται με κάποια άλλα σημεία. Το δίκτυο του distance halving αποτελείται από το «διακριτ-ισμό» αυτού του συνεχούς γράφου με βάση τη δυναμική αποδόμηση του υποκείμενου χώρου I σε κελιά, όπου ο κάθε «επεξεργαστής» είναι υπεύθυνος για ένα κελί. Δύο τέτοια κελιά είναι συνδεδεμένα εάν περιέχουν γειτονικά σημεία στον συνεχή γράφο. Ο τεμαχισμός του χώρου σε κελιά διατηρείται με κατανεμημένο τρόπο. Η εκτέλεση μιας λειτουργίας εισόδου οδηγεί στην διάσπαση ενός υπάρχοντος κελιού ενώ αντίστοιχα η εκτέλεση μιας λειτουργίας αποχώρησης οδηγεί στην ένωση δύο κελιών.

Για την ευκολότερη κατανόηση του συστήματος, αρχικά ορίζεται ο συνεχής distance halving γράφος $\mathbf{G_c}$. Το σύνολο κορυφών του $\mathbf{G_c}$ αποτελεί το διάστημα $I \overset{def}{=} [0,1)$. Το σύνολο των ακμών του $\mathbf{G_c}$ ορίζεται από τις εξής συναρτήσεις: $l(y) \overset{def}{=} \frac{y}{2}, r(y) \overset{def}{=} \frac{y}{2} + \frac{1}{2}$, όπου $y \in I$, το l αντιπροσωπεύει το «αριστερά» και r το «δεξιά». Όπως φαίνεται από τα παραπάνω, ο βαθμός συνδέσεων

εξόδου κάθε σημείου είναι 2 ενώ ο βαθμός εισόδου είναι 1. Στο Σχήμα 21 το άνω διάγραμμα απεικονίζει τις ακμές ενός σημείου στο *I*, ενώ το κάτω διάγραμμα απεικονίζει πως ένα διάστημα αντιστοιχίζεται σε δύο διαστήματα, το καθένα μισό μέγεθος από το αρχικό. Επεκτείνοντας το συμβολισμό μπορούμε να γράψουμε $r([y,z]),l([y,z])$ συμβολίζοντας την εικόνα του διαστήματος $[y,z]$ κάτω από τα *r*, *l*. Με βάση αυτά είναι δυνατό να οριστεί ο διακριτός Distance Halving γράφος. Έστω $\vec{x}$ ένα σύνολο από *n* σημεία στο *I*. Το σημείο $\boldsymbol{x}_i$ μπορεί να είναι το αναγνωριστικό του επεξεργαστή $\boldsymbol{u}_i$, ή κάποιος κατατεμαχισμός του αναγνωριστικού του. Τα σημεία του $\vec{x}$ χωρίζουν το *I* σε *n* τμήματα, με κάθε τμήμα του $\boldsymbol{x}_i$ να ορίζεται ως $s(x_i) = [x_i, x_{i+1})(i = 1...n-1)$ και $s(x_n) = [x_{n-1},1) \cup [0,x_1)$.



Σχήμα 21. Οι ακμές και τα διαστήματα στο χώρο *I* του Distance Halving γράφου

Στον διακριτό Distance Halving γράφο $G_{\vec{x}}$, κάθε επεξεργαστής $\boldsymbol{u}_i$ συσχετίζεται με ένα τμήμα $s(\boldsymbol{x}_i)$. Εάν ένα σημείο *y* βρίσκεται στο $s(\boldsymbol{x}_i)$ τότε αναφέρεται πως το $\boldsymbol{u}_i$ καλύπτει το *y*. Ένα ζευγάρι κορυφών $(\boldsymbol{u}_i,\boldsymbol{u}_j)$ είναι ακμή του $G_{\vec{x}}$ εάν υπάρχει ακμή (y,z) στον συνεχή γράφο, τέτοια ώστε το $y \in s(\boldsymbol{x}_i)$ και το $z \in s(\boldsymbol{x}_j)$. Οι ακμές $(\boldsymbol{u}_i, \boldsymbol{u}_{i+1})$ και $(\boldsymbol{u}_n, \boldsymbol{u}_1)$ προστίθενται κατά τέτοιο τρόπο ώστε ο $G_{\vec{x}}$ να περιέχει ένα δακτύλιο. Οι ακμές του δακτυλίου αποτελούν αντι-παράλληλες κατευθυνόμενες ακμές.

Εάν ένας επεξεργαστής $\boldsymbol{u}_i$ επιθυμεί να εισέλθει στο σύστημα, τότε ακολουθεί τα εξής βήματα:

1. Επιλέγει ένα $\boldsymbol{x}_i$ και το θέτει ως αναγνωριστικό του $\boldsymbol{u}_i.id \leftarrow \boldsymbol{x}_i$.
2. Θέτει ως επεξεργαστή το $\boldsymbol{u}_j$, τέτοιο ώστε $\boldsymbol{x}_i \in s(\boldsymbol{x}_j)$. Αναζητεί το $\boldsymbol{u}_j$ και λαμβάνει από αυτόν όλα τα αντικείμενα δεδομένων που αντιστοιχούν στο $s(\boldsymbol{x}_j)$ και τις διευθύνσεις όλων των γειτόνων που πρέπει να έχει ο $\boldsymbol{u}_i$.

3. Ενημερώνει όλους τους γείτονες του $\boldsymbol{u}_j$ ώστε να αλλάξουν τους πίνακες διευθύνσεών τους κατάλληλα.

Θεωρώντας ότι ο γράφος έχει σταθερό βαθμό, το τρίτο βήμα του παραπάνω αλγόριθμου απαιτεί μόνο O(1) μηνύματα. Κατά την αποχώρηση ενός επεξεργαστή, η απλούστερη λύση που υλοποιείται είναι η ανάληψη του αντίστοιχου διαστήματος από τον προκάτοχό του, ο οποίος επεκτείνει το διάστημά του ώστε να περιλαμβάνει το $s(\boldsymbol{u}_i)$.

Η αντιστοίχηση των δεδομένων σε κόμβους γίνεται με τρόπο παρόμοιο με αυτό που συναντούμε σε άλλους αλγορίθμους κατανεμημένων πινάκων κατατεμαχισμού. Αρχικά τα δεδομένα αντιστοιχίζονται στο διάστημα I με τη χρήση μιας συνάρτησης κατατεμαχισμού. Στην συνέχεια, ο επεξεργαστής $\boldsymbol{u}_i$ είναι υπεύθυνος για όλα τα σημεία στο $s(\boldsymbol{u}_i)$. Η συνάρτηση κατατεμαχισμού ορίζεται κατά τη δημιουργία του δικτύου και εν συνεχεία παραμένει σταθερή και γνωστή σε όλους τους επεξεργαστές που εισέρχονται στο δίκτυο.

Η κατασκευή του Distance Halving υλοποιεί το γνωστό γράφο De Bruijn. Ως γνωστό, ο γράφος De Bruijn r-διαστάσεων αποτελείται από 2^r επεξεργαστές και 2^{r+1} κατευθυνόμενες ακμές, με κάθε κόμβο να αντιστοιχεί σε μία δυαδική ακολουθία μεγέθους r και κάθε κόμβο $\boldsymbol{u}_1\boldsymbol{u}_2\ldots\boldsymbol{u}_r$ να έχει δύο κατευθυνόμενες ακμές προς τους κόμβους $\boldsymbol{u}_2\ldots\boldsymbol{u}_{r-1}\mathbf{0}$ και $\boldsymbol{u}_2\ldots\boldsymbol{u}_{r-1}\mathbf{1}$. Το Distance Halving DHT προσομοιώνει το γράφο De Bruijn με τον ακόλουθο τρόπο. Έστω ότι $n=2^r$ και $\vec{x}$ ένα σύνολο από m σημεία τέτοια ώστε $x_i=i/n$. Τότε ο Distance Halving γράφος $G_{\vec{x}}$ είναι ισομορφικός προς τον De Bruijn γράφο r-διαστάσεων.

Το Distance Halving χρησιμοποιεί τις ιδιότητες του συνεχούς γράφου για να ορίσει δύο απλούς αλγορίθμους δρομολόγησης στον διακριτό γράφο. Ο πρώτος αλγόριθμος έχει μικρά μονοπάτια αναζήτησης, ενώ ο δεύτερος αυξάνει το μονοπάτι δρομολόγησης κατά ένα μέγιστο παράγοντα 2 για να βελτιώσει την κατανομή κίνησης. Έστω ότι ο επεξεργαστής $\boldsymbol{u}_i$ επιθυμεί να αναζητήσει το σημείο y. Το σημείο y μπορεί να είναι είτε το αναγνωριστικό ενός επεξεργαστή είτε το κλειδί ενός αρχείου (y=h(όνομα-αρχείου), όπου h η συνάρτηση κατατεμαχισμού) και επομένως μπορεί να είναι ένα μοναδικό σημείο στην γραμμή, οπότε ο $\boldsymbol{u}_i$ αναζητά τον επεξεργαστή $\boldsymbol{u}_j$ για τον οποίο $y \in s(\boldsymbol{x}_j)$.

Άπληστος αλγόριθμος: Εάν για έναν επεξεργαστή ισχύει $y \in s(\boldsymbol{x}_j)$, τότε το $s(\boldsymbol{x}_j)$ θα περιέχει επίσης σημεία που είναι κοντά στο y. Αυτή η παρατήρηση χρησιμοποιείται για την εξομοίωση της αναζήτησης στο $G_{\vec{x}}$. Το πρωτόκολλο δρομολογεί τα μηνύματα μέσω των ακμών που κατευθύνονται προς τα πίσω, και επομένως θεωρείται ότι κάθε ακμή είναι αντι-παράλληλη. Η επικεφαλίδα

του μηνύματος περιέχει τον προορισμό y και την τρέχον θέση του μηνύματος στο I. Ο άπληστος αλγόριθμος ακολουθεί τα εξής δύο βήματα:

1. Έστω w το σημείο που βρίσκεται στην μέση του x_i και **σ** η δυαδική αναπαράσταση του w. Υπολόγισε το ελάχιστο **t** τέτοιο ώστε το $\sigma_t(y)$ να βρίσκεται μέσα στο $s(x_j)$.
2. Έστω $\mathbf{z}=\sigma_t(y)$, με $\mathbf{z} \in s(x_j)$. Προχώρησε από το **z** (δηλαδή από το $s(x_j)$ και επομένως από το u_i) στις ακμές που κατευθύνονται προς τα πίσω. Κάθε φορά που το μήνυμα προωθείται πρέπει να ανανεώνεται η επικεφαλίδα ώστε να περιέχει την τρέχων θέση του μηνύματος στο *I*. Ο επεξεργαστής που περιέχει το y εντοπίζεται μετά από **t** μηνύματα.

Distance Halving αλγόριθμος: Ο αλγόριθμος αναζήτησης Distance Halving επιτυγχάνει μικρή συμφόρηση ακόμα και στις χειρότερες περιπτώσεις. Αποτελείται από δύο στάδια. Στο πρώτο στάδιο στέλνει το μήνυμα σε ένα σχεδόν τυχαίο προορισμό και στο δεύτερο στάδιο δρομολογεί το μήνυμα από τον τυχαίο προορισμό προς το στόχο. Η επικεφαλίδα του μηνύματος πρέπει να περιέχει την πηγή x_i, το στόχο y και μια τυχαία ακολουθία **σ**. Όταν ένας επεξεργαστής λάβει ένα μήνυμα ακολουθεί τα ακόλουθα βήματα για το πρώτο στάδιο:

1. Ελέγχει εάν το $\sigma_t(y)$ καλύπτεται από το παρόν διάστημα ή από ένα από τα γειτονικά διαστήματα. Εάν καλύπτεται τότε το μήνυμα προωθείται στον επεξεργαστή που είναι υπεύθυνος για το $\sigma_t(y)$ και ο αλγόριθμος συνεχίζει στο δεύτερο στάδιο.
2. Επέλεξε τυχαία ένα δυφίο και πρόσθεσε το στην τυχαία ακολουθία του **σ** που βρίσκεται στην επικεφαλίδα του μηνύματος.
3. Υπολόγισε το νέο $\sigma_t(x_i)$ και ανανέωσε την επικεφαλίδα.
4. Προώθησε το μήνυμα στον γείτονα που καλύπτει το σημείο $\sigma_t(x_i)$. (προϋποθέτει την ύπαρξη ακμής)

Στο δεύτερο στάδιο το μήνυμα προωθείται μέσω των ακμών που κατευθύνονται προς τα πίσω από το $\sigma_t(y)$ στο y. Σε κάθε βήμα ο επεξεργαστής που διαχειρίζεται το μήνυμα διαγράφει το τελευταίο δυφίο του **σ**. Η διαδικασία προσομοιώνει την ταυτόχρονη κίνηση δύο μηνυμάτων, ενός από την πηγή και ενός από το στόχο. Και τα δύο μηνύματα κινούνται με βάση την ίδια ακολουθία **σ**. Σε κάθε βήμα, η απόσταση μεταξύ τους μειώνεται στο μισό εωσότου συναντηθούν σε ένα τυχαίο σημείο στη μέση.

3.5.7. Υπερπηδώμενοι Γράφοι

Οι υπερπηδώμενοι γράφοι έχουν ερευνηθεί από δύο διαφορετικές ερευνητικές ομάδες, αυτήν των Aspens και Shah και αυτήν των Jones κ.α. [96],[97]. Η θεωρία των υπερπηδώμενων γράφων βασίζεται στην επέκταση

των υπερπηδώμενων λιστών [98][99]. Σε αντίθεση με τα ισοζυγισμένα δέντρα, οι υπερπηδώμενες λίστες βασίζονται σε θεωρία πιθανοτήτων και οι μηχανισμοί εισαγωγής και διαγραφής δεν απαιτούν αναδιαμόρφωση του δέντρου, οπότε είναι συνήθως γρηγορότεροι κατά ένα σταθερό παράγοντα. Μια υπερπηδώμενη λίστα αποτελείται μια σειρά επιπέδων με ταξινομημένες συνδεδεμένες λίστες. Όλοι οι κόμβοι συμμετέχουν στην λίστα του βασικού επιπέδου 0. Μερικοί κόμβοι συμμετέχουν επίσης στην λίστα επιπέδου 1 με κάποια σταθερή πιθανότητα. Ένα υποσύνολο αυτών συμμετέχουν στην λίστα επιπέδου 2 και ούτως καθεξής. Μια αναζήτηση μπορεί να προσπελάσει γρήγορα τη λίστα διαβαίνοντας τα αραιά υψηλότερα επίπεδα εωσότου εντοπίσει το στόχο της. Δυστυχώς, οι κόμβοι που βρίσκονται στα υψηλότερα επίπεδα μιας υπερπηδώμενης λίστας αποτελούν σημεία κίνησης και σφάλματος. Σε αντίθεση με τις υπερπηδώμενες λίστες, οι υπερπηδώμενοι γράφοι προσφέρουν πολλαπλές λίστες σε κάθε επίπεδο για πλεονασμό και κάθε κόμβος συμμετέχει σε μια λίστα σε κάθε επίπεδο.

Κάθε κόμβος σε έναν υπερπηδώμενο γράφο έχει κατά μέσο όρο O(logn) γείτονες, όπως και στα περισσότερα DHT συστήματα. Το κύριο πλεονέκτημα των υπερπηδώμενων γράφων έναντι των DHT συστημάτων είναι η υποστήριξη αναζητήσεων με βάση προθέματα και γειτνίαση. Οι DHT αλγόριθμοι κατατεμαχίζουν τα αντικείμενα σε ένα τυχαίο σημείο του γράφου. Ως επακόλουθο, δεν προσφέρουν εγγυήσεις ως προς την τοποθεσία αποθήκευσης των δεδομένων ή πως το μονοπάτι ανάκτησης των δεδομένων θα παραμένει πάντα στην ίδια διαχειριστική αρχή, όποτε αυτό είναι εφικτό [97]. Αντιθέτως, οι υπερπηδώμενοι γράφοι προσφέρουν αναζητήσεις ονομάτων που λαμβάνουν υπόψη τους την τοπολογία του δικτύου [97]. Καθώς οι λίστες είναι ταξινομημένες, οι υπερπηδώμενοι μπορούν επίσης να υποστηρίξουν αναζητήσεις διαστημάτων. Τέλος, προσφέρουν μηχανισμούς για την ένωση διασπασμένων τμημάτων [100], οι οποίοι όμως λειτουργούν μόνο σειριακά, καθώς η σχεδίαση παράλληλων αλγορίθμων ένωσης δεν έχει ακόμα υλοποιηθεί.

Τα πλεονεκτήματα των υπερπηδώμενων γράφων έχουν δυστυχώς και κάποιο κόστος. Για να μπορούν να παρέχουν αναζητήσεις διαστημάτων και ελαστικότητα στην τοποθέτηση δεδομένων, οι υπερπηδώμενοι γράφοι απαιτούν πολλούς περισσότερους δείκτες από ότι οι αντίστοιχοι DHT αλγόριθμοι. Ο αυξημένος αριθμός δεικτών μεταφράζεται σε αυξημένη κίνηση διαχείρισης. Ένα άλλο μειονέκτημα που είχαν τουλάχιστον οι πρώτες υλοποιήσεις των υπερπηδώμενων γράφων ήταν η έλλειψη αλγορίθμου για την κατανομή των κλειδιών σε υπολογιστές. Ως επακόλουθο, δεν υπήρχαν εγγυήσεις για συνολική ισοκατανομή των κλειδιών ή την απόσταση μεταξύ γειτονικών κλειδιών [101]. Οι Aspnes, Kirsch κ.α. σχεδίασαν ένα σχήμα για

τη μείωση των δεικτών μεταξύ υπολογιστών από O(m logm), όπου m είναι ο αριθμός των δεδομένων, σε O(nlogn), όπου n είναι ο αριθμός των κόμβων [101]. Στην συνέχεια της ενότητας θα παρουσιάσουμε αναλυτικά τη δομή του συστήματος των Aspens, Kirsch κ.α.. Παρόλο τις βελτιώσεις που σχεδίασαν, υπάρχουν ακόμα αρκετά άλυτα ζητήματα που χαρακτηρίζουν τις υλοποιήσεις βασισμένες σε υπερπηδώμενους γράφους:

- Αρκετές εξομοιώσεις έχουν γίνει αλλά η λεπτομερής ανάλυσή τους εκλείπει
- Οι μηχανισμοί εισαγωγής και αποχώρησης απαιτούν αρκετή βελτίωση
- Δεν έχει υπάρξει αρκετή έρευνα ως προς τον τρόπο χειρισμού κόμβων με διαφορετικές χωρητικότητες

3.5.7.1. Aspens, Kirsch και Krishnamurthy

Ο αλγόριθμος των Aspens, Kirsch και Krishnamurthy [101] επιχειρεί να λύσει το πρόβλημα σχεδίασης μιας πολιτικής κατανεμημένης εξισορρόπησης κίνησης για την ανάθεση στοιχείων ενός υπερπηδώμενου γράφου, όπου κάθε μηχάνημα ελέγχει ένα συγκεκριμένο διάστημα στο χώρο κλειδιών και με την προϋπόθεση ότι παραπλήσια κλειδιά αποθηκεύονται στο ίδιο μηχάνημα. Αυτή η ιδιότητα επιτρέπει την αντικατάσταση ολόκληρου του υπερπηδώμενου γράφου από μια συντομευμένη δομή δεδομένων που περιέχει μόνο ένα σταθερό αριθμό δειγματοληπτικών στοιχείων από κάθε μηχάνημα. Με τον τρόπο αυτό μειώνεται ο αριθμός των δεικτών μεταξύ των μηχανημάτων από O(nlogn), όπου n ο αριθμός των στοιχείων ή των κλειδιών, σε O(mlogm), όπου m είναι ο αριθμός των μηχανημάτων στο σύστημα (όπου προφανώς $m<n$).

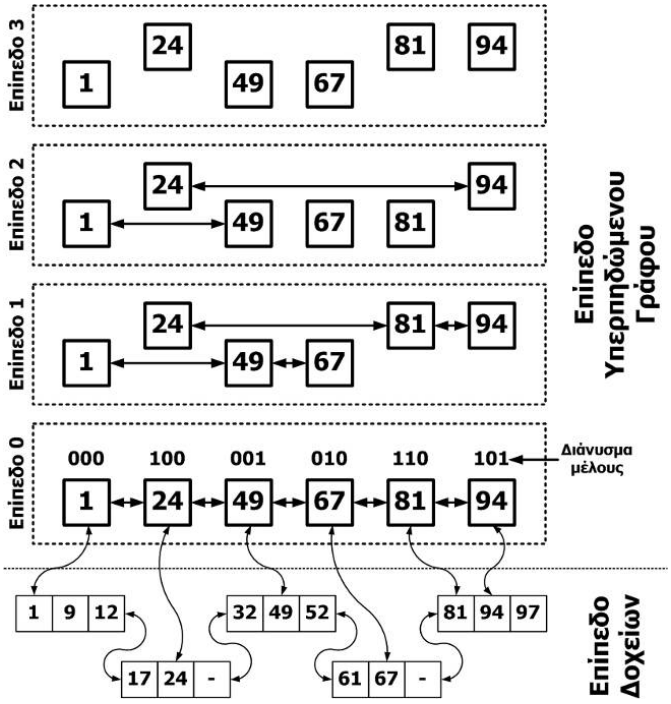
Η ανάθεση παραπλήσιων κλειδιών στο ίδιο μηχάνημα γίνεται ώστε κλειδιά που λογικά σχετίζονται να βρίσκονται σε κοντινή γεωγραφική θέση και οι περισσότεροι δείκτες να είναι τοπικοί. Πιο συγκεκριμένα, ο αλγόριθμος ομαδοποιεί κλειδιά σε δοχεία, με κάθε μηχάνημα να κατέχει έναν αριθμό δοχείων και να είναι υπεύθυνο για όλα τα αντικείμενα των οποίων τα κλειδιά βρίσκονται σε αυτά τα δοχεία. Ο αλγόριθμος δε θέτει περιορισμούς στα κλειδιά με εξαίρεση ότι πρέπει να μπορούν να διαταχθούν. Εν συνεχεία, οι εφαρμογές μπορούν να αναθέσουν κλειδιά σε στοιχεία με όποιο τρόπο επιθυμούν.

Κάθε δοχείο επιλέγει ένα αντιπροσωπευτικό κλειδί να εμφανίζεται στο υπερπηδώμενο γράφο. Αυτά τα αντιπροσωπευτικά κλειδιά χρησιμοποιούνται κατά τις λειτουργίες αναζήτησης για να δρομολογούν προς το κατάλληλο δοχείο. Αυτή η λογική μειώνει την πολυπλοκότητα του χώρου ενός

υπερπηδώμενου γράφου σε O(blogb) δείκτες, όπου b ο αριθμός των δοχείων στο σύστημα, μέσω της μείωσης των κλειδιών που εισέρχονται στο γράφο (περίπου ένα κλειδί για κάθε δοχείο). Αυτός ο μηχανισμός απαιτεί ελάχιστες αλλαγές στους αλγορίθμους των υπερπηδώμενων γράφων, μειώνοντας σημαντικά τον αριθμό των καταστάσεων που απαιτούνται από την ίδια τη δομή. Τα αντιπροσωπευτικά κλειδιά επιλέγονται κυρίως από τα κεντρικά στοιχεία ενός δοχείου. Αυτό επιτρέπει σε αντικείμενα που βρίσκονται προς το τέλος να μεταναστεύουν σε διπλανά δοχεία χωρίς να απαιτείται ανανέωση της δομής του υπερπηδώμενου γράφου εκτός και αν μετακινηθεί το αντιπροσωπευτικό κλειδί, οπότε πρέπει να αντικατασταθεί και στον γράφο.

Η παραπάνω χρήση των δοχείων προσφέρει επίσης το πλεονέκτημα διαίρεσης του το συστήματος σε δύο (σχεδόν) ορθογώνιες δομές δεδομένων. Με αυτόν τον τρόπο το σύστημα μπορεί να παρουσιαστεί ότι αποτελείται από δύο επίπεδα. Αυτός ο συνδυασμός του επιπέδου του υπερπηδώμενου γράφου και του επιπέδου των δοχείων αναφέρεται ως «σύστημα δύο επιπέδων», το οποίο απεικονίζεται στο Σχήμα 22.

Το άνω επίπεδο αποτελείται από τον υπερπηδώμενο γράφο, όπου κάθε κλειδί αποθηκεύει δείκτες προς τους γείτονές του και ένα δείκτη προς το δοχείο στο οποίο ανήκει. Αυτή η δομή λειτουργεί ως υπερκείμενο δίκτυο που υποστηρίζει τόσο αναζητήσεις όσο και τη διαχείριση μις ελεύθερης λίστας. Το κάτω επίπεδο αποτελείται από την αλυσίδα των δοχείων, κατανεμημένη στα μηχανήματα του συστήματος. Η διαίρεση αυτή είναι ιδιαίτερα χρήσιμη καθώς επιτρέπει την έμμεση βελτιστοποίηση του υπερπηδώμενου γράφου μέσω της διαχείρισης της αλυσίδας των δοχείων και της διεπαφής μεταξύ των δύο επιπέδων, δίχως να διαταράσσονται οι επιθυμητές ιδιότητες του υπερκείμενου υπερπηδώμενου γράφου.



Σχήμα 22. Το «σύστημα δύο επιπέδων» των Aspens, Kirsch και Krishnamurthy

Ενώ η παραπάνω δομή δίνει τη δυνατότητα προσαρμογής με βάση τη γεωγραφική θέση των μηχανημάτων, παρουσιάζει προβλήματα κατανομής κίνησης και δεδομένων καθώς αυτά εισέρχονται και αποχωρούν από το σύστημα. Για αυτόν το λόγο οι Aspens κ.α. παρουσίασαν έναν αλγόριθμο ισοκατανομής κίνησης που επιλύει αυτά τα προβλήματα με βάση τη δομή των δύο επιπέδων.

Ο αλγόριθμος αναζήτησης για τη δομή δύο επιπέδων είναι όμοιος με αυτόν της αναζήτησης του κοντινότερου στοιχείου προς το στόχο στους υπερπηδώμενους γράφους, με την προσθήκη μιας επιπλέον αναζήτησης μέσα στο δοχείο που περιέχει το στοιχείο και τα δύο γειτονικά του δοχεία. Επομένως, το κόστος της αναζήτησης είναι λογαριθμικό ως προς τον αριθμό των στοιχείων του υπερπηδώμενου γράφου, που είναι περίπου ίσος προς τον αριθμό των δοχείων.

Ο αλγόριθμος των Aspens κ.α. χρησιμοποιεί τοπικές αλλαγές για την εξισορρόπηση του φόρτου κίνησης. Η βασική ιδέα έγκειται στο ότι ένα μηχάνημα που επιφορτίζεται με κίνηση μπορεί να προσπαθήσει να μετατοπίσει μερικά από τα κλειδιά του σε δοχεία γειτονικών μηχανημάτων που έχουν λιγότερη κίνηση. Για το λόγο αυτό διατηρείται μια «ελεύθερη λίστα» από δοχεία, ώστε αν όλα τα δοχεία είναι επιφορτισμένα με κίνηση, ένα νέο άδειο δοχείο εγγράφεται ώστε να επωμιστεί ένα μέρος της κίνησης, ταξινομώντας τα δοχεία σε ενεργά ή ελεύθερα. Η υλοποίηση των ελεύθερων λιστών γίνεται μέσω ενός ξεχωριστού υπερπηδώμενου γράφου. Κάθε ελεύθερο

δοχείο συσχετίζεται με μια τυχαία τιμή κλειδιού, εισάγει ένα δείκτη προς το δοχείο στον υπερπηδώμενο γράφο και ικανοποιεί τις αιτήσεις για ελεύθερα δοχεία επιστρέφοντας ένα οποιοδήποτε στοιχείο που είναι αποθηκευμένο στον υπερπηδώμενο γράφο. Αυτή η στρατηγική προσφέρει εντοπισμό ελεύθερων δοχείων με κόστος O(logb) από οποιοδήποτε σημείο εισόδου στον υπερπηδώμενο γράφο.

Κάθε δοχείο κατηγοριοποιείται επιπλέον ως ανοιχτό ή κλειστό. Υπάρχουν διάφοροι τρόποι για τη δημιουργία αυτής της διαφοροποίησης. Ένας από αυτούς είναι η χρήση ενός ορίου ως προς τον αριθμό των κλειδιών, πέρα του οποίου ένα δοχείο θεωρείται κλειστό, ή η χρήση ενός ορίου με βάση το ποσοστό κίνησης που δέχεται.

Εν συνεχεία, η λίστα των ενεργών δοχείων χωρίζεται σε ομάδες ανά δύο ή τρία, διατηρώντας την ιδιότητα πως κάθε κλειστό δοχείο βρίσκεται δίπλα σε ένα ανοικτό δοχείο και πως κάθε ανοιχτό δοχείο έχει ένα κλειστό δοχείο στα αριστερά του. Η ιδιότητα αυτή απαιτεί κάθε σύστημα να έχει τουλάχιστον δύο δοχεία. Στην αρχική φάση του συστήματος, όπου υπάρχει μόνο ένα μηχάνημα, θεωρείται ότι το μηχάνημα διατηρεί και τα δύο δοχεία. Κάθε ομάδα πρέπει να περιέχει έναν από τους δύο παρακάτω τύπους, όπου *C* αντιπροσωπεύει ένα κλειστό δοχείο και *O* ένα ανοιχτό δοχείο:

1. *C-O*
2. *C-O-C*

Η παραπάνω δομή διατηρείται μέσω της μεταφοράς κλειδιών μεταξύ γειτονικών δοχείων, όποτε αυτό χρειάζεται. Επομένως, εάν ένα κλειδί πρέπει να προστεθεί σε ένα κλειστό δοχείο, ένα κλειδί μεταφέρεται προς το διπλανό ανοιχτό δοχείο. Με τον ίδιο τρόπο, η διαγραφή ενός κλειδιού από ένα κλειστό δοχείο προϋποθέτει τη μεταφορά ενός κλειδιού από ένα ανοιχτό δοχείο. Καθώς ένα ανοιχτό δοχείο δέχεται περισσότερα κλειδιά, μπορεί να δηλώσει τον εαυτό του κλειστό, απαιτώντας αναδιοργάνωση της δομής των δοχείων. Ο τρόπος λειτουργίας αυτών των αναδιοργανώσεων περιγράφεται στην συνέχεια για τις λειτουργίες εισαγωγής και αποχώρησης από το σύστημα.

Εισαγωγή: Οι εισαγωγές σε κλειστά δοχεία που δεν προκαλούν το κλείσιμο διπλανών ανοιχτών δοχείων είναι αρκετά απλή και απαιτεί απλά τη μεταφορά κλειδιών όπως αυτή ορίστηκε παραπάνω. Υπάρχουν δύο πιθανοί τύποι εισαγωγής που απαιτούν αναδιοργάνωση της δομής των δοχείων. Σε κάθε έναν από αυτούς τους τύπους, το *O'* αντιπροσωπεύει ένα καινούργιο δοχείο από την ελεύθερη λίστα και ένα κλειδί εισάγεται σε ένα κλειστό δοχείο *C1*:

1. *C1-O2* $\Rightarrow$ *C1-O'-C2*
2. *C1-O2-C3* $\Rightarrow$ *C1-O2* | *C3-O'*

Στην πρώτη περίπτωση, το νέο κλειδί εισάγεται στο *C1*, προκαλώντας τη μεταφορά ενός κλειδιού από το *C1* στο *O2*. Αυτό προκαλεί το *O2* να γίνει κλειστό. Το νέο δοχείο *O'* αποσπάται από την ελεύθερη λίστα ώστε να αποκατασταθεί η δομή. Στην δεύτερη περίπτωση, το νέο κλειδί εισάγεται στο *C1*, προκαλώντας τη μεταφορά ενός κλειδιού από το *C1* στο *O2*. Εν συνεχεία, το *O2* μεταφέρει ένα κλειδί στο *C3*, ώστε να παραμείνει ανοιχτό. Το *C3* πρέπει στην συνέχεια να μεταφέρει ένα κλειδί στο άδειο δοχείο *O'*.

Αποχώρηση: Κατά τη λειτουργία της αποχώρησης υπάρχουν τρεις πιθανοί συνδυασμοί που να απαιτούν αναδιοργάνωση των δοχείων. Σε όλες τις περιπτώσεις ένα δοχείο επανατοποθετείται στην ελεύθερη λίστα μόνο όταν γίνεται αναδιοργάνωση. Η ύπαρξη ενός ανοιχτού δοχείου σε μια ομάδα, το οποίο δεν περιέχει κανένα κλειδί, θεωρείται έγκυρη. Στις παρακάτω περιπτώσεις, το *O** αντιπροσωπεύει ένα άδειο δοχείο το οποίο θα επανατοποθετηθεί στην ελεύθερη λίστα:

1. *C1-O*-C3* $\Rightarrow$ *C1-C3*
2. *C1-O2* | *C3-O** $\Rightarrow$ *C1-O2-C3*
3. *C1-O2-C3* | *C4-O** $\Rightarrow$ *C1-O2* | *C3-O4*

Στην πρώτη περίπτωση, το *C3* μεταφέρει ένα από τα κλειδιά του στο *C1* και δημιουργούν μια ομάδα 2 δοχείων. Στην δεύτερη περίπτωση, αφού το κλειδί διαγραφεί από το *C3*, το *O2* μεταφέρει ένα κλειδί στο *C3* με αποτέλεσμα τη δημιουργία μιας ομάδας 3 δοχείων. Εάν το *O2* είναι επίσης άδειο, τότε το *C1* μπορεί να ομαδοποιηθεί με το *C3*, το οποίο είναι πλέον ανοιχτό, για να δημιουργήσει μια ομάδα 2 δοχείων. Στην τρίτη περίπτωση, εκτελείται παρόμοια μεταφορά για τη δημιουργία δύο ομάδων 2 δοχείων.

3.6. Βασικά ζητήματα στα DHT συστήματα

Έχοντας δει τους βασικότερους DHT αλγόριθμους, μέσω των συστημάτων που αναφέραμε, μπορούμε να επικεντρωθούμε σε μερικά βασικά ζητήματα που χαρακτηρίζουν τη σωστή λειτουργία τους. Με βάση αυτά θα είμαστε σε θέση να αξιολογήσουμε τους αλγόριθμους που παρουσιάσαμε και κυρίως να αναπτύξουμε ορθότερα την αρχιτεκτονική του δικού μας συστήματος, που παρουσιάζεται στο επόμενο κεφάλαιο. Τα κυριότερα από αυτά τα προβλήματα αποτελούν η απόδοση του συστήματος, η ανεκτικότητα σε ξαφνικές εξόδους, τα σημεία έντονης κίνησης, η χρήση γεωγραφικών πληροφοριών των χρηστών και η ετερογένεια του περιεχομένου.

3.6.1. Απόδοση συστήματος

Το βασικότερο μέτρο απόδοσης αποτελεί για τους DHT αλγορίθμους το μήκος του μονοπατιού, που απαιτείται για μια επιτυχή αναζήτηση. Οι περισσότεροι αλγόριθμοι έχουν μήκος μονοπατιού O(log n) βήματα, όπου n ο αριθμός των κόμβων/χρηστών, με εξαίρεση το CAN που έχει $O(dn^{1/d})$, όπου d η διάσταση του d-τόρου . Το μήκος όμως αυτό εξαρτάται άμεσα από τον αριθμό των γειτονικών κόμβων που απαιτείται να διατηρείται από κάθε κόμβο. Όσο μεγαλύτερο το μέγεθος τόσο περισσότερη μνήμη πρέπει να διατίθεται από τους χρήστες και τόσες περισσότερες ενημερώσεις πρέπει να γίνονται για να διατηρείται ο πίνακας των γειτόνων ενήμερος καθώς χρήστες εισέρχονται και εξέρχονται από το σύστημα. Όσον αφορά τη διάθεση μνήμης, λόγω της ραγδαίας πτώσης των τιμών στον εξοπλισμό μνήμης, αυτό δεν αποτελεί ιδιαίτερο περιορισμό. Όσον αφορά το δεύτερο όμως, η συνεχής αλλαγή του πληθυσμού των ομότιμων συστημάτων περιορίζει το μέγεθος των γειτόνων που μπορούν να είναι διαρκώς ενημερωμένοι, καθώς απαιτείται συχνή ανταλλαγή μηνυμάτων. Οι περισσότεροι αλγόριθμοι απαιτούν O(logn) γείτονες, με εξαίρεση τον CAN που απαιτεί O(d). Χρειάζεται επομένως να υπάρξει ένας συνδυασμός των δύο παραπάνω μέτρων για την ανάπτυξη ενός αλγορίθμου με μικρό μονοπάτι αναζήτησης και όσο το δυνατόν λιγότερους γείτονες.

3.6.2. Ανεκτικότητα σε αποτυχίες

Η απόδοση των συστημάτων όπως ορίστηκε προηγουμένως, αναφέρεται μόνο σε θεωρητική βάση. Στην πράξη, οι κόμβοι και οι αλγόριθμοι δρομολόγησης μέσω αυτών παρουσιάζουν αρκετές αποτυχίες, που οφείλονται στην ίδια τη φύση των ομότιμων συστημάτων. Υπάρχουν τρία ζητήματα όσον αφορά τα σφάλματα που μπορεί να προκύψουν και να οδηγήσουν έναν αλγόριθμο σε ανεπιτυχή αναζήτηση.

Το πρώτο αφορά τη δυνατότητα δρομολόγησης καθώς κόμβοι εξέρχονται από το σύστημα και οι γειτονικοί κόμβοι μπορούν μεν να αντιληφθούν αυτήν την αποχώρηση αλλά δεν έχουν προλάβει να αποκτήσουν νέους γείτονες στην θέση τους. Θα αποκαλέσουμε αυτήν την περίπτωση ως *στατική ανεκτικότητα* και θα τη μετρήσουμε με το ποσοστό των κλειδιών που είναι ακόμη εφικτά και το μήκος του μονοπατιού αναζήτησης.

Το δεύτερο αφορά στην δυνατότητα δρομολόγησης όταν κόμβοι εξέρχονται και οι γειτονικοί κόμβοι έχουν προλάβει να αποκτήσουν μερικούς γείτονες στην θέση τους, κυρίως μέσω μιας εναλλακτικής λίστας που διατηρούν, όπως η *λίστα διαδόχων* ή το *σετ φύλλου*. Η ύπαρξη τέτοιων λιστών

μπορεί μεν να επιβεβαιώνει την ορθή λειτουργία των αλγορίθμων αλλά κοστίζει σε μνήμη και κυρίως σε κίνηση που προκαλείται από την επικαιροποίησή της.

Τέλος, υπάρχει και το ζήτημα του χρόνου που απαιτείται ώστε το σύστημα να επανέλθει σε πλήρη ομαλή λειτουργία. Αυτό βέβαια απαιτεί την ανανέωση των πινάκων όλων των γειτόνων, οπότε ένα μέτρο σύγκρισης αποτελεί ο αριθμός των μηνυμάτων που πρέπει να ανταλλαχθούν μέχρι να αποκατασταθεί η σωστή λειτουργία.

3.6.3. Δρομολόγηση σε σημεία έντονης κίνησης

Όταν υπάρχει μεγάλη ζήτηση για ένα συγκεκριμένο κλειδί υπάρχει περίπτωση κάποιοι κόμβοι να υπερφορτωθούν με κίνηση και να μη μπορούν να λειτουργήσουν σωστά. Πολλοί αλγόριθμοι χρησιμοποιούν μεθόδους αντιγραφής, για να δημιουργήσουν πολλαπλά αντίγραφα τέτοιων κλειδιών σε διάφορους κόμβους, ώστε να διανέμουν την κίνηση πιο εξισορροπημένα. Αυτή η περίπτωση αποκαλείται «σημείων έντονης αναζήτησης».

Υπάρχει όμως και μια πιο δύσκολη περίπτωση, που αφορά έντονη κίνηση σε έναν κόμβο όχι διότι είναι υπεύθυνος για ένα κλειδί αλλά διότι πολλά μηνύματα αναζήτησης άλλων κλειδιών διαδίδονται μέσω αυτού. Η περίπτωση αυτή ονομάζεται «σημείων έντονης δρομολόγησης», καθώς αφορούν τη διαδρομή που ακολουθούν τα μηνύματα αναζήτησης. Η περίπτωση αυτή είναι αρκετά πιο πολύπλοκη από την προηγούμενη καθώς αφορά τον ίδιο τον αλγόριθμο δρομολόγησης και δεν επιλύεται με την απλή δημιουργία αντιγράφων κάποιων κλειδιών.

3.6.4. Γεωγραφική πληροφορία

Όλα τα παραπάνω μέτρα απόδοσης αφορούν βήματα σε επίπεδο εφαρμογής και όχι σε πραγματικό IP επίπεδο. Είναι λοιπόν σημαντικό να εξερευνηθεί η καθυστέρηση μιας αναζήτησης όχι μόνο σε βήματα μονοπατιού αλλά και σε πραγματικό χρόνο. Καθώς οι κόμβοι βρίσκονται διάσπαρτοι στο Διαδίκτυο, ένα βήμα σε επίπεδο εφαρμογής μπορεί να απέχει απλά όσο δύο υπολογιστές σε ένα τοπικό LAN ή να πρόκειται για δυο υπολογιστές σε διαφορετικές ηπείρους (που απαιτεί αρκετά πραγματικά βήματα και εμπεριέχει σημαντική χρονική καθυστέρηση). Οι θεωρητικές εκδόσεις όλων των παραπάνω αλγορίθμων δεν ασχολούνται με το πρόβλημα αυτό, στην πράξη όμως (όταν αναπτύχθηκαν ως δοκιμαστικές εφαρμογές) προσπάθησαν να συμπεριλάβουν λειτουργίες που εξετάζουν την γεωγραφική απόσταση των

κόμβων. Υπάρχουν τρεις βασικοί τρόποι για την επίλυση του γεωγραφικού αυτού προβλήματος.

3.6.4.1. Δρομολόγηση με βάση την εγγύτητα

Η δρομολόγηση με βάση την εγγύτητα ενσωματώνει την καθυστέρηση μεταξύ δυο κόμβων στην επιλογή του επόμενου κόμβου που πρέπει να δρομολογηθεί ένα μήνυμα αναζήτησης. Έτσι, ο αλγόριθμος δεν ελέγχει απλά ποιος κόμβος βρίσκεται πλησιέστερα στο κλειδί προς αναζήτηση αλλά και ποιος είναι ταυτόχρονα και πιο κοντά γεωγραφικά (ή χρονικά). Οι μέχρι τώρα ενδείξεις δείχνουν πως αυτή η τεχνική μειώνει το χρόνο δρομολόγησης.

3.6.4.2. Γείτονες με βάση την εγγύτητα

Αυτή η τεχνική ενσωματώνει την εγγύτητα όχι στην δρομολόγηση αλλά στον τρόπο που επιλέγονται οι γείτονες κάθε κόμβου και επομένως στην διάταξη του συστήματος. Με αυτόν τον τρόπο η δρομολόγηση των αναζητήσεων μέσω των γειτόνων ενσωματώνει τη γεωγραφική πληροφορία. Μόνο ο αλγόριθμος των Plaxton/Tapestry (θυμίζουμε ότι το Tapestry βασίζεται στο Plaxton) χρησιμοποιεί αυτήν την τεχνική, καθώς ο τρόπος κατασκευής του υπερκείμενου δικτύου στους άλλους αλγορίθμους δεν το επιτρέπει.

3.6.4.3. Γεωγραφική διάταξη

Σε αυτήν την τεχνική, τα αναγνωριστικά των κόμβων δεν αποδίδονται μέσω μιας τυχαίας κατανομής (μέσω της συνάρτησης κατατεμαχισμού) αλλά με βάση τη γεωγραφική τους θέσης, ώστε κόμβοι με μικρή γεωγραφική απόσταση να έχουν και μικρή απόσταση στον χώρο των αναγνωριστικών και επομένως στην διάταξη του συστήματος. Μια τέτοια τεχνική έχει επιχειρηθεί από το CAN με αρκετά καλά αποτελέσματα. Μελέτες έχουν δείξει [102] ότι η καθυστέρηση στο Διαδίκτυο μπορεί να μοντελοποιηθεί σε ένα γεωμετρικό σύστημα d-διαστάσεων. Επομένως, αλγόριθμοι που χρησιμοποιούν πολυδιάστατα αναγνωριστικά μπορούν εύκολα να ενσωματώσουν αυτήν την τεχνική.

Σε όλα τα παραπάνω θα πρέπει να επισημάνουμε ότι πολλές φορές η ενσωμάτωση γεωγραφικών πληροφοριών και η αναδιάταξη του συστήματος ή του τρόπου δρομολόγησης μπορεί να έχει αντίκτυπο στα υπόλοιπα χαρακτηριστικά που αναφέρουμε, όπως τη δημιουργία σημείων έντονης κίνησης ή την ανεκτικότητα σε αποτυχίες.

3.7. Σύνοψη - Συμπεράσματα

Στην ενότητα αυτή παρουσιάσαμε αναλυτικά την υπάρχουσα ερευνητική βιβλιογραφία σε ομότιμα συστήματα. Αρχικά προβάλλαμε τους διαφορετικούς τρόπους δεικτοδότησης με βάση την τοπολογία (τοπικούς, κεντρικούς και κατανεμημένους) και στην συνέχεια διαχωρίσαμε τους εννοιολογικούς δείκτες από τους μη-εννοιολογικούς. Η διατριβή αυτή βασίζεται σε μη-εννοιολογικούς δείκτες και για το λόγο αυτό παραθέσαμε στην συνέχεια διεξοδικά την υπάρχουσα βιβλιογραφία και έρευνα στον τομέα αυτό, παρουσιάζοντας τις διαφορετικές αρχιτεκτονικές που έχουν αναπτυχθεί με βάση τις προτεινόμενες τοπολογίες. Για κάθε ερευνητική εργασία αλλά και τοπολογία παραθέσαμε τα θετικά και αρνητικά χαρακτηριστικά, που μας βοήθησαν στην συνέχεια να καταλήξουμε στην χρήση δενδρικής δομής για τη σχεδίαση της αρχιτεκτονικής μας. Τέλος, παρουσιάσαμε κάποια βασικά ζητήματα που εντοπίστηκαν κατά τη μελέτη μας και τα οποία είναι κοινά στα περισσότερα DHT συστήματα.

3.8. Αναφορές

[1] Rowstron, A., Druschel, P., "Pastry: Scalable, decentralized object location, and routing for large-scale Peer-to-Peer systems," in Middleware 2001, Nov., 2001.

[2] Zhao, B. Y., Huang, L., Stribling, J., Rhea, S.C., Joseph, A.D., Kubiatowicz, J.D., "Tapestry: A resilient global-scale overlay for service deployment," IEEE Journal on Selected Areas in Communications, Jan., 2004.

[3] Hellerstein, J. M., " Toward network data indepence", in ACM SIGMOD Record, 32(3), 2003.

[4] Huebsch, R. , Hellerstein, J. M., Lanham, N., Loo, B. T., Shenker, S. , Stoica, I.,"Querying the Internet with PIER", in Proceedings of 19th International Conference on Very Large Databases (VLDB), Sep 2003.

[5] Cooper, B. F., Garcia-Molina, H., "SIL: Modeling and measuring scalable peer-to-peer search networks", in Proceedings of the International Workshop on Databases, Information Systems and Peer-to-Peer Computing, 2003.

[6] Manber, U., " Forward. Modern Information Retrieval", R. Baeza-Yates and B. Ribeiro-Neto, Reading, MA, Addison- Wesley: 5-8.

[7] Sen, S., Wang, J., "Analyzing Peer-to-Peer Traffic Across Large Networks", IEEE/ACM Transactions on Networking, 12(2): 219-232, 2004.

[8] Rodrigues, R., Liskov, B., Shrira, L., "The design of a robust peer-to-peer system", in SIGOPS European Workshop, 2002.

[9] Gupta, A., Liskov, B., Rodrigues, R. "One Hop Lookups for Peer-to-Peer Overlays", in 9th Workshop on Hot Topics in Operating Systems (HotOS), Lihue, Hawaii, 18-21 May, 2003

[10] Yang, B., Vinograd, P., Garcia-Molina, H., "Evaluating GUESS and Non-Forwarding Peer-to-Peer Search", in the 24th International Conference on Distributed Computing Systems (ICDCS-2004), Mar 23-26, Tokyo University of Technology, Hachioji, Tokyo, Japan, 2004.

[11] Mizrak, A., Cheng, Y., Kumar, V., Savage, S. "Structured superpeers: Leveraging heterogeneity to provide constant-time lookup", IEEE Workshop on Internet Applications, 2003.

[12] Gupta, A., Liskov, B., Rodrigues, R., "Efficient Routing for Peer-to-Peer Overlays", in First Symposium on Networked Systems Design and Implementation (NSDI), San Francisco, CA, March, 2004.

[13] Klingberg, T., Manfredi , R., "Gnutella 0.6", June, 2002

[14] Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S., "Search and replication in unstructured peer to peer networks", in Proceedings of the 16th international conference on supercomputing: 84-95, 2002.

[15] Yang, B., Garcia-Molina, H., "Efficient Search in Peer-to-Peer Networks", in Proceedings of the 22nd International Conference on Distributed Computing Systems, 2-5 July, Vienna, Austria, 2002.

[16] Adamic, L., Lukose, R., Puniyani, A., Huberman, B., "Search in power-law networks", Physical review, The American Physical Society 64(046135), 2001.

[17] Banaei-Kashani, F., Shahabi, C., "Criticality-based analysis and design of unstructured peer-to-peer networks as complex systems", in Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid: 351-358, 2003.

[18] Jovanovic, M., "Modeling Large-scale Peer-to-Peer Networks and a Case Study of Gnutella", Master's Thesis, Department of Electrical and Computer Engineering and Computer Science, College of Engineering, University of Cincinnati, 2001.

[19] KaZaa Media Desktop, Sharman Networks Ltd, 2001, www.kazaa.com

[20] Sen, S., Wang, J., "Analyzing peer-to-peer traffic across large networks", in Proceedings of the second ACM SIGCOMM workshop on Internet measurement: 137-150, 2002.

[21] Sen, S., Wang, J., "Analyzing Peer-to-Peer Traffic Across Large Networks", IEEE/ACM Transactions on Networking 12(2): 219-232, 2004.

[22] Direct Connect, http:www.neo-modus.com

[23]Saroiu, S., Gummadi, K., Dunn, R., Gribble, S., Levy, H., "An analysis of Internet content delivery systems", ACMSIGOPS Operating Systems Review 36: 315-327, 2002.

[24]Massoulie, L., Kermarrec, A., Ganesh, A., "Network awareness and failure resilience in self-organizing overlay networks", in Proceedings of the 22nd International Symposium on Reliable Distributed Systems, 6-8 Oct: 47-55, 2003.

[25]Loo, A., "The Future or Peer-to-Peer Computing", Communications of the ACM 46(9): 56-61. Sept 2003.

[26]Yang, B., Garcia-Molina, H., "Comparing Hybrid Peer-to-Peer Systems (extended)", in the 27th International Conference on Very Large Data Bases, Rome, Italy, 11-14 September, 2001.

[27]Scholl, D., "OpenNap Home Page", http://opennap.sourceforge.net/, 2001.

[28]Saroiu, S., Gummadi, P., Gribble, S., "A measurement study of peer-to-peer file sharing systems", in Proceedings of Multimedia Computing and Networking 2002 (MMCN'02), January, 2002.

[29]Chawathe, Y., Ratnasamy, S., Breslau, L., Lanham, N., Shenker, S. ""Making gnutella-like P2P systems scalable", in Proceedings of the 2003 conference on Applications, Technologies, Architectures and Protocols for Computer Communications, Karlsruhe, Germany, August 25-29: 407-418, 2003.

[30]Ghemawat, S., Gobioff, H., Leung, S.-T., "The Google file system", in Proceedings of the 19th ACM symposium on operating systems principles, Bolton Landing, New York, USA: 29-43, 2003.

[31]Clarke, I., "A Distributed Decentralised Information Storage and Retrieval System", Undergraduate Thesis, Division of Informatics, University of Edinburgh, 1999.

[32]Clarke, I., Sandberg, O., Wiley, B., Hong, T., "Freenet: A Distributed Anonymous Information Storage and Retrieval System", International Workshop on Design Issues in Anonymity and Unobservability. H. Federrath. New York, USA, Springer, 2001.

[33]Clarke, I., Miller,S., Hong, T., Sandberg, O., Wiley, B., ""Protecting Free Expression Online with Freenet", IEEE Internet Computing 6(1). Jan/Feb, 2002.

[34]Clarke, I., "Freenet's Next Generation Routing Protocol", http://freenetproject.org/index.php?page=ngrouting. 20th July, 2003.

[35]Mache, J., Gilbert, M., Guchereau, J., Lesh, J., Ramli, F., Wilkinson, M., "Request algorithms in Freenet-style peer-topeer systems", in Proceedings

of the Second IEEE International Conference on Peer to Peer Computing (P2P'02), September 5-7, Linkoping, Sweden, 2002.

[36]Kronfol, A. Z., "FASD: A fault-tolerant, Adaptive scalable distributed search engine", Master's Thesis, Princeton University, http://www.cs.princeton.edu/~akronfol/fasd/, 2002.

[37]Gribble, S., Brewer, E., Hellerstein, J. M., Culler, D., "Scalable, Distributed Data Structures for Internet Service Construction", in Proceedings of the 4th Symposium on Operating Systems Design and Implementation (OSDI 2000), San Diego, California, USA, October, 2000.

[38]Gray, J., "The transaction concept: Virtues and limitations", in Proceedings of VLDB, Cannes, France, September, 1981.

[39]Stonebraker, M., Aoki, P., Litwin, W., Pfeffer, A., Sah, A., Sidell, J., Staelin, C., Yu, A., "Mariposa: a wide-area distributed database system", THE VLDB Journal - The International Journal of Very Large Data Bases(5): 48-63, 1996.

[40]Loo, B. T., Hellerstein, J. M., Huebsch, R., Shenker, S., Stoica, I., "Enhancing P2P File-Sharing with Internet-Scale Query Processor", in Proceeings of the 30th International Conference on Very Large Data Bases (VLDB 2004), Toronto, Canada, 29 August - 3 September, 2004.

[41]Hellerstein, J. M., "Toward network data independence", ACM SIGMOD Record 32(3): 34-40. Sept, 2003.

[42]Walfish, M., Balakrishnan, H., Shenker, S., "Untangling the Web from DNS", in Proceedings of the First Symposium on Networked Systems Design and Implementation (NSDI'04), San Francisco, California, March 29-31: 225-238, 2004.

[43]Plaxton, C., Rajaraman, R., Richa, A., "Accessing nearby copies of replicated objects in a distributed environment", ACM Syposium on Parallel Algorithms and Architectures, June, 1997.

[44]Karger, D., Lehman, E., Leighton, T., Panigraphy, R., Levin, M., Lewin, D., "Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web", ACM Symposium on Theory of Computing, May, 1997.

[45]Litwin, W., Neimat, M.-A., Schneider, D., "RP*: A Family of Order-Preserving Scalable Distributed Data Structures", in Proceedings of the 20th International Conference on Very Large Data Bases (VLDB), 2004.

[46]Rowstron, A., Druschel, P., "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems", IFIP/ACM Middleware 2001, Heidelberg, Germany, Nov, 2001.

[47]Stoica, I., Morris, R., Liben-Nowell, D., Karger, D., Kaashoek, M., Dabek, F., Balakrishnan, H., "Chord: A scalable peer-to-peer lookup service for

internet applications", in Proceedings of ACM SIGCOMM 2001, San Diego, CA, USA, 2001.

[48]Zhao, B., Kubiatowicz, J., Joseph, A., "Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing", Report No. UCB/CSD-01-1141, University of California, Berkeley, California, USA, 2001.

[49]Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Weimer, W., Wells, C., Zhao, B., "Oceanstore: An Architecture for global-scale persistent storage", in Proceedings of the Ninth International Conference on Architecture Support for Programming Languages and Operating Systems (ASPLOS 2000), November: 190-201, 2000.

[50]Karger, D. , Ruhl, M., "Simple Efficient Load Balancing Algorithms for Peer-to-Peer Systems", The 3rd International Workshop on Peer-to-Peer Systems, San Diego, CA, USA, February 26-27, 2004.

[51]Lei, S., Grama, A., "Extended Consistent Hashing: A Framework for Distributed Servers", in Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS 2004), March 23-26, Tokyo, Japan, 2004.

[52]Devine, R., "Design and Implementation of DDH: A Distributed Dynamic Hashing Algorithm", in Proceedings of 4th International Conference on Foundations of Data Organizations and Algorithms, 1993.

[53]Litwin, W., Niemat, M.-A., Schneider, D., "LH* - Linear Hashing for Distributed Files", in Proceedings of ACM International Conference on Management of Data (SIGMOD), May, 1993.

[54]Honicky, R., Miller, E., "A fast algorithm for online placement and reorganization of replicated data", in Proceedings of the 17th International Parallel and Distributed Processing Symposium, Nice, France, April, 2003.

[55]Litwin, W., "Re: Chord & LH*", Email to Ion Stoica, 23rd March, 2004.

[56]Christin, N., Chuang, J., "On the cost of participating in a peer-to-peer network", Technical Report, University of California, Berkeley, http://p2pecon.berkeley.edu/pub/TR-2003-12-cc.pdf, 2003.

[57]Christin, N., Chuang, J., "On the Cost of Participating in a Peer-to-Peer Network", The 3rd International Workshop on Peer-to-Peer Systems, San Diego, CA, USA, February 26-27, 2004.

[58]Rhea, S., Roscoe, T., Kubiatowicz, J., "Structured Peer-to-Peer Overlays Need Application-Driven Benchmarks", in Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03), February, 2003.

[59]Gummadi, K., Gummadi, R., Gribble, S., Ratnasamy, S., Shenker, S., Stoica, I., "The impact of DHT routing geometry on resilience and

proximity", in Proceedings of the 2003 conference on Applications, Technologies, Architectures and Protocols for Computer Communications: 381-394, 2003.

[60] Loguinov, D., Kumar, A., Ganesh, S., "Graph-theoretic analysis of structured peer-to-peer systems: routing distances and fault resilience", in Proceedings of the 2003 conference on Applications, Technologies, Architectures and Protocols for Computer Communications, Karlsruhe, Germany, August 25-29: 395-406, 2003.

[61] Hsiao, H., King, C.-T., "A tree model for structure peer-to-peer protocols", in Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid: 336-343, 2003.

[62] Li, J., Stribling, J., Gil, T., Morris, R., Kaashoek, F., "Comparing the performance of distributed hash tables under churn", The 3rd International Workshop on Peer-to-Peer Systems, San Diego, CA, USA, February 26-27, 2004.

[63] Gil, T., Kaashoek, F., Li, J., Morris, R., Stribling, J., "p2psim, a simulator for peer-to-peer protocols", http://www.pdos.lcs.mit.edu/p2psim/, 2003.

[64] Hildrum, K., Kubiatowicz, J. D. , Rao, S., Zhao, B. Y., "Distributed Object Location in a Dynamic Network, Theory of Computing Systems, November, 2002.

[65] Maymounkov, P., Mazieres, D., "Kademlia: A peer-to-peer information system based on the XOR metric", in Proceedings of the First International Workshop on Peer to Peer Systems (IPTPS 2002), March, kademia.scs.cs.nyu.edu, 2002.

[66] Li, X., Plaxton, C., "On name resolution in peer to peer networks", in Proceedings of the second ACM international workshop on principles of mobile computing: 82-89, 2002.

[67] Lynch, N., Malkhi, D., Ratajczak, D., "Atomic Data Access in Distributed Hash Tables", in Proceedings of the International Peer-to-Peer Symposium, March, 2002.

[68] Doval, D., O'Mahony, D., "Overlay Networks: A Scalable Alternative for P2P," IEEE INTERNET COMPUTING , July-August, 2003.

[69] Mahajan, R., Castro, M., Rowstron, A., "Controlling the cost of reliability in peer-to-peer overlays", Second International Workshop on Peer-to-Peer Systems (IPTPS 03), Berkeley, CA, USA, 20-21 February, 2003.

[70] Rhea, S., Geels, D., Roscoe, T., Kubiatowicz, J., "Handling Churn in a DHT", Report No. UCB/CSD-03-1299, University of California, Berkeley, California, USA, also Proceedings of the USENIX Annual Technical Conference, June, 2004.

[71] Castro, M., Costa, M., Rowstron, A., "Performance and dependability of structured peer-to-peer overlays", Microsoft Research Technical Report MSR-TR-2003-94, December, 2003. Also 2004 International Conference on Dependable Systems and Networks, Palazzo dei Congressi, Florence, Italy, June 28-July, 2004.

[72] van Renesse, R., Bozdog, A., "Willow: DHT, Aggregation and Publish/Subscribe in One Protocol", The 3rd International Workshop on Peer-to-Peer Systems, San Diego, CA, USA, February 26-27, 2004.

[73] Liben-Nowell, D., Balakrishnan, H., Karger, D., "Analysis of the evolution of peer-to-peer systems", Annual ACM Sumposium on Principles of Distributed Computing, Monterey, California, United States, ACM Press, 2002.

[74] Alima, L., El-Ansary, S., Brand, P., Haridi, S., "DKS(N,k,f): a family of low communication, scalable and fault-tolerant infrastructures for P2P applications", in Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid: 344-350, 2003.

[75] National Institute of Standards and Technology, "FIPS Pub 180-1: Secure Hash Standard (SHA-1)," Federal Information Processing Standards Publication, April ,1995.

[76] Ratsanamy, S., Francis, P., Handley, M., Karp, R., "A scalable content-addressable network," in ACM SIGCOMM Conference, ACM Press, San Diego (CA), August, 2001.

[77] McCanne, S., Floyd, S., "The LBNL/UCB Network Simulator", Lawrence Berkeley Laboratory, University of California, Berkeley.

[78] Ratnasamy, S., "A Scalable Content-Addressable Network", Doctoral Dissertation, University of California at Berkeley, Berkeley, California, USA, 2002.

[79] Ratnasamy, S., Shenker, S., Stoica, I., "Routing Algorithms for DHTs: Some Open Questions", in Proceedings of the First International Workshop on Peer to Peer Systems (IPTPS 2002), March, Cambridge, MA, 2002.

[80] Malkhi, D., Naor, M., Ratajczak, D., "Viceroy: a scalable and dynamic emulation of the butterfly", in Proceedings of the 21st annual symposium on principles of distributed computing: 183-192, 2002.

[81] Kaashoek, F., Karger, D., "Koorde: A Simple Degreeoptimal Hash Table", Second International Workshop on Peer-to-Peer Systems (IPTPS 03), Berkeley, CA, USA, 20-21, February, 2003.

[82] Li, X., Plaxton, C., "On name resolution in peer to peer networks", in Proceedings of the second ACM international workshop on principles of mobile computing: 82-89, 2002.

[83]Abraham, I., Malkhi, D., Dubzinski, O., "LAND:Stretch (1+epsilon) Locality Aware Networks for DHTs", in Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA04), New Orleans, LA, 2004.

[84]Naor, M., Wieder, U., "Novel architectures for P2P applications: the continuous-discrete approach", in Proceedings of the fifteenth annual ACM Symposium on Parallel Algorithms and Architectures (SPAA 2003), San Diego, California, USA, June 7-9: 50-59, 2003.

[85]de Bruijn, N. D., "A combinatorial problem", Koninklijke Netherlands: Academe Van Wetenschappen 49: 758-764, 1946.

[86]Mao, J.-W., "The Coloring and Routing Problems on de Bruijn Interconnection Networks", Doctoral Dissertation, National Sun Yat-sen University, July 18, 2003.

[87]Schlumberger, M. L., "De Bruijn communication networks", Doctoral Dissertation, Department of Computer Science, Stanford University, Palo Alto, CA, 1974.

[88]Reddy, S. M., Pradhan, D. K. , Kuhl, J. G.,"Direct graphs with minimal and maximal connectivity", Technical Report, School of Engineering, Oakland University, July, 1980.

[89]Imase, M., Itoh, M., "Design to minimized diameter on build-block network", IEEE Transactions on Computers C-30(6): 439-442, June, 1981.

[90]Rowley, R. A., Bose, B., "Fault-tolerant ring embedding in de Bruijn networks", IEEE Transactions on Computers 42(12): 1480-1486, December, 1993.

[91]Lee, K. Y., Liu, G., Jordan, H. F., "Hierarchical networks for optical communications", Journal of Parallel and Distributed Computing 60: 1-16, 2000.

[92]Naor, M., Wieder, U., "Know thy Neighbor's Neighbor: Better Routing for Skip-Graphs and Small Worlds", The 3rd International Workshop on Peer-to-Peer Systems, San Diego, CA, USA, February 26-27, 2004.

[93]Fraigniaud, P., Gauron, P., "The content-addressable networks D2B", Technical Report 1349, Laboratoire de Recherche en Informatique, Universit´e de Paris Sud, January, 2003.

[94]Datta, A., Girdzijauskas, S., Aberer, K., "On de Bruijn routing in distributed hash tables: there and back again", in Proceedings of the Fourth IEEE International Conference on Peer-to-Peer Computing, Zurich, Switzerland, 25-27, August, 2004.

[95]Naor, M., Wieder, U., "A Simple Fault Tolerant Distributed Hash Table", Second International Workshop on Peer-to-Peer Systems (IPTPS 03), Berkeley, CA, USA, 20-21, February, 2003.

[96] Aspnes, J., Shah, G., "Skip graphs", in Proceedings of the 14th annual ACM-SIAM symposium on discrete algorithms: 384-393. January, 2003.

[97] Harvey, N., Jones, M. B., Saroiu, S., Theimer, M., Wolman, A., "SkipNet: A Scalable Overlay Network with Practical Locality Properties", in Proceedings of the Fourth USENIX Symposium on Internet Technologies and Systems (USITS '03), Seattle, WA, March, 2003.

[98] Pugh, W., "Skip lists: a probabilistic alternative to balanced trees", in Proceedings of the Workshop on Algorithms and Data Structures, pages 437-449, 1989.

[99] Pugh, W., "Skip lists: a probabilistic alternative to balanced trees", Communications of the ACM 33(6): 668-676, June, 1990.

[100] Harvey, N., Jones, M. B., Theimer, M., Wolman, A., "Efficient Recovery from Organization Disconnects in SkipNet", Second International Workshop on Peer-to-Peer Systems (IPTPS03), Berkeley, CA, USA, 20-21, February, 2003.

[101] Aspnes, J., Kirsch, J., Krishnamurthy, A., "Load Balancing and Locality in Range-Queriable Data Structures", in Proceedings of the 23rd Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC 2004), St. John's, Newfoundland, Canada, July 25-28, 2004.

[102] Saroiu, S., Gummadi, P.K., Gribble, S.D., "A measurement study of peer-to-peer le sharing systems," Technical Report UW-CSE-01-06-02, University of Washington, July, 2001.

4 ΓΕΝΙΚΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΥΣΤΗΜΑΤΟΣ

Μέσα από βιβλιογραφική έρευνα (σε επιστημονικά περιοδικά και μέσω του Διαδικτύου), διαπιστώσαμε ότι τα ομότιμα συστήματα για ανταλλαγή περιεχομένου παραμένουν σε ερευνητικό επίπεδο, καθώς καμία πρόταση δεν έχει ακόμα τυποποιηθεί από κάποιο έγκυρο σώμα. Αντίθετα, υπάρχουν ακόμη πολλά ζητήματα που πρέπει να επιλυθούν, όπως αναφέραμε στο προηγούμενο κεφάλαιο. Η επίλυση αυτών των προβλημάτων μέσω βελτιώσεων σε ήδη υπάρχοντες αλγόριθμους θεωρείται δύσκολη ενώ μια νέα προσέγγιση είναι πιθανό να προσφέρει μια πιο ολοκληρωμένη λύση. Με βάση αυτήν τη διαπίστωση προχωρήσαμε στην πρόταση μιας νέας αρχιτεκτονικής για ομότιμα συστήματα, που θα αντιμετωπίζει σε ικανοποιητικό επίπεδο τα περισσότερα από τα θέματα που αναφέραμε.

4.1. Στόχοι του συστήματος και επιλογή της αρχιτεκτονικής

Στόχος του συστήματος που θέλουμε να αναπτύξουμε είναι η παροχή μιας υποδομής για τη δημιουργία εφαρμογών ανταλλαγής περιεχομένου, είτε αυτό το περιεχόμενο είναι στατικό είτε δυναμικό (π.χ. ζωντανή ροή). Το σύστημα αυτό θα πρέπει να παρέχει τη δυνατότητα στους χρήστες να συνδέονται ή να αποσυνδέονται δίχως περιορισμούς και να μπορεί να υποστηρίζει μεγάλο αριθμό χρηστών και μεγάλο όγκο περιεχομένου. Για το λόγο αυτό επιλέχτηκε να σχεδιαστεί με βάση τα ομότιμα συστήματα, παρέχοντας παράλληλα χαμηλό κόστος λειτουργίας και περισσότερη ελευθερία στους χρήστες.

Θέλουμε επίσης να έχει ο χρήστης τη δυνατότητα να αναζητά γρήγορα το περιεχόμενο που επιθυμεί και να μπορεί και ο ίδιος να εκδίδει περιεχόμενο με τον ίδιο απλό τρόπο, ανεξάρτητα από τη φύση του περιεχομένου. Επίσης δε θέλουμε το σύστημά μας να απαιτεί αλλαγές στην υπάρχουσα δικτυακή δομή, στον τεχνολογικό εξοπλισμό ή νέες ρυθμίσεις από πλευράς χρήστη και δικτύου. Για το λόγο αυτό επιλέξαμε το σύστημά μας να βασίζεται στην δημιουργία ενός υπερκείμενου δικτύου, που θα λειτουργεί σε επίπεδο εφαρμογής, πάνω στο οποίο θα αναπτύξουμε ένα DHT αλγόριθμο για την

έκδοση, αναζήτηση και δρομολόγηση του περιεχομένου αλλά και των ίδιων των χρηστών.

4.2. Η αρχιτεκτονική «Umbrella»

Η αρχιτεκτονική μας, την οποία έχουμε ονομάσει «Umbrella», βασίζεται στην δημιουργία ενός υπερκείμενου δικτύου, όπου όλοι οι κόμβοι εισάγονται σε αυτό και τοποθετούνται στην κατάλληλη θέση με βάση ένα μοναδικό κωδικό που έχουν. Ο κωδικός αυτός δίνεται με βάση τη συνάρτηση κατατεμαχισμού SHA-1 [1] στο IP του κόμβου (ή συνδυασμό IP και ονόματος υπολογιστή), που επιστρέφει έναν αριθμό με 160 bits. Η βασική αρχή στην οποία βασίζεται η αρχιτεκτονική αυτή (και τη διαφοροποιεί από αρκετές άλλες) είναι ότι η τοποθέτηση των κόμβων μέσα στο δίκτυο γίνεται με πλήρη οργάνωση, με αποτέλεσμα τόσο η έκδοση όσο και η μετέπειτα αναζήτηση περιεχομένου να είναι απλή και σύντομη. Επιπλέον, όπως θα γίνει στην συνέχεια αντιληπτό, ο κάθε κόμβος χρειάζεται να γνωρίζει μόνο ένα πολύ περιορισμένο και σταθερό αριθμό γειτονικών κόμβων (ανεξάρτητο από τον αριθμό των κόμβων που απαρτίζουν το δίκτυο) και να διατηρεί ενήμερες πληροφορίες μόνο για αυτούς.

Ο κάθε κόμβος εισάγεται στο δίκτυο μέσω της σύνδεσής του σε έναν κόμβο που βρίσκεται ήδη σε αυτό (δεν απαιτείται δηλαδή η ύπαρξη ειδικών κόμβων εισόδου). Στην συνέχεια διαδίδεται η είσοδος του νέου κόμβου εωσότου βρεθεί η θέση στην οποία πρέπει να τοποθετηθεί και ενημερωθούν όλοι οι γειτονικοί κόμβοι. Εν συνεχεία (και καθώς ο νέος κόμβος βρίσκεται πλέον στο δίκτυο) δημοσιεύει τα περιεχόμενα που έχει στην διάθεσή του.

Η διαδικασία δημοσίευσης είναι παρόμοια με αυτής της εισαγωγής. Κάθε περιεχόμενο χαρακτηρίζεται από έναν αριθμό λέξεων-κλειδιών (keywords) και το ίδιο το όνομα του περιεχομένου. Σε όλες τις λέξεις-κλειδιά καθώς και στο όνομα δίνεται ένας κωδικός με βάση τη συνάρτηση κατατεμαχισμού και στην συνέχεια αυτά διαδίδονται μέσα στο δίκτυο προσπαθώντας να βρουν τον κόμβο που έχει τον πλησιέστερο κωδικό ως προς το δικό τους. Μόλις εντοπιστεί αυτός ο κόμβος τότε δημοσιεύεται σε αυτόν το κλειδί μαζί με πληροφορίες για τον κόμβο που το περιέχει (καθώς σε άλλον κόμβο υπάρχει η δημοσίευση του περιεχομένου και σε άλλον το περιεχόμενο καθαυτό).

Εν συνεχεία, όποιος κόμβος αναζητά κάποιο περιεχόμενο, απλά χρησιμοποιεί τη συνάρτηση κατατεμαχισμού για να παράγει τον κατάλληλο κωδικό είτε του ονόματος του περιεχομένου προς αναζήτηση είτε μιας από τις λέξεις κλειδιά που το χαρακτηρίζουν. Κατόπιν στέλνει μήνυμα αναζήτησης

για τον κόμβο με το πλησιέστερο κωδικό ως προς τον κωδικό αναζήτησης. Τα τρία αυτά βασικά βήματα αντικατοπτρίζουν και τα τρία βασικά είδη μηνυμάτων που έχουν αναπτυχθεί για την παρούσα αρχιτεκτονική :

- Μήνυμα εισόδου

  Αποστέλλεται όποτε ένας νέος κόμβος εισέρχεται στο δίκτυο και προωθείται κατάλληλα ώστε να τοποθετηθεί αυτός στην σωστή θέση.
- Μήνυμα δημοσίευσης

  Αποστέλλεται όταν κάποιος κόμβος (που έχει ήδη τοποθετηθεί στο δίκτυο) θέλει να δημοσιεύσει ένα νέο περιεχόμενο και προωθείται προς τον κόμβο με το πλησιέστερο ως προς αυτό κωδικό.
- Μήνυμα αναζήτησης

  Αποστέλλεται όταν γίνεται αναζήτηση ενός περιεχομένου και προωθείται προς τον κόμβο με το πλησιέστερο κωδικό ως προς τον κωδικό του περιεχομένου που αναζητείται.

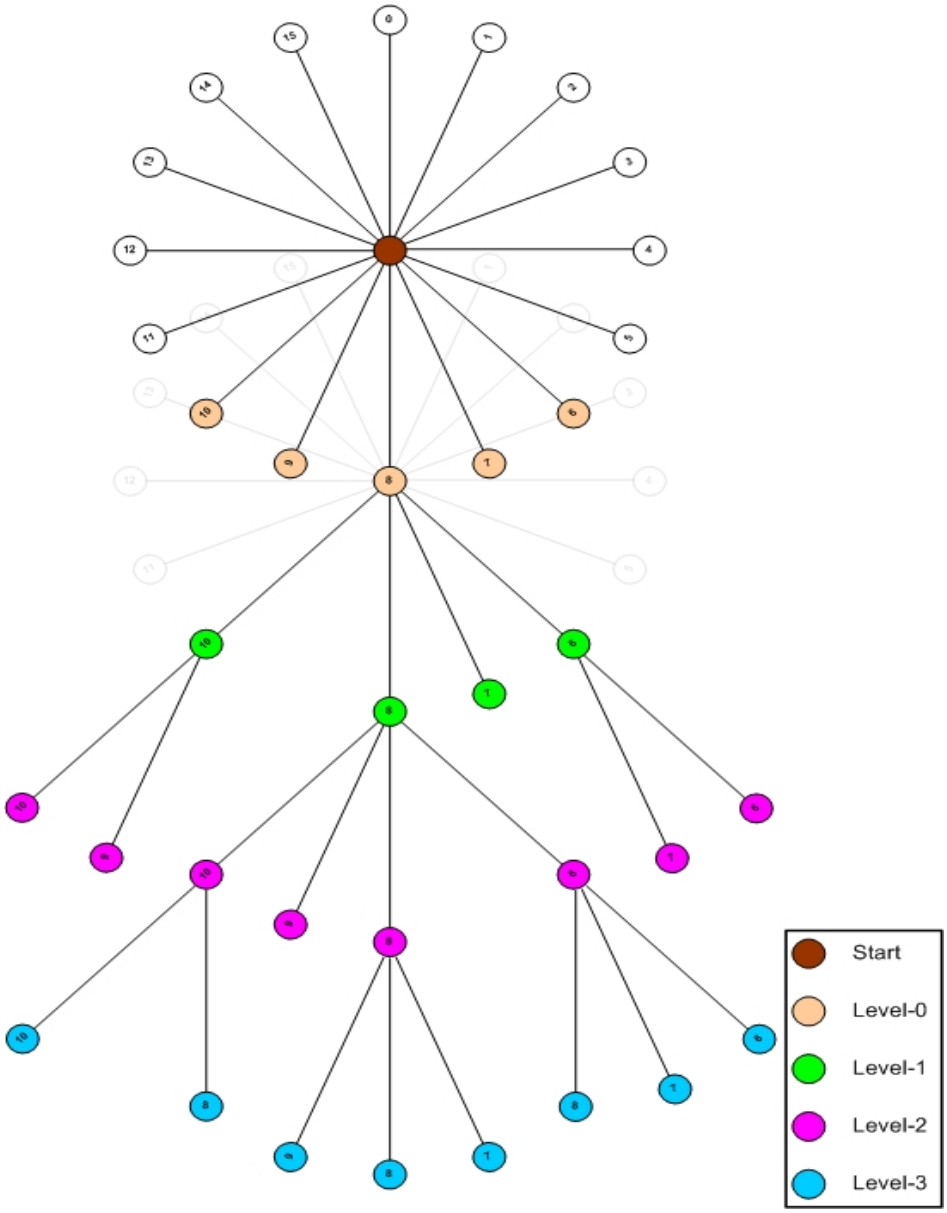
Όπως ήδη αναφέραμε, η αρχιτεκτονική αυτή βασίζεται στην καλή οργάνωση του δικτύου, ώστε να μην απαιτείται ο κάθε κόμβος να έχει πολλές γνώσεις για το δίκτυο αλλά και να μη χρειάζεται να πλημμυρίζουμε το δίκτυο με μηνύματα ,όπως στην περίπτωση του Gnutella [2] .

Η οργάνωση του δικτύου γίνεται με βάση τον κωδικό που παράγεται από τη συνάρτηση κατατεμαχισμού. Κάθε κόμβος διατηρεί πληροφορίες για τα εξής :

- Το επίπεδο στο οποίο βρίσκεται
- Τους κόμβους που βρίσκονται δεξιά και αριστερά του
- Τον κόμβο που βρίσκεται στο επάνω επίπεδο
- Τους κόμβους που βρίσκονται στο κάτω επίπεδο
- Ένα σύνολο από κόμβους που χρησιμοποιούνται ως εναλλακτικές συνδέσεις σε περίπτωση που κάποιος από τους ανωτέρω κόμβους εγκαταλείψει απότομα (χωρίς να στείλει δηλαδή μήνυμα εξόδου) το δίκτυο

Πιο συγκεκριμένα, κάθε κόμβος βρίσκεται σε ένα επίπεδο ν, στο οποίο βρίσκονται $16^{\nu+1}$ άλλοι κόμβοι. Κάθε κόμβος έχει έναν πατέρα, που βρίσκεται ένα επίπεδο πιο πάνω, και 16 παιδιά (ή λιγότερα, καθώς κάποια μπορεί να είναι κενά), που βρίσκονται ένα επίπεδο κάτω από αυτόν. Η σχέση που συνδέει έναν κόμβο πατέρα επιπέδου ν με έναν κόμβο παιδί (που βρίσκεται αναγκαστικά στο επίπεδο $\nu+1$) είναι :

- Οι *ν+1* πρώτοι δεκαεξαδικοί αριθμοί (4*ν+4* bits) του κωδικού[1] του πατέρα είναι ίδιοι με τους αντίστοιχους του παιδιού.
- Ο *ν+2* δεκαεξαδικός αριθμός του κωδικού του παιδιού καθορίζει τη θέση του παιδιού στη λίστα των *16* παιδιών που έχει ο πατέρας. Όλα τα παιδιά του πατέρα πρέπει δηλαδή να έχουν τους *ν+1* πρώτους δεκαεξαδικούς αριθμούς του αναγνωριστικού τους κοινούς και τον *ν+2* αριθμό διαφορετικό από όλα τα άλλα παιδιά.



Σχήμα 23 : Η αρχιτεκτονική Umbrella

[1] Η SHA-1 συνάρτηση κατακερματισμού που χρησιμοποιούμε παράγει έναν αριθμό μήκους 160 bits που στην συνέχεια της εργασίας θα ονομάζουμε αναγνωριστικό του κόμβου και θα τον αναπαριστούμε είτε σε bits είτε ως συμβολοσειρά 40 δεκαεξαδικών αριθμών.

Αν ορίσουμε ως *f* τη συνάρτηση που επιστρέφει το ψηφίο στην θέση *n* ενός αναγνωριστικού, τότε για να ανήκουν δύο κόμβοι *A* και *B* στον ίδιο πατέρα *Π*, θα πρέπει να ισχύει η παρακάτω σχέση :

$$f_A(n) = f_B(n), \quad 1 \le n \le \nu + 1$$

$$f_A(n) \ne f_B(n), \quad n = \nu + 2$$

Και από τον ορισμό που έχουμε δώσει για τη σχέση που συνδέει έναν πατέρα με ένα παιδί, θα πρέπει επίσης να ισχύει:

$$f_\Pi(n) = f_A(n) = f_B(n), \quad 1 \le n \le \nu + 1$$

Με βάση αυτόν τον απλό κανόνα (που δεν ισχύει μόνο για τον πρώτο κόμβο που εισέρχεται και στην ουσία δημιουργεί το δίκτυο και ο οποίος δεν έχει αριθμό επιπέδου) δημιουργείται όλο το δίκτυο Umbrella. Μια εικόνα του δικτύου Umbrella απεικονίζεται στο Σχήμα 23. Στην συνέχεια θα περιγράψουμε πιο αναλυτικά τον τρόπο που δημιουργείται το δίκτυο Umbrella, τη συνάρτηση κατατεμαχισμού και σύγκρισης, τον πίνακα γειτόνων που πρέπει να διατηρεί κάθε κόμβος και τις τρεις βασικές λειτουργίες του συστήματος, δηλαδή την εισαγωγή στο δίκτυο, τη δημοσίευση του περιεχομένου και την αναζήτηση.

4.2.1. Συνάρτηση κατατεμαχισμού και σύγκρισης

Όπως ήδη αναφέραμε, τα αναγνωριστικά των κόμβων αλλά και των κλειδιών δίνονται μέσω της συνάρτησης κατατεμαχισμού SHA-1. Η συνάρτηση αυτή μας δίνει τη δυνατότητα να έχουμε μια ομοιόμορφη κατανομή των περιεχομένων και επομένως των κλειδιών στους κόμβους, ώστε να μην παρουσιάζονται κόμβοι με αυξημένη κίνηση. Επιπλέον, προσφέρει ανωνυμία στο σύστημα, μέσω εξαναγκαστικής τοποθέτησης των δημοσιεύσεων του περιεχομένου. Στον Πίνακας 5 φαίνονται μερικά παραδείγματα αναγνωριστικών κόμβων και περιεχομένου με τα αντίστοιχα αναγνωριστικά τους από τη συνάρτηση SHA-1. Όπως φαίνεται και από τον πίνακα, ακόμα και οι μικρότερες αλλαγές στο πεδίο οδηγεί σε τελείως διαφορετικό κλειδί.

Για την αναζήτηση του κατάλληλου κόμβου και γενικότερα τη δρομολόγηση στο σύστημα χρησιμοποιούμε μια συνάρτηση σύγκρισης *comp*, η οποία συγκρίνει δυο αναγνωριστικά και επιστρέφει τη διαφορά τους. Η συνάρτηση αυτή έχει ως είσοδο τα δύο αναγνωριστικά (id_1 και id_2) και επιστρέφει μια δεκαδική τιμή ως εξής :

$$comp(id_1, id_2) = \sum_{i=1}^{i=160} g(i)$$

$$g(i) = \begin{cases} 2^{160-i}, & \alpha\nu \quad f_{id_1}(i) = f_{id_2}(i) \\ 0, & \alpha\nu \quad f_{id_1}(i) \neq f_{id_2}(i) \end{cases}$$

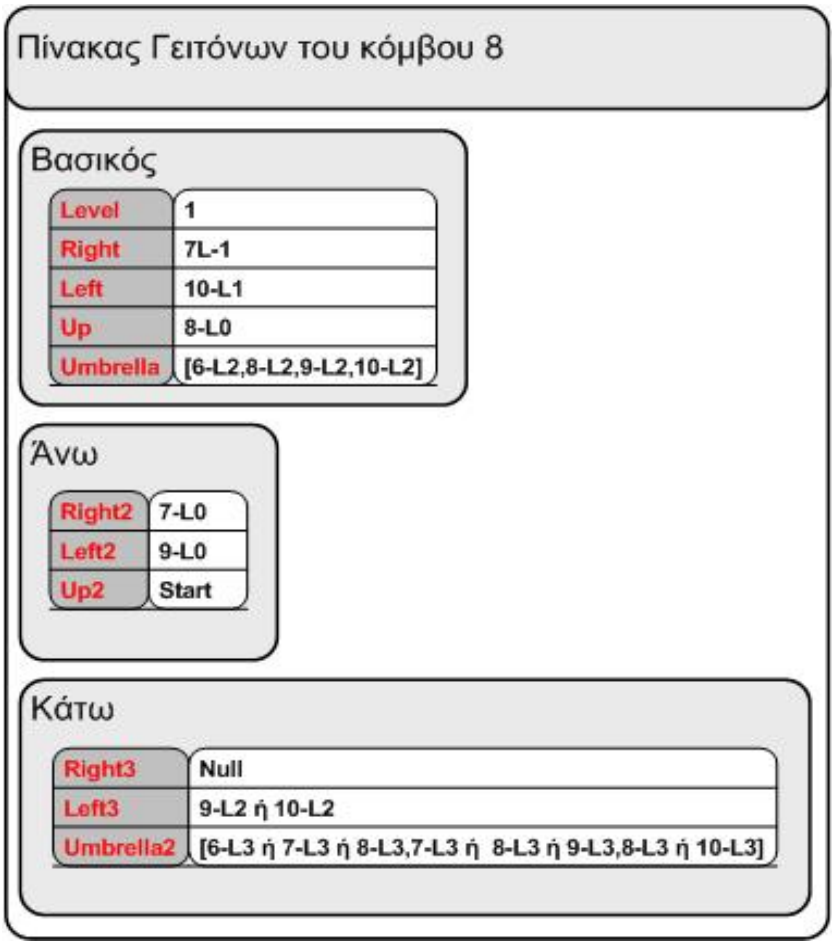
Πεδίο (κόμβος ή περιεχόμενο)	Αναγνωριστικό (κόμβου ή κλειδιού)
computer1@147.102.7.123	288fb2ea28273abf2825220fbd034a2b47cf7f6f
computer2@147.102.7.123	F70060ad01b5c437e0b4b9feea6dfbc6b476ae81
homecom@114.45.43.3	eddbe5ddcc4f1db6c095fead79fa4c0cf6585bac
One Hundred Years of Solitude	4657006cf87107c2dd979890b6227b0c62ba2f61
Sigur Ros	1fb14d169bda3f04819b2b7ad3f27ba64b69bfcf

Πίνακας 5. Πεδία και αντίστοιχα αναγνωριστικά μέσω της συνάρτησης SHA-1

4.2.2. Πίνακας γειτόνων

Όπως αναφέραμε ήδη, η σωστή οργάνωση του δικτύου επιτρέπει τη λειτουργία του με απλούς κανόνες και διαδικασίες. Για να γίνει αυτό εφικτό ο κάθε κόμβος διατηρεί έναν πίνακα γειτόνων, και είναι υπεύθυνος ώστε να διατηρείται όσο γίνεται πιο ενήμερος. Στον πίνακα αυτό διατηρούνται τρία διαφορετικά σετ πληροφοριών, το βασικό, το άνω και το κάτω. Οι βασικές πληροφορίες περιλαμβάνουν τη γνώση των κόμβων μέσω των οποίων δρομολογεί τα μηνύματα ένας κόμβος, υπό κανονικές συνθήκες. Για να μπορέσει όμως το δίκτυο να είναι ανεκτικό σε εξόδους κόμβων (απότομες ή μη) ο κάθε κόμβος περιέχει πληροφορίες για επιπλέον γειτονικούς κόμβους, που διατηρούνται στο άνω σετ όσον αφορά κόμβους που βρίσκονται πιο ψηλά στην δομή του συστήματος, και στο κάτω σετ για κόμβους που βρίσκονται σε μεγαλύτερο επίπεδο. Έτσι κάθε κόμβος διατηρεί έναν πίνακα γειτόνων, όπως αυτός φαίνεται στο Σχήμα 24 και επεξηγείτε στον Πίνακας 6, με βάση το Σχήμα 23 και πιο συγκεκριμένα τον κόμβο 8 στο επίπεδο-1[2].

[2] Στην συνέχεια της εργασίας αυτής οι αναφορές στους κόμβους μέσα στην διάταξη του συστήματος θα γίνονται με την εξής μορφή <αριθμός-Lαριθμός>, όπου ο πρώτος αριθμός διατυπώνει τη θέση του κόμβου ανάμεσα στα παιδιά και ο δεύτερος αριθμός (με το γράμμα L να προηγείται) το επίπεδο στο οποίο βρίσκεται. Έτσι, ο κόμβος 9-L2 αναφέρεται στο παιδί στην θέση 9, που βρίσκεται στο επίπεδο 2. Ο αριθμός αυτός δεν είναι μοναδικός για όλο το σύστημα αλλά είναι μοναδικός για ένα συγκεκριμένο πατέρα (καθώς υπάρχουν πολλά διαφορετικά παιδιά στην ίδια θέση του ίδιου επιπέδου αλλά με διαφορετικό πατέρα). Για να



Σχήμα 24. Τυπικός Πίνακας Γειτόνων για τον κόμβο 8

Τα παραπάνω στοιχεία επαρκούν για να κάνουν το δίκτυό μας ανεκτικό σε απότομες εξόδους κόμβων. Πιο συγκεκριμένα, η γνώση των κόμβων Right2,Left2 και Up2 (άνω σετ) μας επιτρέπει να ανεβαίνουμε επίπεδο ακόμα και αν κάποιοι κόμβου εγκαταλείψουν, ενώ η γνώση των κόμβων Right3,Left3 και Umbrella2 (κάτω σετ) μας επιτρέπει να κατεβαίνουμε επίπεδο σε περίπτωση απότομης διακοπής συνδέσεων. Οι δε κόμβοι Right,Left,Up,Umbrella και η πληροφορία του Level (βασικό σετ) αποτελούν τον κορμό του δικτύου μας. Έτσι, σε περίπτωση που κάποιος κόμβος έχει εγκαταλείψει το δίκτυο και δεν είναι δυνατή η επικοινωνία ώστε να εφαρμοσθούν οι αλγόριθμοι εισαγωγής, δημοσίευσης ή αναζήτησης, τότε αυτοί προσαρμόζονται και σε περίπτωση που δεν είναι δυνατή η επικοινωνία προς το ανώτερο επίπεδο (όπου αυτό χρειάζεται στον αντίστοιχο αλγόριθμο) το μήνυμα προωθείται αντί για τον πατέρα στους κόμβους Right2,Left2 και Up2, ενώ όταν δεν είναι δυνατή η επικοινωνία προς τα παιδιά (κατώτερο

περιγράψουμε πλήρως τη θέση του με την παραπάνω μορφή θα πρέπει να ορίσουμε την ακολουθία με την οποία οδηγούμαστε στον κόμβο, με αφετηρία τον πρώτο κόμβο του συστήματος. Για παράδειγμα, με βάση το Σχήμα 23, ο κόμβος 9-L2 θα περιγραφτεί με μοναδικό τρόπο ως <8-L0/8-L1/9-L2>.

επίπεδο), τότε τα μηνύματα προωθούνται προς τους κόμβους Right3,Left3 και Umbrella2.

Πεδίο	Τιμή	Επεξήγηση
Level	1	Το επίπεδο στο οποίο βρίσκεται.
Right	7-L1	Τον κόμβο που βρίσκεται στα δεξιά του και δεν είναι κενός
Left	10-L1	Τον κόμβο που βρίσκεται στα αριστερά του και δεν είναι κενός.
Up	8-L0	Τον κόμβο πατέρα
Right2	7-L0	Τον κόμβο που βρίσκεται δεξιά από τον κόμβο πατέρα.
Left2	9-L0	Τον κόμβο που βρίσκεται αριστερά από τον κόμβο πατέρα.
Up2	Start	Τον κόμβο πατέρα του πατέρα.
Right3	Null	Ένα από τα παιδιά (τυχαία επιλογή) του κόμβου που βρίσκεται δεξιά του.
Left3	9-L2 ή 10-L2	Ένα από τα παιδιά (τυχαία επιλογή) του κόμβου που βρίσκεται αριστερά του.
Umbrella	[6-L2,8-L2,9-L2,10-L2]	Όλοι οι κόμβοι που είναι παιδιά του.
Umbrella2	[6-L3 ή 7-L3 ή 8-L3, 7-L3 ή 8-L3 ή 9-L3, 8-L3 ή 10-L3]	Ένα κόμβος παιδί (τυχαία επιλογή) από κάθε παιδί του.

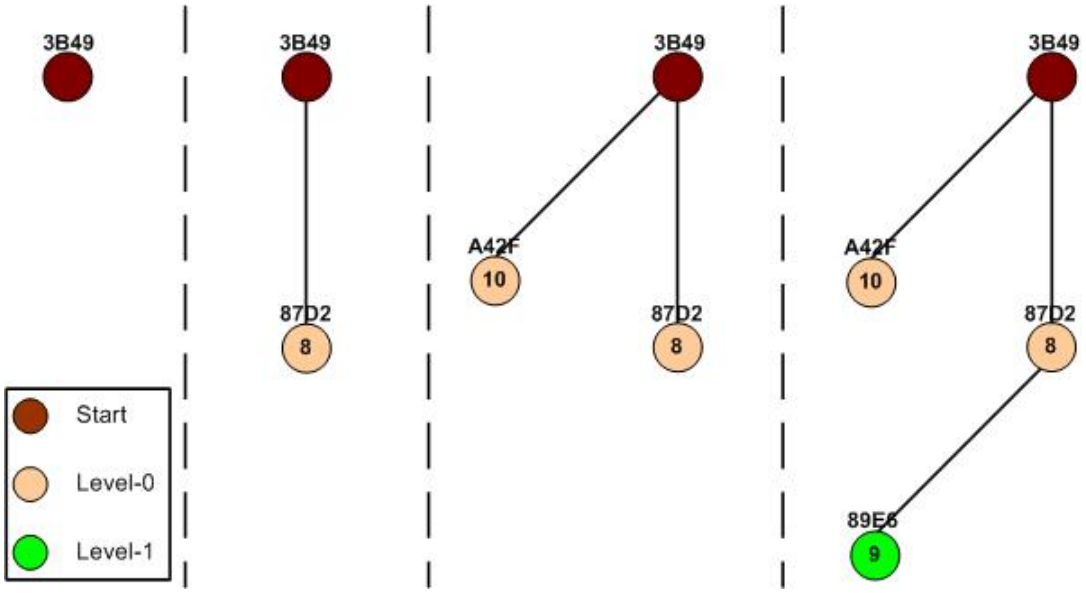
Πίνακας 6. Επεξήγηση των πεδίων του πίνακα γειτόνων για τον κόμβο 8

4.2.3. Δημιουργία του συστήματος

Όταν το σύστημα δημιουργείται ο πρώτος κόμβος που εισέρχεται (ο οποίος είναι και αυτός που πρακτικά το δημιουργεί), τοποθετείται στην κορυφή του συστήματος. Ο κόμβος αυτός δεν πρόκειται ποτέ να έχει πατέρα, θα έχει μόνο παιδιά και δεν έχει επίσης και επίπεδο (μπορούμε να το αναφέρουμε ως -1). Καθώς νέοι κόμβοι συνδέονται στο σύστημα, αυτοί τοποθετούνται με βάση το αναγνωριστικό τους όπως θα περιγράψουμε στην συνέχεια. Ο τρόπος με τον οποίο βρίσκουν οι νέοι κόμβοι έναν κόμβο που βρίσκεται ήδη στο σύστημα δεν περιγράφεται καθώς έχουν ήδη προταθεί

αρκετές τέτοιες τεχνικές που μπορούν να ενσωματωθούν στο σύστημά μας, όπως η δημιουργία ενός δακτυλίου εκκίνησης [3] ή η χρήση ενός κατανεμημένου συστήματος εκπομπής [4] .

Πρέπει να τονίσουμε ότι όλοι οι κόμβοι στο σύστημα είναι τοποθετημένοι σε καλά οργανωμένη δομή εκτός από τον πρώτο κόμβο, ο οποίος είναι τυχαία τοποθετημένος στην κορυφή. Αυτό γίνεται ώστε να μη χρειάζεται να αναδιοργανώνουμε το σύστημά μας κάθε φορά που εισέρχεται νέος κόμβος (να γίνεται αναδιάταξη) καθώς αυτό θα απαιτούσε πολλά μηνύματα χωρίς να υπάρχει κάποια αντίστοιχη βελτίωση στην απόδοση του συστήματος. Στο Σχήμα 25 φαίνεται μια τυπική εκκίνηση του συστήματος κατά τη δημιουργία του. Στο σχήμα αυτό τα αναγνωριστικά έχουν μέγεθος 16 bits (για συντομία), που αντιστοιχούν σε 4 δεκαεξαδικούς αριθμούς. Ο κόμβος 3B49 εισέρχεται πρώτος και δημιουργεί το σύστημα. Στην συνέχεια εισέρχεται ο κόμβος 87D2, οπότε τοποθετείται στην θέση 8 στο επίπεδο 0. Εν συνεχεία εισέρχονται διαδοχικά οι κόμβοι A42F και 89E6. Ο πρώτος τοποθετείται ως παιδί στο επίπεδο 0 και στην θέση 10 (λόγω του πρώτου ψηφίου που είναι A) ενώ ο δεύτερος στην θέση 9 στο επίπεδο 1, ως παιδί του 87D2, καθώς έχει το πρώτο ψηφίο ίδιο με αυτό του 8-L0 κόμβου και το δεύτερο ψηφίο είναι 9, οπότε τοποθετείται στην θέση 9. Η θέση του κάθε κόμβου είναι ανεξάρτητη από τον κόμβο τον οποίο χρησιμοποίησε για να συνδεθεί στο σύστημα, με μόνη εξαίρεση τον πρώτο κόμβο, που δημιουργεί το σύστημα και αυτόματα τοποθετείται στην κορυφή.



Σχήμα 25. Τυπική εκκίνηση του δικτύου και τοποθέτηση κόμβων

4.2.4. Διαδικασία εισαγωγής

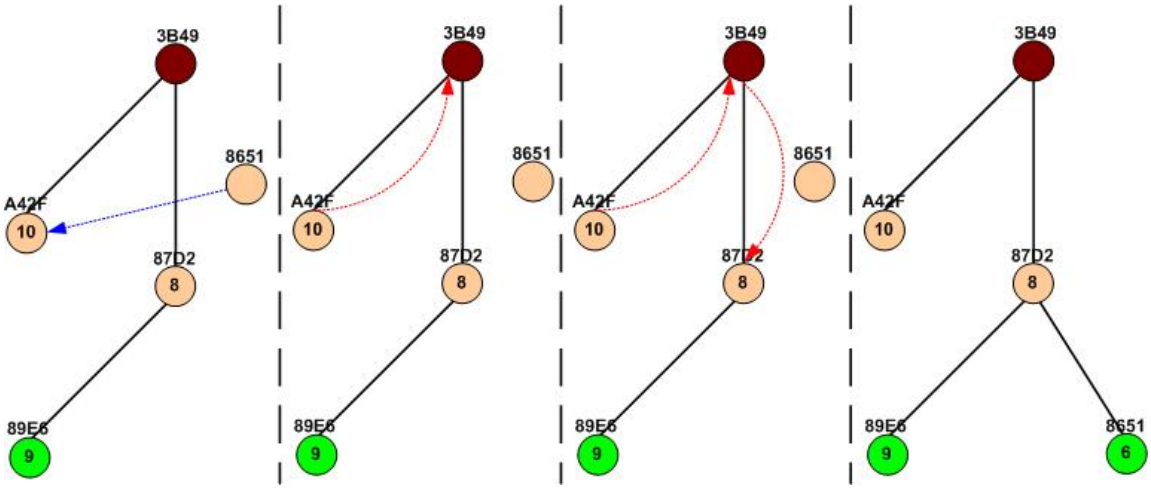
Η διαδικασία εισαγωγής ενός νέου κόμβου είναι αρκετά απλή και ανάγεται στα ακόλουθα βήματα :

- Ο νέος κόμβος συνδέεται σε έναν υπάρχοντα κόμβο και στέλνει μήνυμα για εισαγωγή στο δίκτυο
- Ο κόμβος ελέγχει εάν οι *ν+1* αριθμοί του κωδικού του νέου κόμβου συμπίπτουν με τα δικά του(*ν* το επίπεδο του κόμβου).
- Εάν όχι τότε προωθεί το μήνυμα στον κόμβο πατέρα του.
- Εάν ναι τότε το προωθεί στον κόμβο παιδί που έχει τον *ν+2* αριθμό ίδιο με αυτόν του νέου κόμβου.
- Εάν δεν υπάρχει τέτοιο παιδί (κενή θέση) τότε ο νέος κόμβος γίνεται παιδί του κόμβου.
- Ενημερώνεται ο νέος κόμβος για τους γείτονές του και στέλνει πληροφορίες σε αυτούς για τον εαυτό του.

```
(1)   insert ( new_node , new_node_id )
(2)       if ( check_same_up_to( new_node_id , this_node.id , this_node.lv+1 ) )
(3)           num = new_node_id.get_number( this_node.lv+2 )
(4)           if ( kid_exists( num ) )
(5)               get_kid_at( num ).insert( new_node , new_node_id )
(6)           else
(7)               set_kid_at( new_node , num )
(8)               new_node.inform_neighbours( this_node )
(9)       else
(10)          this_node.father.insert( new_node , new_node_id )
```

Σχήμα 26. Ψευδοκώδικας για εισαγωγή κόμβου στο σύστημα

Στο Σχήμα 26 δίνεται ο αλγόριθμος εισαγωγής ενός νέου κόμβου σε μορφή ψευδοκώδικα. Στο Σχήμα 27 φαίνεται ένα παράδειγμα εισαγωγής ενός νέου κόμβου στο σύστημα και πως αυτός τοποθετείται στην κατάλληλη θέση. Ο κόμβος 8651 συνδέεται στο υπερκείμενο δίκτυο μέσω του κόμβου A42F και στέλνει μήνυμα για εισαγωγή στο δίκτυο. Ο κόμβος A42F ελέγχει αν το πρώτο ψηφίο είναι ίδιο με αυτόν, όπως αυτό ορίζεται στη γραμμή (2). Το επίπεδο του A42F είναι 0, οπότε ελέγχει τα 0+1=1 ψηφία, που είναι διαφορετικά, οπότε καταλήγουμε στην γραμμή (10) και προωθούμε το μήνυμα στον πατέρα. Ο πατέρας λαμβάνει το μήνυμα και ελέγχει αν τα πρώτα -1+1=0 ψηφία είναι ίδια. Προφανώς είναι οπότε εκτελείται η γραμμή (3), η οποία επιστρέφει το -1+2=1 ψηφίο, το οποίο είναι 8. Εκτελείτε εν συνεχεία η γραμμή (4), η οποία επιστρέφει θετική και το μήνυμα προωθείται στο αντίστοιχο παιδί, δηλαδή στον κόμβο 87D2. Η εκτέλεση πάλι του αλγορίθμου επιστρέφει στην γραμμή (3), όπου ο αριθμός στην θέση 0+2=2 είναι 6. Στην γραμμή (4) η απάντηση είναι αρνητική, οπότε εκτελούνται οι γραμμές (7)-(8), που τοποθετούν το νέο κόμβο ως παιδί του 87D2 στην θέση 6.



Σχήμα 27. Εισαγωγή του κόμβου 8651 στο σύστημα

Μετά το τέλος της τοποθέτησης του νέου κόμβου, ο πατέρας του προωθεί όλα τα κλειδιά που έχουν δημοσιευθεί σε αυτόν και για τα οποία ο νέος κόμβος είναι καταλληλότερος. Η συνάρτηση ελέγχου των αναγνωριστικών και επιλογής του καταλληλότερου βασίζεται στην συνάρτηση σύγκρισης, καθώς επιλέγεται ο κόμβος που επιφέρει υψηλότερη βαθμολογία όταν συγκριθεί με το κλειδί.

4.2.5. Διαδικασία δημοσίευσης

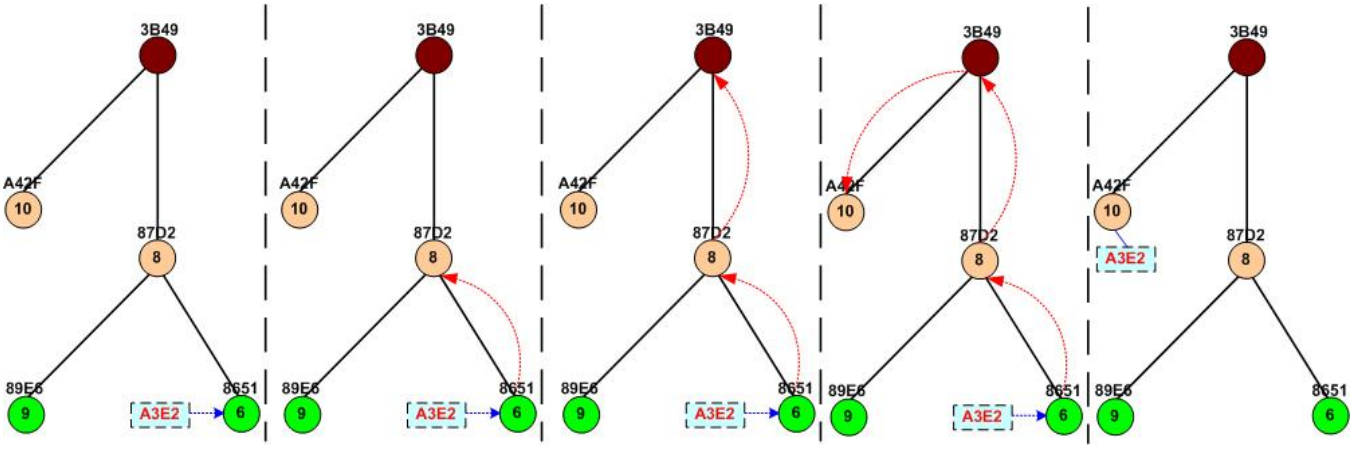
Αντίστοιχα απλή είναι και η διαδικασία δημοσίευσης περιεχομένου στο δίκτυο Umbrella και ανάγεται στα ακόλουθα βήματα :

- Εάν ο κωδικός του περιεχομένου δεν έχει τους v+1 αριθμούς του κωδικού του ίδιους με αυτούς του κόμβου τότε προωθεί το μήνυμα δημοσίευσης στον πατέρα του.
- Εάν τους έχει τότε το προωθεί στον κόμβο παιδί που έχει τον v+2 αριθμό ίδιο με αυτόν του περιεχομένου.
- Εάν δεν υπάρχει τέτοιο παιδί τότε το περιεχόμενο δημοσιεύεται σε αυτόν.

```
(1)  publish ( content , content_id )
(2)      if ( check_same_up_to( content_id , this_node.id , this_node.lv+1 ) )
(3)          num = content_id.get_number( this_node.lv+2 )
(4)          if ( kid_exists( num ) )
(5)              get_kid_at( num ).publish( content , content_id )
(6)          else
(7)              publish_here( content )
(8)      else
(9)          this_node.father.publish( content , content_id )
```

Σχήμα 28. Ψευδοκώδικας για δημοσίευση περιεχομένου

Ο αλγόριθμος δημοσίευσης σε μορφή ψευδοκώδικα δίνεται στο Σχήμα 28 ενώ στο Σχήμα 29 φαίνεται διαγραμματικά μια τυπική δημοσίευση. Ο κόμβος 8651 θέλει να δημοσιεύσει το περιεχόμενο A3E2, οπότε με βάση τη γραμμή (2) του αλγορίθμου δημοσίευσης, ελέγχει τα πρώτα 1+1=2 ψηφία αν είναι ίδια. Καθώς δεν είναι, εκτελείται η γραμμή (9) και το μήνυμα προωθείται στον πατέρα (κόμβο 87D2). Αντίστοιχα ο κόμβος 87D2 προωθεί το μήνυμα στον 3B49. Στον κόμβο αυτό η γραμμή (2) επιστρέφει θετική απάντηση οπότε εκτελείται η γραμμή (3) που επιστρέφει το πρώτο ψηφίο του περιεχομένου, που είναι A. Η γραμμή (4) επιστρέφει επίσης θετική και το μήνυμα προωθείται στο παιδί στην θέση 10, δηλαδή τον κόμβο A42F. Εκτελείται πάλι ο αλγόριθμος και καταλήγει στην γραμμή (4), όπου αυτήν τη φορά επιστρέφει αρνητική απάντηση και ο κώδικας συνεχίζεται στην γραμμή (7), όπου το περιεχόμενο δημοσιεύεται στον κόμβο A42F.



Σχήμα 29. Δημοσίευση του περιεχομένου με κλειδί AE62

4.2.6. Διαδικασία αναζήτησης

Η διαδικασία για την αναζήτηση ενός περιεχομένου γίνεται με τον εξής απλό τρόπο:

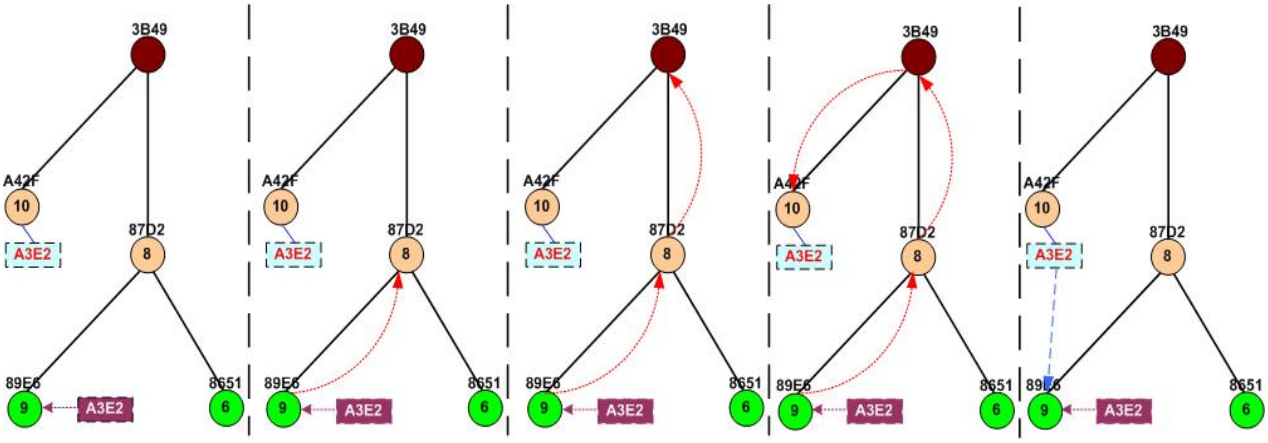
- Ελέγχεται εάν το περιεχόμενο που αναζητείται βρίσκεται δημοσιευμένο στον κόμβο αυτό.
- Εάν ναι, τότε σταματά η αναζήτηση.
- Εάν όχι τότε ελέγχεται εάν ο κωδικός του περιεχομένου που αναζητείται έχει τους ν+1 αριθμούς του κωδικού του ίδιους με αυτούς του κόμβου.
- Σε αρνητική περίπτωση προωθεί το μήνυμα αναζήτησης στον πατέρα του.
- Εάν τους έχει, τότε το προωθεί στον κόμβο παιδί που έχει τον ν+2 αριθμό ίδιο με αυτόν του περιεχομένου.

- Εάν δεν υπάρχει τέτοιο παιδί τότε το περιεχόμενο προφανώς δεν υπάρχει στο δίκτυο.

```
(1)   search ( search_start , content_id )
(2)       if ( content_exists_here( content_id ) )
(3)           search_start.reply_positive( this_node.id , content_id )
(4)       else
(5)           if ( check_same_up_to( content_id , this_node.id , this_node.lv+1 ) )
(6)               num = content_id.get_number( this_node.lv+2 )
(7)               if ( kid_exists( num ) )
(8)                   get_kid_at( num ).search( search_start , content_id )
(9)               else
(10)                  search_start.reply_negative( content_id )
(11)          else
(12)              this_node.father.search( search_start , content_id )
```

Σχήμα 30. Ψευδοκώδικας για αναζήτηση περιεχομένου

Ο αντίστοιχος ψευδοκώδικας του παραπάνω αλγορίθμου φαίνεται στο Σχήμα 30. Στο Σχήμα 31 φαίνεται πως γίνεται η αναζήτηση για το περιεχόμενο με κλειδί A3E2 (το οποίο είναι αποθηκευμένο στον κόμβο A42F) ,η οποία ξεκινάει από τον κόμβο 89E6. Ο κώδικας στην γραμμή (2) επιστρέφει αρνητική και συνεχίζει στην (5), η οποία είναι επίσης αρνητική και το μήνυμα προωθείται στον πατέρα (87D2) λόγω της γραμμής (12). Ο ίδιος αλγόριθμος προωθεί πάλι το μήνυμα στον πατέρα (3B49). Εδώ, η εκτέλεση της γραμμής (5) επιστρέφει θετική και εκτελείται η (6) που επιστρέφει το πρώτο ψηφίο του κλειδιού, που είναι A. Η γραμμή (7) επιστρέφει θετική και η (8) προωθεί το μήνυμα στο παιδί στην θέση 10, δηλαδή τον κόμβο A42F. Εδώ η εκτέλεση της γραμμής (2) είναι θετική, οπότε η (3) επιστρέφει θετικό μήνυμα αναζήτησης στον κόμβο 89E6. Αν ο κόμβος A42F δεν είχε το ζητούμενο περιεχόμενο τότε με τη διαδοχική εκτέλεση των γραμμών του αλγορίθμου θα καταλήγαμε στην (10), η οποία θα έστελνε αρνητικό μήνυμα αναζήτησης στον κόμβο 89E6, καθώς δεν υπάρχουν άλλοι κατάλληλοι κόμβοι όπου θα μπορούσε να είχε δημοσιευθεί το περιεχόμενο.



Σχήμα 31. Αναζήτηση για το περιεχόμενο A3E2 από τον κόμβο 89E6

4.2.7. Διαδικασία εξόδου

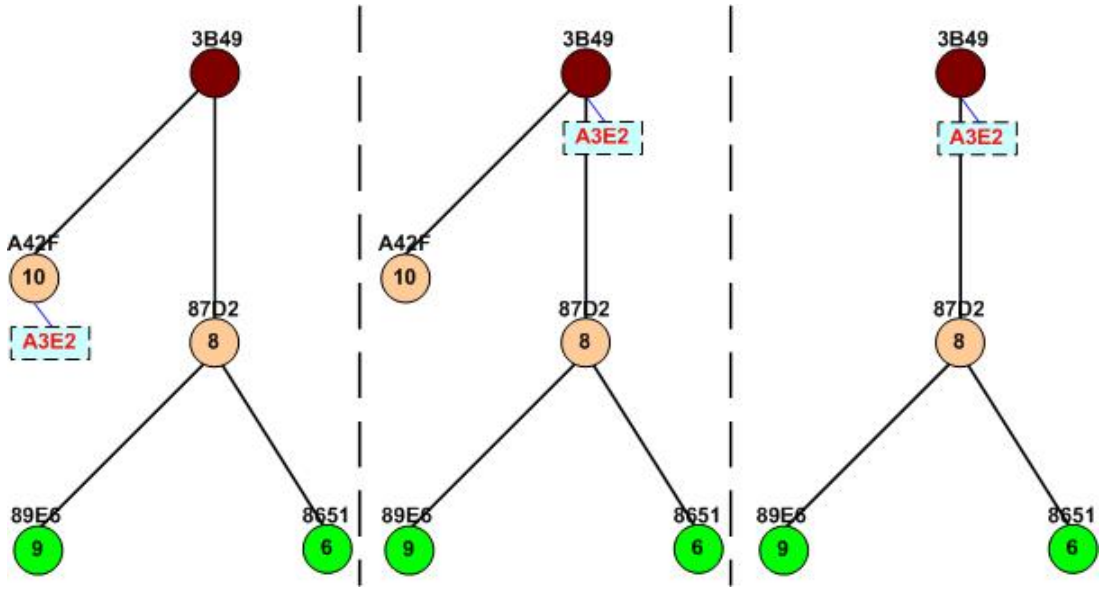
Τέλος, σε περίπτωση που ένας κόμβος θέλει να εγκαταλείψει το δίκτυο (όχι απότομα), αποστέλλει ένα μήνυμα εξόδου και στην συνέχεια εκτελείται ο παρακάτω αλγόριθμος, που διοργανώνει κατάλληλα το σύστημα ώστε να ενημερωθούν όλοι οι γειτονικοί κόμβοι :

- Εάν ο κόμβος δεν έχει παιδιά τότε μετάφερε όλες τις δημοσιεύσεις περιεχομένων στον πατέρα και ενημέρωσε όλους του γειτονικούς κόμβους ώστε να σε διαγράψουν.
- Εάν έχει παιδιά τότε διάλεξε ένα παιδί (τυχαία) και αντίγραψε όλες τις πληροφορίες που είχες ώστε ο κόμβος παιδί να είναι ίδιος με σένα προτού φύγεις.
- Εάν το παιδί έχει παιδιά τότε επανέλαβε το ίδιο βήμα με ένα τυχαίο παιδί του (αναδρομικά) μέχρι να φτάσεις σε κόμβο που δεν έχει παιδιά, οπότε εκτελείται το πρώτο βήμα.

```
(1)   delete ( )
(2)       if ( has_kids( ) )
(3)           rand_kid = choose_random_kid( )
(4)           if ( rand_kid.has_kids( ) )
(5)               rand_kid.delete( )
(6)           else
(7)               rand_kid.move_published( )
(8)               rand_kid.copy_neighbors( )
(9)               inform_neighbors( rand_kid )
(10)              disconnect( )
(11)      else
(12)          this_node.father.move_published( )
(13)          inform_neighbors( this_node.father )
(14)          disconnect( )
```

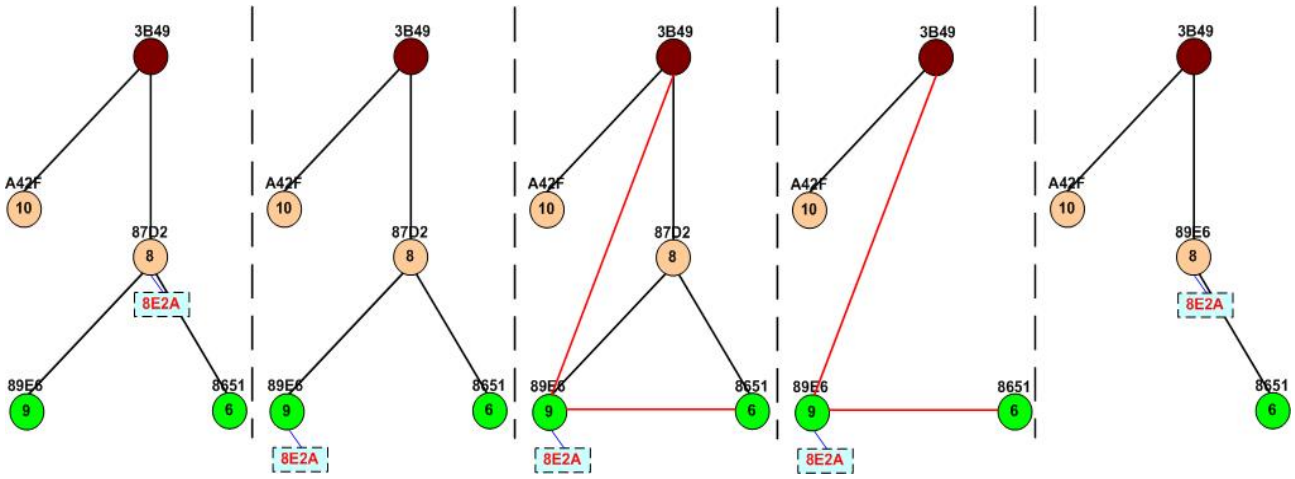
Σχήμα 32. Ψευδοκώδικας για διαγραφή κόμβου από το σύστημα

Στο Σχήμα 32 παρουσιάζεται ο αντίστοιχος ψευδοκώδικας, ενώ στο Σχήμα 33 και στο Σχήμα 34 αναπαρίσταται η αναδιάταξη του συστήματος μετά την έξοδο ενός κόμβου με παιδιά και ενός χωρίς. Στην πρώτη περίπτωση (Σχήμα 33) η εκτέλεση του κώδικα αρχικά δίνει αρνητική απάντηση στην γραμμή (2), οπότε εκτελούνται οι γραμμές (12)-(14) και τερματίζει η διαδικασία. Οι γραμμές αυτές αρχικά μεταφέρουν όλα τα δημοσιευμένα περιεχόμενα του κόμβου A42F (δηλαδή το κλειδί A3E2) στον κόμβο πατέρα (3B49), στην συνέχεια ενημερώνουν τους γείτονες ότι υπεύθυνος θα είναι ο πατέρας (γραμμή (13)) και τελικά ο κόμβος αποσυνδέεται από το σύστημα.



Σχήμα 33. Διαγραφή του κόμβου A42F, ο οποίος δεν έχει παιδιά

Στην δεύτερη περίπτωση (Σχήμα 34), ο κόμβος 87D2 εκτελεί τον αλγόριθμο εξόδου από το σύστημα και η γραμμή (2) είναι θετική καθώς έχει παιδιά. Στην (3) επιλέγεται τυχαία ο κόμβος 89E6 και καθώς δεν έχει παιδιά η γραμμή (4) οδηγεί στην (7). Εάν ο κόμβος είχε παιδία τότε θα είχαμε εκτέλεση της γραμμής (5), η οποία θα καλούσε αναδρομικά τον αλγόριθμο διαγραφής για τον κόμβο που επιλέχτηκε τυχαία. Η αναδρομή αυτή θα τερμάτιζε όταν θα εντοπιζόταν κόμβος χωρίς παιδιά. Στην συγκεκριμένη περίπτωση η ροή του αλγορίθμου τερματίζει με την εκτέλεση των εντολών (7)-(10). Οι εντολές αυτές αρχικά μεταφέρουν όλα τα κλειδιά που έχουν δημοσιευθεί στο παιδί, στην συνέχεια μεταφέρουν όλες τις συνδέσεις με γειτονικούς κόμβους στο παιδί, ενημερώνει τους γείτονες ότι υπεύθυνος αλλά και γείτονας θα αποτελεί το παιδί και τέλος ο κόμβος αποσυνδέεται.



Σχήμα 34. Διαγραφή του κόμβου 87D2, ο οποίος έχει παιδιά

Στο σημείο αυτό μπορούμε να προτείνουμε μια βελτιωμένη εκδοχή του αλγορίθμου εξόδου, και πιο συγκεκριμένα ως προς τη γραμμή (3), όπου επιλέγεται τυχαία ένα παιδί του κόμβου. Επειδή ο αλγόριθμος λειτουργεί αναδρομικά στην περίπτωση επιλογής κόμβου που έχει παιδιά, θα μπορούσαμε αντί για τυχαία επιλογή να επιλέγαμε πάντα τον κόμβο που δεν έχει παιδιά ή (σε περίπτωση μη εύρεσης τέτοιου κόμβου) του κόμβου που η υπόλοιπη δομή είναι μικρότερη, καθώς αυτό θα οδηγούσε σε συντομότερο τερματισμό της αναδρομικής διαδικασίας.

4.3. Πολυπλοκότητα Αναζήτησης

Η δομή του πρωτοκόλλου μας και του πίνακα γειτόνων που έχουμε περιγράψει εγγυώνται πως ένα δημοσιευμένο κλειδί μπορεί να εντοπιστεί μέσω της κατάλληλης αναζήτησης σε λογαριθμικό αριθμό (υπερκείμενων) βημάτων σε σχέση με το μέγεθος του δικτύου. Αυτό αποδεικνύεται με βάση τα δύο παρακάτω θεωρήματα:

Θεώρημα 1. Δεδομένου ενός δικτύου Umbrella με πληθυσμό N κόμβους και βάση αναγνωριστικών β που έχουν κατανεμηθεί με μια συνεπή συνάρτηση κατατεμαχισμού, το μέγιστο ύψος της δομής Umbrella είναι λογαριθμικού μεγέθους.

Απόδειξη : Έστω β η βάση των αναγνωριστικών, N ο συνολικός αριθμός κόμβων και κ ένα συγκεκριμένο επίπεδο στην δομή Umbrella. Τότε σύμφωνα με το πρωτόκολλο Umbrella, σε κάθε επίπεδο επιτρέπονται το πολύ β^{κ} κόμβοι, με $\beta^0=0$, δεδομένου ότι η δομή μας αρχίζει από το επίπεδο 1. Επομένως, εάν μ είναι ο αριθμός των απαιτούμενων επιπέδων για το συγκεκριμένο πληθυσμό, τότε έχουμε την εξής σχέση:

$$N = \sum_{\kappa=0}^{\mu} \beta^{\kappa} = \frac{\beta^0 - \beta^{\mu+1}}{1-\beta} = \frac{\beta^{\mu+1}}{\beta-1} \Leftrightarrow \mu = \{\log_{\beta}[N(\beta-1)]\} - 1$$

Επομένως το μέγιστο ύψος μ της δομής είναι λογαριθμικού μεγέθους $O(\log_{\beta}N)$. ▪

Θεώρημα 2. Μια πετυχημένη αναζήτηση στο δίκτυο Umbrella απαιτεί λογαριθμικό αριθμό βημάτων $O(\log_{\beta}N)$.

Απόδειξη : Έστω ότι ένας κόμβος ρ, ο οποίος βρίσκεται στο επίπεδο λ_{ρ}, αναζητεί ένα συγκεκριμένο κλειδί κ, το οποίο βρίσκεται στο δίκτυο σε έναν άλλο κόμβο φ, στο επίπεδο λ_{φ}. Εάν μ είναι ο αριθμός των επιπέδων του δικτύου, N ο αριθμός των κόμβων και β η βάση των αναγνωριστικών, τότε μπορούμε να διαπιστώσουμε πως η χειριστη περίπτωση θα απαιτούσε και τους δύο κόμβους να βρίσκονται στο επίπεδο μ με μέγιστη απόσταση μεταξύ τους (δηλαδή ο κόμβος ρ είναι το μ-επιπέδου παιδί του πρώτου παιδιού στο

επίπεδο 1 και ούτω καθεξής, ενώ ο κόμβος φ είναι το μ-επιπέδου παιδί του β παιδιού στο επίπεδο 1 και ούτω καθεξής). Σε αυτήν την περίπτωση, ο μηχανισμός αναζήτησης πρέπει πρώτα να διαβεί προς τα άνω όλη τη δομή (δηλαδή μ βήματα) και εν συνεχεία να κατέλθει μέχρι τη βάση (μ βήματα επίσης). Συνολικά απαιτούνται δηλαδή 2μ βήματα. Επομένως, από το προηγούμενο θεώρημα συνεπάγεται ότι ο αριθμός των βημάτων είναι λογαριθμικός $O(\log_{\beta}N)$. ▪

4.4. Σύνοψη - Συμπεράσματα

Στην ενότητα αυτή έγινε αναλυτική παρουσίαση της πρωτότυπης αρχιτεκτονικής μας. Αρχικά αναφέραμε τους στόχους της σχεδίασής μας και τους λόγους που οδήγησαν στην χρήση DHT αλγορίθμου. Εν συνεχεία παρουσιάσαμε τη δομή της αρχιτεκτονικής, περιγράφοντας αναλυτικά τη γεωμετρία, τον τρόπο χρήσης της συνάρτησης κατατεμαχισμού SHA-1 και τον πίνακα γειτόνων. Κατόπιν παραθέσαμε τον τρόπο δημιουργίας του συστήματος και όλες τις βασικές λειτουργίες (εισαγωγή, δημοσίευση, αναζήτηση και έξοδο) τόσο περιγραφικά όσο και με τη σχηματική απεικόνιση αλλά και την παράθεση ψευδοκώδικα. Τέλος, αναλύσαμε την πολυπλοκότητα της λειτουργίας της αναζήτησης, αποδεικνύοντας ότι το σύστημά μας μπορεί να δρομολογήσει σε λογαριθμικό αριθμό βημάτων.

4.5. Αναφορές

[1] National Institute of Standards and Technology, "FIPS Pub 180-1: Secure Hash Standard (SHA-1)," Federal Information Processing Standards Publication, April 1995.

[2] Ritter, J., "Why Gnutella Can't Scale. No, Really", http://www.darkridge.com/~jpr5/doc/gnutella.html, February, 2001.

[3] Castro, M., Druschel, P., Kermarrec, A.M., Rowsron, A., "One Ring to Rule Them All: Service Discovery and Binding in Structured Peer-to-peer Overlay Networks", Sept., 2002.

[4] Freedman, M.J., Lakshminarayanan, K., Mazieres, D., "Oasis: Anycast for any service", 2006.

5 ΕΞΟΜΟΙΩΣΗ ΚΑΙ ΠΡΩΤΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

Στο κεφάλαιο αυτό θα παρουσιάσουμε αρχικά τον εξομοιωτή που χρησιμοποιήσαμε κατά τη διάρκεια των πειραμάτων μας και εν συνεχεία θα αναλύσουμε τα αποτελέσματα μιας σειράς εξομοιώσεων, που αποδεικνύουν την ορθή λειτουργία της αρχιτεκτονικής ως προς τις βασικές λειτουργίες και κυρίως την επιτυχή και αποδοτική αναζήτηση.

5.1. Εισαγωγή

Για την εξομοίωση της αρχιτεκτονικής χρησιμοποιήθηκε η πλατφόρμα neurogrid [1] , που έχει ειδικά αναπτυχθεί για τη μελέτη ομότιμων συστημάτων. Για την προσαρμογή της στην δική μας αρχιτεκτονική χρειάστηκε να γίνουν πολλές τροποποιήσεις και να αναπτυχθεί ένα νέο μοντέλο, που να εξομοιώνει κάθε κόμβο, τις πληροφορίες που θα διατηρεί για τους γειτονικούς κόμβους και τα μηνύματα που θα δέχεται ή θα αποστέλλει (με βάση πάντα το μοντέλο που μόλις περιγράψαμε). Επειδή το σύστημά μας , όπως και όλα τα αντίστοιχα ομότιμα συστήματα που αναφέρονται στην βιβλιογραφία, βασίζεται πάνω στο υπάρχον IP δίκτυο χωρίς καμία επιπλέον απαίτηση, δεν κρίθηκε αναγκαίο η χρησιμοποίηση εξομοιωτών όπως το OPNET [2] ή NS-2 [3] , που είναι αρκετά αναλυτικοί και δε θα επέτρεπαν την εξομοίωση μεγάλου αριθμού κόμβων. Αντίθετα, στον εξομοιωτή που χρησιμοποιούμε, και ο οποίος είναι σε Java και μας δίνει τη δυνατότητα να εργασθούμε τόσο από Windows όσο και από Linux και έχουμε τη δυνατότητα να εξομοιώσουμε χιλιάδες κόμβους σε αρκετά σύντομο χρονικό διάστημα.

5.2. Ο εξομοιωτής neurogrid

Ο εξομοιωτής neurogrid αρχικά σχεδιάστηκε για να συγκριθούν οι αλγόριθμοι Freenet, Gnutella και NeuroGrid [4] , αλλά βασικό στοιχείο της σχεδίασης αποτέλεσε η δυνατότητα επέκτασης και σε άλλα μοντέλα. Ο εξομοιωτής χρησιμοποιεί abstract classes, οι οποίες είναι γενικές και

υπάρχουν στα περισσότερα ομότιμα συστήματα. Για παράδειγμα, η abstract κλάση Network προσφέρει μεθόδους για τη δημιουργία και προσθήκη κόμβων με συγκεκριμένη τοπολογία. Στην συνέχεια, τα διάφορα ομότιμα συστήματα που προστίθενται στον εξομοιωτή μπορούν να υλοποιήσουν την κλάση αυτήν και να κάνουν αλλαγές ή προσθήκες στις μεθόδους που παρέχουν.

Κάθε σύστημα μπορεί να μεταβάλλει όσες abstract classes απαιτούνται για τη σωστή μοντελοποίηση του κάθε αλγορίθμου. Το neurogrid παρέχει μια σειρά από βασικές abstract ή μη classes, που αποτελούν τον κορμό κάθε ομότιμου συστήματος. Οι classes αυτές είναι :

1. Keyword

   Αναπαριστά τα κλειδιά του συστήματος, τα οποία μπορούν να δημιουργηθούν με οποιοδήποτε τρόπο επιθυμούμε (αν και συνήθως χρησιμοποιείται μια συνάρτηση κατατεμαχισμού), και με βάση τα οποία γίνονται οι αναζητήσεις των περιεχομένων.

2. Document

   Αποτελεί το περιεχόμενο που έχει κάθε κόμβος. Κάθε document μπορεί να περιέχει έναν αριθμό από κλειδιά (keywords) και κάθε κόμβος να περιέχει έναν αριθμό από περιεχόμενα (documents).

3. Message

   Υλοποιεί ένα μήνυμα, ανεξάρτητα από το περιεχόμενό του. Η κλάση αυτή χρησιμοποιείται ώστε όλα τα μηνύματα να δρομολογούνται από τον εξομοιωτή.

4. Node

   Αποτελεί την υλοποίηση του κόμβου. Κάθε κόμβος λαμβάνει και στέλνει μηνύματα, τα επεξεργάζεται, διατηρεί μια λίστα από γνωστούς κόμβους και όποιες άλλες λειτουργίες απαιτεί ο αλγόριθμος από κάθε κόμβο.

5. MessageHandler

   Υλοποιεί τη μέθοδο λήψης μηνυμάτων. Κάθε κόμβος έχει ένα μόνο MessageHandler, από τον οποίο λαμβάνει σειριακά μηνύματα από άλλους κόμβους και επίσης στέλνει μηνύματα σε άλλους κόμβους.

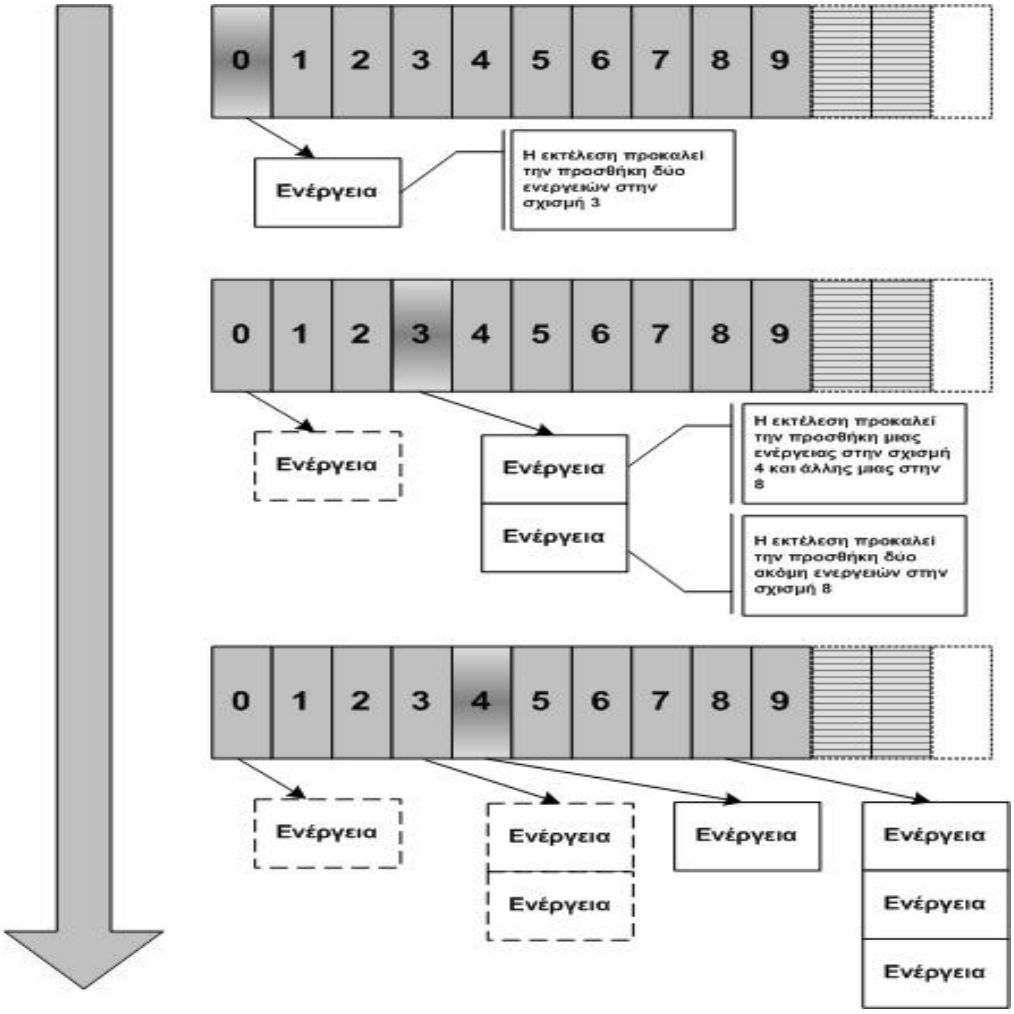
6. Network

Υλοποιεί το υπερκείμενο δίκτυο. Στην κλάση αυτή ορίζεται ο τρόπος δημιουργίας και εισόδου των κόμβων, η τοπολογία του δικτύου και γενικά όλες οι ενέργειες που θέλουμε να εξομοιώσουμε.

Κάθε αντικείμενο που δημιουργείται στο neurogrid έχει ένα μοναδικό αναγνωριστικό, το οποίο δημιουργείται μέσω στατικής ανάθεσης κάθε νέου αντικειμένου της ίδιας κλάσης. Έτσι κάθε κόμβος, περιεχόμενο, κλειδί ή μήνυμα μπορεί να καταγραφεί και να εντοπιστεί μέσω του αναγνωριστικού του. Αυτό μας βοηθάει σημαντικά στην στατιστική ανάλυση της εξομοίωσής μας αλλά και στον έλεγχο ορθής λειτουργίας του συστήματός μας, καθώς έχουμε τη δυνατότητα να εντοπίζουμε όλες τις ενέργειες κάθε αντικειμένου. Χρειάζεται όμως προσοχή, ώστε ο αλγόριθμος που σχεδιάζουμε και εξομοιώνουμε να μην κάνει χρήση αυτών των δυνατοτήτων, καθώς η φύση των ομότιμων συστημάτων δεν επιτρέπει τη γνώση συγκεντρωτικών και καθολικών πληροφοριών, αφού οι λειτουργίες είναι κατανεμημένες στους χρήστες.

Ο εξομοιωτής ξεκινάει από την κλάση Simulation, που αποτελεί την αρχική κλάση, η οποία διαβάζει τις ρυθμίσεις που έχουμε ορίσει για την εξομοίωσή μας, εκτελεί την εξομοίωση, καταγράφει (μέσω άλλων κλάσεων) στατιστικά στοιχεία και τελικά τα παρουσιάζει μέσω της βιβλιοθήκης JFreeChart [5] .

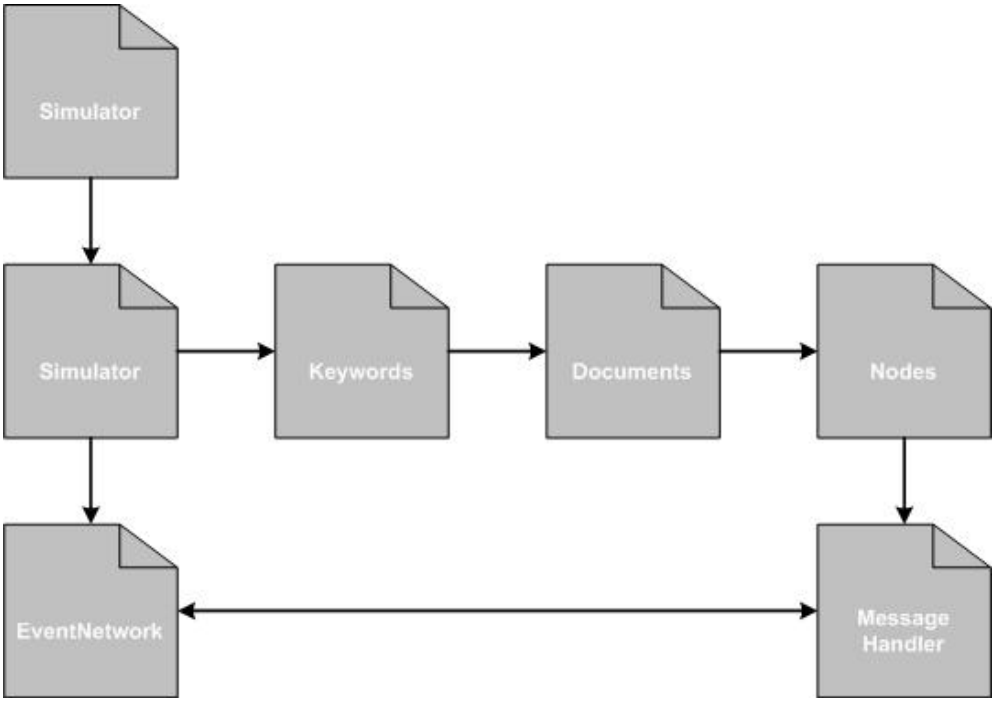
Κατά τη διάρκεια της εξομοίωσης, κόμβοι δημιουργούνται και εισέρχονται στο δίκτυο σύμφωνα με συγκεκριμένη κατανομή και τοπολογία, και μηνύματα ανταλλάσσονται και δρομολογούνται μεταξύ τους. Η ανταλλαγή αυτών των μηνυμάτων γίνεται μέσω της χρήσης της κλάσης EventNetwork, η οποία υλοποιεί και εξομοιώνει την έννοια του χρόνου. Έτσι, σε κάθε μήνυμα που αποστέλλεται από έναν κόμβο δίνεται μια χρονική σφραγίδα και τοποθετείται σε μια χρονική σχισμή του EventNetwork, όπως φαίνεται στο Σχήμα 35. Η κλάση αυτή εκτελεί τις ενέργειες που υπάρχουν σε κάθε χρονική σχισμή (οι οποίες μπορούν να απαιτούν την προσθήκη ενεργειών σε μετέπειτα σχισμές) και όταν όλες έχουν εκτελεί, τότε προχωράει στην επόμενη. Όταν όλες οι ενέργειες στις χρονικές σχισμές έχουν εκτελεστεί, τότε τερματίζει ένας κύκλος εξομοίωσης και η κλάση Simulator συλλέγει στατιστικά στοιχεία.



Σχήμα 35. Παράδειγμα χρονικής ακολουθίας που υλοποιεί η κλάση EventNetwork

5.3. Η εξομοίωση του συστήματος Umbrella

Με βάση τον εξομοιωτή neurogrid μοντελοποιήσαμε την αρχιτεκτονική του συστήματος Umbrella για να επιβεβαιώσουμε την ορθή λειτουργία του και να ελέγξουμε την απόδοσή του υπό διαφορετικές ελεγχόμενες συνθήκες. Για το λόγο αυτό υλοποιήσαμε εκ νέου αρκετές από τις classes του εξομοιωτή neurogrid.



Σχήμα 36. Διάγραμμα λειτουργίας της εξομοίωσης του συστήματος Umbrella

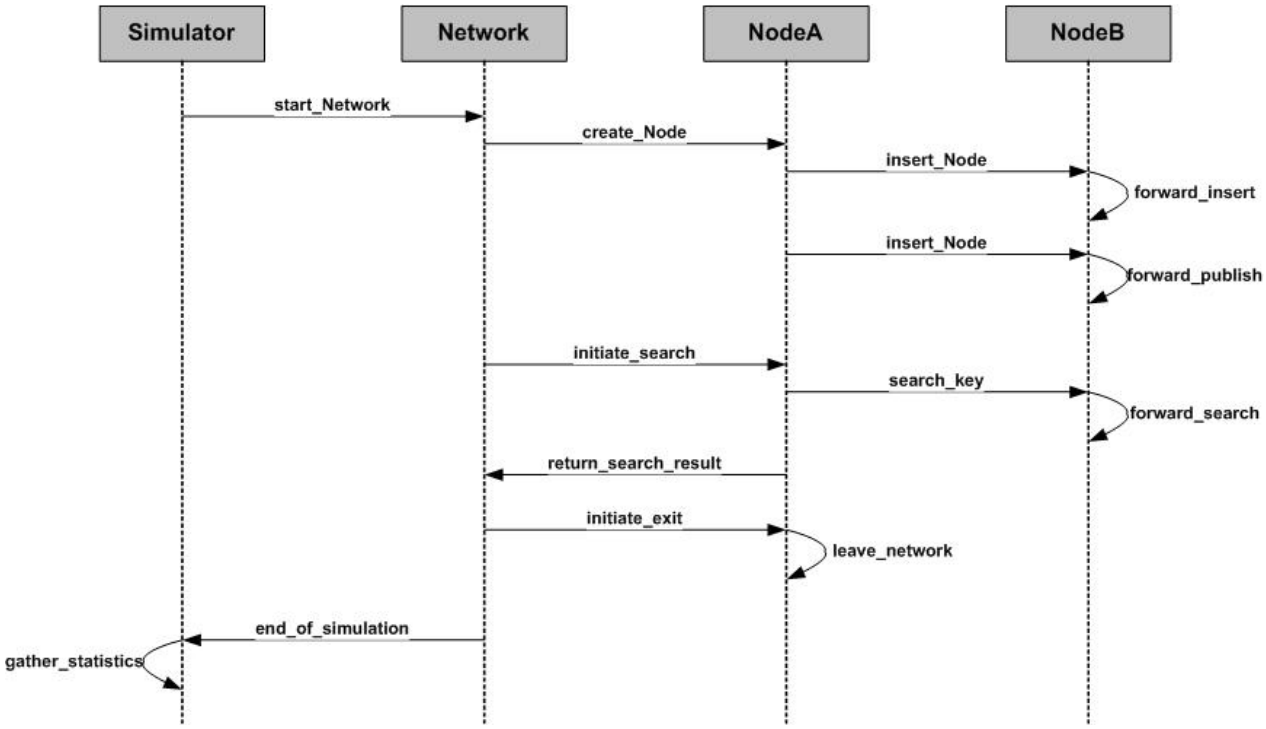
Ο τρόπος με τον οποίο λειτουργεί η εξομοίωση που δημιουργήσαμε φαίνεται στο Σχήμα 36. Αρχικά, η κλάση Simulator διαβάζει τις παραμέτρους της εξομοίωσης που θέλουμε να εκτελέσουμε, που περιλαμβάνει τα στοιχεία που φαίνονται στον Πίνακας 7 (παρατίθενται μόνο τα βασικά). Εν συνεχεία καλείται η κλάση Network, που θα εξομοιώσει το δίκτυο, και η EventNetwork, που θα εξασφαλίσει τη χρονική αλληλουχία των ενεργειών. Μέσα από αυτήν την κλάση αρχικά θα δημιουργηθούν τα κλειδιά και στην συνέχεια θα δημιουργηθούν τα περιεχόμενα. Κατόπιν, θα ανατεθούν κλειδιά στα περιεχόμενα σύμφωνα με τις παραμέτρους της εξομοίωσης. Στην συνέχεια, και με βάση τυχαία κατανομή, θα δημιουργηθούν κόμβοι, στους οποίους αναθέτουμε περιεχόμενα κατά τη φάση δημιουργία τους, και κατόπιν εισέρχονται στο υπερκείμενο δίκτυό μας. Κάθε κόμβος δημιουργεί και έναν MessageHandler για να δέχεται και να στέλνει μηνύματα. Ο κάθε κόμβος έχει επίσης μια κλάση UmbrellaNext, όπου διατηρούνται πληροφορίες για γειτονικούς κόμβους, υλοποιώντας τον πίνακα γειτόνων που έχουμε σχεδιάσει.

Παράμετρος	Επεξήγηση
NO_KEYWORDS	Ο αριθμός των κλειδιών που θα υπάρχουν
NO_DOCUMENTS	Ο αριθμός των περιεχομένων που θα δημιουργηθούν
NO_KEYWORDS_PER_DOCUMENT	Ο αριθμός των κλειδιών που θα περιέχει κάθε περιεχόμενο
NO_DOCUMENTS_PER_NODE	Ο αριθμός των περιεχομένων που θα έχει κάθε κόμβος
NO_NODES	Ο αριθμός των κόμβων
NO_NODES_PUBLISHED	Ο αριθμός των κόμβων που θα δημοσιεύσουν τα περιεχόμενά τους
NO_NODES_INS_PUB	Ο αριθμός των κόμβων που θα εισέλθουν μετά την δημιουργία του αρχικού δικτύου και οι οποίοι θα δημοσιεύσουν το περιεχόμενό τους
NO_NODES_LEAVE_INT	Ο αριθμός των κόμβων που θα εγκαταλείψουν ορθά το δίκτυο
NO_NODES_LEAVE_UNINT	Ο αριθμός των κόμβων που θα εγκαταλείψουν απότομα το δίκτυο (εξομοίωση αποτυχίας)
START_TTL	Ο αριθμός των βημάτων προτού ένα μήνυμα σταματήσει να προωθείται
INTERNAL_LOOP	Ο αριθμός των εξομοιώσεων που θα επαναληφτούν με αυτές τις ρυθμίσεις (για την εξαγωγή έγκυρων αποτελεσμάτων)

Πίνακας 7. Παράμετροι της εξομοίωσης

Καθώς δημιουργείται το δίκτυο, κάθε νέος κόμβος επιλέγει τυχαία έναν υπάρχοντα κόμβο, στέλνει αρχικά ένα μήνυμα εισαγωγής σε αυτόν και κατόπιν δημοσιεύει (μέσω μηνυμάτων δημοσίευσης) όλα τα περιεχόμενά του. Κατά τη διάρκεια της εξομοίωσης γίνεται ένας αριθμός αναζητήσεων, από έναν τυχαίο κόμβο του δικτύου με ένα τυχαίο κλειδί (που ήδη υπάρχει δημοσιευμένο στο δίκτυο) και το αποτέλεσμα της αναζήτησης καθώς και ο αριθμός των βημάτων που απαιτήθηκαν καταγράφονται. Επιπλέον, αν αυτό έχει οριστεί στις παραμέτρους, κόμβοι εξέρχονται από το δίκτυο (ομαλά ή μέσω αποτυχίας). Στο τέλος της εξομοίωσης συλλέγονται στατιστικά στοιχεία για την ανάλυση της απόδοσης. Σε όλες τις παραπάνω ενέργειες, έχουν υλοποιηθεί οι αλγόριθμοι που περιγράψαμε στην αρχιτεκτονική του

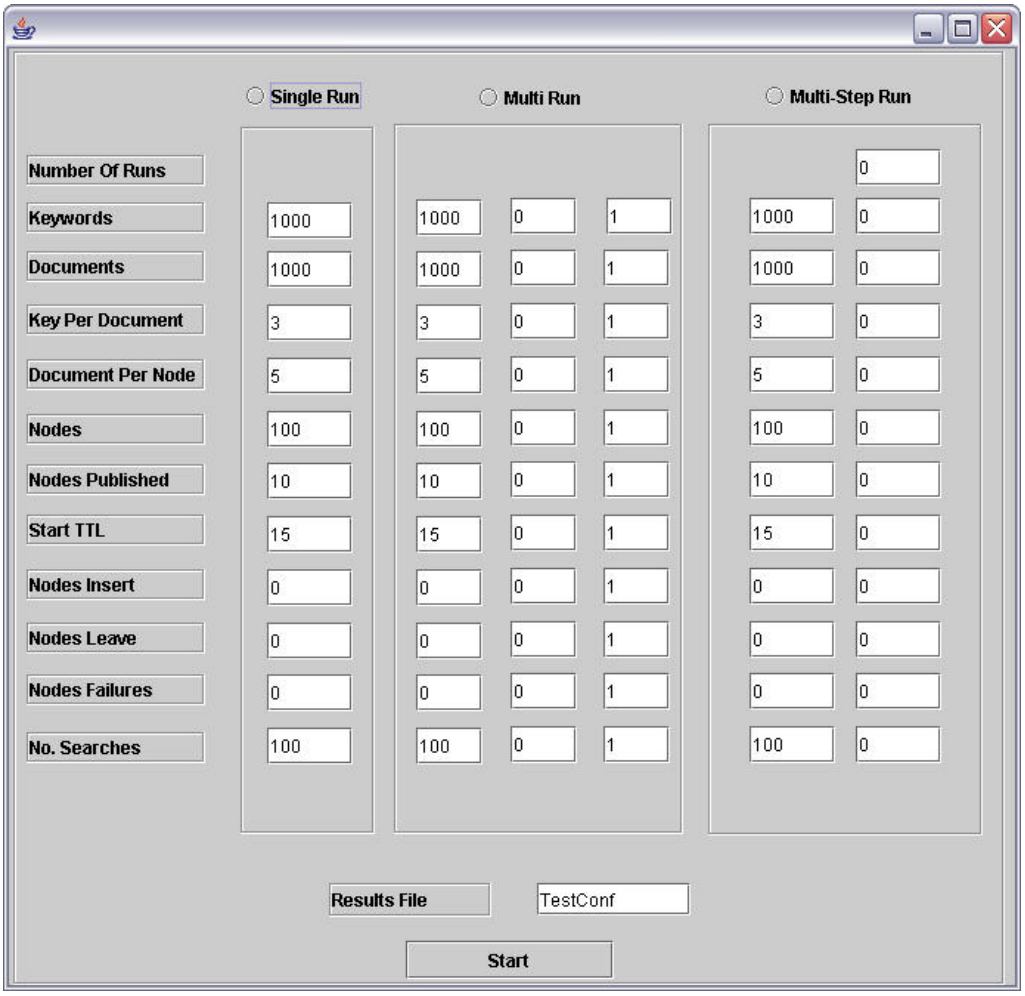
συστήματος. Στο Σχήμα 37 δίνεται η UML ακολουθία των ενεργειών που εκτελούνται κατά τη διάρκεια της εξομοίωσης (σε απλοποιημένη μορφή καθώς ο πραγματικός αριθμός των κλάσεων και μεθόδων είναι αρκετά μεγάλος για να περιγραφεί αναλυτικά).



Σχήμα 37. UML διάγραμμα της λειτουργίας της εξομοίωσης

Για να μπορέσουμε να εξάγουμε περισσότερα αποτελέσματα με τη χρήση του εξομοιωτή, αναπτύξαμε μια επιπλέον κλάση που προσφέρει τη δυνατότητα εκτέλεσης πολλαπλών εξομοιώσεων του συστήματος με διαφορετικές παραμέτρους. Στο Σχήμα 38 φαίνεται η διεπαφή που σχεδιάστηκε και η οποία δίνει τη δυνατότητα εκτέλεσης:

- μιας απλής εξομοίωσης (με συγκεκριμένες παραμέτρους)
- μιας σειράς εξομοιώσεων όπου οι παράμετροι μεταβάλλονται
    - σε όλους τους δυνατούς συνδυασμούς
    - αυξάνονται με βάση συγκεκριμένο βήμα για κάθε μία



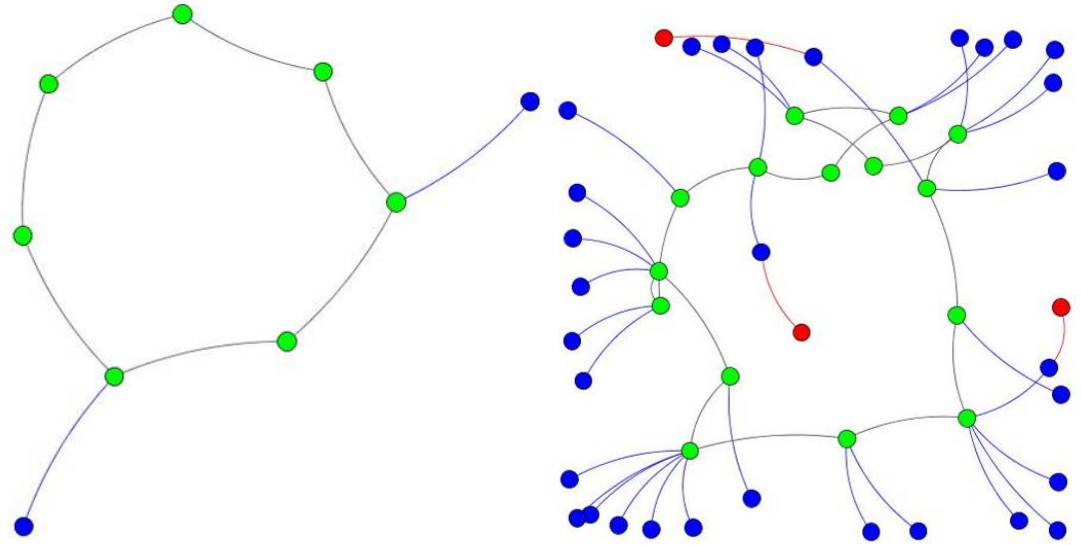
Σχήμα 38. Διεπαφή για την εκτέλεση πολλαπλών εξομοιώσεων

5.4. Τοπολογία

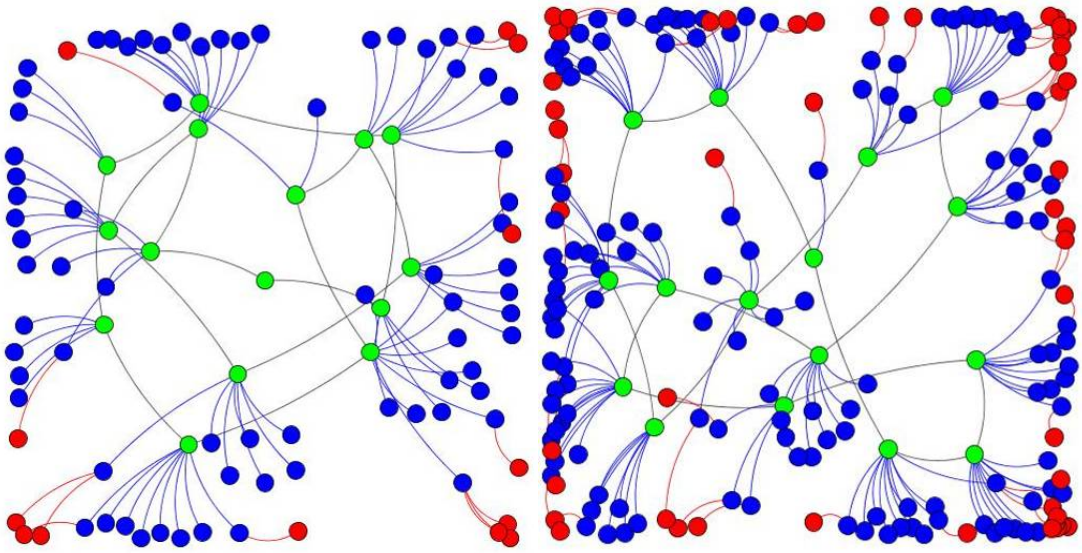
Προτού παρουσιάσουμε τα αποτελέσματα των εξομοιώσεων και την αποδοτικότητα της αρχιτεκτονικής μας, θα δώσουμε μια οπτική απεικόνιση της τοπολογίας Umbrella, όπως αυτή ορίζεται από τη σχεδίαση της αρχιτεκτονικής. Για το σκοπό αυτό χρησιμοποιήσαμε τη βιβλιοθήκη JUNG [6] ,η οποία επιτρέπει την οπτική απεικόνιση δικτύων. Αφού πρώτα δημιουργήσαμε το υπερκείμενο δίκτυο Umbrella και εισαγάγαμε μια σειρά από κόμβους, στην συνέχεια δημιουργήσαμε μια εικόνα του δικτύου απεικονίζοντας κάθε κόμβο και τις συνδέσεις πατέρα-παιδί που έχει. Επιπλέον, χρωματίσαμε κάθε κόμβο ανάλογα με το επίπεδο στο οποίο βρίσκεται, ώστε να γίνεται καλύτερα αντιληπτό το συνολικό δίκτυο.

Στο Σχήμα 39 απεικονίζονται μια σειρά από δίκτυα, όπου ο πληθυσμός μεταβάλλεται από 10 έως 1.000 κόμβους. Όπως φαίνεται από τα σχήματα, οι κόμβοι στο δίκτυο Umbrella κατανέμονται σε όλο το φάσμα της δομής και παρότι δεν έχουμε θέσει αυστηρό περιορισμό στο ελάχιστο αριθμό παιδιών

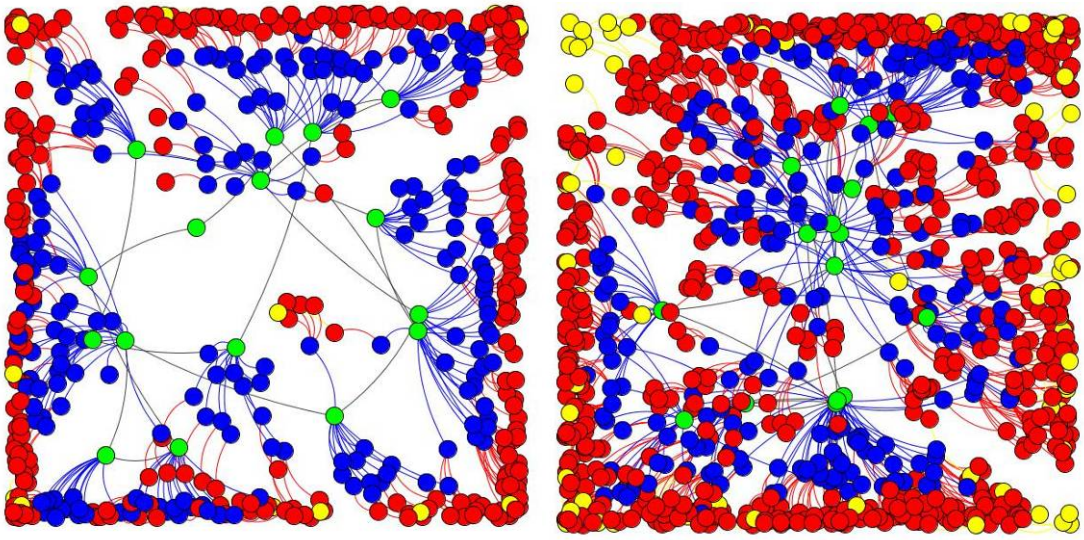
που απαιτείται προτού προστεθεί κάποιο νέο παιδί σε ανώτερο επίπεδο, η δομή διατηρείται αρκετά συμπιεσμένη. Ενώ στην συνέχεια θα παρουσιάσουμε αποτελέσματα εξομοιώσεων με πληθυσμό που φτάνει τους 100.000 κόμβους, η οπτική απεικόνιση που παρουσιάζουμε περιορίζεται σε 1.000 κόμβους καθώς είναι δύσκολο να απεικονίσουμε οπτικά μεγαλύτερο αριθμό.



(α) **(β)**



(γ) **(δ)**

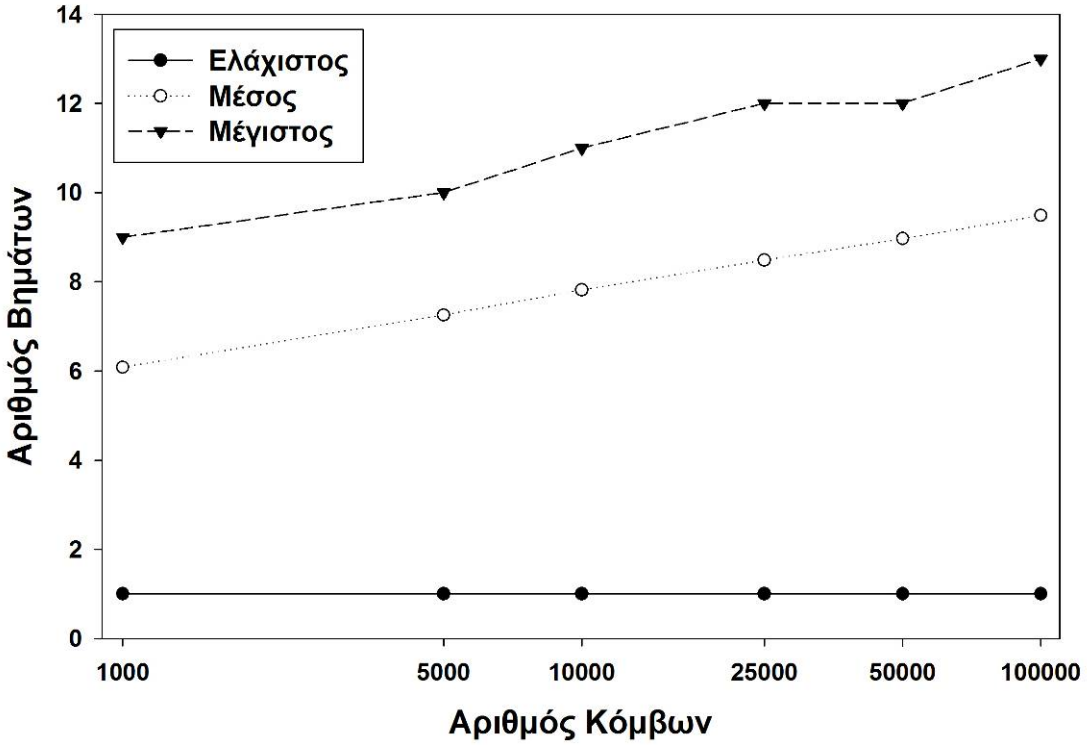


(ε) **(ζ)**

Σχήμα 39. Τοπολογία συστήματος για (α) 10 , (β) 50 , (γ) 100 , (δ) 200 , (ε) 500 και (ζ) 1.000 κόμβους

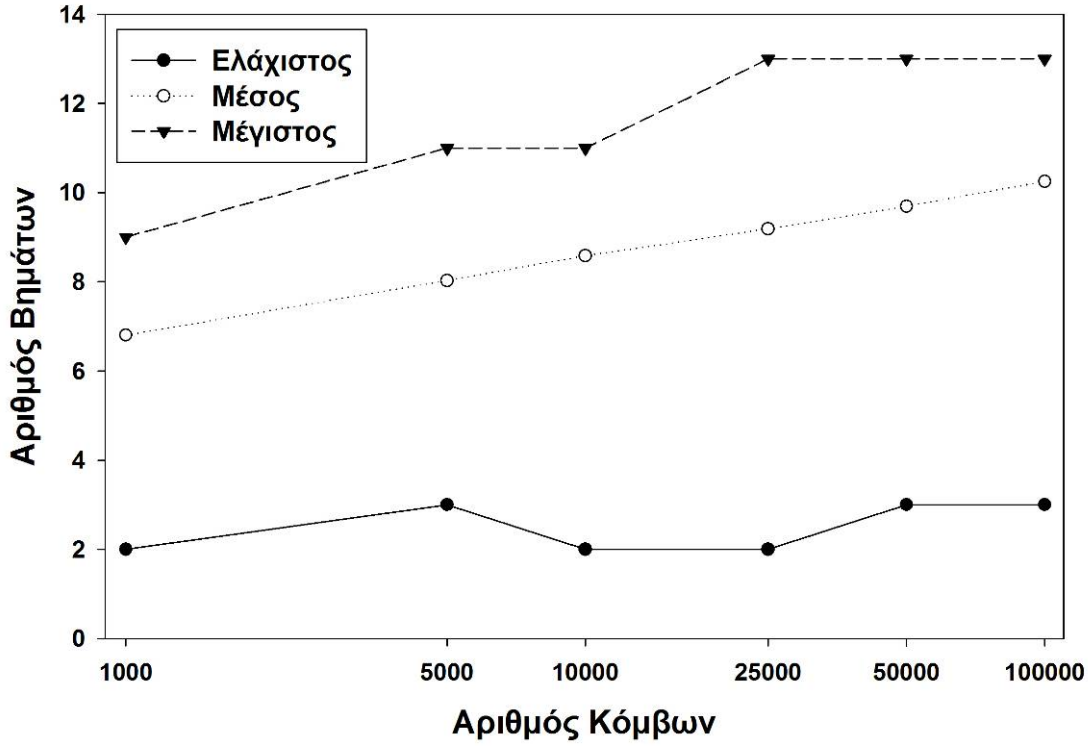
5.5. Πρώτα αποτελέσματα

Τα πρώτα αποτελέσματα της αρχιτεκτονικής μας είναι αρκετά ενθαρρυντικά. Το σύστημά μας έχει την ικανότητα να διοργανώνεται σωστά ανεξάρτητα από τον αριθμό των κόμβων, να εισάγει, δημοσιεύει και αναζητά περιεχόμενα με απόλυτη επιτυχία (100% των αναζητήσεων είναι επιτυχείς) όταν οι κόμβοι εισέρχονται ή εξέρχονται κανονικά από το δίκτυο, διατηρώντας έτσι την καλά οργανωμένη δομή του.



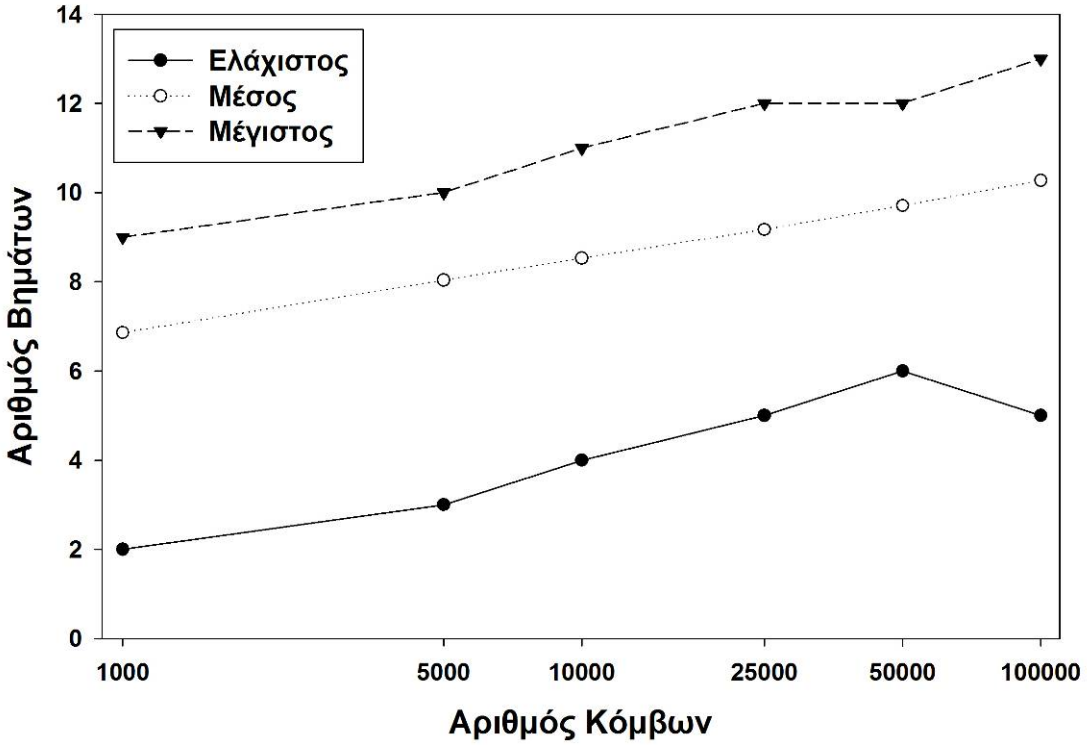
Σχήμα 40. Αριθμός βημάτων για την εισαγωγή κόμβων

Στο Σχήμα 40 φαίνεται ο αριθμός των βημάτων που απαιτούνται για να γίνει εισαγωγή ενός νέου κόμβου στο σύστημα. Στο σχήμα παρουσιάζονται διαδοχικές εξομοιώσεις για 1.000 έως 100.000 κόμβους και όπως φαίνεται ο αριθμός των βημάτων που απαιτούνται διακυμαίνεται από 1 έως 13. Ο μέσος όρος ξεκινάει από 6 και αυξάνει λογαριθμικά καθώς ο αριθμός των κόμβων αυξάνεται, με αργό όμως ρυθμό, με αποτέλεσμα για 100.000 κόμβους να απαιτούνται μόλις 8 περίπου βήματα.



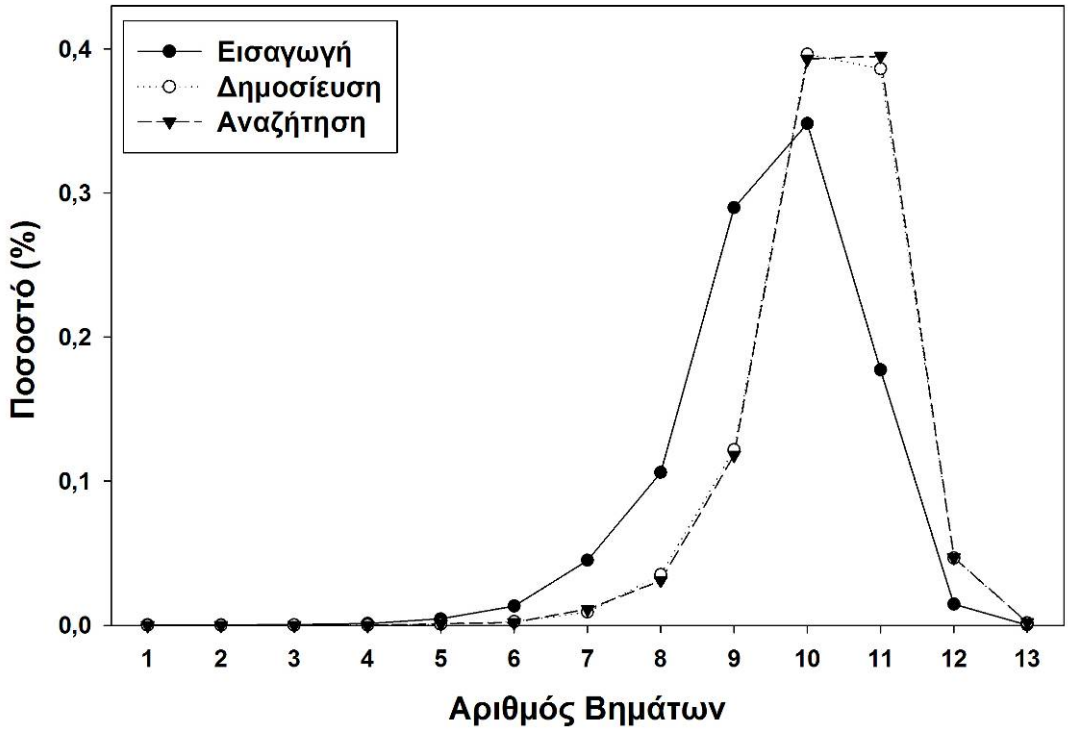
Σχήμα 41. Αριθμός βημάτων για τη δημοσίευση κλειδιών

Στο Σχήμα 41 παρουσιάζεται το ίδιο σχεδιάγραμμα για τον αριθμό των βημάτων που απαιτούνται για τη δημοσίευση ενός περιεχομένου. Στην εξομοίωσή μας όλοι οι κόμβοι περιέχουν 1 έγγραφο που περιέχει 1 κλειδί. Αφού οι κόμβοι εισέλθουν στο δίκτυο εν συνεχεία δημοσιεύουν τα περιεχόμενά τους, δηλαδή τα κλειδιά τους. Όπως φαίνεται ο αριθμός των απαιτούμενων βημάτων διακυμαίνεται από 2 έως 13 βήματα και ο μέσος όρος αυξάνει λογαριθμικά από 7 σε 9 βήματα καθώς ο αριθμός των κόμβων αυξάνει από 1.000 σε 100.000.



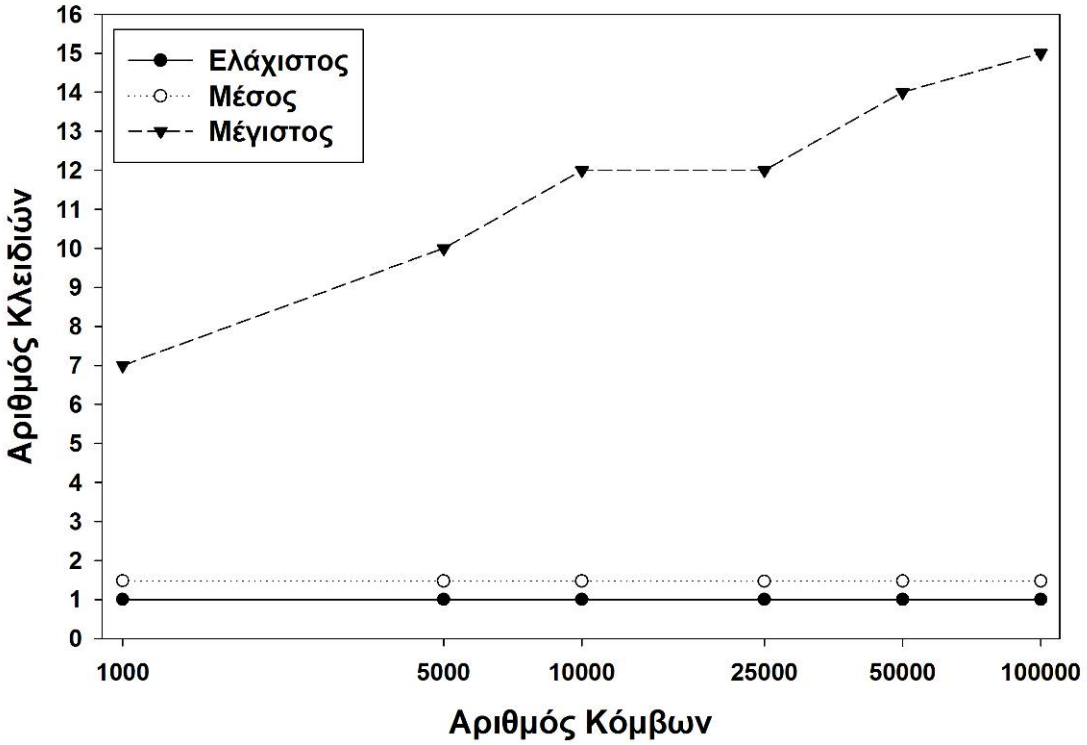
Σχήμα 42. Αριθμός βημάτων για αναζήτηση κλειδιού

Αντίστοιχη εξομοίωση έγινε και για τον αριθμό των βημάτων που απαιτούνται για να γίνει μια αναζήτηση. Τα αποτελέσματα φαίνονται στο Σχήμα 42 και πρέπει να σημειώσουμε ότι όλες οι αναζητήσεις ήταν επιτυχείς. Στην περίπτωση των αναζητήσεων, ο αριθμός των βημάτων διακυμάνθηκε από 2 έως 13 και ο μέσος όρος μεταξύ 7 και 9, αυξανόμενος λογαριθμικά.



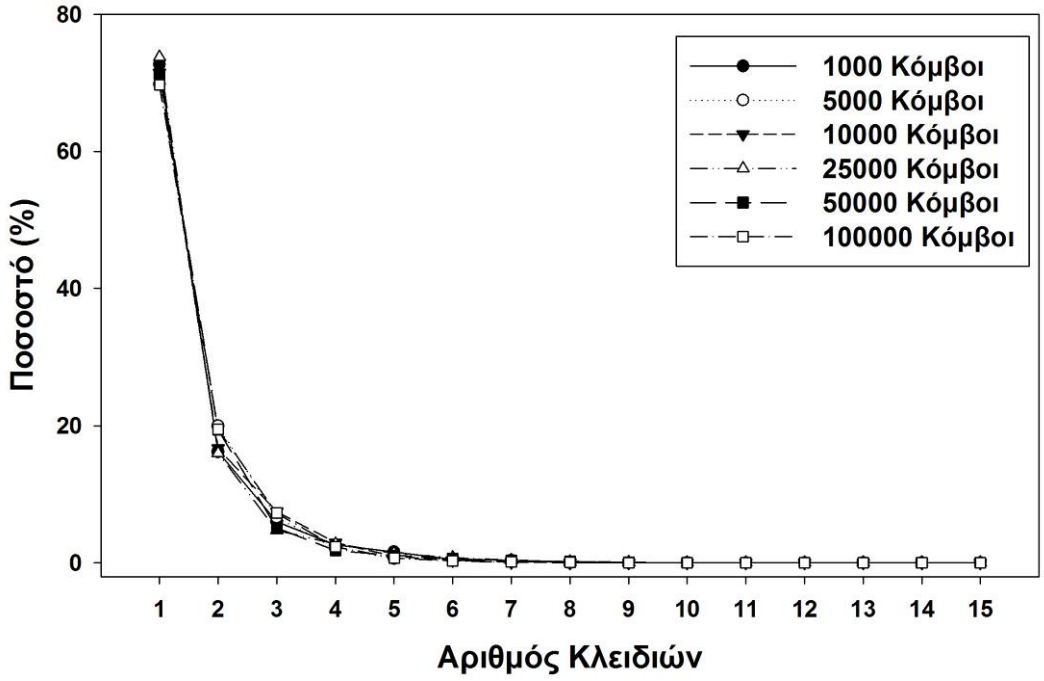
Σχήμα 43. Συνάρτηση κατανομής πιθανότητας για την περίπτωση 100.000 κόμβων

Πιο αναλυτικά, για την περίπτωση των 100.000 κόμβων, στο Σχήμα 43 παρουσιάζεται η συνάρτηση κατανομής πιθανότητας για τον αριθμό των βημάτων που απαιτούνται για την εισαγωγή ενός κόμβου, τη δημοσίευση ενός κλειδιού και την αναζήτηση. Όπως φαίνεται, η εισαγωγή απαιτεί λιγότερα βήματα από ότι η δημοσίευση και η αναζήτηση κατά περίπου ένα βήμα, γεγονός λογικό καθώς η εισαγωγή τερματίζει ένα βήμα πριν την δημοσίευση και την αναζήτηση. Το γεγονός ότι η δημοσίευση και η αναζήτηση απαιτούν τον ίδιο αριθμό βημάτων είναι επίσης αναμενόμενο, καθώς και οι δύο αλγόριθμοι βασίζονται στην ίδια λογική. Όπως φαίνεται από την κατανομή, το μέγιστο πλήθος των μηνυμάτων δημοσίευσης και αναζήτησης εντοπίζεται μεταξύ 9 και 11 βημάτων ενώ για την εισαγωγή μεταξύ 8 και 10.



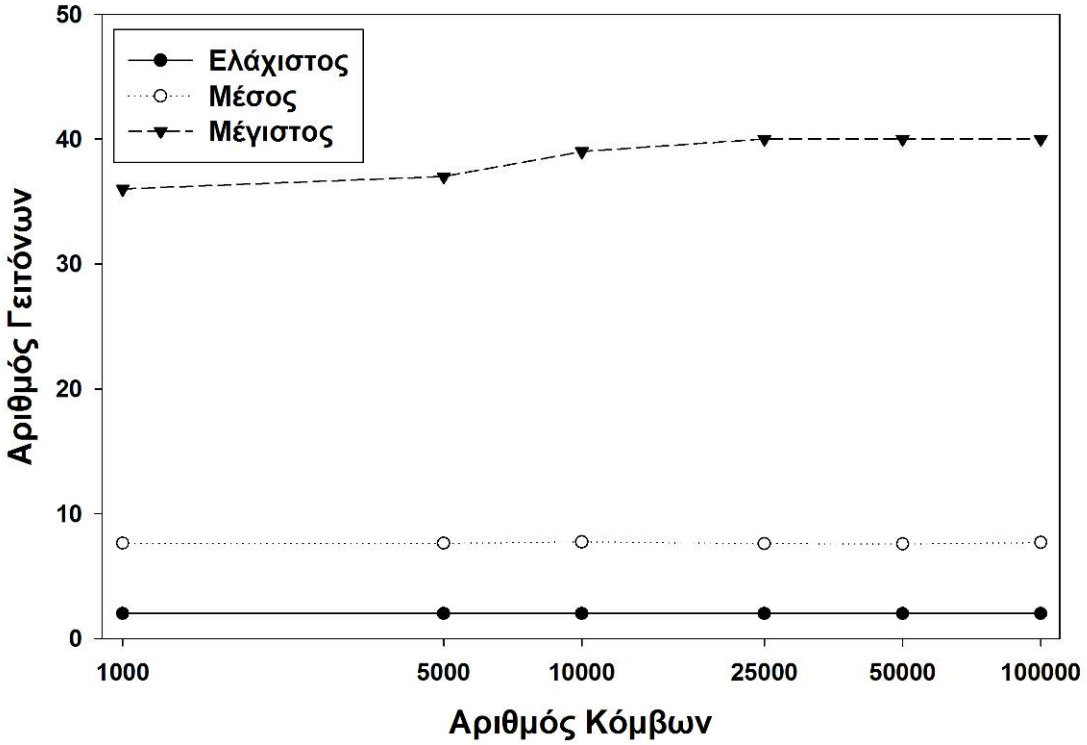
Σχήμα 44. Αριθμός κλειδιών στους κόμβους

Στο Σχήμα 44 φαίνεται η κατανομή των κλειδιών μετά τη δημοσίευσή τους από τους κόμβους. Παρατηρούμε ότι η κατανομή των κλειδιών είναι απολύτως ομαλή, με αποτέλεσμα ο μέσος όρος να διατηρείται γύρω στο 1,4. Καθώς αυξάνεται ο αριθμός των κόμβων στο δίκτυο, η κατανομή παραμένει ίδια με μόνη διαφορά το μέγιστο, όπου ελάχιστοι κόμβοι είναι υπεύθυνοι για περισσότερα κλειδιά, χωρίς αυτό να επηρεάζει ουσιαστικά το μέσο όρο. Έτσι, ενώ για 1.000 κόμβους έχουμε διακύμανση από 1 έως 7 κλειδιά, στο τέλος για 100.000 κόμβους η διακύμανση αυξάνει από 1 έως 14 κλειδιά. Η συμπεριφορά αυτή οφείλεται στην συνάρτηση κατατεμαχισμού, η οποία κατανέμει ομοιόμορφα τα αναγνωριστικά στους κόμβους και στα κλειδιά.



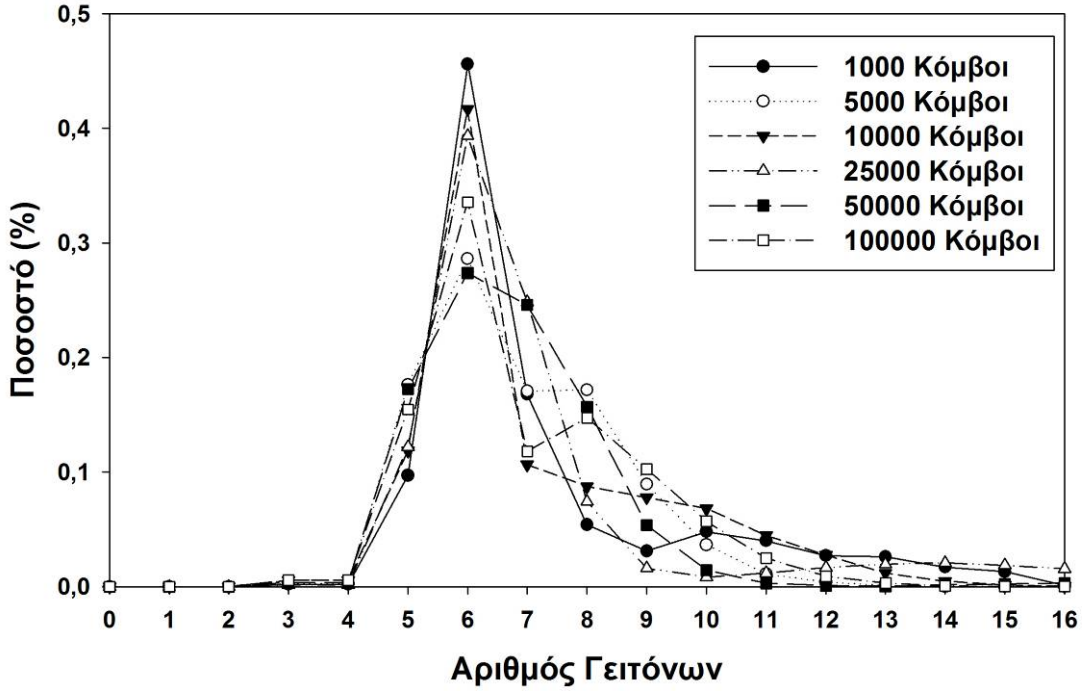
Σχήμα 45. Συνάρτηση κατανομής πιθανότητας κλειδιών

Στο Σχήμα 45 φαίνεται αναλυτικά η συνάρτηση κατανομής πιθανότητας για τον αριθμό των κλειδιών για τον οποίο κάθε κόμβος είναι υπεύθυνος. Όπως φαίνεται, ο αριθμός των κλειδιών κυμαίνεται ουσιαστικά από 1 έως 6 κλειδιά, με πάνω από 90% να έχει το πολύ 2 κλειδιά, σε σύνολο 2n κλειδιών, όπου n ο πληθυσμός. Επίσης είναι προφανές ότι ο αριθμός των κόμβων δεν επηρεάζει ουσιαστικά την κατανομή, καθώς αυτή παραμένει περίπου ίδια για όλες τις εξομοιώσεις.



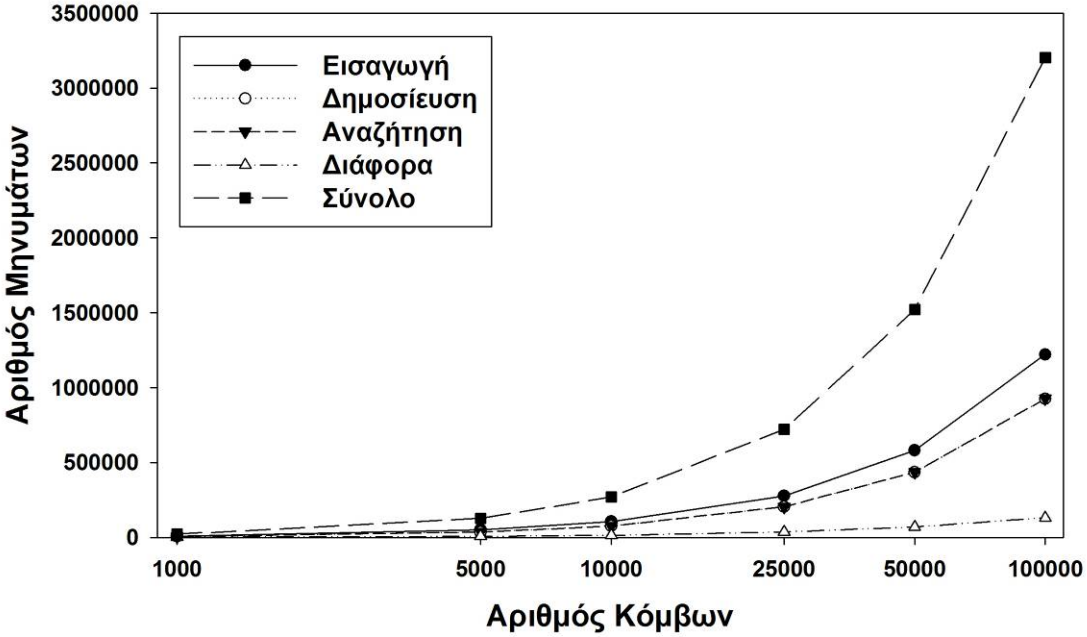
Σχήμα 46. Αριθμός γειτονικών κόμβων

Αν εξετάσουμε στην συνέχεια τον αριθμό των γειτονικών κόμβων που διατηρείται στον πίνακα γειτόνων σε σχέση με τον αριθμό των κόμβων του δικτύου, όπως φαίνεται στο Σχήμα 46, θα δούμε ότι ο μέσος όρος διατηρείται σταθερός και απλά μεγαλώνει ελάχιστα η διακύμανση των τιμών. Έτσι έχουμε ένα μέσο όρο 9 γειτόνων, γεγονός που αποδεικνύει ότι το σύστημά μας απαιτεί ελάχιστη γνώση του συνολικού δικτύου από κάθε κόμβο, η οποία διατηρείται σχεδόν ανεξάρτητη από το συνολικό αριθμό των κόμβων.



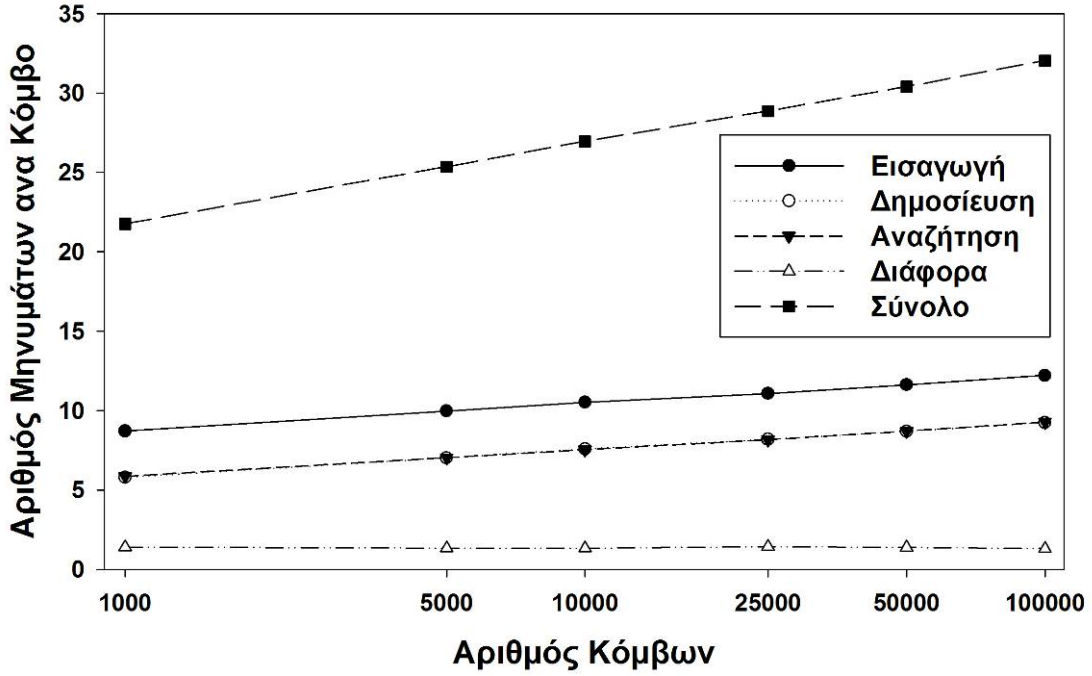
Σχήμα 47. Συνάρτηση κατανομής πιθανότητας γειτόνων

Οι παραπάνω διαπιστώσεις για τον αριθμό των γειτόνων που διατηρεί κάθε κόμβος επιβεβαιώνονται και στο Σχήμα 47, όπου φαίνεται η συνάρτηση κατανομής πιθανότητας για όλους τους συνδυασμούς πληθυσμού. Όπως παρατηρούμε, οι περισσότεροι κόμβοι διατηρούν γνώση μεταξύ 4 και 10 γειτονικών κόμβων, αριθμοί αρκετά χαμηλοί για DHT αλγόριθμο. Επίσης φαίνεται πως το μέγεθος του πληθυσμού δεν επηρεάζει ουσιαστικά την κατανομή, καθώς όλες οι εξομοιώσεις παρουσιάζουν παρόμοια συμπεριφορά.



Σχήμα 48. Αριθμός μηνυμάτων στο σύστημα

Τέλος, το σύστημά μας έχει καταφέρει να διατηρήσει τον αριθμό των μηνυμάτων που απαιτούνται για τη σωστή λειτουργία και οργάνωση σε αρκετά χαμηλό επίπεδο. Στο Σχήμα 48 φαίνεται ο αριθμός των μηνυμάτων και πως αυτός αυξάνει με τον αριθμό των κόμβων. Όπως φαίνεται η αύξηση αυτή είναι λογαριθμική και αποτελείται περισσότερο από μηνύματα εισόδου από ότι δημοσίευσης και αναζήτησης. Εδώ πρέπει να επισημάνουμε ότι στις εξομοιώσεις μας, είχαμε μια είσοδο, μια δημοσίευση όλων των περιεχομένων και μια αναζήτηση για κάθε κόμβο. Έτσι, στην περίπτωση των 100.000 κόμβων, είχαμε 100.000 εισαγωγές, 100.000 δημοσιεύσεις όλων των περιεχομένων και 100.000 αναζητήσεις. Σε πραγματική λειτουργία το σύστημα αναμένεται να έχει μεγαλύτερο ποσοστό αναζητήσεων από ότι εισαγωγές και δημοσιεύσεις, γεγονός που θα μειώσει το συνολικό αριθμό μηνυμάτων, καθώς όπως φαίνεται και από το Σχήμα 48, ο αριθμός των μηνυμάτων αναζήτησης είναι ίδιος με αυτών των δημοσιεύσεων και λιγότερος από αυτών των εισαγωγών. Στο Σχήμα 49 φαίνεται τέλος ο αριθμός των μηνυμάτων ανά κόμβο, ο οποίος αυξάνει λογαριθμικά και για 100.000 κόμβους διατηρείται περίπου στα 31 μηνύματα ανά κόμβο, μέγεθος ιδιαίτερα μικρό καθώς κάθε μήνυμα δεν ξεπερνάει τα 10 kilobytes, δηλαδή έχουμε συνολική κίνηση λιγότερη από 310 kilobytes για κάθε χρήστη, στην διάρκεια όλης της εξομοίωσης.



Σχήμα 49. Αριθμός μηνυμάτων ανά κόμβο

5.6. Σύνοψη - Συμπεράσματα

Η ενότητα αυτή παρουσίασε τα πρώτα αποτελέσματα της αρχιτεκτονικής μας. Αρχικά έγινε η παρουσίαση του εξομοιωτή που χρησιμοποιήσαμε (neurogrid) και η ανάλυση τόσο του τρόπου λειτουργίας του όσο και του μοντέλου που αναπτύχθηκε για την εξομοίωση του συστήματός μας. Στην συνέχεια, παραθέσαμε γραφικά την τοπολογία της αρχιτεκτονικής μας, προτού αρχίσουμε την παρουσίαση των αποτελεσμάτων των πρώτων εξομοιώσεων. Η πρώτη αυτή σειρά εξομοιώσεων επικεντρώθηκε στον μέσο αριθμό βημάτων για κάθε λειτουργία, τη γνώση γειτονικών κόμβων, την κατανομή των κλειδιών καθώς και το μέσο αριθμό μηνυμάτων ανά κόμβων. Τα αποτελέσματα επιβεβαίωσαν τη θεωρητική μας ανάλυση για λογαριθμική συμπεριφορά των λειτουργιών, όσον αφορά το μέσο αριθμό βημάτων, καθώς και για την περιορισμένη απαίτηση γνώσης γειτόνων αλλά και παραγόμενων μηνυμάτων ανά κόμβο (που κυμαίνονται από 20 έως 35 για μεταβλητό πληθυσμό από 1.000 έως και 100.000 κόμβους).

5.7. Αναφορές

[1] Joseph, S., "An Extendible Open Source P2P Simulator," P2PJournal, 2003.

[2] OPNET-Technologies, "The OPNET Modeler," ttp://www.opnet.com/products/modeler/home.html , 2003.
[3] UCB, LBLN, VINT, "The Network Simulator - ns-2," http://www.isi.edu/nsnam/ns, 2003.
[4] Joseph S., "Adaptive Routing in Distributed Decentralized Systems: NeuroGrid, Gnutella and Freenet", in Proceedings of Workshop on Infrastructure for Agents, MAS, and Scalable MAS, at Autonomous Agents, Montreal, Canada, 2001.
[5] JFreeChart, www.jfree.org/jfreechart/index.html
[6] Madadhain, J., Fisher, D., Smyth, P., White, S. Boey, Y., "Analysis and Visualization of Network Data using JUNG", http://jung.sourceforge.net, 2005

6 ΒΕΛΤΙΩΣΗ ΑΛΓΟΡΙΘΜΟΥ ΚΑΙ ΠΕΡΑΙΤΕΡΩ ΑΠΟΤΕΛΕΣΜΑΤΑ

Στην συνέχεια θα παρουσιάσουμε μια σειρά από αλλαγές στους αλγόριθμους εισαγωγής, δημοσίευσης, αναζήτησης και διαγραφής, ώστε η αρχιτεκτονική μας να είναι ανεκτική σε αποτυχίες κόμβων. Κατόπιν θα αναλύσουμε τα αποτελέσματα διαφόρων εξομοιώσεων που εκτελέσαμε για τον έλεγχο της απόδοσης της αρχιτεκτονικής σε πραγματικές συνθήκες.

6.1. Εισαγωγή

Στις εξομοιώσεις που παρουσιάσαμε στο προηγούμενο κεφάλαιο επιβεβαιώσαμε τη λειτουργία της αρχιτεκτονικής μας σε ιδανικές συνθήκες, όπου κόμβοι εισέρχονται στο δίκτυο, δημοσιεύουν περιεχόμενα και εκτελούν αναζητήσεις. Σε πραγματικές όμως συνθήκες λειτουργίας το σύστημά μας θα πρέπει να αντιμετωπίσει δύο επιπρόσθετες διαδικασίες :

1. Κανονική έξοδο κόμβων από το σύστημα
   Στην περίπτωση αυτή ο κόμβος που εξέρχεται καλεί τον αλγόριθμο εξόδου, που αναλύθηκε σε προηγούμενο κεφάλαιο, και αφού επιβεβαιώσει ότι η διαδικασία ολοκληρώθηκε εξέρχεται από το σύστημα.
   Ο αλγόριθμος εξόδου επιβεβαιώθηκε μέσα από μια σειρά από εξομοιώσεις, όπου ένας αριθμός (από 0% έως 80% επί του συνόλου) τυχαία επιλεγμένων κόμβων εξέρχονταν από το δίκτυο και κατόπιν γίνονταν αναζητήσεις για κλειδιά. Όλα τα αποτελέσματα των αναζητήσεών μας ήταν επιτυχή (100% επιτυχία) ακόμα και όταν το 80% των κόμβων του δικτύου είχε εξέλθει από το σύστημα.
2. Απότομη έξοδο κόμβων ή αποτυχίες επικοινωνίας
   Λόγω της φύσης του συστήματος, αναμένεται διάφοροι κόμβοι να μη βρίσκονται διαθέσιμοι είτε λόγω προσωρινής διακοπής επικοινωνίας είτε λόγω απότομης εξόδου από το δίκτυο. Στις καταστάσεις αυτές θα αναφερόμαστε ως αποτυχία του κόμβου. Οι αλγόριθμοι που παρουσιάσαμε σε προηγούμενο κεφάλαιο

δεν είναι σε θέση να αντιμετωπίσουν τέτοιου είδους αποτυχίες, παρότι ο πίνακας γειτόνων δίνει τη δυνατότητα παράκαμψης ενός κόμβου. Για το λόγο αυτό στην συνέχεια θα περιγράψουμε τις αλλαγές που έγιναν στους αλγορίθμους της αρχιτεκτονικής και θα παρουσιάσουμε μια σειρά από αποτελέσματα εξομοιώσεων, όπου καταγράφεται η απόδοση του συστήματος σε περιπτώσεις αποτυχιών.

6.2. Τροποποίηση αλγορίθμων

Στους αλγορίθμους, που αναφέραμε σε προηγούμενο κεφάλαιο, απαιτούνται αλλαγές ώστε να μπορούν να χρησιμοποιήσουν τις επιπλέον πληροφορίες που διατηρούνται στον πίνακα γειτόνων. Με βάση τη μορφή που έχει ήδη δοθεί, οι αλγόριθμοι χρησιμοποιούσαν μόνο το βασικό σετ γειτόνων και καθόλου το άνω και κάτω σετ.

Οι αλλαγές που κάναμε στους αλγόριθμους χρησιμοποιούν το πάνω και κάτω σετ γειτόνων σε περίπτωση που είναι αδύνατη η επικοινωνία με κάποιον κόμβο, ο οποίος ήταν αποθηκευμένος στο βασικό σετ γειτόνων. Όταν διαπιστωθεί αυτή η αδυναμία επικοινωνίας, τότε τα μηνύματα προωθούνται είτε σε γειτονικό κόμβο που βρίσκεται στο άνω σετ, σε περίπτωση που δεν ήταν εφικτή η επικοινωνία με τον κόμβο πατέρα, είτε σε γειτονικό κόμβο που βρίσκεται στο κάτω σετ, όταν δεν ήταν εφικτή η επικοινωνία με έναν κόμβο παιδί. Οι αλλαγές αυτές, που παρουσιάζονται στην συνέχεια, εφαρμόστηκαν σε όλους τους αλγορίθμους (εισαγωγή, δημοσίευση, εύρεση και διαγραφή) και αντικατέστησαν τις εντολές ελέγχου για ύπαρξη κόμβου πατέρα και κόμβου παιδιού αντίστοιχα.

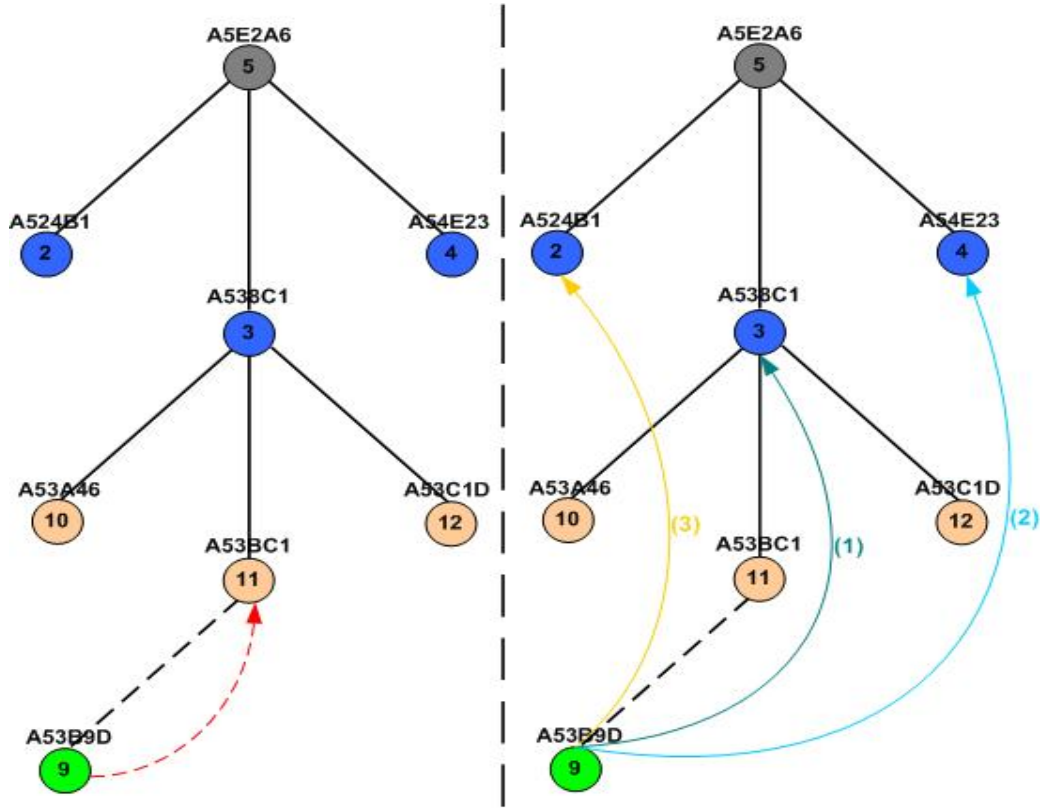
6.2.1. Αποτυχία σύνδεσης με κόμβο πατέρα

Σε περίπτωση που ένας κόμβος δε μπορεί να επικοινωνήσει με τον κόμβο πατέρα του, τότε προσπαθεί αρχικά να επικοινωνήσει με τον κόμβο πατέρα του πατέρα (καταχώρηση Up2 στο άνω σετ γειτόνων), εν συνεχεία με τον κόμβο δεξιά του κόμβου πατέρα (καταχώρηση Right2) και τέλος με τον κόμβο αριστερά του κόμβου πατέρα (καταχώρηση Left2). Στο Σχήμα 50 φαίνεται η αλλαγή αυτή υπό μορφή ψευδοκώδικα. Όπως παρατηρούμε, ο κώδικας ελέγχει με σειρά προτεραιότητας τις παραπάνω τρεις πληροφορίες από το άνω σετ (επιβεβαιώνοντας ότι οι κόμβοι υπάρχουν και είναι εφικτή η επικοινωνία) και ο πρώτος κόμβος που θα προωθήσει το μήνυμα τερματίζει και τον αλγόριθμο.

```
(1)     if ( this_node.father == unreachable )
(2)             if ( ( this_node.father2.exists( ) ) && ( this_node.father2 != unreachable ) )
(3)                     forward( this_node.father2 )
(4)             else
(5)             if ( ( this_node.right2.exists( ) ) && ( this_node.right2 != unreachable ) )
(6)                     forward( this_node.right2 )
(7)             else
(8)             if ( ( this_node.left2.exists( ) ) && ( this_node.left2 != unreachable ) )
(9)                     forward( this_node.left2 )
```

Σχήμα 50. Ψευδοκώδικας σε περίπτωση αποτυχίας κόμβου πατέρα

Στο Σχήμα 51 δίνεται ένα παράδειγμα εφαρμογής του αλγορίθμου. Αρχικά ο κόμβος A53B9D προσπαθεί να επικοινωνήσει με τον πατέρα αλλά αυτό είναι αδύνατο λόγω αποτυχίας. Στην συνέχεια θα επιχειρήσει με τη σειρά να επικοινωνήσει με τον κόμβο πατέρα του πατέρα (A538C1 μέσω της σύνδεσης 1), τον κόμβο δεξιά του προαναφερθέντος κόμβου (A53E23 μέσω σύνδεσης 2) και τέλος τον κόμβο αριστερά (A524B1 μέσω σύνδεση 3). Αν κάποια από τις παραπάνω απόπειρες είναι επιτυχής, τότε οι επόμενες δεν υλοποιούνται.



Σχήμα 51. Παράδειγμα αλγορίθμου αποτυχίας κόμβου πατέρα

6.2.2. Αποτυχία σύνδεσης με κόμβο παιδί

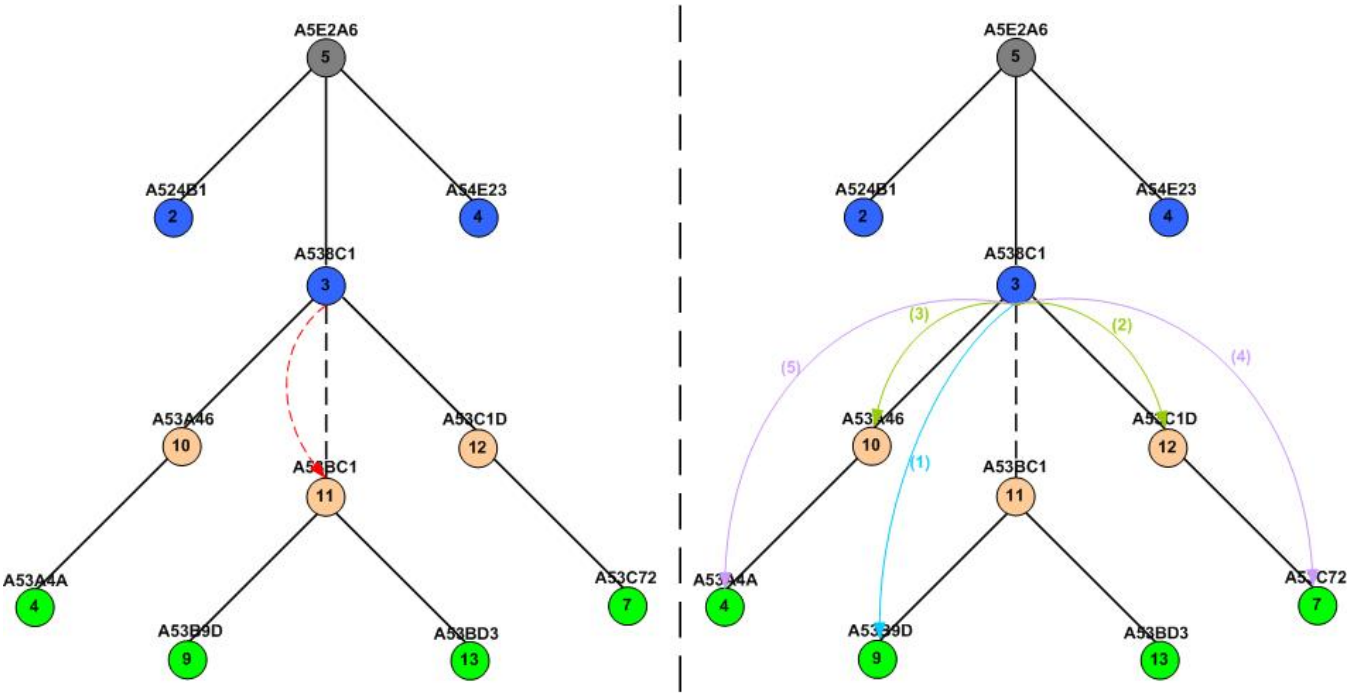
Στην περίπτωση όπου δεν είναι εφικτή η επικοινωνία με έναν κόμβο παιδί τότε αρχικά επιχειρείται επικοινωνία με ένα παιδί του παιδιού

(καταχώρηση Umbrella2 στο κάτω σετ γειτόνων), εν συνεχεία με το παιδί δεξιά του κόμβου-παιδί (καταχώρηση Umbrella στο βασικό σετ), κατόπιν με το παιδί αριστερά του κόμβου παιδί (καταχώρηση Umbrella στο βασικό σετ) και τέλος με ένα από τους δεξιά ή αριστερά κόμβους παιδιά των γειτόνων του κόμβου (καταχωρήσεις Right3 και Left3 αντίστοιχα στο κάτω σετ γειτόνων). Ο ψευδοκώδικας του αλγορίθμου αυτού δίνεται στο Σχήμα 52. Όπως και στην περίπτωση αποτυχίας του πατέρα, ο παρακάτω αλγόριθμος ελέγχει με συγκεκριμένη σειρά προτεραιότητας την εύρεση ενός εναλλακτικού κόμβου για προώθηση του μηνύματος και ο πρώτος κόμβος που επιτύχει τερματίζει τον αλγόριθμο.

```
(1)     num = query.get_number( this_node.lv+2 )
(2)     if ( ( kid_exists( num ) ) && ( this_node.kid(num) == unreachable)
(3)             if ( ( this_node.Umbrella2.exists( ) ) && ( this_node.Umbrella2 != unreachable ) )
(4)                     forward( this_node.Umbrella2.random_kid( ) )
(5)             else
(6)             if ((this_node.kid_exists(num+1)) && (this_node.kid_exists(num+1)!= unreachable))
(7)                     forward( this_node.kid( num+1 ) )
(8)             else
(9)             if ((this_node.kid_exists(num-1)) && (this_node.kid_exists(num-1)!= unreachable))
(10)                    forward( this_node.kid( num-1 ) )
(11)            else
(12)            if ( ( this_node.right3.exists( ) ) && ( this_node.right3 != unreachable ) )
(13)                    forward( this_node.right3 )
(14)            else
(15)            if ( ( this_node.left3.exists( ) ) && ( this_node.left3!= unreachable ) )
(16)                    forward( this_node.left3 )
```

Σχήμα 52. Ψευδοκώδικας σε περίπτωση αποτυχίας κόμβου-παιδί

Στο Σχήμα 53 παρουσιάζουμε ένα παράδειγμα εφαρμογής του παραπάνω αλγορίθμου. Ο κόμβος A53BC1 προσπαθεί ανεπιτυχώς να επικοινωνήσει με τον κόμβο-παιδί λόγω αποτυχίας. Αρχικά θα γίνει απόπειρα επικοινωνίας με έναν κόμβο παιδί του παιδιού (A53B9D ή A53BD3 μέσω της σύνδεσης 1) και σε περίπτωση επιτυχίας ο αλγόριθμος τερματίζει. Αν αποτύχει τότε επιχειρεί να επικοινωνήσει με τον κόμβο δεξιά και κατόπιν αριστερά του παιδιού (κόμβοι A53C1D και A53A46 μέσω συνδέσεων 2 και 3 αντίστοιχα). Αν και αυτές οι προσπάθειες αποτύχουν τότε επιχειρεί να επικοινωνήσει με τους κόμβους παιδιά των κόμβων δεξιά και αριστερά του παιδιού (κόμβοι A53C72 και A53A3A μέσω συνδέσεων 4 και 5). Και ο αλγόριθμος αυτός τερματίζει μετά την εύρεση ενός από τους παραπάνω κόμβους.



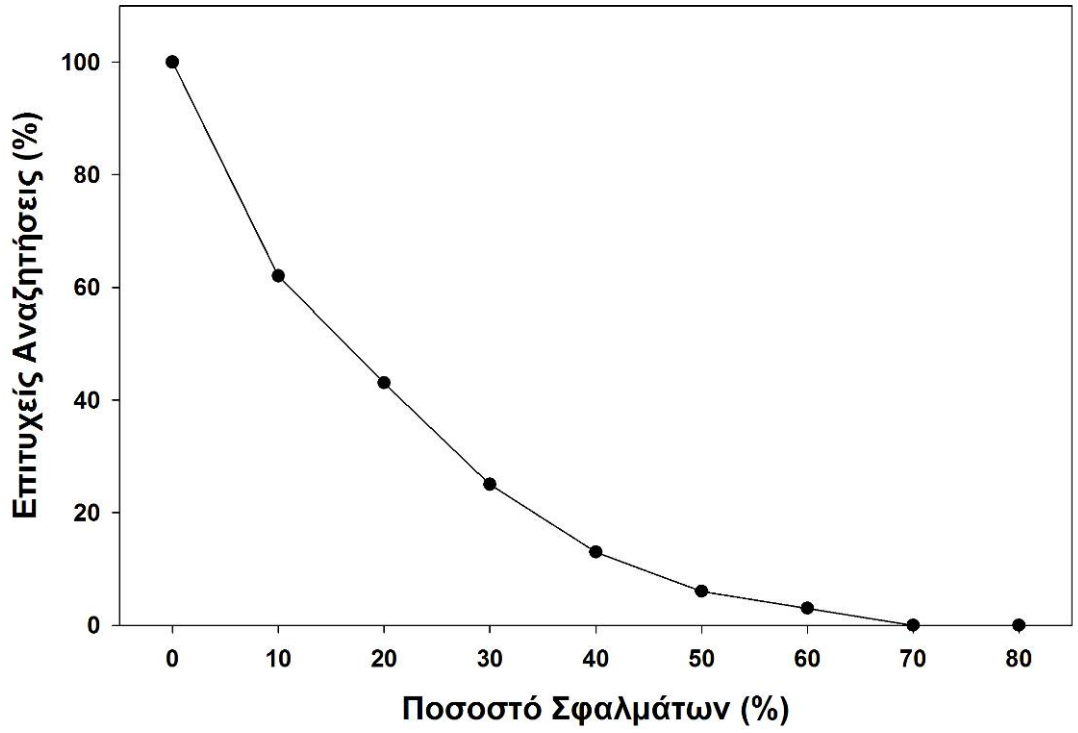
Σχήμα 53. Παράδειγμα αλγορίθμου αποτυχίας κόμβου παιδιού

6.3. Αποτελέσματα εξομοιώσεων αποτυχιών

Για τον έλεγχο των αλγορίθμων αποτυχίας και γενικότερα τη μελέτη της απόδοσης του συστήματος όταν κόμβοι αποχωρούν ξαφνικά από αυτό ή δεν είναι διαθέσιμοι, εκτελέσαμε μια σειρά από εξομοιώσεις όπου τυχαία επιλεγμένοι κόμβοι σταματούν να επικοινωνούν με τους υπόλοιπους.

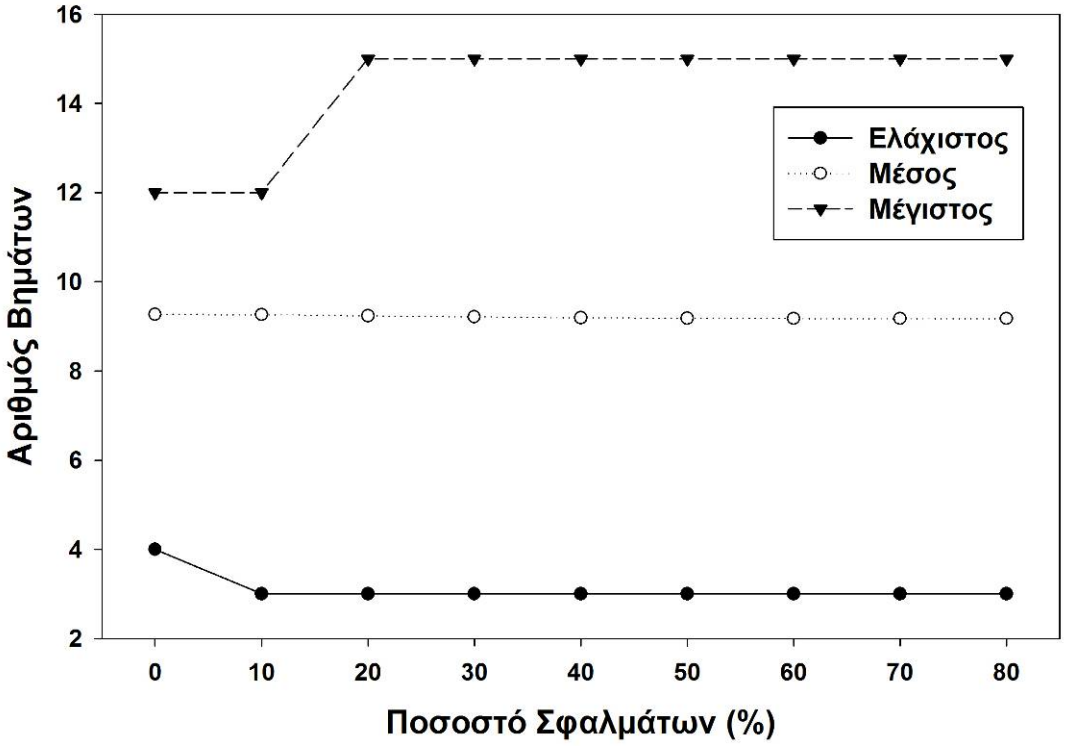
Στην πρώτη σειρά εξομοιώσεων χρησιμοποιήσαμε ένα δίκτυο 100.000 κόμβων, όπου όλοι τους δημοσιεύουν ένα περιεχόμενο που περιέχει ένα κλειδί και σταδιακά προκαλούμε αποτυχίες κόμβων, από ποσοστό 0 έως 80% του συνολικού δικτύου. Σε κάθε περίπτωση εκτελούμε 10.000 αναζητήσεις και καταγράφουμε τα ποσοστά επιτυχίας τους. Στο Σχήμα 54 φαίνονται τα ποσοστά επιτυχίας καθώς αυξάνει το ποσοστό των αποτυχιών κόμβων.

Παρατηρούμε πως το ποσοστό επιτυχών αναζητήσεων μειώνεται με λογαριθμικό ρυθμό. Έτσι παρατηρούμε πως εφόσον αποχωρήσει περίπου 20% των κόμβων το σύστημα επιτυγχάνει τις μισές από τις αναζητήσεις που επιχειρούνται. Τελικά, το σύστημα δεν μπορεί πλέον να δρομολογήσει όταν πάνω από το70% κόμβων αποτυγχάνουν.



Σχήμα 54. Ποσοστά επιτυχίας αναζήτησης καθώς αυξάνει το ποσοστό σφαλμάτων

Όσον αφορά τις επιτυχημένες αναζητήσεις των παραπάνω εξομοιώσεων, όπως φαίνεται στο Σχήμα 55, τα βήματα που απαιτούνται για την επιτυχή αναζήτηση ενός κλειδιού παραμένουν σταθερά, ανεξάρτητα από το ποσοστό των κόμβων που δεν είναι διαθέσιμοι. Στο σχήμα παρουσιάζουμε τα ποσοστά έως και 80% αποτυχίας κόμβων και παράλληλα τη διακύμανση των βημάτων. Αν συγκρίνουμε το μέσο όρο των βημάτων που απαιτούνται για επιτυχή αναζήτηση στο Σχήμα 55 με τα αποτελέσματα που παρουσιάσαμε στο Σχήμα 42, για 100.000 κόμβους, θα παρατηρήσουμε ότι και στις δύο περιπτώσεις ο μέσος όρος κυμαίνεται στα 9 βήματα. Αυτό αποδεικνύει ότι οι αλγόριθμοι που παρουσιάσαμε δεν προκαλούν καθυστέρηση στην διαδικασία αναζήτησης και διατηρούν το μέσο όρο βημάτων για επιτυχή αναζήτηση σταθερό, ανεξάρτητα από το ποσοστό των κόμβων που δεν είναι διαθέσιμοι.

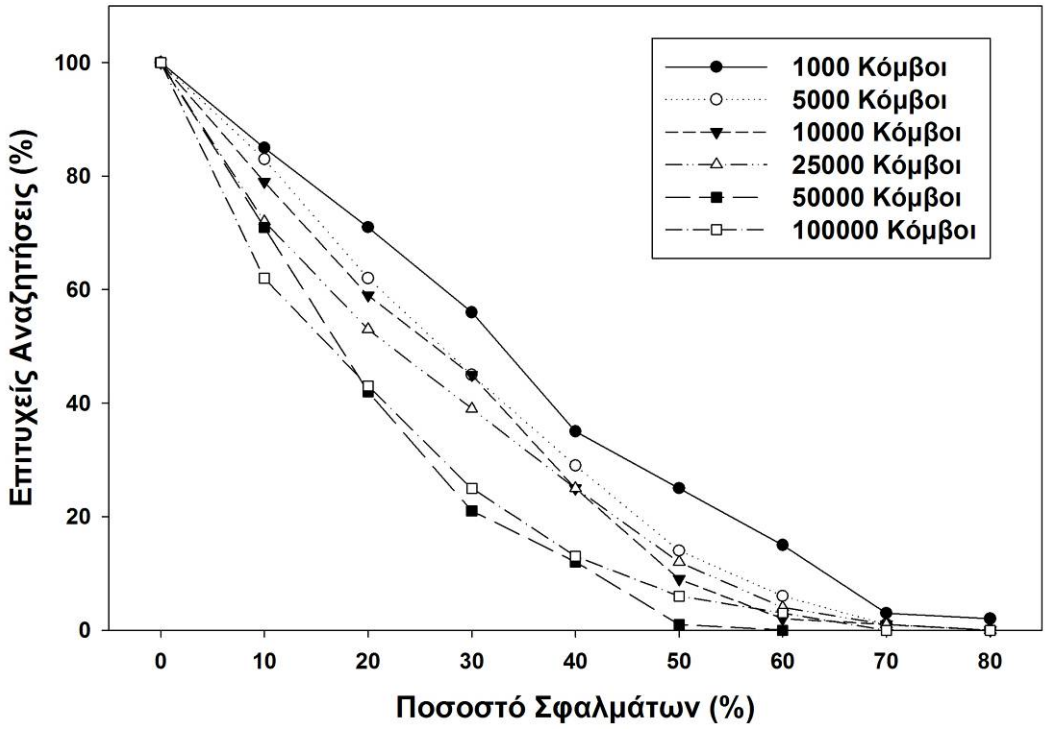


Σχήμα 55. Αριθμός βημάτων των επιτυχών αναζητήσεων καθώς αυξάνουν οι αποτυχίες

Στην δεύτερη σειρά εξομοιώσεων χρησιμοποιήσαμε ένα δίκτυο με 1.000, 5.000, 10.000, 25.000 , 50.000 και 100.000 κόμβους και διεξοδικά δημιουργήσαμε αποτυχίες κόμβων από ποσοστό 0% έως και 80%, με βήματα 10%. Σε όλες τις εξομοιώσεις, το 100% των αρχικών κόμβων δημοσίευσε τα περιεχόμενα του. Πρέπει να σημειώσουμε πως με βάση την αρχιτεκτονική μας, τα κλειδιά των περιεχομένων που δημοσιεύονται δεν αποθηκεύονται στους κόμβους που έχουν τα περιεχόμενα αλλά σε κόμβους με βάση τον αλγόριθμο δημοσίευσης. Επιπλέον, κάθε κόμβος που δημοσιεύει περιεχόμενα έχει 1 έγγραφο, στο οποίο αντιστοιχεί 1 κλειδί (με βάση τις παραμέτρους που ορίσαμε κατά τη διάρκεια των εξομοιώσεων). Έτσι κάθε κόμβος που δημοσιεύει δημιουργεί 1 κλειδί προς δημοσίευση το οποίο θα αποθηκευτεί διάσπαρτο στο δίκτυο με βάση τον αλγόριθμό μας.

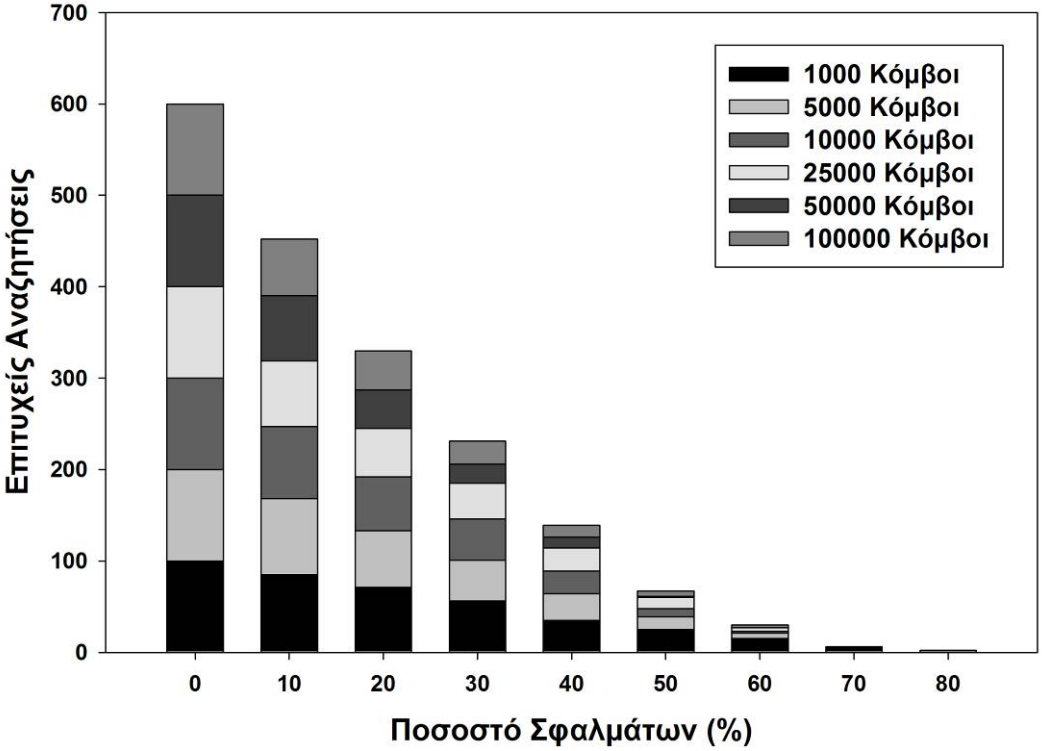
Στο Σχήμα 56 παρουσιάζεται ο αριθμός των επιτυχών αναζητήσεων καθώς αυξάνει το ποσοστό των κόμβων που δεν είναι διαθέσιμοι για διαφορετικό μέγεθος δικτύου (από 1.000 έως 100.000). Όπως παρατηρούμε, όσο αυξάνει ο αριθμός των κόμβων τόσο το σύστημά μας γίνεται πιο ασταθές καθώς αυξάνουν οι αποτυχίες, αλλά με μικρή διαφορά. Επίσης, καθώς αυξάνουν οι κόμβοι μετατοπίζεται το οριακό σημείο, πέραν του οποίου το σύστημα δε λειτουργεί πλέον καθόλου. Έτσι, ενώ για 1.000 κόμβους το σημείο

αυτό είναι γύρω στο 80% μετατοπίζεται καθώς αυξάνει ο αριθμός των κόμβων και γίνεται 70% για τους 100.000 κόμβους.



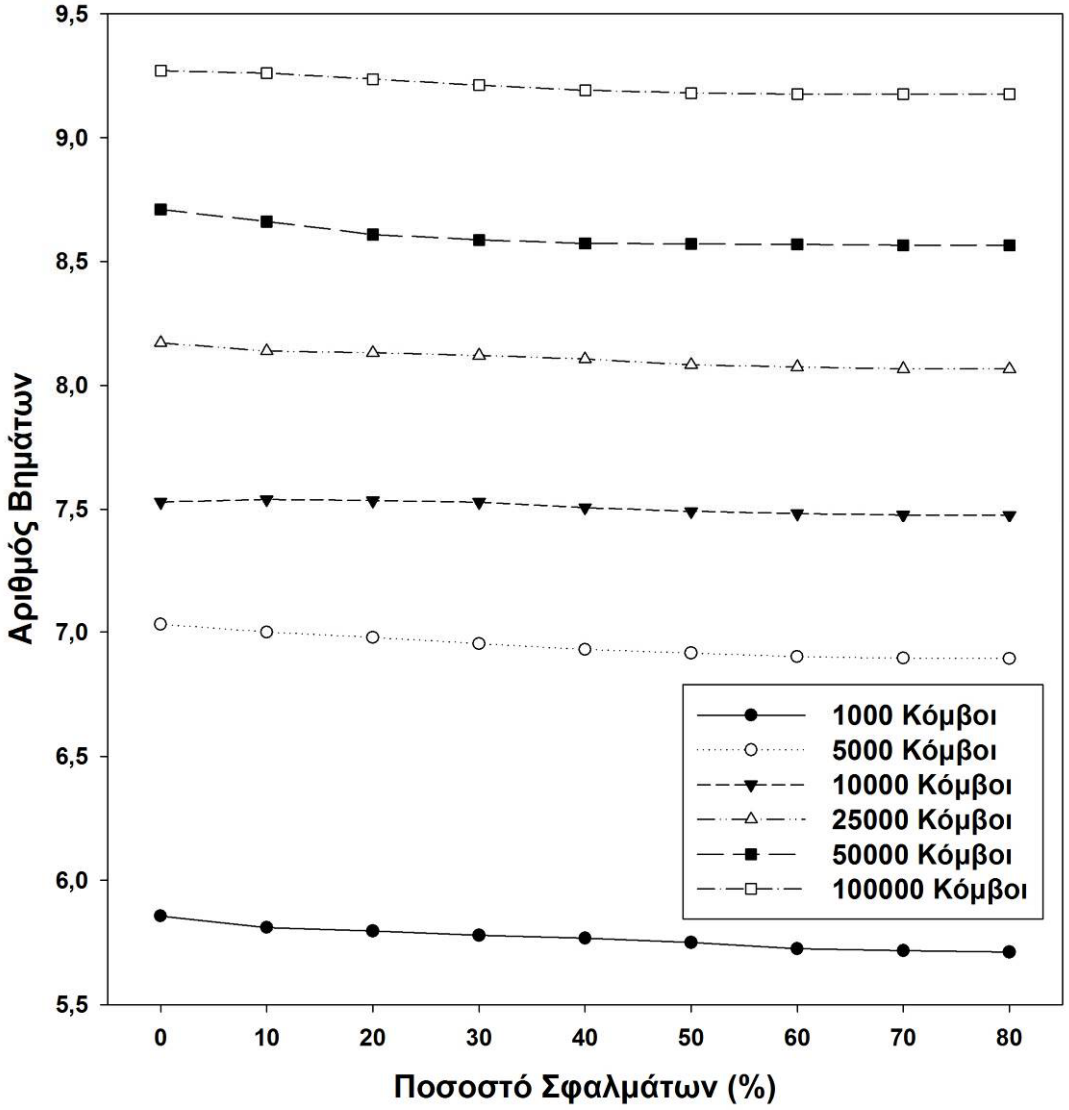
Σχήμα 56. Ποσοστό επιτυχών αναζητήσεων καθώς αυξάνουν οι αποτυχίες για διαφορετικό αριθμό κόμβων

Τα παραπάνω συμπεράσματα επιβεβαιώνονται και από το Σχήμα 57, όπου παρουσιάζονται τα αθροιστικά αποτελέσματα επιτυχών αναζητήσεων για διαφορετικό αριθμό κόμβων. Όπως παρατηρούμε, καθώς αυξάνει το ποσοστό των αποτυχιών μειώνεται το άθροισμα των επιτυχημένων αναζητήσεων και παράλληλα μειώνεται και η αναλογία μεταξύ των εξομοιώσεων με λιγότερους και περισσότερους κόμβους. Οι διαφορές αυτές όμως πρέπει να επισημάνουμε ότι είναι αρκετά χαμηλές Έτσι, ενώ αρχικά για τις 5 εξομοιώσεις είχαμε 1/6 επί του συνόλου επιτυχείς αναζητήσεις για κάθε εξομοίωση, καθώς τα ποσοστά αποτυχίας αυξάνουν βλέπουμε ότι τα ποσοστά για τις εξομοιώσεις με 100.000, 50.000 και 25.000 κόμβους μειώνονται περισσότερο ενώ για τις εξομοιώσεις με 1.000, 5.000 και 10.000 κόμβους μειώνονται μεν αλλά με κάπως μικρότερο ρυθμό.



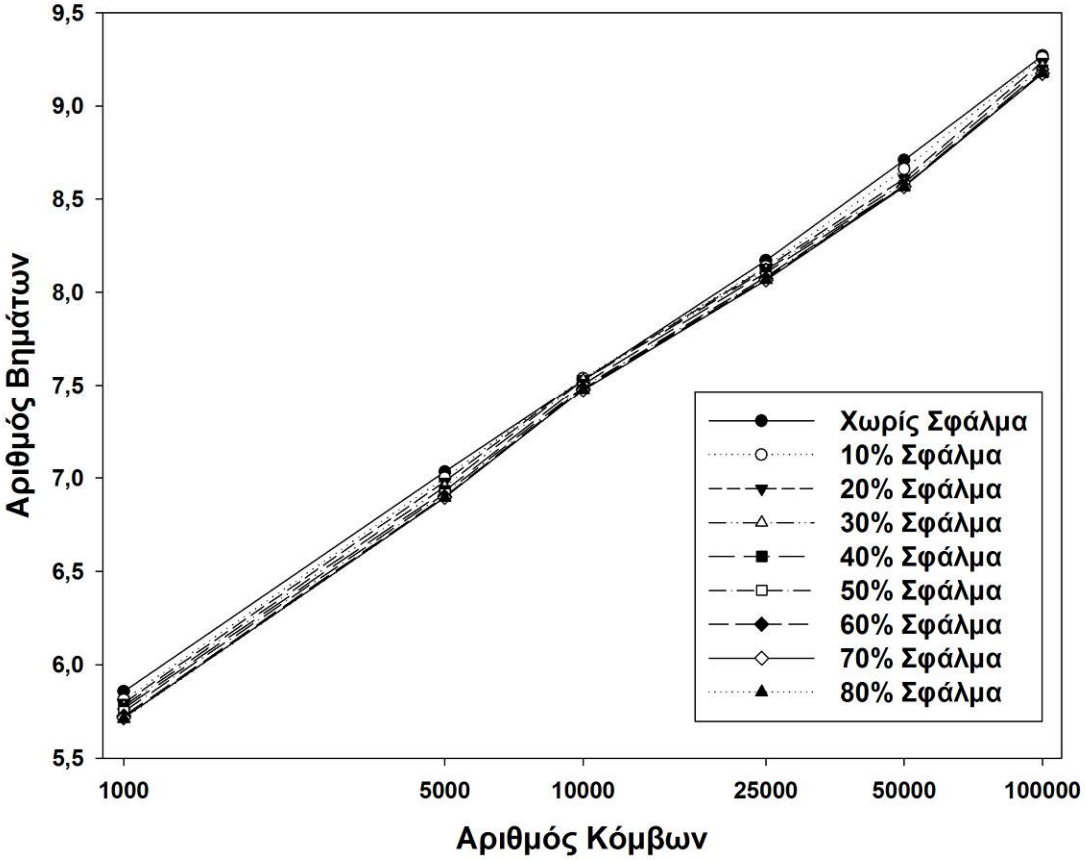
Σχήμα 57. Αθροιστικά αποτελέσματα επιτυχημένων αναζητήσεων

Αν στην συνέχεια παρατηρήσουμε τα βήματα που απαιτούνται για επιτυχείς αναζητήσεις, όπως αυτά δίνονται στο Σχήμα 58, θα διαπιστώσουμε ότι ο μέσος όρος τους κυμαίνεται μεταξύ 5,5 και 9,5 βήματα, αριθμός αρκετά ικανοποιητικός. Παρατηρούμε επίσης, ότι καθώς αυξάνουν οι αποτυχίες, ο μέσος όρος παραμένει σταθερός, για όλες τις περιπτώσεις πληθυσμού κόμβων.



Σχήμα 58. Αριθμός βημάτων των επιτυχών αναζητήσεων καθώς αυξάνουν οι αποτυχίες για διαφορετικό αριθμό κόμβων

Το ίδιο θα παρατηρήσουμε και στο Σχήμα 59, όπου παρουσιάζεται ο αριθμός βημάτων ως προς τον πληθυσμό κόμβων για διαφορετικά ποσοστά σφαλμάτων. Όπως φαίνεται, το ποσοστό των σφαλμάτων δεν επηρεάζει ουσιαστικά τον αριθμό των βημάτων. Αντίθετα, ο πληθυσμός είναι αυτός που ουσιαστικά ορίζει τον αριθμό των βημάτων, όπως παρατηρήσαμε και στο προηγούμενο κεφάλαιο.



Σχήμα 59. Αριθμός βημάτων των επιτυχών αναζητήσεων καθώς αυξάνει ο αριθμός κόμβων για διαφορετικό αριθμό σφαλμάτων

6.4. Σύνοψη - Συμπεράσματα

Στην ενότητα αυτή παρουσιάσαμε μια σειρά από αλλαγές στους αλγόριθμους εισαγωγής, δημοσίευσης, αναζήτησης και διαγραφής, ώστε η αρχιτεκτονική μας να είναι ανεκτική σε αποτυχίες κόμβων. Αρχικά οριοθετήσαμε τη διαφορά μεταξύ κανονικής εξόδου και απότομης εξόδου (αποτυχία κόμβου). Κατόπιν παραθέσαμε αναλυτικά τις αλλαγές στους αλγορίθμους με βάση το διαχωρισμό σε αποτυχίες κόμβου πατέρα ή παιδιού. Εν συνεχεία αναλύσαμε τα αποτελέσματα διαφόρων εξομοιώσεων που εκτελέσαμε για τον έλεγχο της απόδοσης της αρχιτεκτονικής σε πραγματικές συνθήκες, με μεταβλητό αριθμό κόμβων αλλά και ποσοστών αποτυχίας. Τα αποτελέσματα έδειξαν πως το ποσοστό επιτυχών αναζητήσεων μειώνεται με λογαριθμικό ρυθμό σε σχέση με το ποσοστό των αποτυχιών. Επίσης παρατηρήσαμε πως οι αλγόριθμοι που παρουσιάσαμε δεν προκαλούν καθυστέρηση στην διαδικασία αναζήτησης και διατηρούν το μέσο όρο

βημάτων για επιτυχή αναζήτηση σταθερό, ανεξάρτητα από το ποσοστό των κόμβων που δεν είναι διαθέσιμοι.

7 ΑΛΓΟΡΙΘΜΟΣ ΕΠΙΔΙΟΡΘΩΣΗΣ

Στην προηγούμενη ενότητα παρουσιάσαμε μια σειρά από αλλαγές στους βασικούς αλγορίθμους του συστήματός μας για την καλύτερη λειτουργία σε συνθήκες αποτυχιών κόμβων, είτε αυτές οφείλονται σε απότομη αποχώρηση ενός κόμβου είτε σε σφάλματα δικτύου. Οι βελτιώσεις αυτές αυξάνουν την ανεκτικότητα του συστήματος σε σφάλματα αλλά δεν έχουν επιδιορθωτικό χαρακτήρα. Με τη χρήση των επιπλέον συνδέσεων στον πίνακα γειτόνων επιτυγχάνουν την εύρεση νέων μονοπατιών όταν κάποια μονοπάτια δεν είναι προσβάσιμα. Παρόλα αυτά, οι αλλαγές αυτές απλά αποφεύγουν το πρόβλημα των σφαλμάτων και δεν το επιδιορθώνουν. Όταν το ποσοστό των σφαλμάτων είναι μικρό τότε αυτοί οι αλγόριθμοι θεωρούνται ικανοποιητικοί για τη σωστή και ομαλή λειτουργία του συστήματος. Όταν όμως το ποσοστό των σφαλμάτων αυξάνεται, η έλλειψη αλγορίθμου επιδιόρθωσης οδηγεί στον κατακερματισμό του συστήματος και στην δυσλειτουργία.

7.1. Αλγόριθμος επιδιόρθωσης

Για να επιλύσουμε αυτό το πρόβλημα αναπτύξαμε έναν αλγόριθμο επιδιόρθωσης, ο οποίος επιτυγχάνει την επιδιόρθωση σφαλμάτων όποτε εντοπίζεται σφάλμα κόμβου. Ο αλγόριθμος αυτός αποτελείται από μια παραλλαγή του αλγόριθμου αποχώρησης, που παρουσιάστηκε σε προηγούμενη ενότητα, και βασίζεται στην επιδιόρθωση σφαλμάτων που παρουσιάζονται σε κόμβους-παιδιά.

Εύκολα αποδεικνύεται ότι κάθε σφάλμα μπορεί να μετατραπεί σε σφάλμα παιδιού μέσα από την προώθηση κατάλληλων μηνυμάτων σε κόμβους που βρίσκονται στον πίνακα γειτόνων. Η διαδικασία περιλαμβάνει την προώθηση του μηνύματος επιδιόρθωσης είτε προς τα άνω επίπεδα είτε προς τα κάτω μέχρι να φτάσει το μήνυμα στον κόμβο πατέρα του κόμβου που παρουσιάζει το σφάλμα. Πιο αναλυτικά, οι ενέργειες που απαιτούνται για την εύρεση του κατάλληλου κόμβου πατέρα παρουσιάζονται στον Πίνακας 8.

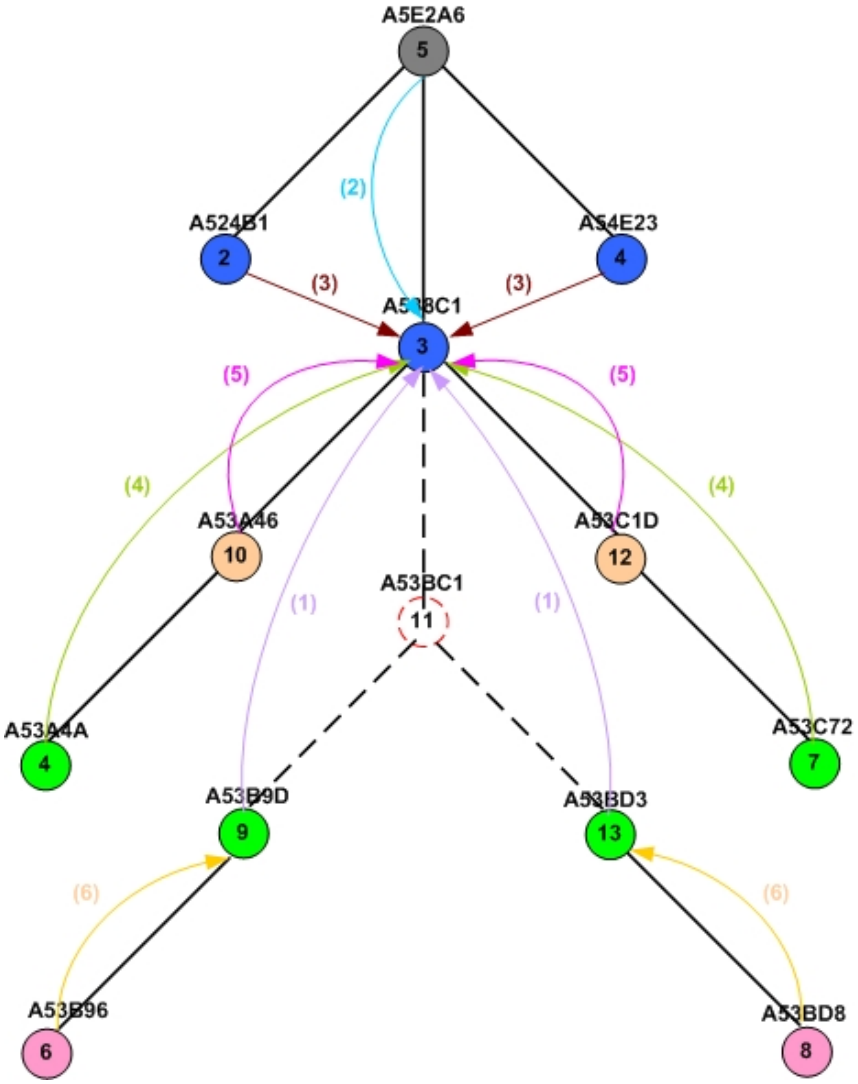
Στο Σχήμα 61 απεικονίζονται οι ενέργειες που εκτελούνται ώστε να ευρεθεί ο πατέρας του παιδιού που παρουσιάζει το σφάλμα. Πιο αναλυτικά, η σύνδεση (1) χρησιμοποιείται για την προώθηση του μηνύματος επιδιόρθωσης

στον κόμβο Up2 όταν το σφάλμα παρουσιάζεται στον κόμβο Up, η σύνδεση (2) όταν το σφάλμα παρουσιάζεται στον κόμβο Node2, οι συνδέσεις (3) όταν το σφάλμα παρουσιάζεται είτε στους κόμβους Left3 είτε Right3, οι συνδέσεις (4) όταν το σφάλμα παρουσιάζεται στους κόμβους Left2 ή Right2, οι συνδέσεις (5) όταν το σφάλμα εντοπίζεται στους κόμβους Left ή Right και τέλος η σύνδεση (6) όταν το σφάλμα εντοπίζεται στον κόμβο Up2.

Κόμβος που παρουσιάζει το σφάλμα	**Ενέργεια που εκτελείται**
Up	Προώθησε στο Up2
Node2	Προώθησε στο Node
Left3 ή Right3	Προώθησε στο Left ή στο Right αντίστοιχα
Left2 ή Right2	Προώθησε στο Up2
Left ή Right	Προώθησε στο Up
Up2	Προώθησε στο Up

Πίνακας 8. Ενέργειες προώθησης του μηνύματος επιδιόρθωσης εωσότου εντοπιστεί ο πατέρας του κόμβου που παρουσιάζει το σφάλμα.

Εάν κάποια από τις παραπάνω συνδέσεις δεν είναι διαθέσιμη, είτε λόγω απουσίας τέτοιας σύνδεσης είτε λόγω άλλου σφάλματος (ταυτόχρονα σφάλματα (churn)), τότε το μήνυμα επιδιόρθωσης δρομολογείται προς όλους τους γειτονικούς κόμβους ώστε να μπορέσει επιτυχώς να δρομολογηθεί στον πατέρα του κόμβου που αποτυγχάνει. Αν το πρόβλημα εντοπίζεται και στον πατέρα του κόμβου που αποτυγχάνει, τότε επιδιορθώνεται πρώτα ο κόμβος πατέρα και μετά ο κόμβος παιδί. Με αυτόν τον τρόπο μπορούμε πάντα να εφαρμόζουμε τον απλό αλγόριθμο επιδιόρθωσης χωρίς επιπρόσθετες πολύπλοκες διαδικασίες.



Σχήμα 60. Παράδειγμα εύρεσης του κόμβου πατέρα σε περίπτωση σφάλματος κόμβου

Όταν εντοπιστεί ο κατάλληλος κόμβος και ενημερωθεί για το σφάλμα του παιδιού του, μια παραλλαγή του αλγορίθμου αποχώρησης εκτελείται ώστε να επιδιορθωθεί το σφάλμα και να αντικατασταθεί ο κόμβος που δεν είναι πλέον προσβάσιμος από ένα από τα υπόλοιπα παιδιά του ή απλά να διαγραφεί, εάν ο κόμβος δεν έχει άλλα παιδιά. Ο αλγόριθμος έχει ως εξής:

- Έλεγξε εάν ο κόμβος έχει παιδιά. Εάν έχει, τότε ένα από αυτά πρέπει να βρίσκεται στην καταχώρηση Umbrella2
- Εάν δεν έχει παιδιά, τότε ενημέρωσε όλους τους γείτονες, μέσω των συνδέσεων του πίνακα γειτόνων, ώστε να διαγράψουν τον κόμβο που παρουσιάζει σφάλμα και να ανανεώσουν τη δομή τους
- Εάν έχει παιδιά, τότε επίλεξε τυχαία ένα παιδί από την καταχώρηση Umbrella2 και ενημέρωσε όλους του γείτονες για την αλλαγή

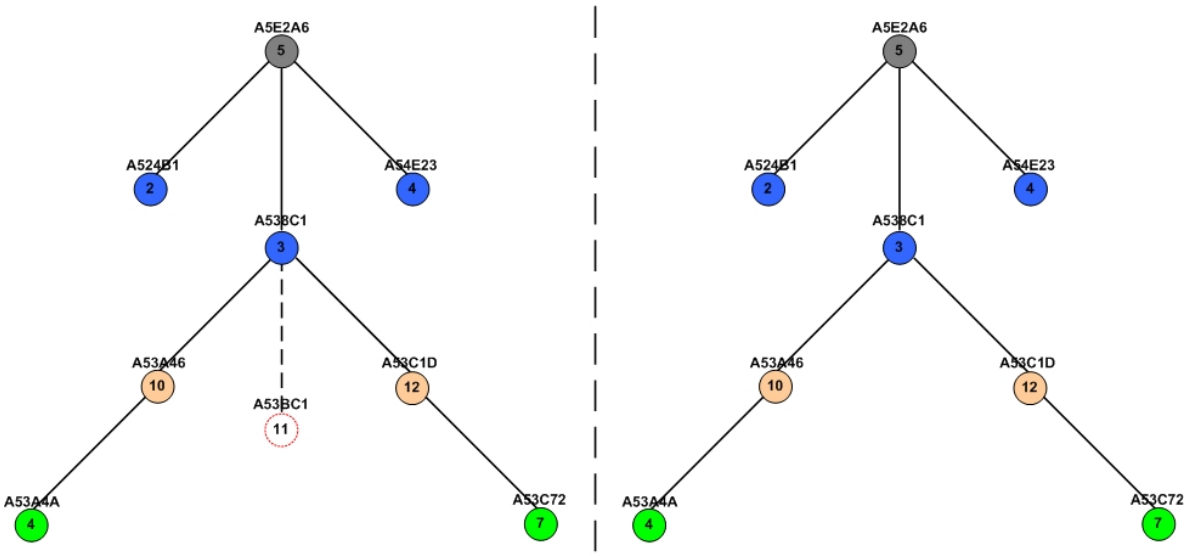
- Το νέο παιδί ενημερώνεται για την αλλαγή και ανανεώνει κατάλληλα τον πίνακα γειτόνων του μέσω των γειτονικών του κόμβων

Ο ψευδοκώδικας του αλγορίθμου επιδιόρθωσης δίνεται αναλυτικά στο Σχήμα 61.

```
(1)   repair ( )
(2)         if ( has_Umbrella2( ) )
(3)               kid = get_appropriate_Umbrella2()
(4)               contact_neighboors_of_change(kid)
(5)               kid.gather_new_neighboors()
(6)         else
(7)               contact_neighboors_of_change()
```

Σχήμα 61. Ψευδοκώδικας του αλγορίθμου επιδιόρθωσης

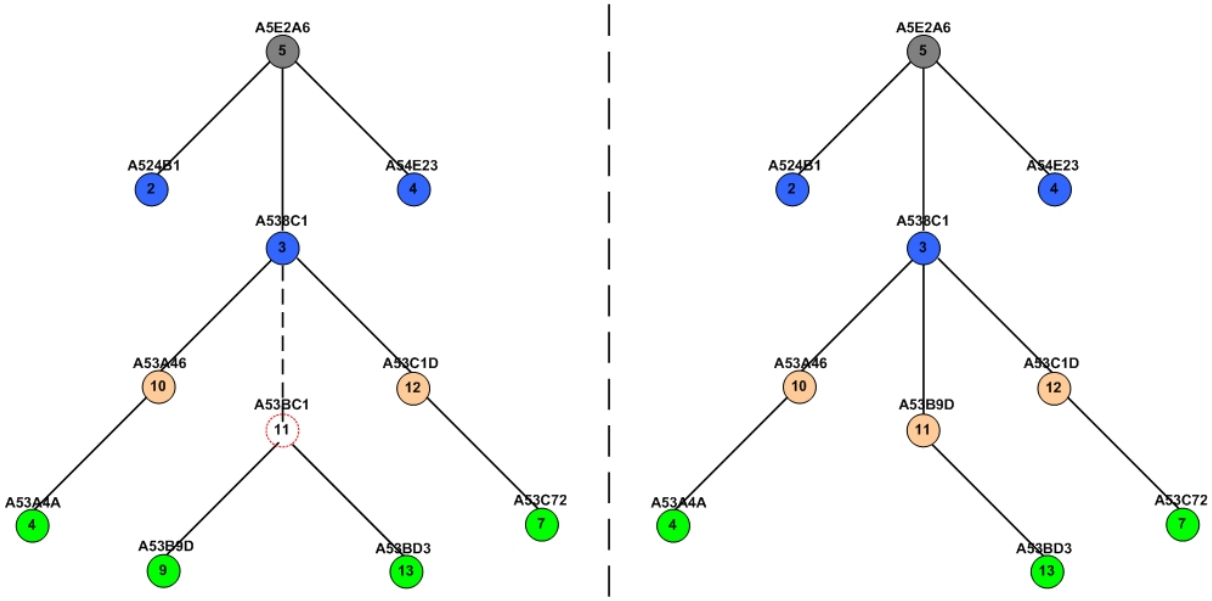
Στα ακόλουθα δύο σχήματα παρουσιάζουμε διαγραμματικά τον τρόπο λειτουργίας του αλγόριθμου επιδιόρθωσης. Πιο συγκεκριμένα, στο Σχήμα 62 παρουσιάζουμε την περίπτωση όπου ο κόμβος που αποτυγχάνει δεν έχει παιδιά, ενώ στο Σχήμα 63 την περίπτωση που ο κόμβος έχει παιδιά. Στην πρώτη περίπτωση, ο κόμβος A53BC1 παρουσιάζει σφάλμα και έχει ενημερωθεί κατάλληλα ο κόμβος πατέρας του A538C1, ο οποίος και εκτελεί τον αλγόριθμο επιδιόρθωσης. Καθώς το παιδί δεν έχει παιδιά (και ο πατέρας του επομένως δεν έχει καταχωρήσεις στο πεδίο Node2), απλά ο κόμβος A538C1 διαγράφει τον κόμβο A53BC1 από τη λίστα των παιδιών του και ενημερώνει κατάλληλα όλους τους γείτονές του.



Σχήμα 62. Παράδειγμα εκτέλεσης αλγορίθμου επιδιόρθωσης όταν ο κόμβος δεν έχει παιδιά

Στην δεύτερη περίπτωση, όπου ο κόμβος A53BC1 που παρουσιάζει σφάλμα έχει παιδιά, ο πατέρας A538C1 εκτελεί τον αλγόριθμο επιδιόρθωσης όπως αυτός παρουσιάζεται στο Σχήμα 63. Σύμφωνα με τον αλγόριθμό μας, ο

A538C1 επιλέγει ένα τυχαίο κόμβο από τις καταχωρήσεις του στο πεδίο Node2, στην συγκεκριμένη περίπτωση τον κόμβο A53B9D, και στην θέση του παλαιού του παιδιού τοποθετεί το νέο κόμβο. Ως συνέπεια, ενημερώνει όλους του γείτονές του και κατόπιν τον κόμβο A53B9D για την αλλαγή. Ο κόμβος A53B9D αλλάζει με τη σειρά του όλες τις κατάλληλες καταχωρήσεις στον πίνακα γειτόνων του και ενημερώνεται από τον πατέρα και τους παλαιούς του γείτονες για τις νέες καταχωρήσεις, ενημερώνοντας επίσης και όλους τους γείτονες που χρειάζεται να ανανεώσουν τις καταχωρήσεις τους.



Σχήμα 63. Παράδειγμα εκτέλεσης αλγορίθμου επιδιόρθωσης όταν ο κόμβος έχει παιδιά

Κάθε κόμβος είναι υπεύθυνος για τον περιοδικό έλεγχο του πίνακα γειτόνων του μέσω της χρήσης μηνυμάτων ping προς όλους τους κόμβους που βρίσκονται στον πίνακα. Κάθε φορά που εντοπίζεται ένα σφάλμα καλείται ο αλγόριθμος επιδιόρθωσης. Όπως θα δείξουμε στην συνέχεια μέσα από τα πειραματικά αποτελέσματα, ο μηχανισμός αυτός αυξάνει σημαντικά την ανεκτικότητα του συστήματός μας σε σφάλματα. Σε συνδυασμό μάλιστα με τη χρήση των σχημάτων αντιγράφων και των εικονικών δικτύων, που θα παρουσιάσουμε αναλυτικά στην επόμενη ενότητα, επιτυγχάνεται η υποστήριξη υψηλού ποσοστού σφαλμάτων κόμβων δίχως να μειώνεται η αποδοτικότητα τους συστήματος.

7.2. Αποτελέσματα εξομοιώσεων αποτυχιών και επιδιόρθωσης

Σε αυτήν τη σειρά των εξομοιώσεων δοκιμάσαμε τον αλγόριθμο επιδιόρθωση σε περιπτώσεις πολλαπλών σφαλμάτων κόμβων, με σκοπό την αξιολόγηση της αποδοτικότητάς του σε επιτυχείς αναζητήσεις. Στις

εξομοιώσεις που πραγματοποιήσαμε μεταβάλαμε τον πληθυσμό του δικτύου από 1.000 έως 100.000 κόμβους και εν συνεχεία προκαλέσαμε περιοδικές αποτυχίες τυχαίων κόμβων σε βήματα του 10% του πληθυσμού, από 0 έως 80% του συνολικού πληθυσμού του δικτύου.

Όταν ένας κόμβος παρουσιάζει σφάλμα, δεν είναι δυνατή καμία επικοινωνία προς ή από τον κόμβο αυτόν ή τον πίνακα γειτόνων του. Ο μηχανισμός επιδιόρθωσής μας παρακάμπτει τον κόμβο που παρουσιάζει σφάλμα και τον αντικαθιστά με ένα γειτονικό κόμβο, ενώ ο πίνακας γειτόνων του αναδημιουργείται μέσω των γειτονικών κόμβων, στον βαθμό που αυτό είναι εφικτό. Οποιαδήποτε πληροφορία χαθεί κατά τη διάρκεια αυτής της διαδικασίας παραμένει κενή και αντικαθίσταται από νέους κόμβους/συνδέσεις στην συνέχεια της εξομοίωσης. Αυτά τα κενά μπορεί να προκαλέσουν προσωρινή έλλειψη διαθεσιμότητας για μερικούς κόμβους αλλά στις περισσότερες περιπτώσεις αποκαθίστανται αυτόματα καθώς η δομή του δικτύου αλλάζει διαρκώς.

Μια σημαντική παράμετρος στον αλγόριθμο επιδιόρθωσής μας αποτελεί ο ρυθμός με τον οποίο καλείται ο μηχανισμός. Πιο συγκεκριμένα, κάθε κόμβος καλεί το μηχανισμό σε δύο περιπτώσεις:

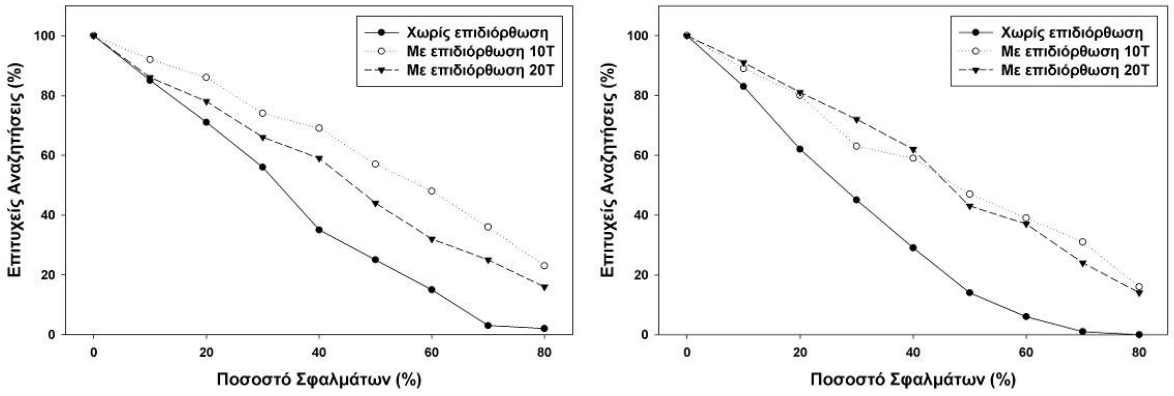
- Κατά τη διάρκεια εκτέλεσης ενός αλγορίθμου (εισαγωγή, δημοσίευση, αποχώρηση ή αναζήτηση), όπου δεν είναι δυνατή η επικοινωνία με κάποιον κόμβο που βρίσκεται στον πίνακα γειτόνων
- Κατά τη διάρκεια του ελέγχου των καταχωρήσεων στον πίνακα γειτόνων, ο οποίος εκτελείται ανά περιοδικά διαστήματα από κάθε κόμβο

Η πρώτη περίπτωση είναι σταθερή και εκτελείται σε όλη τη διάρκεια των εξομοιώσεων όποτε εντοπίζεται ένα τέτοιο σφάλμα. Η δεύτερη περίπτωση είναι μεταβλητή καθώς εκτελέσαμε εξομοιώσεις με διαφορετικές περιόδους ελέγχου. Στα αποτελέσματα που ακολουθούν η περίοδος αυτή είναι είτε 10T είτε 20T, όπου T είναι μια σταθερά που αντιπροσωπεύει τη δραστηριότητα επικοινωνίας κάθε κόμβου. Στην περίπτωση των εξομοιώσεών μας το T ισούται με 100 μηνύματα αποστολής ή παραλαβής από κάθε κόμβο. Με αυτόν τον τρόπο διασφαλίζουμε πως οι κόμβοι που δεν είναι ενεργοί δεν θα πλημμυρίζουν το δίκτυο με μηνύματα επιδιόρθωσης και μόνο οι κόμβοι που έχουν αυξημένη πιθανότητα να παρουσιάζουν σφάλματα στον πίνακα γειτόνων τους (κόμβοι με αυξημένο όγκο κίνησης) θα εκτελούν τους περιοδικούς ελέγχους.

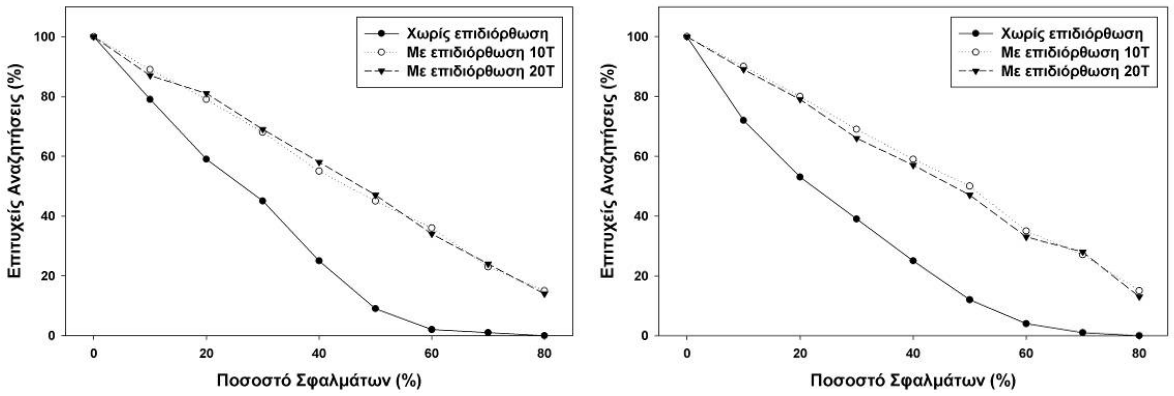
Αρχικά εξετάζουμε την επίπτωση του μηχανισμού επιδιόρθωσης στο ποσοστό επιτυχών αναζητήσεων. Στο Σχήμα 64 φαίνονται τα ποσοστά

επιτυχών αναζητήσεων για τις περιπτώσεις όπου δε χρησιμοποιείται ο μηχανισμός επιδιόρθωσης και για τις περιπτώσεις όπου χρησιμοποιείται επιδιόρθωση με παράμετρο 10T και 20T, για διαφορετικούς πληθυσμούς κόμβων που κυμαίνονται από 1.000 έως 100.000 κόμβους. Όπως φαίνεται από τα σχήματα, ο μηχανισμός επιδιόρθωσης βελτιώνει σημαντικά την απόδοση του συστήματος ανεξάρτητα από τον πληθυσμό του δικτύου.

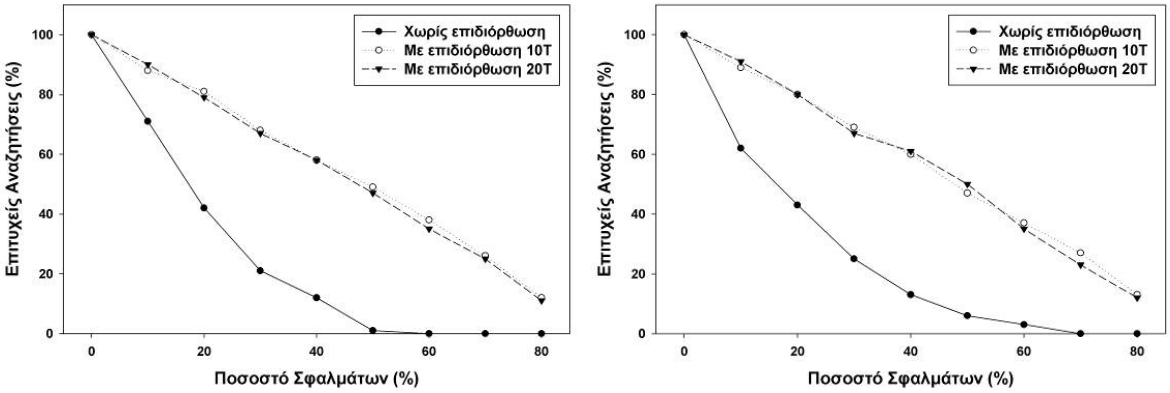
Αν ελέγξουμε τη συμπεριφορά του δικτύου προσεκτικά θα δούμε ότι ο αλγόριθμος επιδιόρθωσης επιτυγχάνει να αλλάξει τον τρόπο που μειώνονται οι επιτυχείς αναζητήσεις και από λογαριθμική μείωση πλέον το σύστημά μας παρουσιάζει γραμμική μείωση ως προς το ποσοστό σφαλμάτων. Αυτό σημαίνει ότι ακόμα και όταν οι μισοί κόμβοι στο δίκτυό μας αποτυγχάνουν, οι μισές αναζητήσεις είναι ακόμα επιτυχείς. Αυτή η συμπεριφορά είναι ιδιαίτερα σημαντική καθώς στις εξομοιώσεις που εκτελέσαμε οι κόμβοι δεν αποτυγχάνουν σειριακά (ένας-ένας) αλλά μαζικά (churn) με βήματα 10% του πληθυσμού. Οι συνθήκες αυτές είναι ιδιαίτερα δύσκολες και συνήθως στα πραγματικά ομότιμα συστήματα η αποχώρηση των χρηστών γίνεται κάτω από πιο ομαλές συνθήκες.



(α) **(β)**



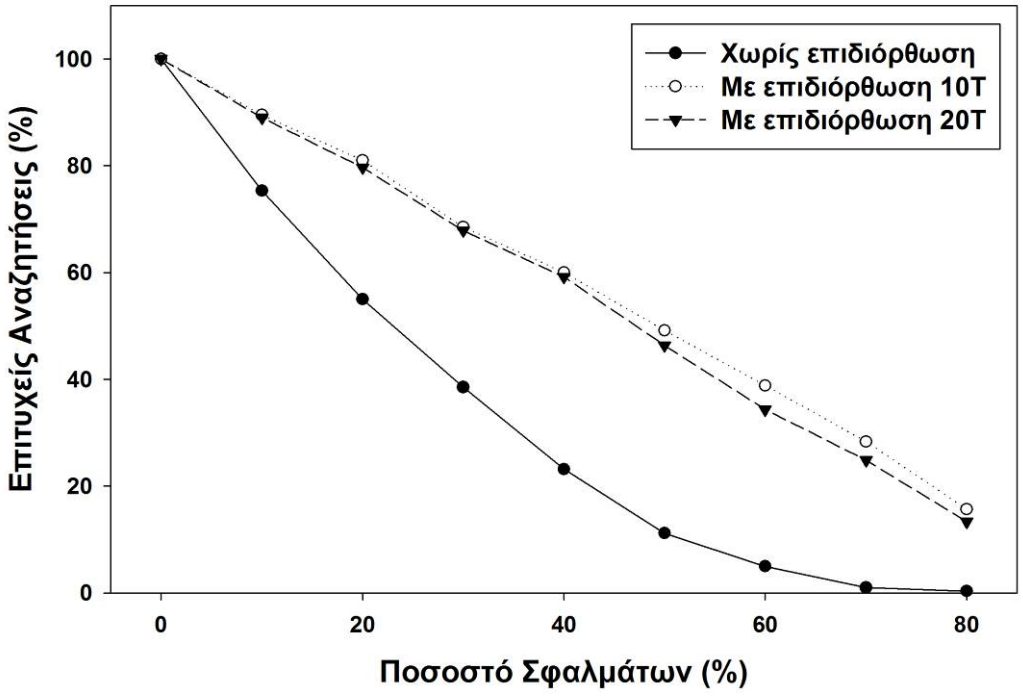
(γ) **(δ)**



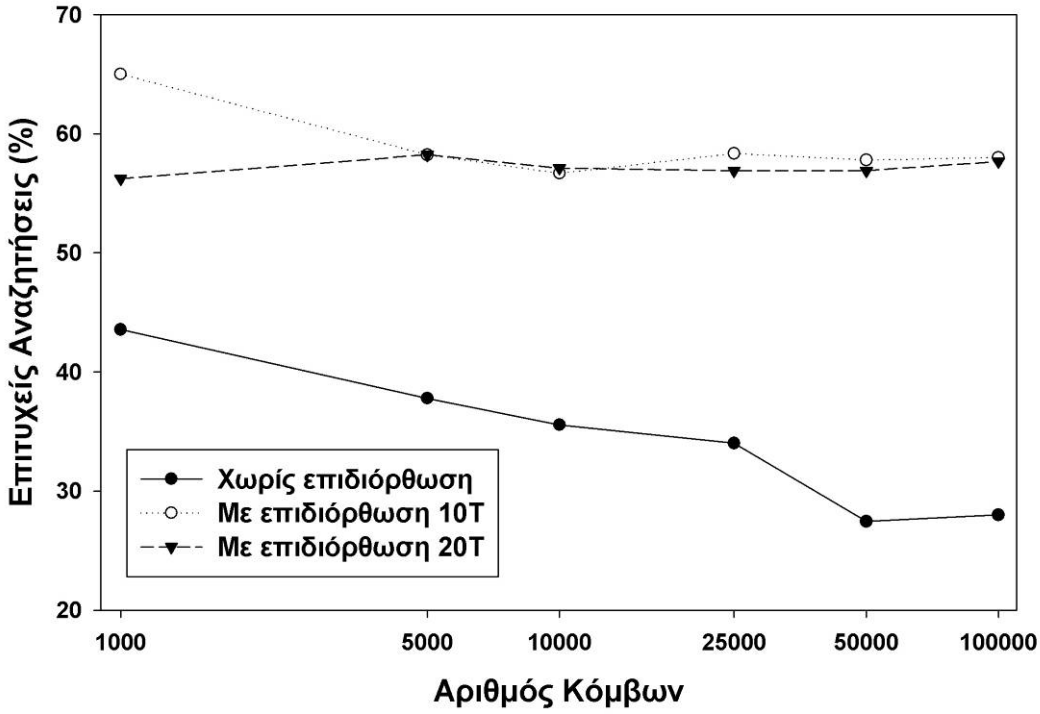
(ε) **(ζ)**

Σχήμα 64. Ποσοστά επιτυχών αναζητήσεων χωρίς επιδιόρθωση και με επιδιόρθωση περιόδου 10T και 20T για διαφορετικούς πληθυσμούς (α) 1.000 , (β) 5.000 , (γ) 10.000 , (δ) 25.000 , (ε) 50.000 και (ζ) 100.000 κόμβων

Για να κατανοήσουμε καλύτερα την επίπτωση του αλγορίθμου επιδιόρθωσης στο σύστημά μας στο Σχήμα 65 έχουμε απεικονίσει το μέσο όρο των ποσοστών επιτυχών αναζητήσεων για διαφορετικές περιπτώσεις του αλγορίθμου επιδιόρθωσης χρησιμοποιώντας (α) το μέσο όρο όλων των πληθυσμών ως προς τα ποσοστά σφαλμάτων και (β) το μέσο όρο όλων των ποσοστών σφαλμάτων ως προς τον πληθυσμό του δικτύου. Από το πρώτο σχήμα παρατηρούμε πως ο αλγόριθμος επιδιόρθωσης βελτιώνει σημαντικά την απόδοση του συστήματος αλλά οι διαφορές μεταξύ 10T και 20T δεν επηρεάζουν σημαντικά την απόδοση, ανεξάρτητα από το ποσοστό σφαλμάτων. Εν συνεχεία, από το δεύτερο σχήμα παρατηρούμαι ότι ο αλγόριθμος επιδιόρθωσης βελτιώνει σημαντικά την απόδοση ανεξάρτητα από τον πληθυσμό του δικτύου. Όπως φαίνεται από το σχήμα, ενώ για μικρό πληθυσμό 1.000 κόμβων ο αλγόριθμος με περίοδο 10T επιφέρει καλύτερη απόδοση από ότι ο αλγόριθμος με περίοδο 20T, η διαφορά αυτή εξαφανίζεται καθώς αυξάνει ο πληθυσμός. Επομένως, συνοψίζοντας όλες τις παραπάνω παρατηρήσεις, μπορούμε να ισχυριστούμε ότι ο αλγόριθμος επιδιόρθωσης βελτιώνει σημαντικά την απόδοση του συστήματός μας ανεξάρτητα από το ποσοστό σφαλμάτων του δικτύου αλλά και από τον πληθυσμό. Επιπλέον, η χαλάρωση της περιόδου επιδιόρθωσης από 10T σε 20T δεν επιφέρει σημαντική μείωση στην απόδοση τους συστήματος.



(α)



(β)

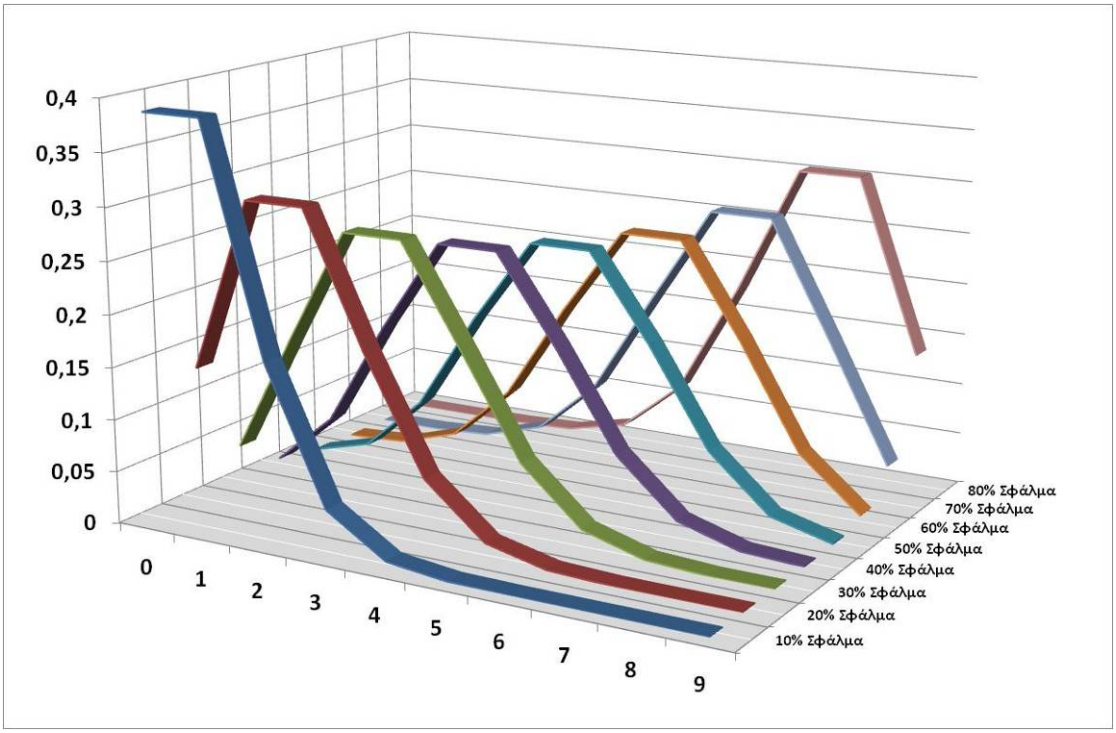
Σχήμα 65. Μέσος όρος ποσοστών επιτυχών αναζητήσεων χωρίς επιδιόρθωση και με επιδιόρθωση περιόδου 10T και 20T ως προς (α) τα ποσοστά σφαλμάτων και (β) τον πληθυσμό κόμβων

Για να κατανοήσουμε καλύτερα τα παραπάνω αποτελέσματα και την ακριβή επίπτωση του αλγορίθμου επιδιόρθωσης αλλά και των σφαλμάτων πιο συγκεκριμένα σε κάθε κόμβο, θα αναλύσουμε το πρόβλημα των σφαλμάτων μέσω της χρήσης πιθανοτήτων. Έστω κόμβος n που βρίσκεται ήδη στο δίκτυο, N ο αριθμός των κόμβων στο δίκτυο και D το σύνολο των κόμβων που βρίσκονται στον πίνακα γειτόνων του n, με $D \subset N$.

Εάν προκαλέσουμε σφάλμα σε m κόμβους, με $m \leq N$, δηλαδή τους αφαιρέσουμε χωρίς να τους αντικαταστήσουμε, τότε επιθυμούμε να υπολογίσουμε την πιθανότητα k από αυτούς τους κόμβους που αφαιρέσαμε να ανήκουν στον πίνακα γειτόνων του n. Η παραπάνω διατύπωση περιγράφει τον αριθμό των επιτυχιών σε μια ακολουθία από επιλογές από έναν περιορισμένο πληθυσμό δίχως αντικατάσταση, δηλαδή μια υπεργεωμετρική κατανομή. Επομένως η πιθανότητα που αναζητούμε δίνεται από τον παρακάτω τύπο:

$$p_k = P(X = k) = Hypergeometric(k; N, m, D) = \frac{\binom{D}{k}\binom{N-D}{m-k}}{\binom{N}{m}}$$

Κατά τη διάρκεια των εξομοιώσεών μας παρατηρήσαμε ότι ένας μέσος όρος 9 κόμβων διατηρείται από κάθε κόμβο στον πίνακα γειτόνων (ανεξάρτητα από τον πληθυσμό). Για αυτόν το λόγο θα εξετάσουμε την περίπτωση πληθυσμού N=100.000 κόμβων και τις πιθανότητες k=0,1,…,9 κόμβοι στον πίνακα γειτόνων να αποτυγχάνουν ταυτόχρονα για ποσοστά σφαλμάτων από 10% έως και 80%, δηλαδή ταυτόχρονη αποτυχία m=10.000, 20.000, …, 80.000 κόμβων. Τα αποτελέσματα της ανάλυσης αυτής δίνονται στο Σχήμα 66 και αναλυτικότερα στον Πίνακας 9. Τα αποτελέσματα αυτά αναπαριστούν την πιθανότητα κάθε κόμβος να μην έχει κανένα, ένα ή μέχρι και όλους τους γείτονες που βρίσκονται στον πίνακα γειτόνων του να παρουσιάζουν σφάλμα, συναρτήσει του συνολικού ποσοστού σφαλμάτων στο δίκτυο. Όπως παρατηρούμε, όταν 50% του συστήματος σφάλει, αυτό συνεπάγεται πως υπάρχει πάνω από 80% πιθανότητα να σφάλουν μεταξύ 3 και 6 κόμβων στον πίνακα γειτόνων κάθε κόμβου στο δίκτυο. Επομένως, οι συνθήκες σφαλμάτων που παρουσιάζουμε στις εξομοιώσεις μας είναι πολύ πιο απαιτητικές από ότι αναμένεται να παρουσιαστούν σε πραγματικές συνθήκες.



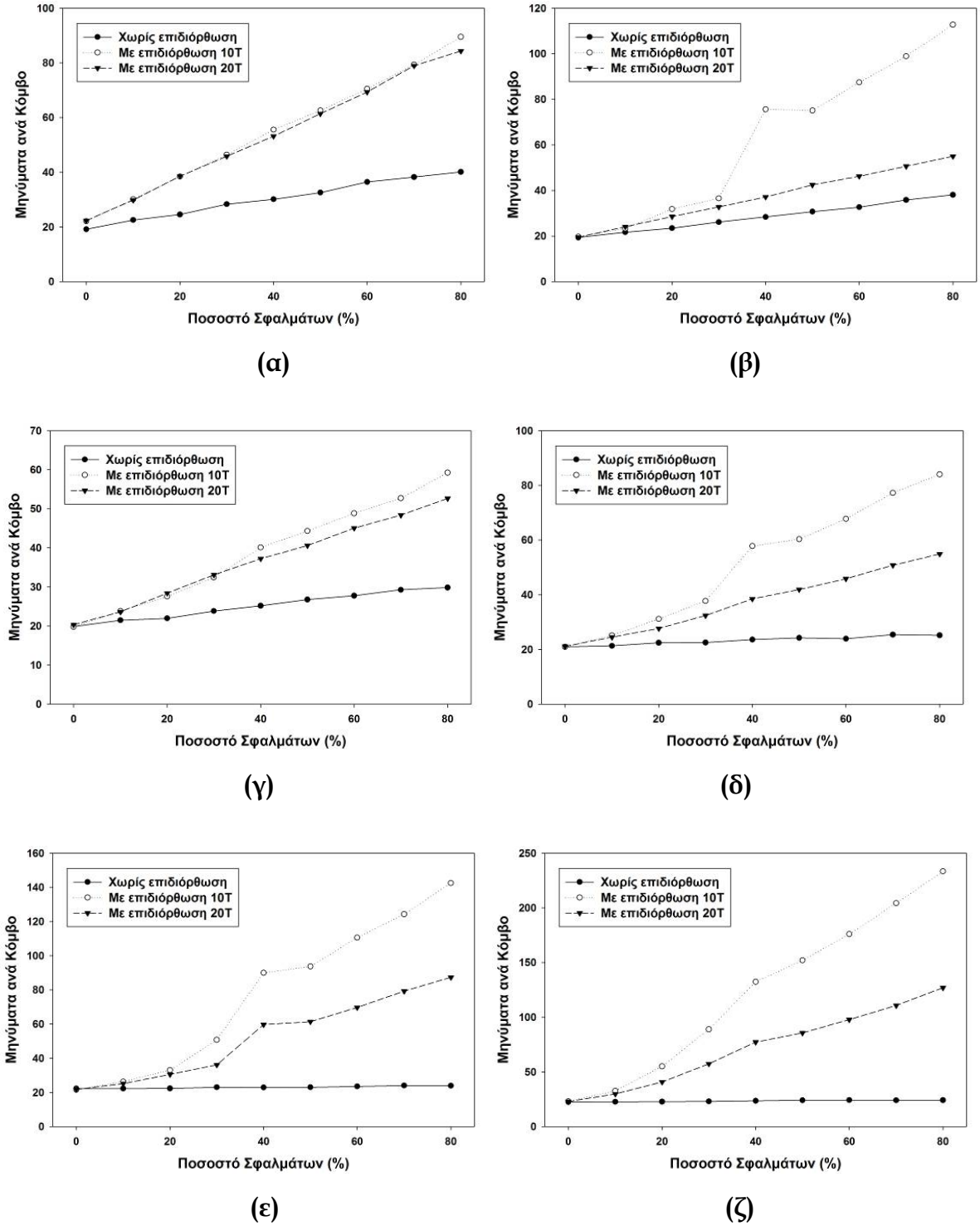
Σχήμα 66. Πιθανότητα ταυτόχρονων σφαλμάτων στον πίνακα γειτόνων ως συνάρτηση του συνολικού ποσοστού σφαλμάτων

	Πιθανότητα σφαλμάτων στον πίνακα γειτόνων									
Σφάλμα	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
10%	0,387	0,387	0,172	0,045	0,007	0,001	0,000	0,000	0,000	0,000
20%	0,134	0,302	0,302	0,176	0,066	0,017	0,003	0,000	0,000	0,000
30%	0,040	0,156	0,267	0,267	0,172	0,074	0,021	0,004	0,000	0,000
40%	0,010	0,060	0,161	0,251	0,251	0,167	0,074	0,021	0,004	0,000
50%	0,002	0,018	0,070	0,164	0,246	0,246	0,164	0,070	0,018	0,002
60%	0,000	0,004	0,021	0,074	0,167	0,251	0,251	0,161	0,060	0,010
70%	0,000	0,000	0,004	0,021	0,074	0,172	0,267	0,267	0,156	0,040
80%	0,000	0,000	0,000	0,003	0,017	0,066	0,176	0,302	0,302	0,134

Πίνακας 9. Πιθανότητα ταυτόχρονων σφαλμάτων στον πίνακα γειτόνων ως συνάρτηση του συνολικού ποσοστού σφαλμάτων

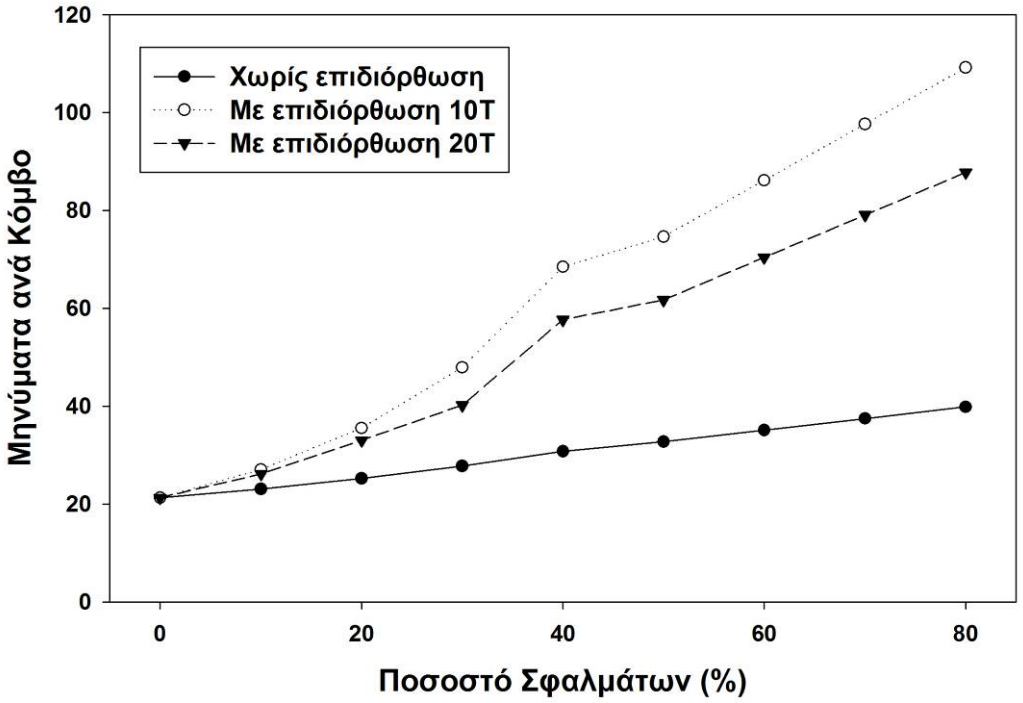
Μια άλλη σημαντική παράμετρος που χαρακτηρίζει την απόδοση του συστήματος είναι ο αριθμός των μηνυμάτων μεταξύ των κόμβων, δηλαδή το παραγόμενο φορτίο κίνησης. Στην περίπτωση του αλγόριθμου επιδιόρθωσης, το φορτίο κίνησης αυξάνει, όπως είναι αναμενόμενο, καθώς ένας αριθμός μηνυμάτων ανταλλάσσετε μεταξύ των γειτονικών κόμβων όταν εντοπιστεί ένα σφάλμα. Τα μηνύματα αυτά είναι κυρίως μηνύματα που απαιτούνται για την ανάκτηση του χαμένου πίνακα γειτόνων του κόμβου που παρουσιάζει το

σφάλμα μέσω των γειτονικών του κόμβων, καθώς και μηνύματα για την αντικατάσταση του από ένα νέο κόμβο και την ενημέρωση των γειτονικών κόμβων για τη νέα δομή.

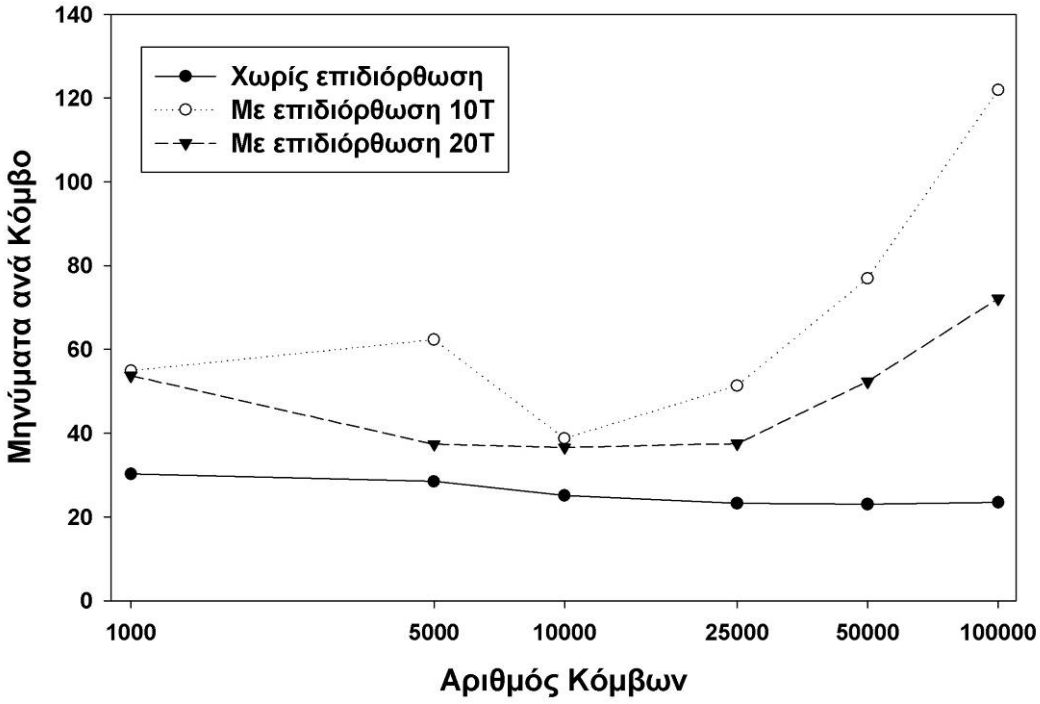


Σχήμα 67. Συνολικά μηνύματα ανά κόμβο χωρίς επιδιόρθωση και με επιδιόρθωση περιόδου 10T και 20T για διαφορετικούς πληθυσμούς (α) 1.000 , (β) 5.000 , (γ) 10.000 , (δ) 25.000 , (ε) 50.000 και (ζ) 100.000 κόμβων

Στο Σχήμα 67 παρουσιάζουμε αναλυτικά το μέσο αριθμό μηνυμάτων ανά κόμβο για διαφορετικούς πληθυσμούς και για τις περιπτώσεις όπου δεν εφαρμόζεται ο αλγόριθμος επιδιόρθωσης, εφαρμόζεται με περίοδο 10Τ και εφαρμόζεται με περίοδο 20Τ. Όπως είναι αναμενόμενο, ο αλγόριθμος επιδιόρθωσης αυξάνει το μέσο αριθμό μηνυμάτων ανά κόμβο ανεξάρτητα από τον πληθυσμό ή την περίοδο επιδιόρθωσης. Από τα σχήματα φαίνεται επίσης ότι όσο μεγαλώνει το ποσοστό σφαλμάτων τόσο περισσότερο αυξάνει και ο φόρτος κίνησης όταν εφαρμόζεται ο αλγόριθμος επιδιόρθωσης σε αντίθεση με την περίπτωση που ο αλγόριθμος επιδιόρθωσης δεν εφαρμόζεται και το συνολικό φορτίο κίνησης παραμένει σταθερό. Και αυτό το χαρακτηριστικό θεωρείται αναμενόμενο καθώς όσο αυξάνουν τα σφάλματα τόσο περισσότερες φορές αναμένεται να εκτελεστεί ο αλγόριθμος επιδιόρθωσης και επομένως να αυξηθεί η κίνηση.



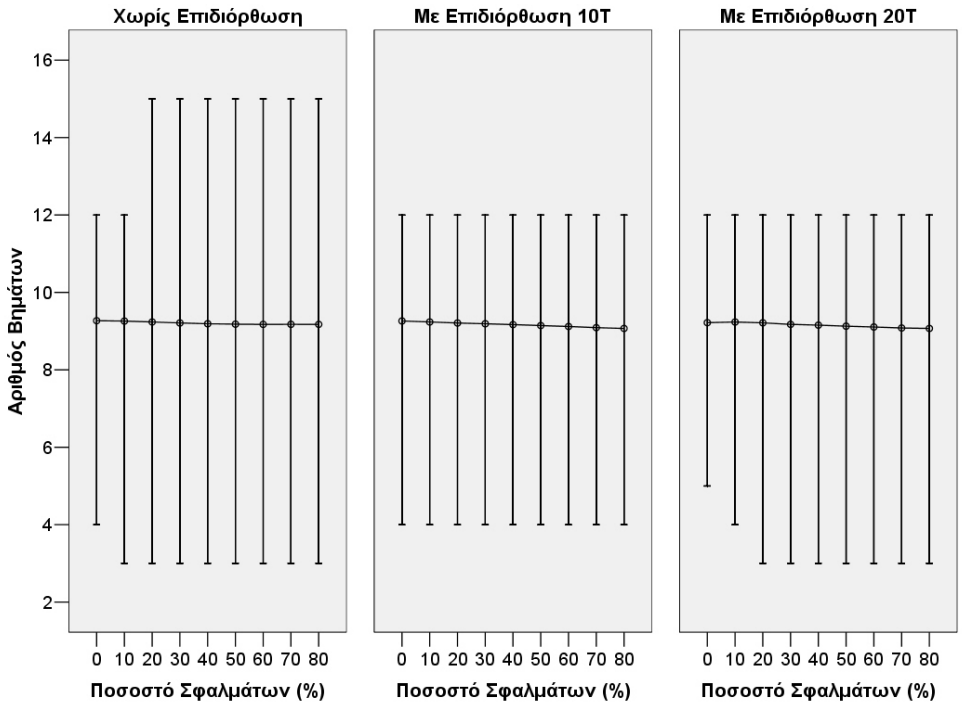
(α)



(β)

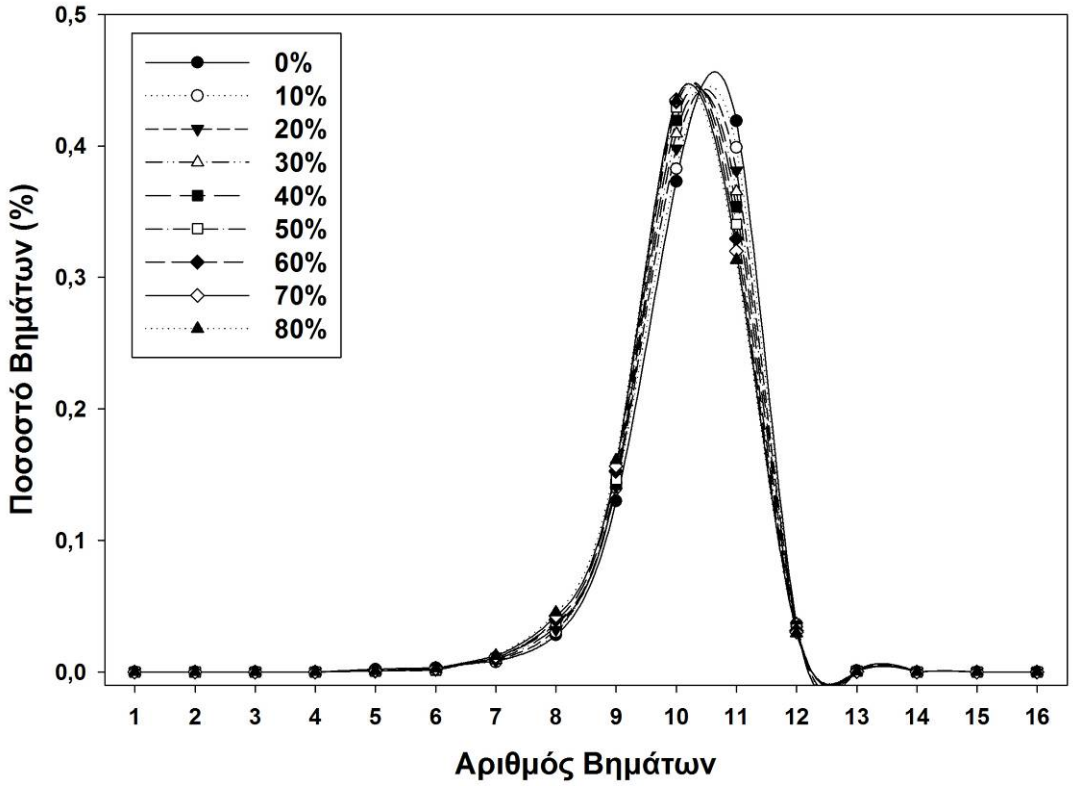
Σχήμα 68. Μέσος όρος μηνυμάτων ανά κόμβο χωρίς επιδιόρθωση και με επιδιόρθωση περιόδου 10T και 20T ως προς (α) τα ποσοστά σφαλμάτων και (β) τον πληθυσμό κόμβων

Οι παραπάνω παρατηρήσεις φαίνονται ακόμη καλύτερα στο Σχήμα 68 όπου παρουσιάζουμε τα αθροιστικά αποτελέσματα για διαφορετικά ποσοστά σφαλμάτων και διαφορετικούς πληθυσμούς κόμβων. Όπως φαίνεται στο πρώτο σχήμα, όσο περισσότερο αυξάνει το ποσοστό σφαλμάτων τόσο περισσότερο αυξάνει και ο μέσος όρος μηνυμάτων. Από το σχήμα παρατηρούμε πως η αύξηση αυτή είναι αρκετά μικρότερη για την περίπτωση επιδιόρθωσης με περίοδο 20Τ, καθώς ο αλγόριθμος καλείται λιγότερες φορές. Με βάση όμως τα αποτελέσματα των επιτυχών αναζητήσεων που αναλύσαμε προηγουμένως, η μικρή διαφορά που παρουσιάζουν οι περιπτώσεις των 10Τ και 20Τ στις επιτυχείς αναζητήσεις και η υπολογίσιμη διαφορά τους στον όγκο κίνησης που παράγουν, μας οδηγούν στο συμπέρασμα ότι η περίπτωση επιδιόρθωσης με περίοδο 20Τ μπορεί να θεωρηθεί ως ο καλύτερος συνδυασμός τόσο για βελτίωση των ποσοστών αναζήτησης όσο και για μη επιφόρτιση του δικτύου με υπερβολικό όγκο κίνησης. Τα ίδια συμπεράσματα επιβεβαιώνονται και από το δεύτερο σχήμα. Πρέπει πάντως να παρατηρήσουμε ότι και στις δύο περιπτώσεις (10Τ και 20Τ), ο αριθμός των μηνυμάτων ανά κόμβο αυξάνει μεν αλλά διατηρείται σε ικανοποιητικά επίπεδα καθώς για την περίπτωση των 100.000 κόμβων αυξάνει από 30 περίπου μηνύματα σε 80 και 120 αντίστοιχα, αριθμός αρκετά μικρός για τόσο μεγάλο δίκτυο.



Σχήμα 69. Μέσος όρος και διακύμανση των βημάτων των επιτυχών αναζητήσεων χωρίς επιδιόρθωση και με επιδιόρθωση περιόδου 10Τ και 20Τ για την περίπτωση των 100.000 κόμβων

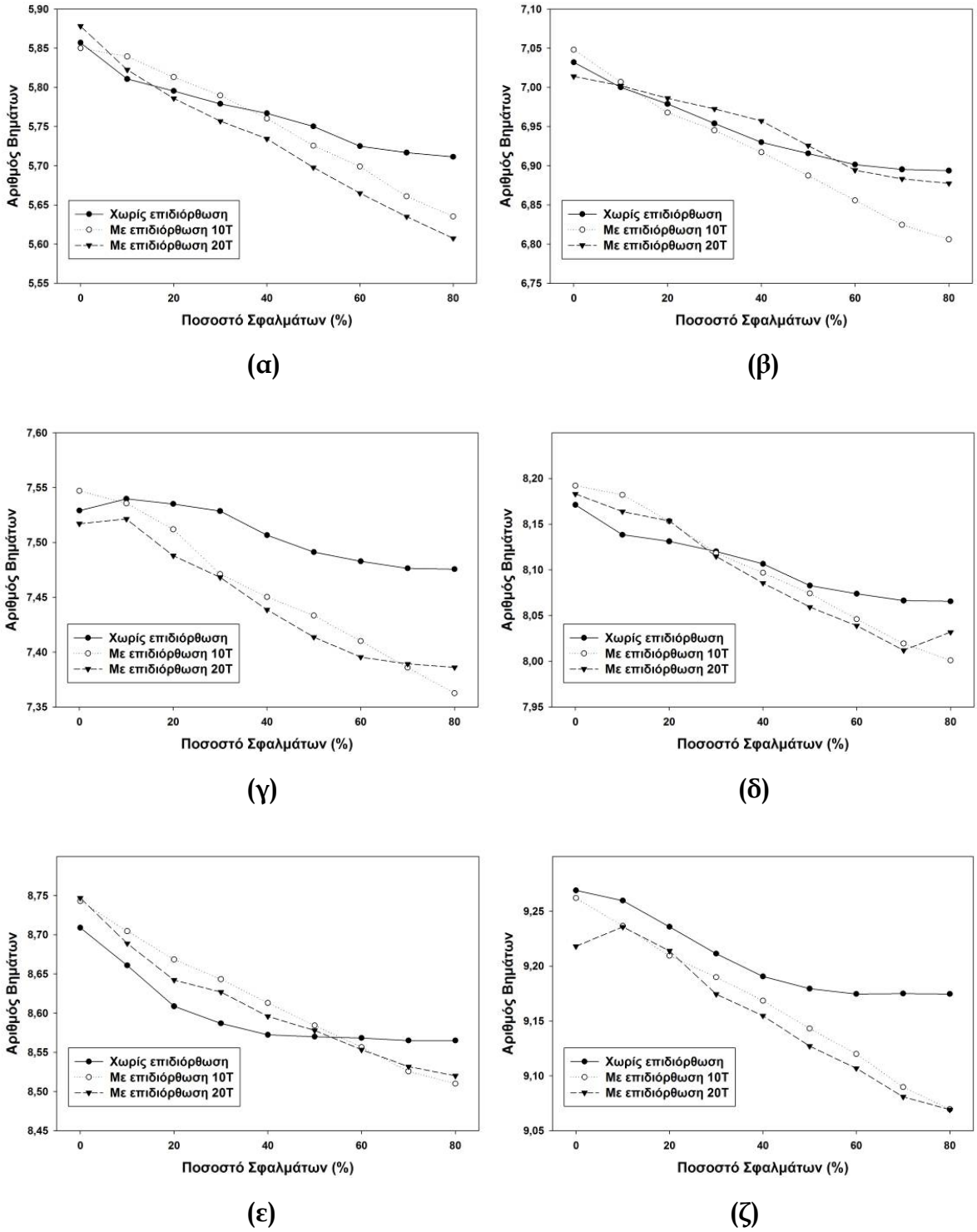
Τέλος, θα εξετάσουμε την επίδραση του αλγόριθμου επιδιόρθωσης στον αριθμό βημάτων των επιτυχών αναζητήσεων. Αρχικά, παρατηρώντας για την περίπτωση των 100.000 κόμβων, από το Σχήμα 69 βλέπουμε πως ο αριθμός των βημάτων δεν αλλάζει σημαντικά, παρά μόνο μειώνεται κάπως η διακύμανση. Στο Σχήμα 70 παρατηρούμε επίσης πως η διακύμανση των βημάτων δεν επηρεάζεται από το ποσοστό σφαλμάτων, όταν στο σύστημα εφαρμόζεται ο αλγόριθμος επιδιόρθωσης. Η συμπεριφορά αυτή θεωρείται σημαντική καθώς αποδεικνύει ότι το σύστημα συνεχίζει να δρομολογεί αναζητήσεις το ίδιο αποδοτικά όσο και πριν την εφαρμογή του αλγόριθμου επιδιόρθωσης. Επομένως, το μόνο αρνητικό του αλγορίθμου είναι η αύξηση του φορτίου κίνησης, η οποία πάντως δεν είναι ιδιαίτερα σημαντική, καθώς όπως είδαμε περιορίζοντας την περίοδο επιδιόρθωσης σε 20T μπορεί να μειωθεί αρκετά χωρίς να μειώσει την απόδοση του συστήματος.



Σχήμα 70. Κατανομή πιθανότητας του μέσου όρου των βημάτων των επιτυχών αναζητήσεων με επιδιόρθωση περιόδου 10T για την περίπτωση των 100.000 κόμβων

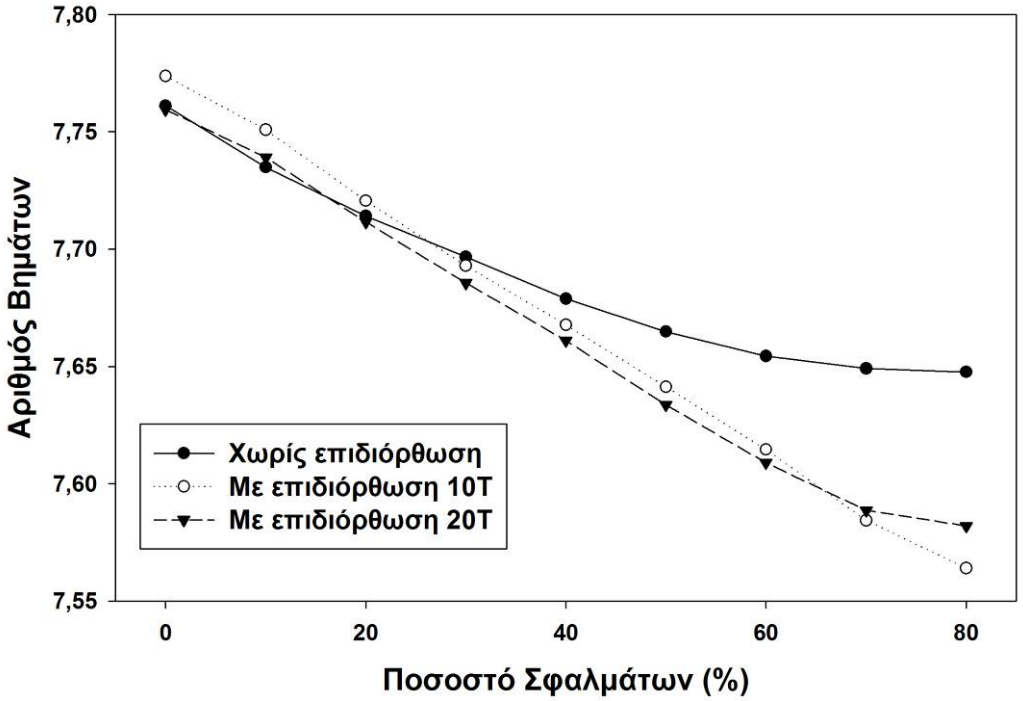
Αν παρατηρήσουμε αναλυτικότερα την επίδραση του αλγορίθμου επιδιόρθωσης στον αριθμό των βημάτων σε σχέση με τα ποσοστά σφαλμάτων και τον πληθυσμό του δικτύου θα δούμε από το Σχήμα 71 πως γενικότερα ο αριθμός των βημάτων μειώνεται ελάχιστα όταν εφαρμόζεται ο αλγόριθμος επιδιόρθωσης και αυξάνουν τα ποσοστά σφαλμάτων. Η μείωση αυτή

κυμαίνεται μεταξύ 0,05% και 4,26% και για αυτόν το λόγο δεν ήταν ιδιαίτερα αντιληπτή στο Σχήμα 69.



Σχήμα 71. Μέσος όρος βημάτων των επιτυχών αναζητήσεων χωρίς επιδιόρθωση και με επιδιόρθωση περιόδου 10Τ και 20Τ για διαφορετικούς πληθυσμούς (α) 1.000 , (β) 5.000 , (γ) 10.000 , (δ) 25.000 , (ε) 50.000 και (ζ) 100.000 κόμβων

Αν μάλιστα αθροίσουμε τα παραπάνω αποτελέσματα ως προς το ποσοστό σφαλμάτων για διαφορετικούς αριθμούς πληθυσμού κόμβων και για τις περιπτώσεις με και χωρίς επιδιόρθωση θα παρατηρήσουμε από το Σχήμα 72 πως ο αριθμός βημάτων μειώνεται σταδιακά καθώς αυξάνουν τα ποσοστά σφαλμάτων. Αυτή η μείωση φτάνει το 1,09% για την περίπτωση της επιδιόρθωσης με περίοδο 10T και το 0,86% για την περίπτωση του 20T. Οι μειώσεις αυτές είναι αρκετά μικρές και ουσιαστικά επιβεβαιώνουν την προηγούμενη διαπίστωσή μας πως ο αλγόριθμος επιδιόρθωσης δεν επηρεάζει τη δρομολόγηση στο δίκτυό μας και απλά επιβαρύνει το σύστημα με ένα επιπρόσθετο φορτίο κίνησης.



Σχήμα 72. Μέσος όρος βημάτων των επιτυχών αναζητήσεων ως συνάρτηση του ποσοστού σφαλμάτων για το μέσο όρο του πληθυσμού κόμβων χωρίς επιδιόρθωση και με επιδιόρθωση περιόδου 10T και 20T

7.3. Σύνοψη - Συμπεράσματα

Στην ενότητα αυτή παρουσιάσαμε τον αλγόριθμο επιδιόρθωσης, ο οποίος επιτρέπει στο σύστημά μας να αναδιοργανώνεται καθώς οι κόμβοι αποχωρούν απότομα. Αρχικά διατυπώσαμε πως κάθε αποτυχία κόμβου μπορεί να μετατραπεί σε αποτυχία παιδιού με βάση την εκτέλεση μιας σειράς ενεργειών, χρησιμοποιώντας τις πληροφορίες που βρίσκονται στον πίνακα γειτόνων. Εν συνεχεία παραθέσαμε τον αλγόριθμο επιδιόρθωσης και

εξηγήσαμε τη λειτουργία του τόσο όταν ο κόμβος παιδί είναι τερματικός κόμβος όσο και όταν έχει παιδιά. Κατόπιν εκτελέσαμε μια σειρά εξομοιώσεων για την ανάλυση της απόδοσης του αλγορίθμου αλλά και του δικτύου συνολικά. Παρατηρήσαμε πως ο αλγόριθμος επιδιόρθωσης επιτυγχάνει να αλλάξει τον τρόπο που μειώνονται οι επιτυχείς αναζητήσεις και από λογαριθμική μείωση πλέον το σύστημά μας παρουσιάζει γραμμική μείωση ως προς το ποσοστό σφαλμάτων. Η βελτίωση αυτή είναι πολύ σημαντική και επιπλέον δεν επιφέρει αλλαγή στην απόδοση της δρομολόγησης καθώς από τα αποτελέσματα των εξομοιώσεων παρατηρήθηκε ότι ο μέσος αριθμός βημάτων παραμένει ουσιαστικά σταθερός. Το μοναδικό αρνητικό του αλγορίθμου επιδιόρθωση, αλλά πλήρως αναμενόμενο, είναι η αύξηση του μέσου αριθμού μηνυμάτων ανά κόμβο, η οποία όμως ακόμα και για την περίπτωση 100.000 κόμβων είναι αρκετά ικανοποιητική καθώς από 30 περίπου μηνύματα αυξάνει σε μόλις 80 με 120.

8 ΣΧΗΜΑΤΑ ΑΝΤΙΓΡΑΦΩΝ, ΕΙΚΟΝΙΚΑ ΔΙΚΤΥΑ ΚΑΙ ΒΕΛΤΙΩΜΕΝΗ ΑΠΟΔΟΣΗ

Έχοντας παρουσιάσει τους βασικούς αλγορίθμους του συστήματός μας καθώς και τον αλγόριθμο επιδιόρθωσης, σε αυτό το κεφάλαιο θα παρουσιάσουμε μια σειρά από επεκτάσεις που βελτιώνουν την απόδοση του συστήματος. Πιο συγκεκριμένα, αρχικά θα περιγράψουμε τρία διαφορετικά σχήματα αντιγράφων που επιτρέπουν την ομαλότερη κατανομή κλειδιών στον χώρο και βελτιώνουν την ανεκτικότητα του συστήματος. Στην συνέχεια εισάγουμε την έννοια των εικονικών δικτύων, όπου κάθε κόμβος είναι μέλος ενός ή περισσότερων εικονικών δικτύων με διαφορετική θέση σε καθένα από αυτά. Τέλος παρουσιάζουμε το συνδυασμό των δύο αυτών επεκτάσεων, ο οποίος επιφέρει σημαντική βελτίωση στην απόδοση του αλγορίθμου μας.

8.1. Σχήματα Αντιγράφων

Η χρήση σχημάτων αντιγράφων έχει αποδειχθεί ότι βελτιώνει την ανεκτικότητα των συστημάτων διανομής περιεχομένου [1]. Κατά τη διάρκεια αυτής της διατριβής υλοποιήσαμε τρία διαφορετικά σχήματα αντιγράφων με βάση τη λογική του βασικού αλγορίθμου δρομολόγησης. Θα πρέπει να επισημάνουμε ότι σε αντίθεση με άλλα σχήματα αντιγράφων που παρουσιάζονται σε διαφορετικά πρωτόκολλα, στα δικά μας δεν γίνεται αντιγραφή του ίδιου του περιεχομένου αλλά μόνο των δημοσιευμένων κλειδιών. Η επιλογή αυτή βασίζεται στον αρχικό μας σκοπό να δημιουργήσουμε ένα πρωτόκολλο που είναι σε θέση να δρομολογεί αναζητήσεις σε πλήρως κατανεμημένα περιβάλλοντα και κάτω από αντίξοες συνθήκες. Αντίθετα, συστήματα όπως το Freenet [1] που αντιγράφουν ταυτόχρονα και τα περιεχόμενα έχουν ως στόχο την εγγύηση εύρεσης σπάνιων (μη δημοφιλών) αρχείων εις βάρος της αποδοτικότητας.

Με τον όρο «σχήμα αντιγράφων» αναφερόμαστε στην διαδικασία αποθήκευσης ενός κλειδιού εγγράφου σε περισσότερους από έναν κόμβο, με

βάση έναν αλγόριθμο ορισμού των κόμβων αυτών. Αυτή η διαδικασία επιτρέπει στο σύστημα να εντοπίζει ένα κλειδί σε περισσότερους από έναν κόμβους, με αποτέλεσμα ακόμη και αν ο αρχικός κόμβος στον οποίο είχε δημοσιευθεί το κλειδί παρουσιάσει σφάλμα να υπάρχει αντίγραφο του κλειδιού σε άλλον κόμβο. Στο σημείο αυτό πρέπει να παρατηρήσουμε πως δε μας απασχολεί εάν ο κόμβος που έχει το αρχικό περιεχόμενο (και όχι το κλειδί) παρουσιάζει σφάλμα, καθώς σκοπός μας είναι η δημιουργία ενός δικτύου όπου θα ισχύει η ιδιότητα, «εάν στο δίκτυο υπάρχει το περιεχόμενο x, τότε οποιοσδήποτε κόμβος μέσα στο δίκτυο μπορεί να εντοπίσει που βρίσκεται το περιεχόμενο x, δηλαδή να εντοπίσει τον κόμβο στον οποίο περιέχεται το κλειδί του x».

Με βάση τα παραπάνω, δημιουργήσαμε τρία διαφορετικά σχήματα αντιγράφων. Κάθε ένα από αυτά επιβάλει τη δημοσίευση κάθε κλειδιού σε περισσότερους από έναν κόμβους, με βάση τον ορισμό του σχήματος. Τα σχήματα αυτά δίνονται παρακάτω, ενώ στο Πίνακας 10 δίδονται και παραδείγματα του κάθε σχήματος για την ευκολότερη κατανόησή τους.

1. Χωρίς Σχήμα Αντιγράφων (ΧΣΑ)

   Η περίπτωση αυτή αναφέρεται στο αρχικό μας σύστημα, όπου κάθε κλειδί δημοσιεύεται μόνο στον κόμβο που εντοπίζεται με βάση τον αλγόριθμο δημοσίευσης.

2. Σχήμα Αντιγράφου Τοπικής Διασποράς (ΣΑ1)

   Σε αυτό το σχήμα, όταν εντοπιστεί ο κόμβος που πρέπει να δημοσιεύσει το κλειδί με βάση τον αλγόριθμο δημοσίευσης, το κλειδί δημοσιεύεται επίσης σε όλους τους κόμβους που βρίσκονται στον πίνακα δρομολόγησης του κόμβου αυτού.

3. Σχήμα Αντιγράφου Αντιστροφής (ΣΑ2)

   Το σχήμα αυτό δημοσιεύει τα κλειδιά σε δύο κόμβους. Ο πρώτος κόμβος είναι αυτός που εντοπίζεται με βάση τον αλγόριθμο δημοσίευσης, ενώ ο δεύτερος είναι αυτός που εντοπίζεται με βάση τον αλγόριθμο δημοσίευσης αλλά με την αναζήτηση να γίνεται με κλειδί που είναι αντίστροφο με το αρχικό.

4. Σχήμα Αντιγράφου Αντιστροφής Τοπικής Διασποράς (ΣΑ3)

   Το σχήμα αυτό είναι συνδυασμός των προηγούμενων δύο σχημάτων και υλοποιεί ένα σχήμα αντιγράφου τοπικής διασποράς τόσο στο αρχικό κλειδί όσο και στο αντίστροφό του.

Κατά την εφαρμογή των παραπάνω σχημάτων μεταβάλλαμε τους βασικούς αλγόριθμους του συστήματος ώστε να υλοποιούν τη λογική τους.

Έτσι, για παράδειγμα, κατά την εφαρμογή του σχήματος αντιγράφου αντιστροφής, ο αλγόριθμος δημοσίευσης καλείται δύο φορές, μία για το κανονικό κλειδί και μια για το αντίστροφό του. Εν συνεχεία, ο αλγόριθμος αναζήτησης αρχικά επιχειρεί την αναζήτηση με βάση το κανονικό κλειδί ενώ στην συνέχεια (αν αυτή αποτύχει) αναζητεί το αντίστροφο του κλειδιού.

Σχήμα Αντιγράφου	Κόμβος που δημοσιεύεται
Χωρίς Σχήμα Αντιγράφων (ΧΣΑ)	<a b c d e f g h>
Σχήμα Αντιγράφου Τοπικής Διασποράς (ΣΑ1)	<a b c d e f g h> όλους τους γείτονες του <a b c d e f g h>
Σχήμα Αντιγράφου Αντιστροφής (ΣΑ2)	<a b c d e f g h> < h g f e d c b a>
Σχήμα Αντιγράφου Αντίστροφης Τοπικής Διασποράς (ΣΑ3)	<a b c d e f g h> όλους τους γείτονες του <a b c d e f g h> < h g f e d c b a> όλους τους γείτονες του < h g f e d c b a>

Πίνακας 10. Σχήματα Αντιγράφων

8.2. Εικονικά Δίκτυα

Η δεύτερη επέκταση που εισαγάγαμε στο δίκτυό μας είναι αυτή των εικονικών δικτύων. Σε αντίθεση με τα σχήματα αντιγράφων, η επέκταση με τη χρήση εικονικών δικτύων προκαλεί την παραγωγή αντιγράφων αναγνωριστικών κόμβων έναντι των αντιγράφων αναγνωριστικών κλειδιών. Όπως περιγράψαμε στις βασικές αρχές του αλγορίθμου μας, κάθε κόμβος αποκτά ένα αναγνωριστικό με βάση τη συνάρτηση κατατεμαχισμού SHA-1. Το αναγνωριστικό αυτό καθορίζει τη θέση του κόμβου στο υπερκείμενο δίκτυο. Η θέση αυτή καθορίζει επίσης τη σύνδεση του κόμβου με τους γείτονές του και επιπλέον το δεσμεύει, αν και όχι αποκλειστικά, με έναν περιορισμένο αριθμό κόμβων, αυτούς που βρίσκονται στον πίνακα γειτόνων του.

Με τη χρήση εικονικών δικτύων αποτρέπουμε αυτόν τον περιορισμό επιτρέποντας σε κάθε κόμβο να συμμετέχει σε έναν αριθμό εικονικών δικτύων, με διαφορετικό αναγνωριστικό σε καθένα από αυτά. Αυτό επιτυγχάνεται με τη χρήση μιας σειράς συναρτήσεων καταχώρησης

αναγνωριστικών, οι οποίες μετασχηματίζουν τα αρχικά αναγνωριστικά σε ένα καινούργιο σύνολο με προκαθορισμένο τρόπο, γνωστό σε όλους τους κόμβους του δικτύου. Αυτό το καινούργιο σύνολο χρησιμοποιείται στην συνέχεια για τη δημιουργία του εικονικού δικτύου και τη δρομολόγηση σε αυτό. Όλοι οι αλγόριθμοι του αρχικού μας δικτύου λειτουργούν στην συνέχεια κανονικά χρησιμοποιώντας όμως σε όλες τους τις ενέργειες το καινούργιο σύνολο. Με τον τρόπο αυτό κάθε κόμβος αποκτά διαφορετική θέση σε κάθε εικονικό δίκτυο και διαφορετικό σύνολο γειτόνων, με αποτέλεσμα εάν δε μπορεί να δρομολογήσει στο ένα δίκτυο να επιχειρεί στο επόμενο. Η διαδικασία αυτή επιφέρει σημαντική βελτίωση στην ανεκτικότητα του δικτύου μας σε απότομες αποχωρήσεις κόμβων (σφάλματα) αλλά έχει ως μειονέκτημα την αύξηση του όγκου κίνησης.

Στο σύστημά μας υλοποιήσαμε 7 διαφορετικές συναρτήσεις καταχώρησης αναγνωριστικών για την εξομοίωση 1, 2, 4 ή 8 δικτύων με τη χρήση 0, 1, 3 ή και των 7 συναρτήσεων για το μετασχηματισμό των αρχικών αναγνωριστικών. Οι συναρτήσεις αυτές ορίζονται στον Πίνακας 11, με την πρώτη συνάρτηση να αποτελεί την αρχική συνάρτηση κατατεμαχισμού SHA-1, η οποία παράγει το αρχικό αναγνωριστικό. Μαζί με κάθε συνάρτηση παρέχουμε και ένα παράδειγμα του αποτελέσματος του μετασχηματισμού για καλύτερη κατανόηση.

Τα εικονικά δίκτυα δημιουργούνται παράλληλα με το αρχικό υπερκείμενο δίκτυο και μεγαλώνουν καθώς νέοι κόμβοι εισέρχονται στο δίκτυο. Οι αλγόριθμοι της εισαγωγής, δημοσίευσης, επιδιόρθωσης και αποχώρησης από το δίκτυο παραμένουν ίδιοι και απλά εκτελούνται παράλληλα και ανεξάρτητα σε κάθε ένα από τα δίκτυα. Ο κάθε κόμβος είναι υπεύθυνος για την ταυτόχρονη δρομολόγηση τόσο στο αρχικό δίκτυο όσο και σε κάθε ένα από τα εικονικά δίκτυα. Η μοναδική διαφορά έγκειται στον μηχανισμό αναζήτησης, ο οποίος εκτελείται στο επόμενο εικονικό δίκτυο μόνο εφόσον έχει αποτύχει στο προηγούμενο. Στο σημείο αυτό τονίζουμε πως τα κλειδιά των εγγράφων που δημοσιοποιούνται δεν αλλάζουν αναγνωριστικό, καθώς οι συναρτήσεις καταχώρησης αναγνωριστικών επηρεάζουν μόνο το χώρο αναγνωριστικών των κόμβων. Χρησιμοποιώντας στην συνεχεία ταυτόχρονα τις επεκτάσεις εικονικών δικτύων και σχημάτων αντιγράφων επιτυγχάνουμε βελτιωμένη απόδοση του συστήματος σε σχέση με την ανεκτικότητά του σε σφάλματα καθώς δημιουργούνται αντίγραφα αναγνωριστικά τόσο στον χώρο των κόμβων όσο και στον χώρο των κλειδιών.

Συνάρτηση Καταχώρησης Αναγνωριστικού	Τρόπος Μετασχηματισμού	Παράδειγμα Αναγνωριστικού
0	Η βασική συνάρτηση κατατεμαχισμού SHA-1	a b c d e f g h
1	Η συνάρτηση αυτή αντιστρέφει το αναγνωριστικό	h g f e d c b a
2	Τα ψηφία του αναγνωριστικού αντιστρέφονται ανά ζεύγος	b a d c f e h g
3	Όλα τα ψηφία αντιστρέφονται ανά ζεύγος και το αποτέλεσμα αντιστρέφεται (συνδυασμός της 2 και μετά 1)	g h e f c d a b
4	Η συνάρτηση 1 εφαρμόζεται στο πρώτο και στο δεύτερο μισό του αναγνωριστικού ανεξάρτητα	d c b a h g f e
5	Το πρώτο μισό του αναγνωριστικού γίνεται δεύτερο (δίχως αντιστροφή)	e f g h a b c d
6	Μια τυχαία αναδιάταξη των ψηφίων του αναγνωριστικού	d a e c g h b f
7	Ίδια με την 6 αλλά με διαφορετική τυχαία αναδιάταξη	c f b a h g d e

Πίνακας 11. Συναρτήσεις καταχώρησης αναγνωριστικών για τη δημιουργία των εικονικών δικτύων.

8.3. Ατομικές Επεκτάσεις - Ανεξάρτητα Αποτελέσματα

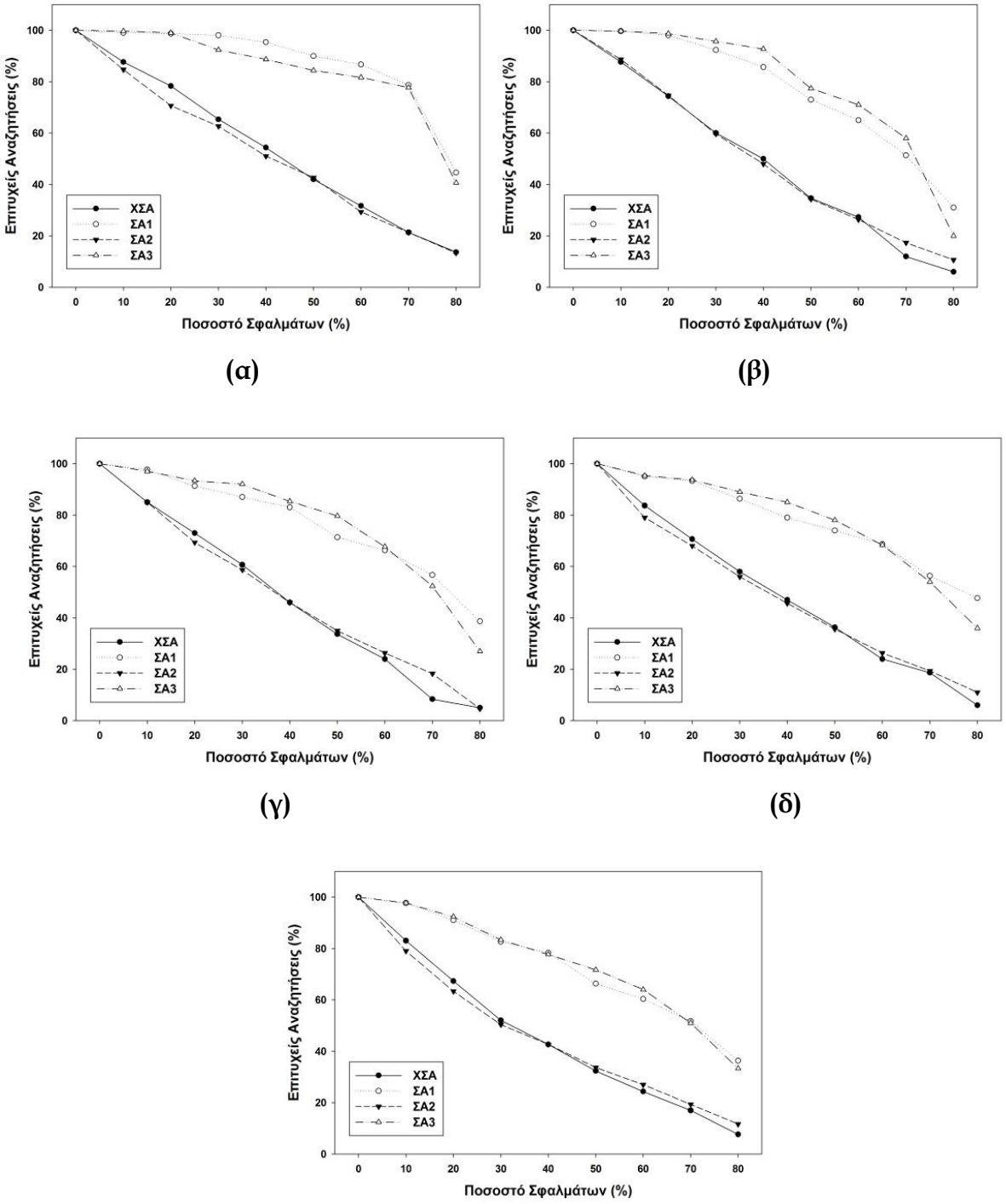
Στην συνέχεια της ενότητας θα παρουσιάσουμε τα αποτελέσματα των εξομοιώσεών μας κατά τη χρήση των επεκτάσεων που παρουσιάσαμε. Αρχικά θα παρουσιάσουμε κάθε επέκταση ατομικά για να δούμε αναλυτικά την επίπτωση που έχει στην απόδοση και ανεκτικότητα του συστήματος. Κατόπιν, στην επόμενη ενότητα, θα παρουσιάσουμε τα αποτελέσματα που καταγράψαμε καθώς εφαρμόσαμε και τις δύο επεκτάσεις ταυτόχρονα.

Κατά τη διάρκεια αυτής της σειράς των εξομοιώσεων ο αριθμός των συνολικών κόμβων των δικτύων περιορίστηκε αρχικά σε 50.000 κόμβους (από 100.000) για την περίπτωση των σχημάτων αντιγράφων και στην συνέχεια σε 25.000 κόμβους για την περίπτωση των εικονικών δικτύων καθώς και τη συνδυασμένη εφαρμογή και των δύο επεκτάσεων. Η μείωση αυτή στο μέγιστο αριθμό εξομοιωμένων κόμβων οφείλεται σε περιορισμούς λόγω έλλειψης αρκετής επεξεργαστικής ισχύς και διαθέσιμης μνήμης των μηχανημάτων που διατίθετο στο εργαστήριο. Παρόλα αυτά, το δείγμα των 50.000 και 25.000 κόμβων αντίστοιχα θεωρείται αρκετά μεγάλο για να εξάγουμε έγκυρα συμπεράσματα.

8.3.1. *Αποτελέσματα Σχημάτων Αντιγράφων*

Αρχικά θα εξετάσουμε την επίδραση των σχημάτων αντιγράφων στην απόδοση του συστήματος. Για το σκοπό αυτό, εξομοιώσαμε διαφορετικά δίκτυα με πληθυσμό από 1.000 έως 50.000 κόμβους και σταδιακά προκαλέσαμε σφάλματα με βήμα 10% επί του συνολικού πληθυσμού έως και 80%.

Στο Σχήμα 73 παρουσιάζονται αναλυτικά τα αποτελέσματα των ποσοστών επιτυχιών για διαφορετικούς πληθυσμούς κόμβους κατά την περίπτωση μη εφαρμογής σχημάτων αντιγράφων (ΧΣΑ), εφαρμογής του σχήματος αντιγράφων τοπικής διασποράς (ΣΑ1), εφαρμογής του σχήματος αντιγράφων αντιστροφής (ΣΑ2) και εφαρμογής σχήματος αντιγράφων αντιστροφής τοπικής διασποράς (ΣΑ3). Όπως φαίνεται καθαρά από τα σχήματα, η εφαρμογή του σχήματος αντιγράφων αντιστροφής βελτιώνει ελάχιστα την ανεκτικότητα του συστήματος. Αντίθετα, τα σχήματα τοπικής διασποράς και αντίστροφης τοπικής διασποράς βελτιώνουν σημαντικά την ανεκτικότητα του συστήματος, χωρίς όμως να παρουσιάζουν σημαντική διαφοροποίηση μεταξύ τους.



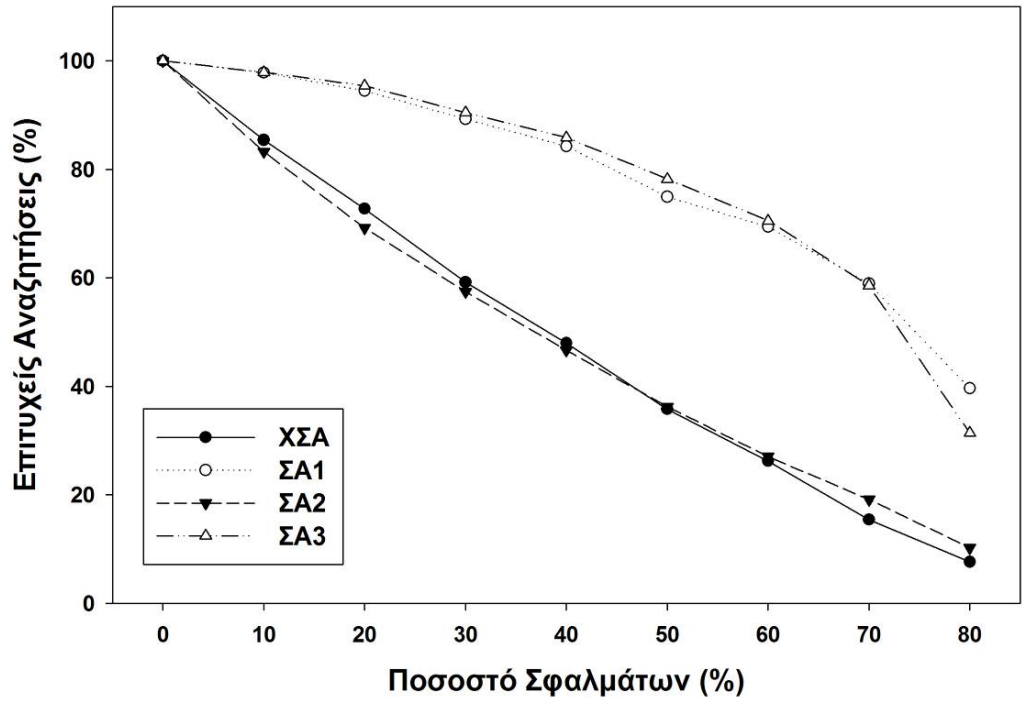
Σχήμα 73. Ποσοστά επιτυχών αναζητήσεων για διαφορετικά σχήματα αντιγράφων για διαφορετικούς πληθυσμούς (α) 1.000 , (β) 5.000 , (γ) 10.000 , (δ) 25.000 και (ε) 50.000 κόμβων

Η επίδραση των σχημάτων αντιγράφων στην ανεκτικότητα του συστήματος φαίνεται καλύτερα στο Σχήμα 74, όπου παρουσιάζονται συγκεντρωτικά αποτελέσματα συναρτήσει του ποσοστού σφαλμάτων και του πληθυσμού των κόμβων αντίστοιχα. Στο σημείο αυτό πρέπει να τονίσουμε

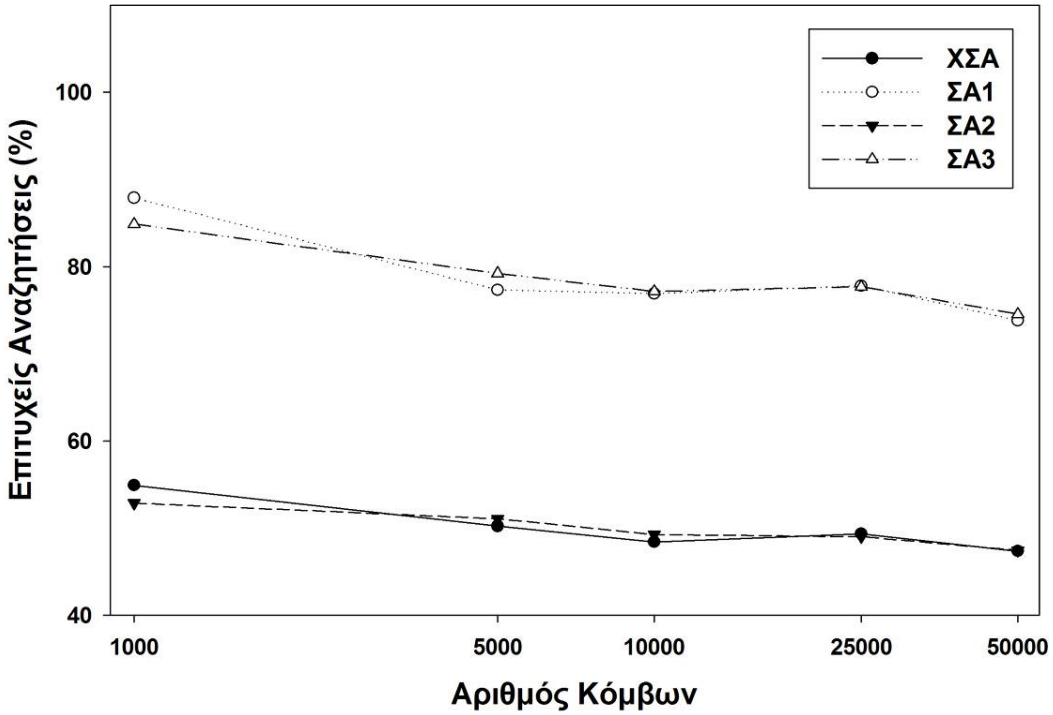
πως σε όλα τα αποτελέσματα που παρουσιάζουμε σε αυτό το κεφάλαιο, όπου δεν γίνεται ρητή αναφορά σε κάποια μεταβλητή, παρουσιάζουμε συγκεντρωτικά αποτελέσματα με βάση το μέσο όρο. Για παράδειγμα, στα αποτελέσματα που απεικονίζονται στο Σχήμα 73 και Σχήμα 74, παρουσιάζονται συγκεντρωτικά αποτελέσματα για τη μεταβλητή περιόδου αλγορίθμου επιδιόρθωσης (μέσος όρος χωρίς επιδιόρθωση, με επιδιόρθωση 10Τ και επιδιόρθωση 20Τ). Με βάση αυτά, τα αποτελέσματα που παρουσιάζουμε δεν αποτελούν τα βέλτιστα δυνατά αλλά είναι ενδεικτικά για την επίδραση των επεκτάσεων μας στο σύστημά μας.

Όπως παρατηρούμε στο Σχήμα 74 (α), η χρήση των σχημάτων αντιγράφων ΣΑ1 και ΣΑ3 βελτιώνουν την ανεκτικότητα του συστήματος έως και ποσοστό που αγγίζει το 400%. Μάλιστα, το ποσοστό αυτό αυξάνεται όσο αυξάνεται το ποσοστό σφαλμάτων. Αντίθετα, η χρήση του σχήματος αντιγράφων ΣΑ2 δε βελτιώνει σημαντικά την απόδοση του συστήματος, που σε μερικές περιπτώσεις είναι και χειρότερη από την αρχική. Εάν και αυτό δε θα έπρεπε να εμφανίζεται, οφείλεται σε στατιστικό λάθος καθώς όλα τα δεδομένα στις εξομοιώσεις μας παράγονται με βάση το μέσο όρο που παράγεται από την εκτέλεση ενός συνόλου εξομοιώσεων με βάση τυχαίες γεννήτριες. Αυτό έχει σαν αποτέλεσμα κάθε εξομοίωση να είναι μοναδική και διαφορετική από κάθε άλλη. Η μικρή αυτή (αρνητική σε μερικές περιπτώσεις) διαφορά καθώς και η μικρή διαφορά μεταξύ της ΣΑ1 και της ΣΑ3 μας οδηγεί στο συμπέρασμα ότι η εφαρμογή της αντιστροφής δε βελτιώνει την ανεκτικότητα του συστήματος. Αντίθετα η τοπική διασπορά είναι αυτή που επιφέρει το επιθυμητό αποτέλεσμα.

Στα ίδια συμπεράσματα οδηγούμαστε παρατηρώντας το Σχήμα 74 (β), όπου βλέπουμε ότι ανεξαρτήτως πληθυσμού τα σχήματα ΣΑ1 και ΣΑ3 παρουσιάζουν βελτίωση που κυμαίνεται γύρω στο 60% σε σχέση με τη μη χρήση σχημάτων αντιγράφων ή τη χρήση του σχήματος ΣΑ1.



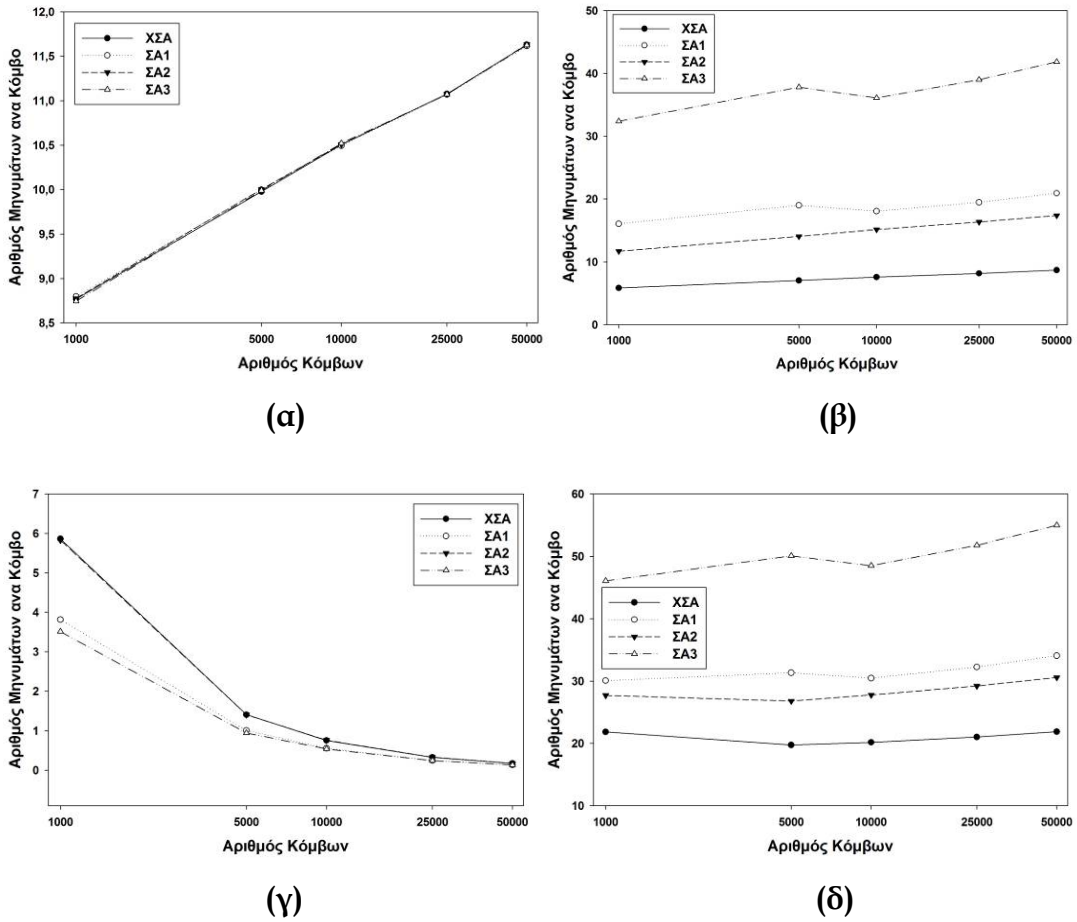
(α)



(β)

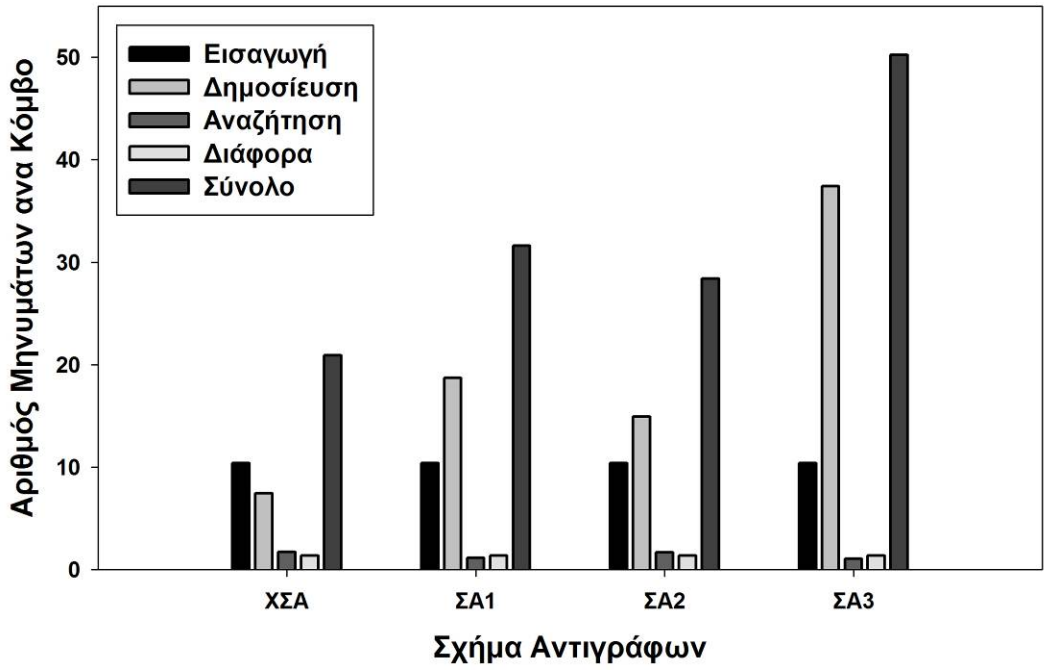
Σχήμα 74. Ποσοστά επιτυχών αναζητήσεων για διαφορετικά σχήματα αντιγράφων συναρτήσει (α) του ποσοστού σφαλμάτων και (β) του πληθυσμού των κόμβων

Στην συνέχεια θα εξετάσουμε την επίπτωση των σχημάτων αντιγράφων στο φόρτο κίνησης του συστήματος. Στο Σχήμα 75 παρουσιάζουμε το μέσο αριθμό μηνυμάτων ανά κόμβο για τις λειτουργίες εισαγωγής, δημοσίευσης και αναζήτησης καθώς και το σύνολο. Όπως φαίνεται από τα σχήματα, τα μηνύματα εισαγωγής παραμένουν ίδια, τα μηνύματα δημοσίευσης αυξάνουν για όλα τα σχήματα ενώ τα μηνύματα αναζήτησης μειώνονται. Όσον αφορά τα μηνύματα εισαγωγής, η μη μεταβολή ήταν αναμενόμενη καθώς δεν επηρεάζεται καθόλου ο αλγόριθμος εισαγωγής. Αντίθετα ο αλγόριθμος δημοσίευσης μεταβλήθηκε καθώς απαιτείτε η δημοσίευση επιπλέων κλειδιών, με λογικό επακόλουθο να αυξηθεί ο αριθμός των μηνυμάτων. Η μείωση στον αριθμό μηνυμάτων αναζήτησης οφείλεται στο ότι υπάρχουν περισσότερες επιτυχείς αναζητήσεις, καθώς οι αποτυχημένες αναζητήσεις απαιτούν το μέγιστο αριθμό βημάτων προτού καταλήξουν σε αποτυχία. Όλα τα παραπάνω έχουν ως αποτέλεσμα την αύξηση του συνολικού αριθμού μηνυμάτων κατά τη χρήση των σχημάτων αντιγράφων, όπως και περιμέναμε.



Σχήμα 75. Μέσος αριθμός μηνυμάτων ανά κόμβο για διαφορετικό αριθμό κόμβων και για τις λειτουργίες (α) εισαγωγής, (β) δημοσίευσης, (γ) αναζήτησης και (δ) συνολική κίνηση

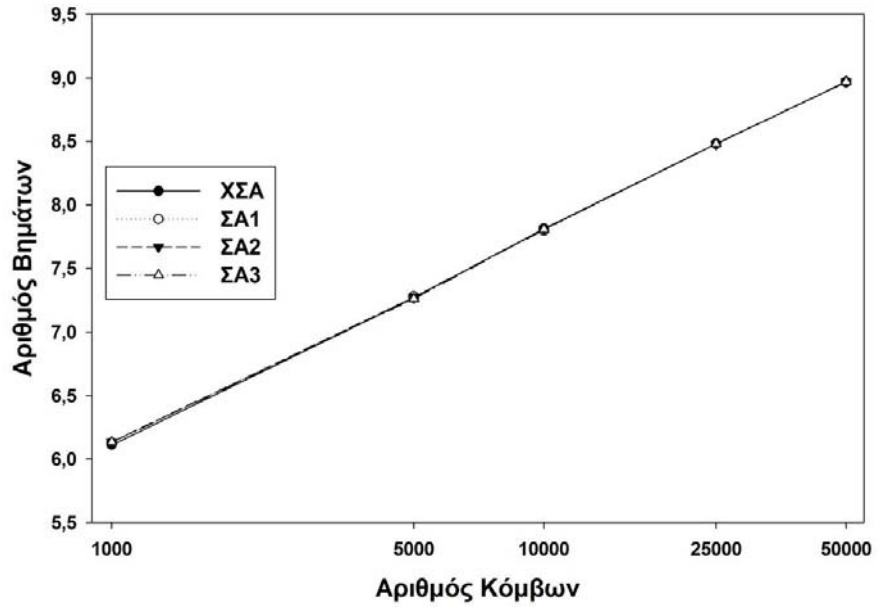
Για να αναλύσουμε καλύτερα την επίδραση των σχημάτων αντιγράφων στον όγκο κίνησης, στο Σχήμα 76 παρουσιάζουμε τα συγκεντρωτικά αποτελέσματα για όλους τους πληθυσμούς. Όπως φαίνεται από το σχήμα, τα σχήματα αντιγράφων ΣΑ1 και ΣΑ2 επιφέρουν συνολική αύξηση 51% και 36% αντίστοιχα ενώ το ΣΑ3 επιφέρει αύξηση 140%. Το συντριπτικό ποσοστό αυτών των αυξήσεων οφείλεται, όπως φαίνεται και αναμενόταν, σε μηνύματα δημοσίευσης. Με βάση και την επίδραση των σχημάτων στην ανεκτικότητα του συστήματος, είναι προφανές πως η χρήση του ΣΑ1 είναι η βέλτιστη, καθώς παρουσιάζει την ίδια ανεκτικότητα σε σφάλματα με το ΣΑ3 αλλά επιφέρει έως και 60% λιγότερο όγκο κίνησης από αυτό. Επίσης πρέπει να τονίσουμε πως το γεγονός ότι το μεγαλύτερο ποσοστό αυτής της αύξησης στον όγκο κίνησης οφείλεται σε μηνύματα δημοσίευσης είναι επίσης θετικό, καθώς η δημοσίευση γίνεται μόνο στην αρχή του συστήματος και στην συνέχεια εκτελούνται κυρίως αναζητήσεις, με αποτέλεσμα αυτό το ποσοστό αύξησης να αναμένεται μικρότερο σε πραγματικές συνθήκες λειτουργίας.



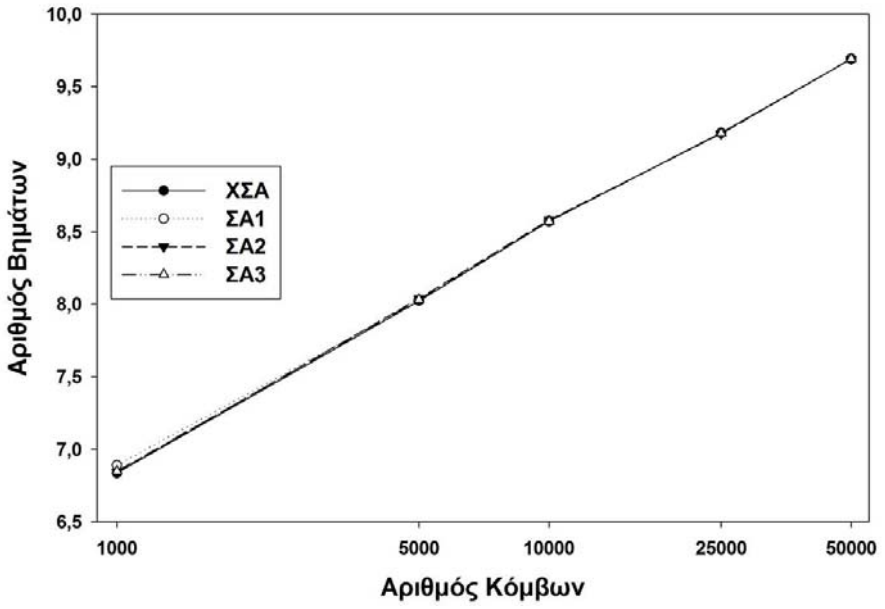
Σχήμα 76. Μέσος αριθμός μηνυμάτων ανά κόμβο για διαφορετικά σχήματα αντιγράφων

Τέλος θα εξετάσουμε την επίπτωση των σχημάτων αντιγράφων στον μέσο αριθμό βημάτων των βασικών αλγορίθμων μας. Για το σκοπό αυτό θα εξετάσουμε ενδεικτικά τους αλγορίθμους εισαγωγής και δημοσίευσης. Όπως φαίνεται από το Σχήμα 77, τόσο η εισαγωγή όσο και η δημοσίευση δεν επηρεάζονται. Ενώ στην πρώτη περίπτωση το αποτέλεσμα ήταν αναμενόμενο,

καθώς όπως παρατηρήσαμε προηγουμένως η εισαγωγή δεν επηρεάζεται από τα σχήματα αντιγράφων, η δεύτερη περίπτωση παρουσιάζει ενδιαφέρον καθώς φαίνεται πως ενώ τα μηνύματα δημοσίευσης αυξάνουν, ο μέσος όρος βημάτων παραμένει σταθερός. Επομένως, η μόνη αρνητική επίπτωση των σχημάτων αντιγράφων στο σύστημά μας είναι αύξηση στον αριθμό μηνυμάτων δημοσίευσης.



(α)

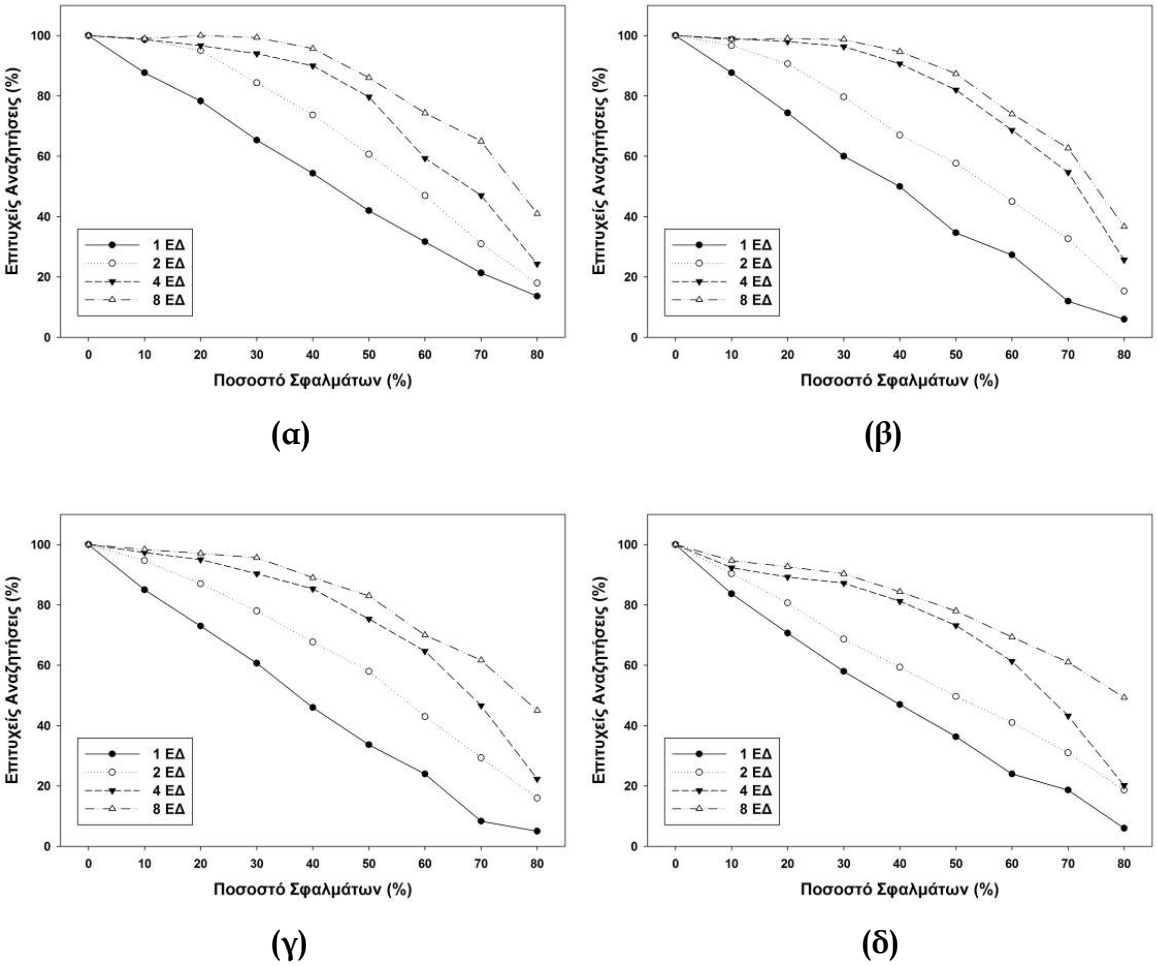


(β)

Σχήμα 77. Μέσος αριθμός βημάτων για διαφορετικούς πληθυσμούς κόμβων για τις λειτουργίες (α) εισαγωγής και (β) δημοσίευσης

8.3.2. Αποτελέσματα Εικονικών δικτύων

Στην συνέχεια των εξομοιώσεών μας θα αξιολογήσουμε την επίδραση των εικονικών δικτύων στο σύστημά μας. Όπως τονίσαμε προηγουμένως, σε αυτήν την ενότητα εφαρμόσαμε μόνο την επέκταση των εικονικών δικτύων δίχως την επέκταση των σχημάτων αντιγράφων. Για τις εξομοιώσεις μας χρησιμοποιήσαμε πληθυσμούς έως και 25.000 κόμβων και πραγματοποιήσαμε σταδιακά σφάλματα με βήμα 10% του πληθυσμού έως και 80% επί του συνόλου. Τονίζουμε επίσης πως σε οποιοδήποτε σημείο δεν γίνεται αναφορά σε κάποια μεταβλητή (πχ. περίοδος αλγορίθμου επιδιόρθωσης) τότε τα αποτελέσματα που απεικονίζονται αφορούν το μέσο όρο αυτής της μεταβλητής.



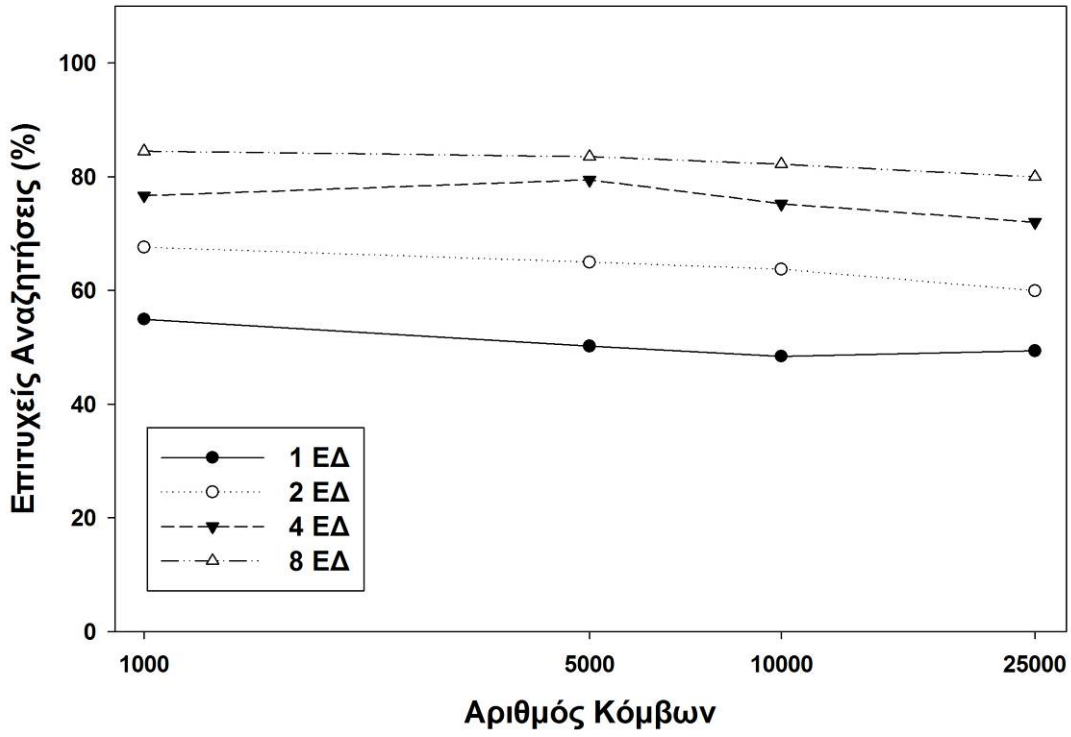
Σχήμα 78. Ποσοστά επιτυχών αναζητήσεων για διαφορετικό αριθμό εικονικών δικτύων (ΕΔ) για διαφορετικούς πληθυσμούς (α) 1.000 , (β) 5.000 , (γ) 10.000 και (δ) 25.000 κόμβων

Αρχικά παρουσιάζουμε την επίδραση των εικονικών δικτύων στην ανεκτικότητα του δικτύου σε σφάλματα. Όπως φαίνεται στο Σχήμα 78, όσο

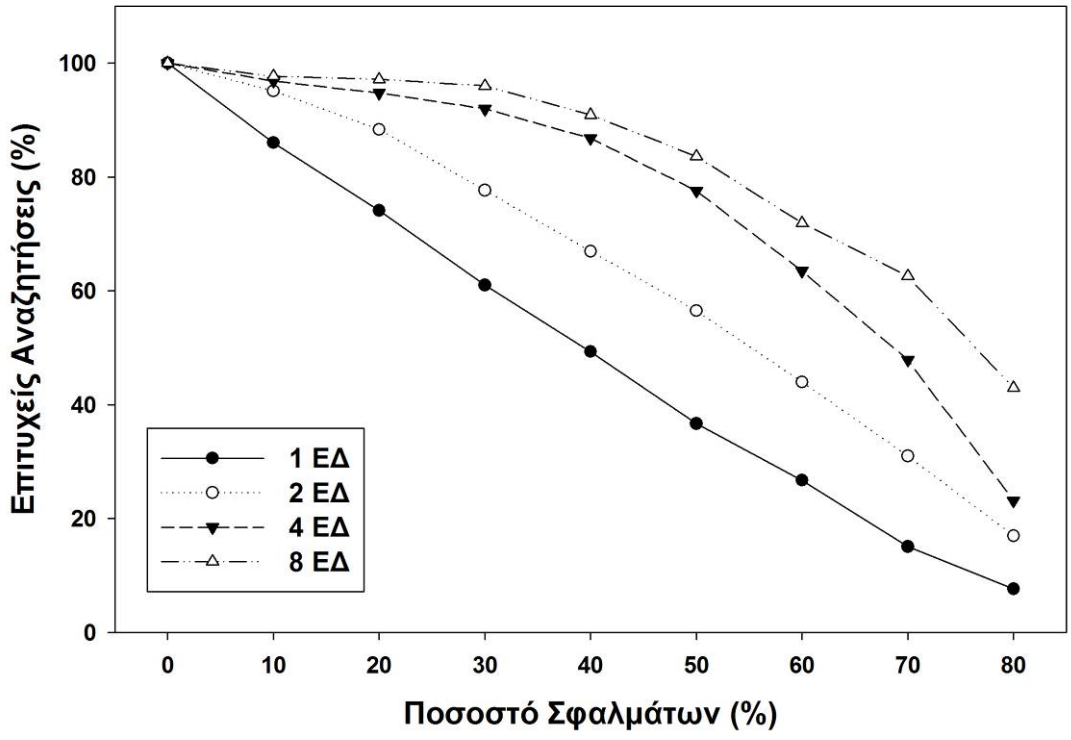
αυξάνει ο αριθμός των εικονικών στοιχείων τόσο αυξάνει και η ανεκτικότητα του συστήματος σε σφάλματα, ανεξάρτητα από τον πληθυσμό του δικτύου. Παρατηρούμε επίσης ότι η αύξηση αυτή μεγαλώνει όσο αυξάνει και το ποσοστό των σφαλμάτων, στοιχείο που αποδεικνύει πως η χρήση των εικονικών δικτύων επιτρέπει στο σύστημά μας να λειτουργεί καλύτερα σε περιβάλλοντα πολλαπλών σφαλμάτων και απότομων αποχωρήσεων. Με βάση το Σχήμα 78 βλέπουμε πως οι διαφορές αυτές φτάνουν έως και το 250% για την περίπτωση των 2 εικονικών δικτύων, το 460% για την περίπτωση των 4 εικονικών δικτύων και το 722% για την περίπτωση των 8.

Για να κατανοήσουμε καλύτερα τα παραπάνω αποτελέσματα στο Σχήμα 79 παρουσιάζουμε το ποσοστό των επιτυχών αναζητήσεων για διαφορετικά εικονικά δίκτυα συναρτήσει του αριθμού κόμβων και του ποσοστού σφαλμάτων. Από το Σχήμα 79 (α) παρατηρούμε πως η χρήση των εικονικών δικτύων βελτιώνει την ανεκτικότητα του συστήματός μας ανεξάρτητα από τον πληθυσμό του δικτύου. Όπως φαίνεται από το διάγραμμα, η βελτίωση αυτή είναι περίπου 26% για τη χρήση 2 εικονικών δικτύων, 50% για τη χρήση 4 εικονικών δικτύων και 63% για τη χρήση 8.

Από το Σχήμα 79 (β) παρατηρούμε τη μέση αύξηση της ανεκτικότητας του δικτύου για διαφορετικά ποσοστά σφαλμάτων, για το μέσο όρο όλων των πληθυσμών. Όπως φαίνεται από το σχήμα, η χρήση των εικονικών δικτύων επιφέρει σημαντική αύξηση στην ανεκτικότητα του δικτύου. Η αύξηση αυτή μάλιστα είναι γραμμική ως προς το ποσοστό των σφαλμάτων και φθάνει έως και το 122% για την περίπτωση των 2 εικονικών δικτύων, το 218% για την περίπτωση των 4 εικονικών δικτύων και το 461% για την περίπτωση των 8, με το μέσο όρο της αύξησης να κυμαίνεται στο 49%, 93% και 140% αντίστοιχα. Οι τιμές αυτές είναι ενδεικτικές για να κατανοήσουμε ότι η χρήση ακόμη και 2 ή 4 εικονικών δικτύων είναι αρκετή για να βελτιώσει σημαντικά την ανεκτικότητα στο υπερκείμενο δίκτυό μας. Όπως έχουμε αναφέρει και προηγουμένως, τα αποτελέσματα θεωρούν το μέσο όρο όλων των μεταβλητών που δεν παρουσιάζονται, επομένως η πραγματική βελτίωση μπορεί να είναι ακόμα μεγαλύτερη από αυτήν που απεικονίζεται. Παρόλα αυτά, αυτό που μας ενδιαφέρει ουσιαστικά δεν είναι τόσο η ποσοτική βελτίωση (καθώς παρουσιάζουμε μέσους όρους και όχι βέλτιστα) αλλά η ποιοτική βελτίωση και γενικότερα η συμπεριφορά και επίδραση που έχουν οι επεκτάσεις που παρουσιάσαμε στο αρχικό δίκτυο.



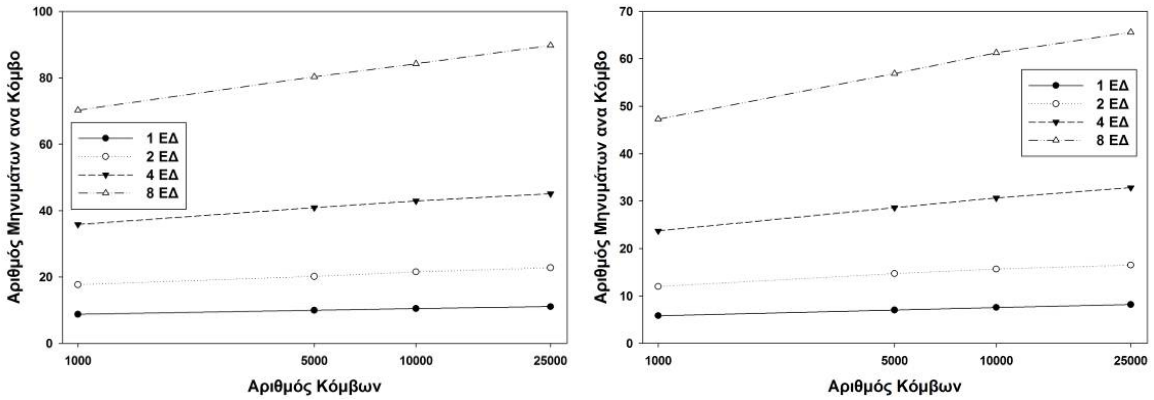
(α)



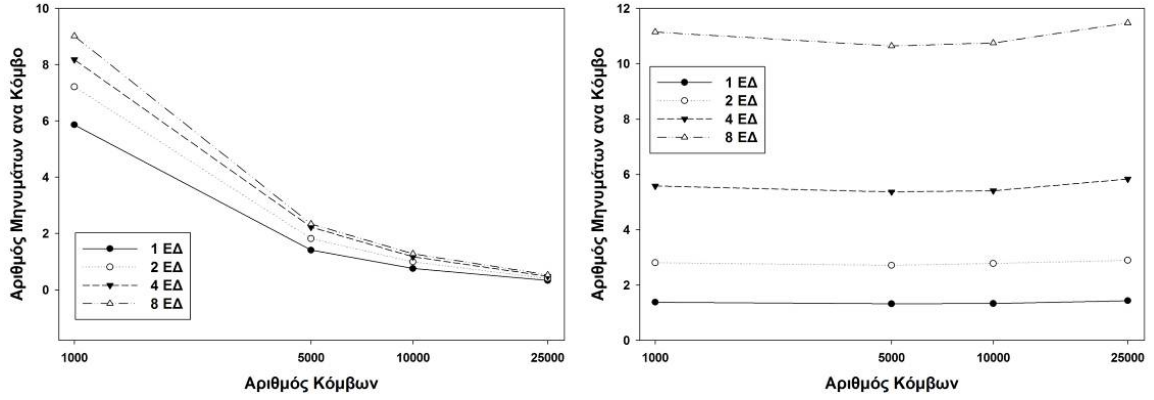
(β)

Σχήμα 79. Ποσοστά επιτυχών αναζητήσεων για διαφορετικό αριθμό εικονικών δικτύων συναρτήσει του (α) πληθυσμού κόμβων και (β) ποσοστό σφαλμάτων

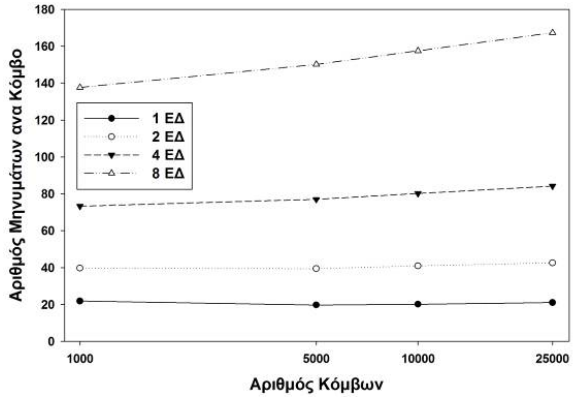
Στην συνέχεια θα εξετάσουμε την επίπτωση των εικονικών δικτύων στο φόρτο κίνησης του δικτύου. Στο Σχήμα 80 απεικονίζεται ο μέσος αριθμός μηνυμάτων ανά κόμβο για διαφορετικό αριθμό κόμβων με βάση το είδος των μηνυμάτων και πιο συγκεκριμένα για τα μηνύματα εισαγωγής, δημοσίευσης, αναζήτησης, διάφορα άλλα μηνύματα και το σύνολο. Τα γραφήματα συμφωνούν με την αναμενόμενη επιρροή με βάση τη λογική των εικονικών δικτύων. Καθώς κάθε εικονικό δίκτυο αποτελεί στην ουσία ένα ξεχωριστό δίκτυο, για τις περισσότερες λειτουργίες ο αριθμό των μηνυμάτων είναι πολλαπλάσιο του αριθμού των εικονικών δικτύων. Έτσι για 2 εικονικά δίκτυα έχουμε διπλασιασμό, για 4 τετραπλασιασμό και για 8 οκταπλασιασμό. Η μόνη διαφοροποίηση εντοπίζεται στα μηνύματα αναζήτησης τα οποία είναι απλώς οριακά περισσότερα και η διαφορά αυτή μειώνεται καθώς αυξάνει ο αριθμός των εικονικών δικτύων. Αυτό είναι επίσης αναμενόμενο, καθώς ο αλγόριθμος αναζήτησης καλείται στα εικονικά δίκτυα σειριακά και μόνο εφόσον δεν υπάρχει επιτυχής αναζήτηση στα προηγούμενα. Έτσι, εάν για παράδειγμα ο αλγόριθμος αποτύχει στο βασικό δίκτυο, τότε και μόνο τότε εκτελείται στο δεύτερο (εικονικό) δίκτυο. Εάν επιτύχει σε αυτό τότε τερματίζεται, ειδάλλως συνεχίζει στο επόμενο.



(α) **(β)**



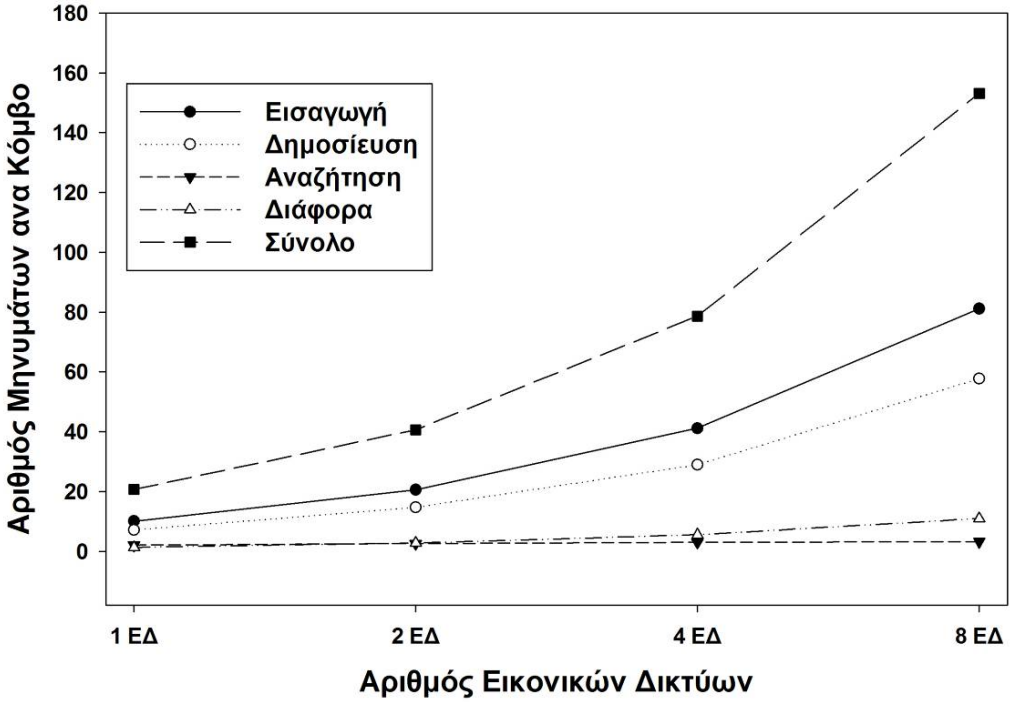
(γ) **(δ)**



(ε)

Σχήμα 80. Μέσος αριθμός μηνυμάτων ανά κόμβο για διαφορετικό αριθμό εικονικών δικτύων συναρτήσει του πληθυσμού των κόμβων για τις λειτουργίες (α) εισαγωγής , (β) δημοσίευσης , (γ) αναζήτησης , (δ) διάφορα άλλα μηνύματα και (ε) το σύνολο

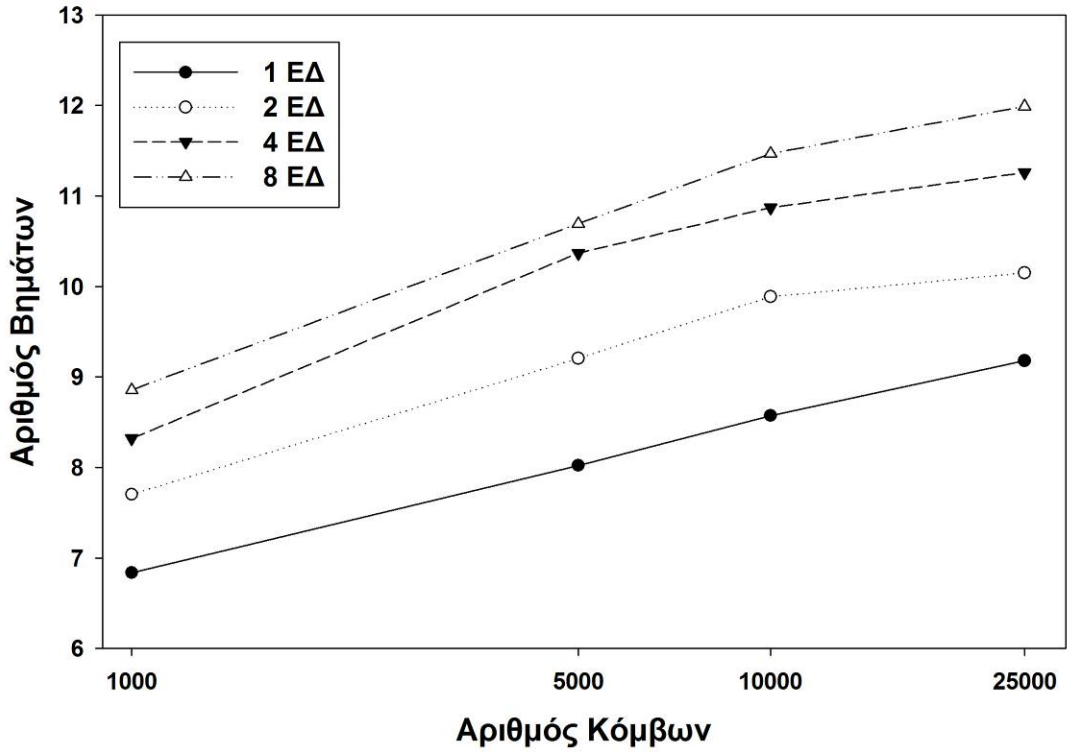
Η παραπάνω διαπιστώσεις φαίνονται καλύτερα στο Σχήμα 81, όπου βλέπουμε τη γραμμική αύξηση του συνόλου σε σχέση με τον αριθμό των εικονικών δικτύων, ενώ τα μηνύματα αναζήτησης παραμένουν σταθερά. Επομένως, η χρήση των εικονικών δικτύων συνιστάται ακόμα περισσότερο σε δίκτυα όπου οι αναζητήσεις είναι πιο συχνές από ότι οι εισαγωγές και δημοσιεύσεις, καθώς δεν επηρεάζουν καθόλου το φόρτο κίνησης.



Σχήμα 81. Αριθμός μηνυμάτων ανά κόμβο συναρτήσει του αριθμού εικονικών δικτύων για τις διαφορετικές λειτουργίες και το σύνολο

Τέλος, στο Σχήμα 82 εξετάσαμε την επίπτωση των εικονικών δικτύων στον μέσο αριθμό βημάτων για επιτυχείς αναζητήσεις. Από το διάγραμμα φαίνεται ότι ο αριθμός αυτός αυξάνει καθώς αυξάνει ο αριθμός των εικονικών δικτύων αλλά η αύξηση είναι αρκετά μικρή, καθώς η διαφορά μεταξύ 1 και 8 εικονικών δικτύων είναι μόλις 2 βήματα, ανεξάρτητα από τον πληθυσμό του δικτύου.

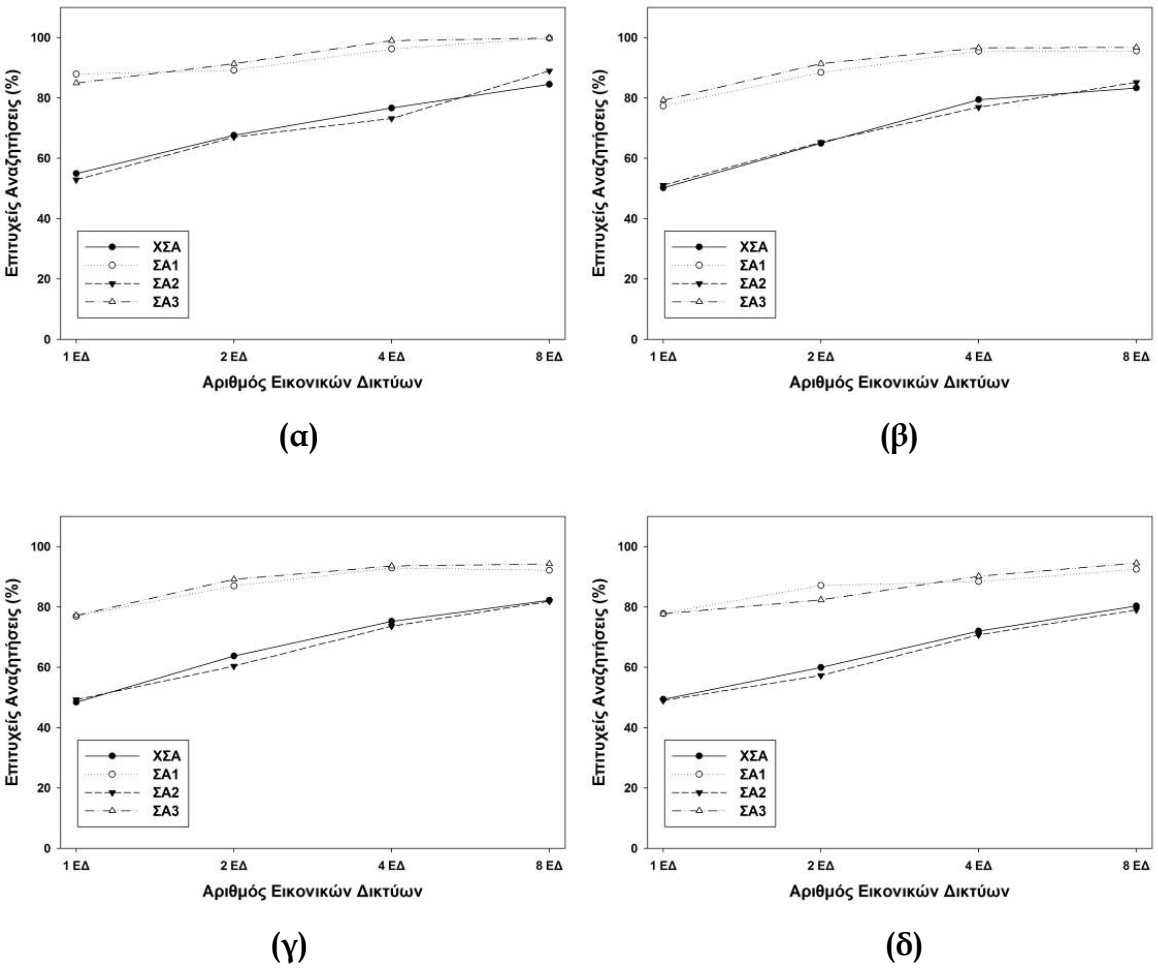
Επομένως, όπως και στην περίπτωση των σχημάτων αντιγράφων, τα εικονικά δίκτυα βελτιώνουν σημαντικά την ανεκτικότητα του συστήματος, με μόνη πρακτικά αρνητική επίπτωση την αύξηση του όγκου κίνησης λόγω της αύξησης του αριθμού μηνυμάτων. Χρειάζεται επομένως σωστός σχεδιασμός κατά την εφαρμογή αυτών των επεκτάσεων ώστε το σύστημα να είναι αρκετά ανεκτικό σε σφάλματα αλλά και ο όγκος κίνησης να παραμένει σε ικανοποιητικά επίπεδα. Αυτό θα εξεταστεί λεπτομερέστερα στη επόμενη ενότητα όπου θα αναλύσουμε τη βελτιωμένη απόδοση του συστήματος κατά την εφαρμογή και των δύο επεκτάσεων ταυτόχρονα.



Σχήμα 82. Μέσος αριθμός βημάτων επιτυχών αναζητήσεων για διαφορετικό αριθμό εικονικών δικτύων ως προς το πληθυσμό του δικτύου

8.4. Συνδυασμός Επεκτάσεων - Βελτιωμένα Αποτελέσματα

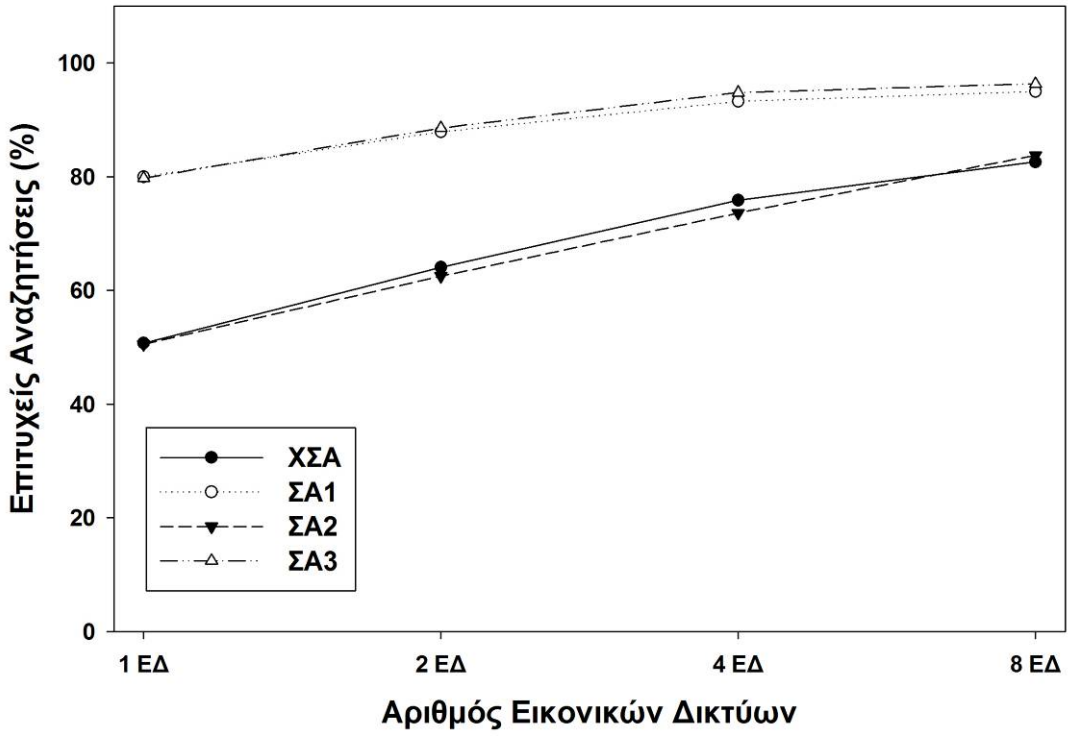
Στην συνέχεια αυτής της ενότητας θα εξετάσουμε τη συμπεριφορά του συστήματός μας κατά την εφαρμογή και των δύο επεκτάσεων ταυτόχρονα. Ο συνδυασμός αυτός, όπως θα δούμε στην συνέχεια μέσα από μια σειρά αποτελεσμάτων, επιφέρει βελτιωμένη απόδοση του συστήματος με βάση την ανεκτικότητα σε σφάλματα αλλά προκαλεί και αυξημένη κίνηση, οπότε απαιτείται συνετή χρήση των επεκτάσεων ώστε να ευρεθεί ο καλύτερος συνδυασμός. Στα αποτελέσματα που θα παρουσιάσουμε εξομοιώσαμε μια σειρά από δίκτυα όπου ο πληθυσμός κυμαίνεται από 1.000 έως 25.000 κόμβους ενώ στο σύστημα εφαρμόζονται τα τρία διαφορετικά σχήματα αντιγράφων και έως και 8 εικονικά δίκτυα. Σε όλα τα σχήματα, όπου δεν αναφέρεται μια μεταβλητή, όπως για παράδειγμα η περίοδος του αλγορίθμου επιδιόρθωσης, τα αποτελέσματα παρουσιάζουν το μέσο όρο.



Σχήμα 83. Ποσοστά επιτυχών αναζητήσεων για διαφορετικά εικονικά σχήματα συναρτήσει του αριθμού εικονικών δικτύων για διαφορετικούς πληθυσμούς (α) 1.000 , (β) 5.000 , (γ) 10.000 και (δ) 25.000 κόμβων

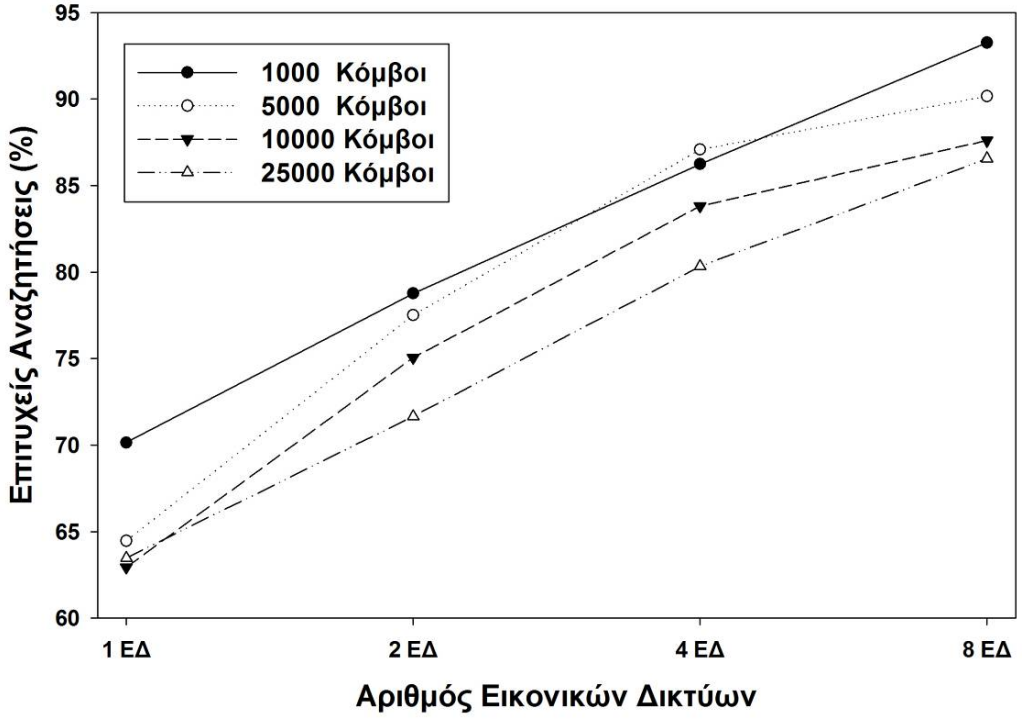
Τα αποτελέσματα που θα παρουσιάσουμε εξετάζουν τα ποσοστά επιτυχών αναζητήσεων και το μέσο αριθμό μηνυμάτων ανά κόμβο καθώς μεταβάλλεται ο πληθυσμός του δικτύου και προκαλούνται σφάλματα κόμβων με βήματα 10% επί του συνόλου του πληθυσμού έως και 80%. Ο αριθμός των βημάτων των διάφορων βασικών λειτουργιών δεν εξετάζεται καθώς όπως είδαμε στις προηγούμενες ενότητες, δεν επηρεάζεται σημαντικά από τις επεκτάσεις (κάτι που επαληθεύτηκε και από τις εξομοιώσεις μας κατά τη συνδυαστική χρήση των επεκτάσεων αλλά παραλείπεται η παρουσίαση αυτών των αποτελεσμάτων καθώς απλά επιβεβαιώνουν τα αποτελέσματα της προηγούμενης ενότητας).

Στο Σχήμα 83 φαίνεται πως η ταυτόχρονη χρήση εικονικών δικτύων και σχημάτων αντιγράφων βελτιώνει την ανεκτικότητα του δικτύου. Όπως φαίνεται, η χρήση των σχημάτων αντιγράφων 1 και 3 βελτιώνει σημαντικά την ανεκτικότητα ανεξάρτητα από τον πληθυσμό του δικτύου. Η βελτίωση αυτή μειώνεται καθώς αυξάνει ο αριθμός των εικονικών δικτύων, καθώς από 57% για 1 δίκτυο μειώνεται σε 37% για 2 εικονικά δίκτυα, σε 23% για 4 και σε 15% για 8. Η αύξηση βέβαια των εικονικών δικτύων επιφέρει αύξηση στην ανεκτικότητα, οπότε παρά τη μείωση αυτή στον ρυθμό βελτίωσης, η συνολική ανεκτικότητα βελτιώνεται σημαντικά καθώς αυξάνονται τα εικονικά δίκτυα.

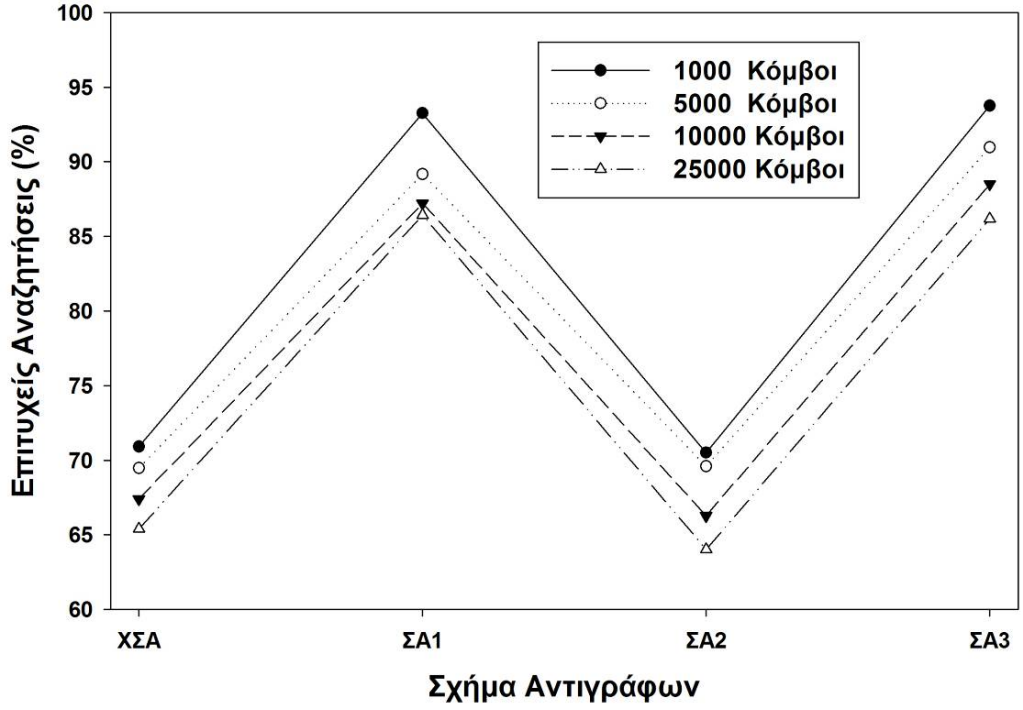


Σχήμα 84. Μέσος όρος ποσοστών επιτυχών αναζητήσεων για διαφορετικά σχήματα αντιγράφων και αριθμό εικονικών δικτύων

Στο Σχήμα 84 απεικονίζεται ο μέσος όρος των επιτυχών αναζητήσεων για διαφορετικά σχήματα αντιγράφων και εικονικά δίκτυα με βάση το μέσο όρο των διαφορετικών πληθυσμών. Όπως φαίνεται, η ταυτόχρονη χρήση των δύο επεκτάσεων βελτιώνει σημαντικά την ανεκτικότητα και από 50% φτάνει έως και 96%. Όπως φαίνεται όμως, η χρήση 4 εικονικών δικτύων είναι αρκετή για να προσφέρει στο σύστημα σχεδόν μέγιστη ανεκτικότητα, καθώς η αύξηση των εικονικών δικτύων σε 8 αυξάνει την ανεκτικότητα λιγότερο από 2%. Στα αποτελέσματα αυτά παρουσιάζουμε το μέσο όρο για σφάλματα από 0% έως 80% και για επιδιόρθωση με περίοδο 10Τ, 20Τ και χωρίς επιδιόρθωση. Επομένως, η βέλτιστη απόδοση του δικτύου είναι μεγαλύτερη από αυτήν που παρουσιάζεται στο σχεδιάγραμμα.



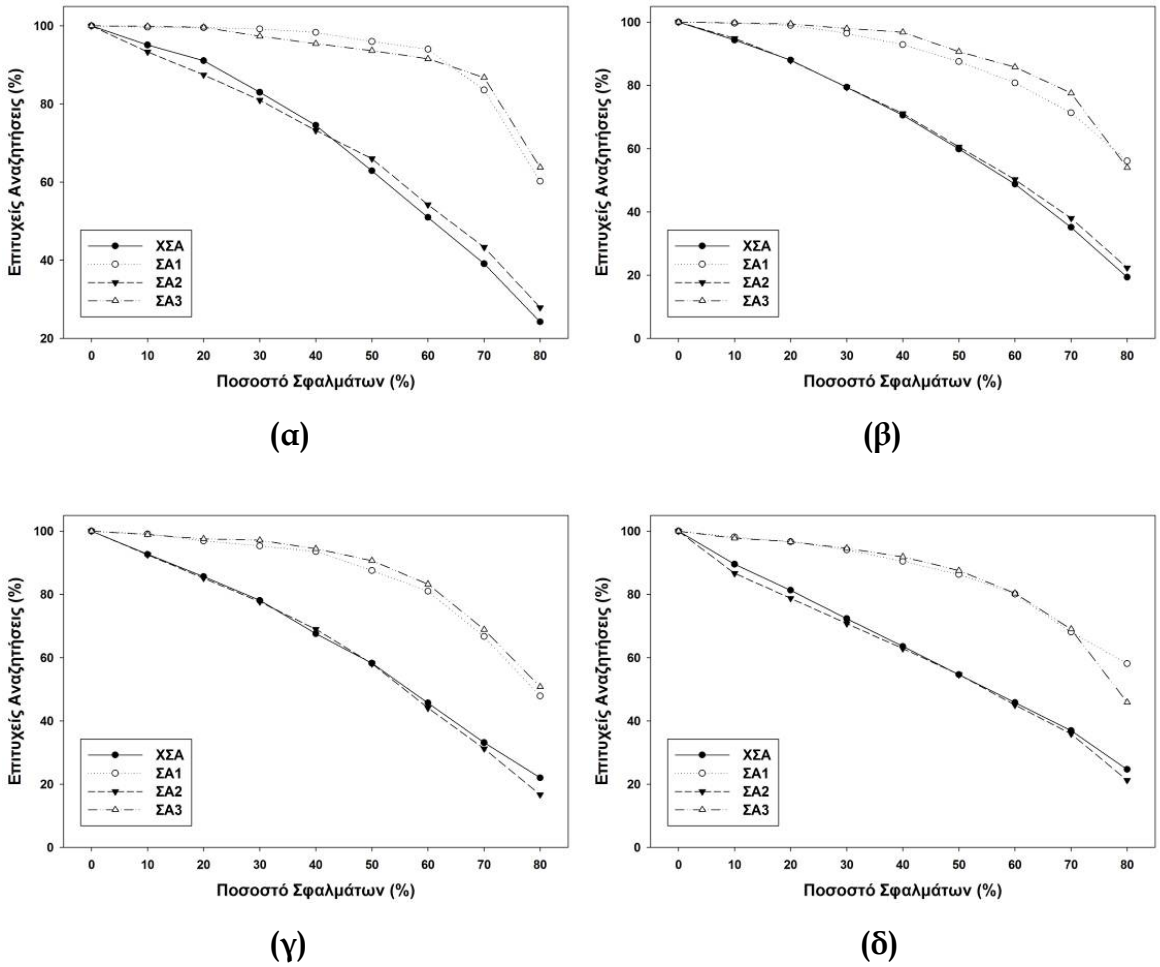
(α)



(β)

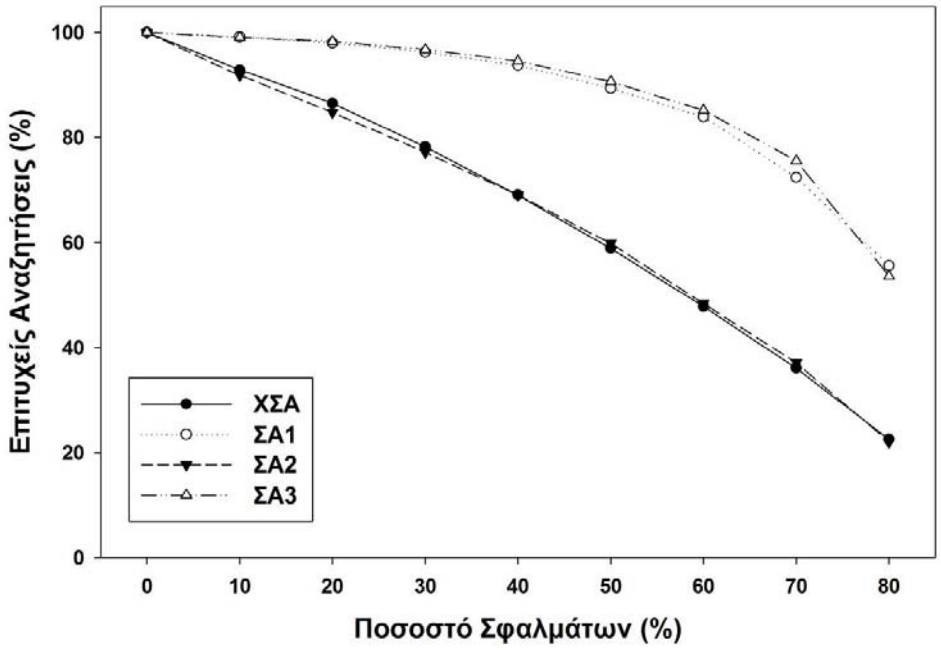
Σχήμα 85. Ποσοστά επιτυχών αναζητήσεων για διαφορετικούς πληθυσμούς ως προς (α) τον αριθμό των εικονικών δικτύων και (β) τα διαφορετικά σχήματα αντιγράφων

Αν αναλύσουμε την επίδραση του συνολικού πληθυσμού στην ανεκτικότητα του συστήματος κατά τη χρήση και των δύο επεκτάσεων, βλέπουμε από τα διαγράμματα του Σχήμα 85 πως η ανεκτικότητα του συστήματος μειώνεται μεν αλλά με πολύ αργό ρυθμό (καθώς στα σχήματα που παρουσιάζονται ο πληθυσμός αυξάνει αλλά όχι γραμμικά καθώς από 1.000 ανεβαίνει σε 5.000, 10.000 και 25.000).



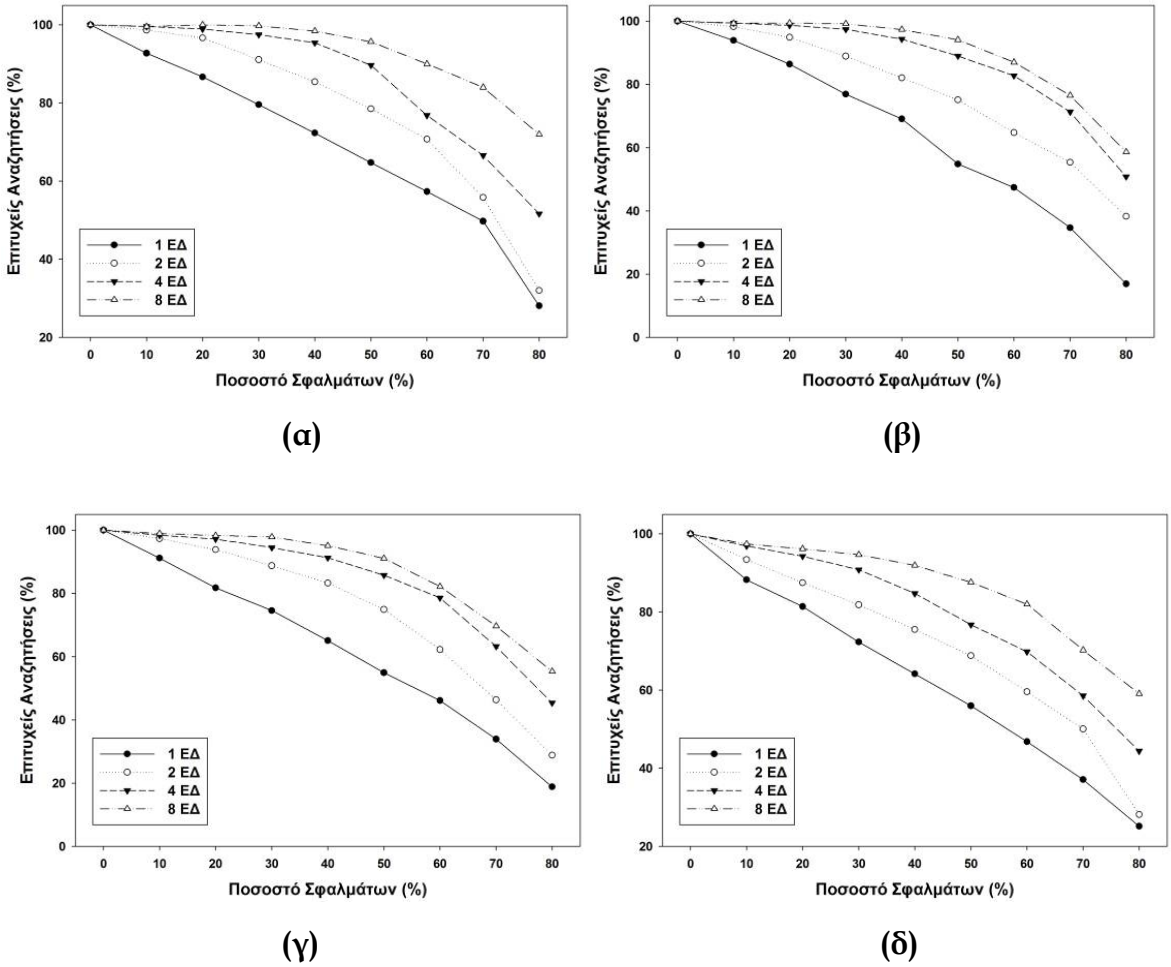
Σχήμα 86. Ποσοστά επιτυχών αναζητήσεων για διαφορετικά σχήματα αντιγράφων συναρτήσει του ποσοστού σφαλμάτων για διαφορετικούς πληθυσμούς (α) 1.000 , (β) 5.000 , (γ) 10.000 και (δ) 25.000 κόμβων

Στο Σχήμα 86 παρουσιάζεται η ανεκτικότητα για διαφορετικούς πληθυσμούς και για διαφορετικά σχήματα αντιγράφων (για το μέσο όρο των εικονικών δικτύων) συναρτήσει του ποσοστού σφαλμάτων. Όπως φαίνεται από τα σχήματα, η χρήση των σχημάτων 1 και 3 επιτυγχάνει σαφέστατη βελτίωση, η οποία για σφάλματα της τάξης του 40% διατηρεί το σύστημα σε σχεδόν τέλεια λειτουργία, με μικρή γραμμική μείωση των ποσοστών επιτυχίας. Η μείωση αυτή αυξάνεται, και μετατρέπεται σε εκθετική μόνο για ποσοστά άνω του 60%.



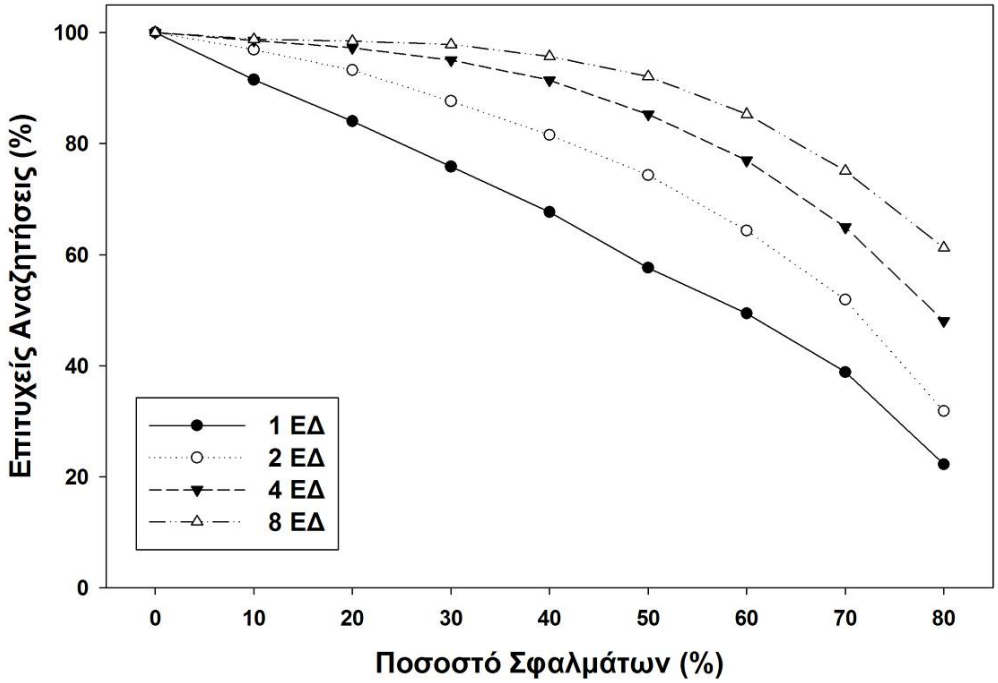
Σχήμα 87. Ποσοστά επιτυχών αναζητήσεων για διαφορετικά σχήματα αντιγράφων ως προς το ποσοστό σφαλμάτων

Το Σχήμα 87 επιβεβαιώνει τα παραπάνω αποτελέσματα και αποδεικνύει την πραγματικά σημαντική δράση των σχημάτων αντιγράφων. Όπως φαίνεται, για ποσοστό σφαλμάτων κάτω του 40% το σύστημα συνεχίζει να δρομολογεί αναζητήσεις με ποσοστά επιτυχίας άνω του 94%. Παρόλο που στην συνέχεια το σύστημα αρχίζει να παρουσιάζει μείωση στην ανεκτικότητα, είναι ακόμη σε θέση να δρομολογεί με ποσοστά άνω του 50% ακόμα και για ποσοστά σφαλμάτων της τάξεως του 80%.



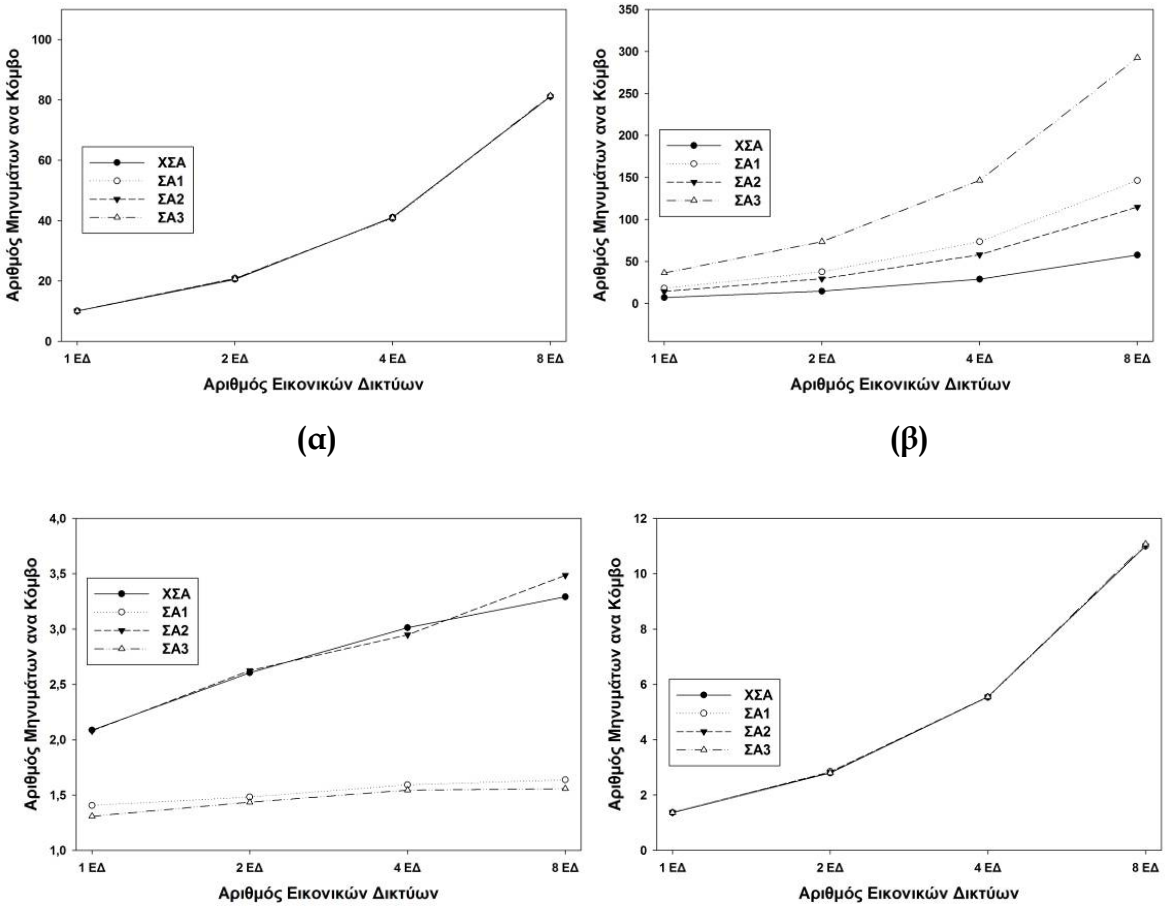
Σχήμα 88. Ποσοστά επιτυχών αναζητήσεων για διαφορετικό αριθμό εικονικών δικτύων συναρτήσει του ποσοστού σφαλμάτων για διαφορετικούς πληθυσμούς (α) 1.000 , (β) 5.000 , (γ) 10.000 και (δ) 25.000 κόμβων

Αν αναλύσουμε την ανεκτικότητα ως προς το ποσοστό σφαλμάτων για την επέκταση των εικονικών δικτύων (με βάση το μέσο όρο των σχημάτων αντιγράφων) θα παρατηρήσουμε στο Σχήμα 88 πως καθώς τα δίκτυα αυξάνονται τα ποσοστά μειώνονται από γραμμικά σε αντίστροφα εκθετικά. Παρατηρούμε επίσης πως ενώ για μικρά ποσοστά (0-20%) οι διαφορές μεταξύ των εικονικών δικτύων (2-8) είναι μικρές, στην συνέχεια αυτές αυξάνονται αλλά παραμένουν σταθερές για ποσοστά σφαλμάτων άνω του 40%.



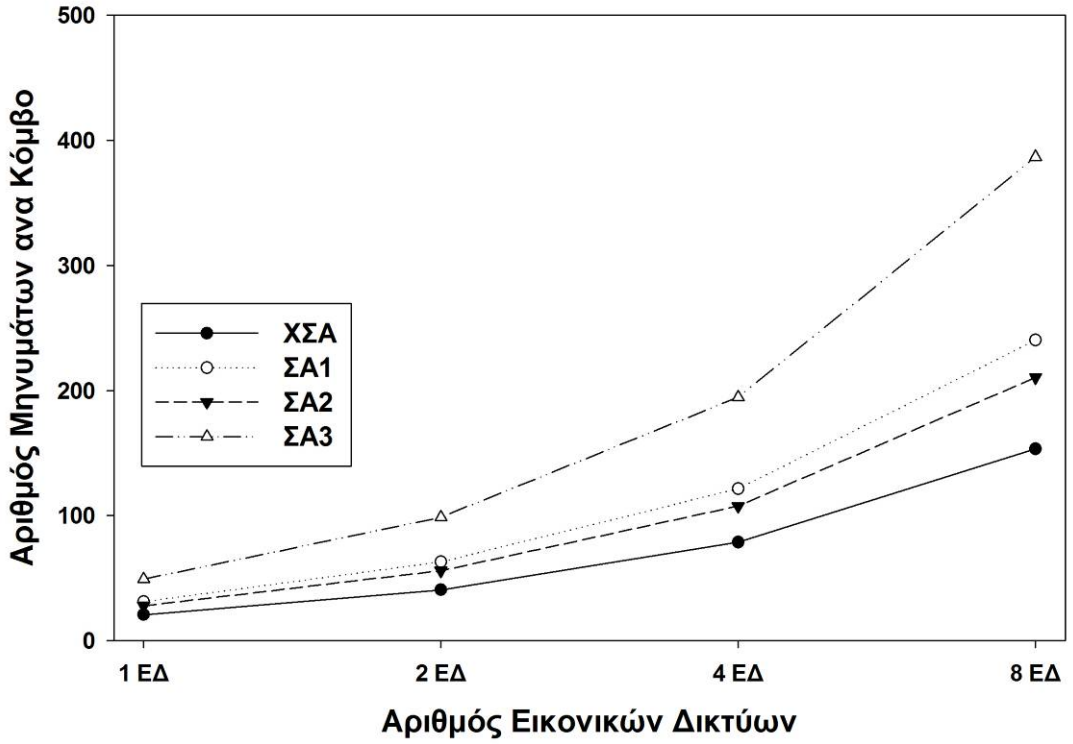
Σχήμα 89. Ποσοστά επιτυχών αναζητήσεων για διαφορετικό αριθμό εικονικών δικτύων ως προς το ποσοστό σφαλμάτων

Τα παραπάνω αποτελέσματα επιβεβαιώνονται και από το Σχήμα 89. Στο σχήμα αυτό παρατηρούμε ακόμα καλύτερα πως η γραμμική μείωση για το 1 δίκτυο μετατρέπεται σε αντίστροφη εκθετική για την περίπτωση των 8 εικονικών δικτύων.



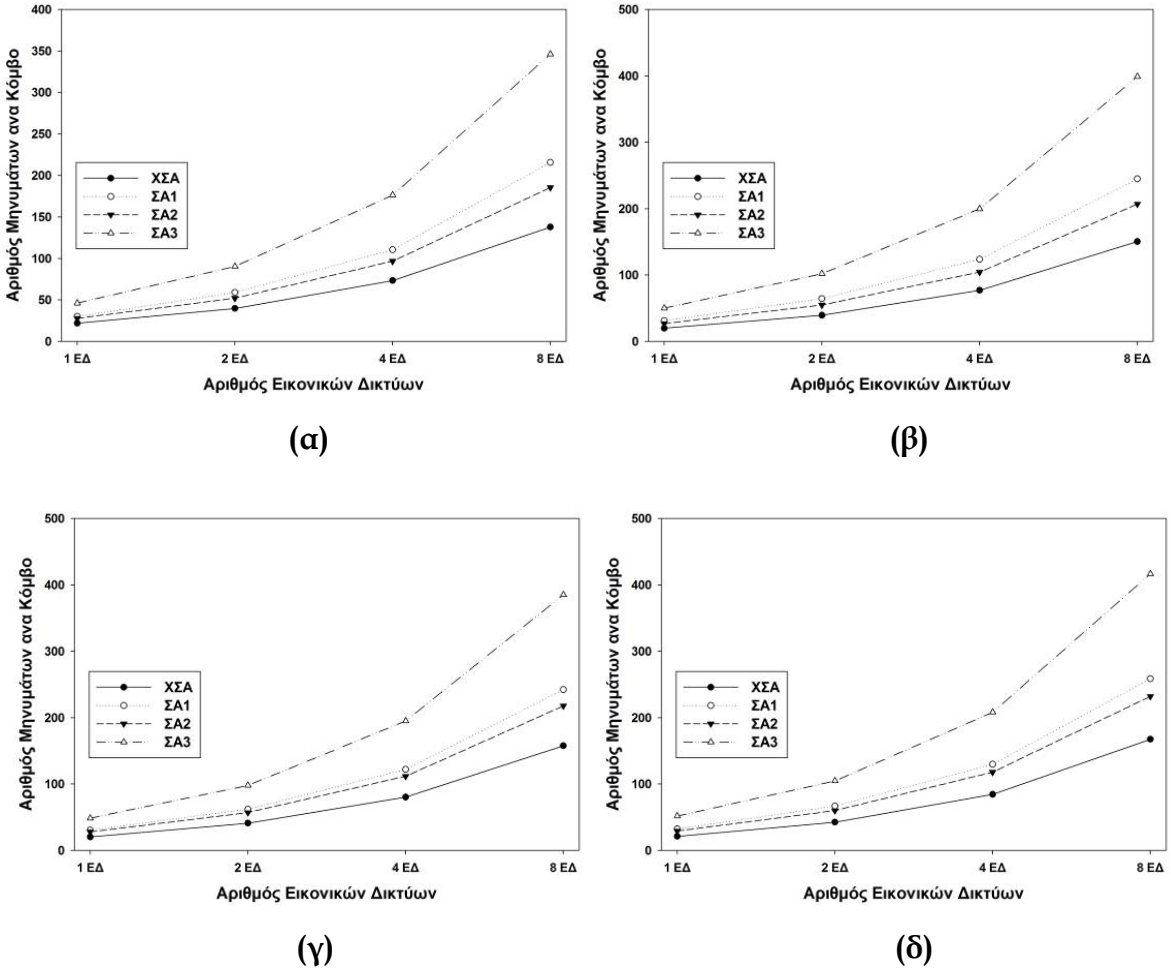
Σχήμα 90. Μέσος αριθμός μηνυμάτων ανά κόμβο για διαφορετικά σχήματα αντιγράφων συναρτήσει του αριθμού εικονικών δικτύων για μηνύματα (α) εισαγωγής, (β) δημοσίευσης , (γ) αναζήτησης και (δ) διάφορα

Στην συνέχεια θα εξετάσουμε την επίδραση του συνδυασμού των επεκτάσεων στην παραγόμενη κίνηση. Αρχικά, στο Σχήμα 90 φαίνεται ο μέσος αριθμός μηνυμάτων για τις διαφορετικές λειτουργίες του δικτύου καθώς μεταβάλλεται ο αριθμός των εικονικών δικτύων και χρησιμοποιούνται διαφορετικά σχήματα αντιγράφων. Όπως φαίνεται, η εισαγωγή δεν επηρεάζεται καθόλου από τα σχήματα αντιγράφων αλλά ορίζεται αποκλειστικά από τον αριθμό εικονικών δικτύων, που αυξάνει γραμμικά με τον αριθμό των δικτύων. Η δημοσίευση αντίθετα διαφοροποιείται τόσο σε σχέση με τα εικονικά δίκτυα όσο και με τα σχήματα αντιγράφων. Η αναζήτηση παραμένει ανεπηρέαστη από τα εικονικά δίκτυα στην περίπτωση εφαρμογής των σχημάτων αντιγράφων 1 και 3 ενώ στην περίπτωση μη χρήσης ή χρήσης σχήματος 2 αυξάνει καθώς αυξάνουν τα εικονικά δίκτυα. Τέλος, τα διάφορα υπόλοιπα μηνύματα ακολουθούν τη συμπεριφορά της εισαγωγής.



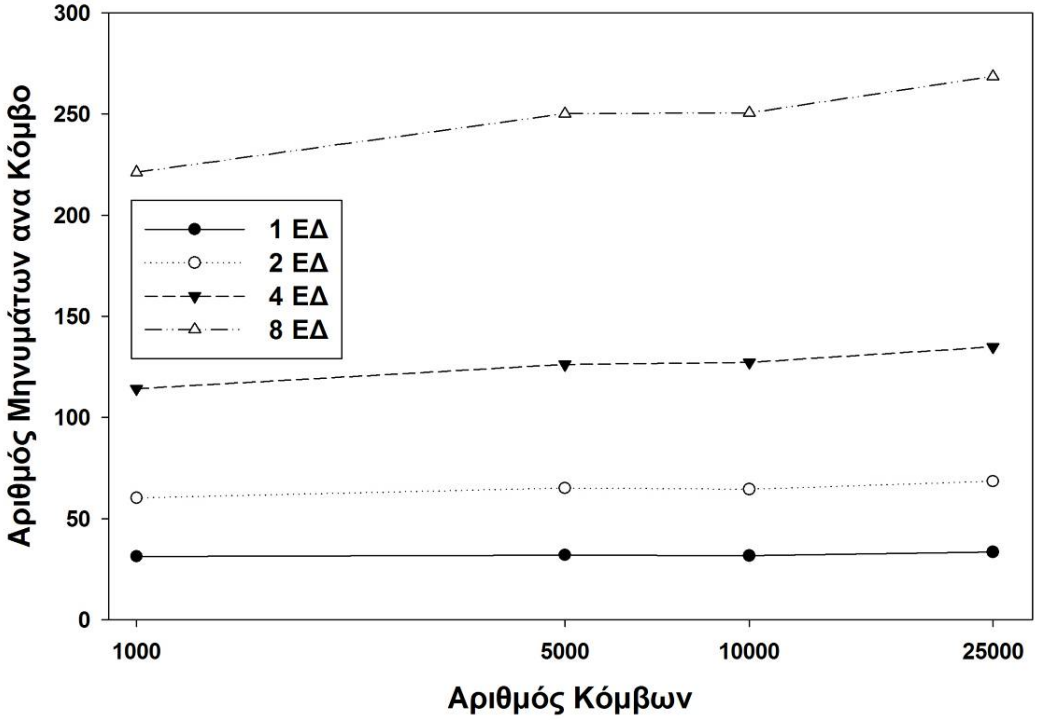
Σχήμα 91. Μέσος αριθμός μηνυμάτων ανά κόμβο για διαφορετικά σχήματα αντιγράφων συναρτήσει του αριθμού εικονικών δικτύων για το σύνολο των μηνυμάτων

Αν παρατηρήσουμε στο Σχήμα 91 τον αριθμό των μηνυμάτων ανά κόμβο για το σύνολο των μηνυμάτων, παρατηρούμε πως η συνολική αύξηση είναι γραμμική ως προς τον αριθμό των εικονικών δικτύων και ενώ τα σχήματα αντιγράφων 1 και 2 δεν αυξάνουν σημαντικά το μέσο όρο σε σχέση με την μη χρήση σχημάτων αντιγράφων, το σχήμα 3 αυξάνει τον όγκο κίνησης κατά περίπου 150% (σε αντίθεση με μέση αύξηση 50% και 35% για τα σχήματα 1 και 2).

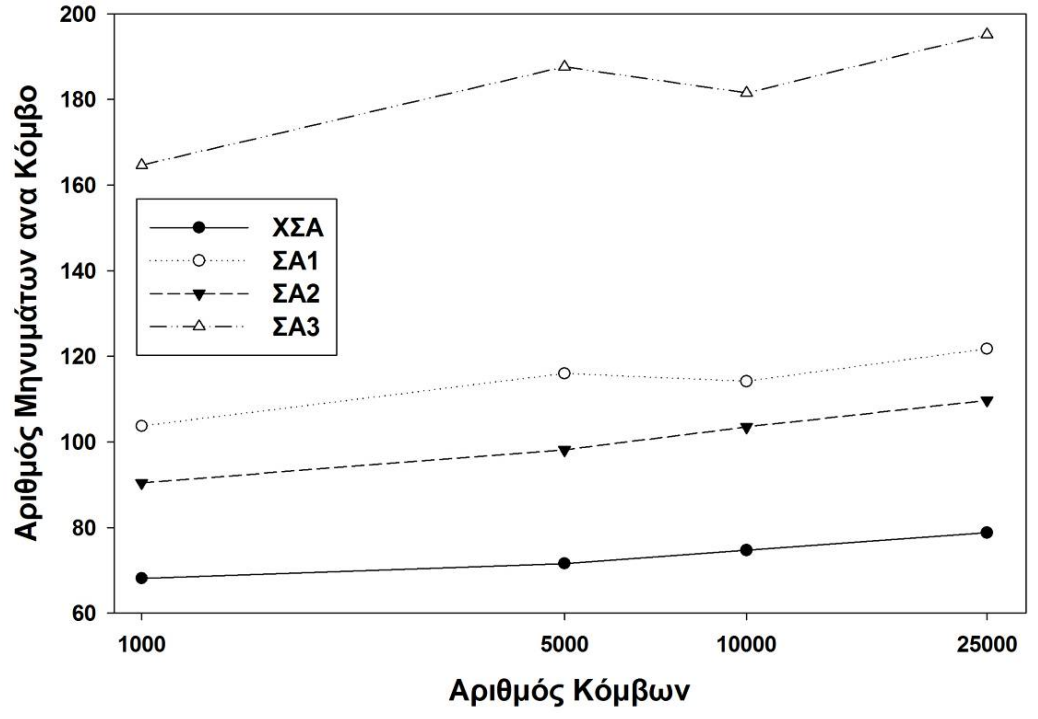


Σχήμα 92. Μέσος αριθμός μηνυμάτων ανά κόμβο για διαφορετικά σχήματα αντιγράφων συναρτήσει του αριθμού εικονικών δικτύων για διαφορετικούς πληθυσμούς (α) 1.000 , (β) 5.000 , (γ) 10.000 και (δ) 25.000 κόμβων

Στο Σχήμα 92 φαίνεται το σύνολο των μηνυμάτων για διαφορετικούς πληθυσμούς κόμβων. Παρατηρώντας τα σχήματα διαπιστώνουμε πως η αύξηση του πληθυσμού δεν αυξάνει ιδιαίτερα το μέσο όρο καθώς τα μηνύματα μπορεί να αυξάνονται σε σύνολο αλλά αντίστοιχα αυξάνει και ο πληθυσμός, με αποτέλεσμα ο μέσος όρος να παραμένει περίπου σταθερός.



(α)

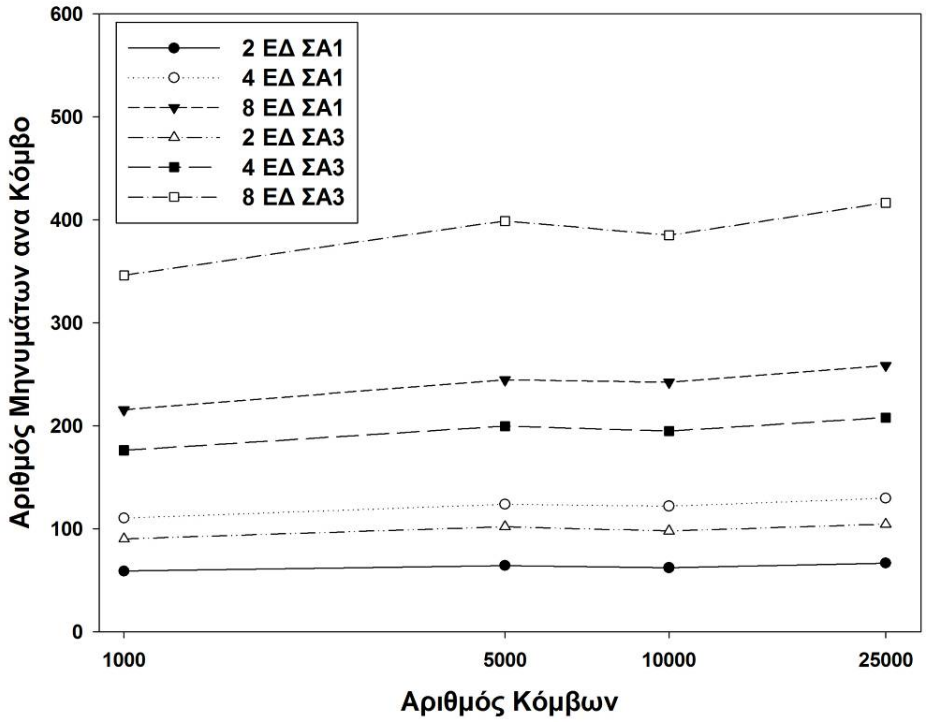


(β)

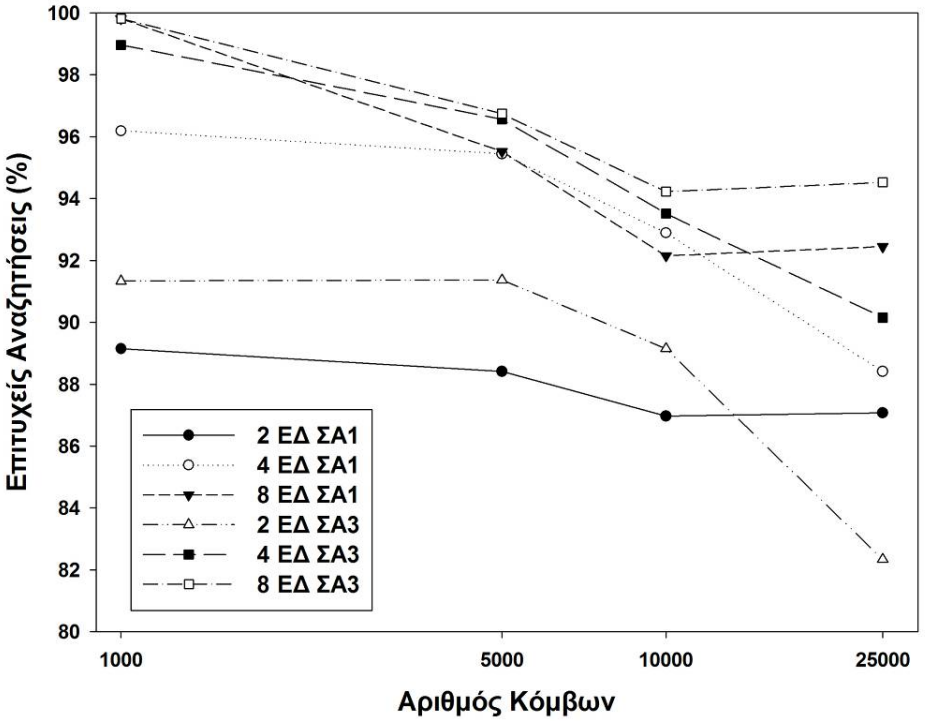
Σχήμα 93. Μέσος αριθμός μηνυμάτων ανά κόμβο συναρτήσει του πληθυσμού των κόμβων για (α) διαφορετικό αριθμό εικονικών δικτύων και (β) διαφορετικά σχήματα αντιγράφων

Για να κατανοήσουμε καλύτερα την επίδραση του πληθυσμού στις επεκτάσεις, στο Σχήμα 93 αρχικά απεικονίζεται η επίδραση για διαφορετικό αριθμό δικτύων (α) και στην συνέχεια για διαφορετικά σχήματα αντιγράφων (β). Όπως φαίνεται και από τα δύο σχήματα, η επίδραση του πληθυσμού είναι μικρή και οι επεκτάσεις είναι αυτές που επηρεάζουν ουσιαστικά το μέσο αριθμό μηνυμάτων.

Τα αποτελέσματα αυτά επιβεβαιώνονται και από το Σχήμα 94 όπου φαίνεται το σύνολο συναρτήσει του πληθυσμού για διαφορετικούς συνδυασμούς των επεκτάσεων. Όπως φαίνεται στο σχήμα (α), ο μέσος όρος παραμένει σχεδόν σταθερός ανεξάρτητα από την αύξηση του πληθυσμού. Στο σχήμα (β) απεικονίζονται τα ποσοστά επιτυχίας για καλύτερη κατανόηση της επίδρασης του συνδυασμού των επεκτάσεων. Όπως φαίνεται, όταν χρησιμοποιείται συνδυασμός άνω των 2 εικονικών δικτύων και σχήμα αντιγράφων είτε 1 είτε 3, το σύστημα διατηρεί τα ποσοστά των αναζητήσεων διαρκώς άνω του 80% (μέσος όρος και όχι βέλτιστη τιμή) και στην ειδικότερη περίπτωση που τα εικονικά δίκτυα είναι 4 ή άνω τότε το ποσοστό αυτό διατηρείται άνω του 90%.



(α)



(β)

Σχήμα 94. (α) Μέσος αριθμός συνολικών μηνυμάτων και (β) ποσοστό επιτυχών αναζητήσεων ως προς τον πληθυσμό των κόμβων για διαφορετικούς συνδυασμούς των επεκτάσεων

Τέλος, στον Πίνακας 12 φαίνονται τα βέλτιστα αποτελέσματα που επιτύχαμε (σε αντίθεση με το μέσο όρο που παρατίθεντο έως τώρα) κατά τη χρήση των δύο επεκτάσεων και του αλγορίθμου επιδιόρθωσης για την περίπτωση δικτύου 25.000 κόμβων. Όπως παρατηρούμε, η ταυτόχρονη χρήση των επεκτάσεων και ο αλγόριθμος επιδιόρθωσης είναι σε θέση να δρομολογούν αναζητήσεις στο σύστημα μας έως και 99% σε περιπτώσεις σφαλμάτων έως και 80%. Επίσης, παρατηρούμε πως η χρήση του σχήματος 3 δε διαφέρει ιδιαίτερα από αυτήν του σχήματος 1 και η χρήση περιόδου 10T διαφέρει ελάχιστα από την περίοδο 20T. Επιπλέον, είναι προφανές ότι 2 εικονικά δίκτυα αρκούν για ποσοστά σφαλμάτων έως και 70% ενώ για σχεδόν μηδενικές απώλειες σε υψηλότερα ποσοστά αποτυχιών απαιτούνται 4 και άνω εικονικά δίκτυα.

Εικονικά Δίκτυα		**2**				**4**				**8**			
Περίοδος Επιδιόρθωσης		**20T**		**10T**		**20T**		**10T**		**20T**		**10T**	
Σχήμα Αντιγράφων		**1**	**3**	**1**	**3**	**1**	**3**	**1**	**3**	**1**	**3**	**1**	**3**
Ποσοστά σφαλμάτων (%)	**0**	100	100	100	100	100	100	100	100	100	100	100	100
	10	100	100	100	100	100	100	100	100	100	100	100	100
	20	100	100	100	100	100	100	100	100	100	100	100	100
	30	100	100	100	100	100	100	100	100	100	100	100	100
	40	100	100	100	100	100	100	100	100	100	100	100	100
	50	99	100	99	99	100	100	100	100	100	100	100	100
	60	99	99	99	98	100	100	100	100	100	100	100	100
	70	97	97	96	93	96	100	99	100	100	96	100	100
	80	77	86	80	89	82	98	97	98	98	97	99	99

Πίνακας 12. Βέλτιστα ποσοστά αναζητήσεων που επιτεύχθηκαν στην περίπτωση δικτύου 25.000 κόμβων και για διαφορετικούς συνδυασμούς σχημάτων αντιγράφων, εικονικών δικτύων και περιόδους επιδιόρθωσης

Με βάση όλα τα παραπάνω αποτελέσματα, μπορούμε να συμπεράνουμε ότι το σύστημά μας είναι σε θέση να δρομολογεί αναζητήσεις με εκπληκτική ανεκτικότητα σε δίκτυα μεγάλου πληθυσμού και υψηλών ποσοστών σφαλμάτων. Επιπλέον, για να ελαχιστοποιήσουμε τον αριθμό των επιπλέον μηνυμάτων που επιφέρουν οι επεκτάσεις που εισάγαμε, συνιστάται η χρήση 2 ή 4 το πολύ εικονικών δικτύων με ταυτόχρονη χρήση του σχήματος

αντιγράφου τοπικής διασποράς και περίοδο επιδιόρθωσης 20T, καθώς με αυτές τις παραμέτρους το σύστημα δρομολογεί με απόλυτη (100%) επιτυχία για ποσοστά σφαλμάτων έως και 40% και με ποσοστό άνω του 95% για ποσοστά σφαλμάτων έως και 70%, χωρίς ο όγκος κίνησης να είναι ιδιαίτερα υψηλός.

8.5. Σύνοψη - Συμπεράσματα

Σε αυτό το κεφάλαιο παρουσιάσαμε μια σειρά από επεκτάσεις που βελτιώνουν την απόδοση του συστήματος. Πιο συγκεκριμένα, αρχικά περιγράψαμε τρία διαφορετικά σχήματα αντιγράφων που επιτρέπουν την ομαλότερη κατανομή κλειδιών στον χώρο καθώς και βελτιώνουν την ανεκτικότητα του συστήματος. Η βελτίωση αυτή αγγίζει το 400% σε πολλές περιπτώσεις ενώ η μέση τιμή κυμαίνεται γύρω στο 60%. Επιπλέον, παρατηρήσαμε πως η βελτίωση αυτή δεν επηρεάζει το μέσο αριθμό βημάτων και η μόνη αρνητική επίπτωση είναι η επιφόρτιση του δικτύου με επιπλέον κίνηση, που κυμαίνεται σε ποσοστά 51%, 36% και 140% αντίστοιχα για τα τρία διαφορετικά σχήματα. Το συντριπτικό ποσοστό αυτών των αυξήσεων όμως οφείλεται σε μηνύματα δημοσίευσης και όχι σε αναζητήσεις. Στην συνέχεια εισάγαμε την έννοια των εικονικών δικτύων, όπου κάθε κόμβος είναι μέλος ενός ή περισσότερων εικονικών δικτύων με διαφορετική θέση σε καθένα από αυτά. Και στην περίπτωση αυτή, η επέκτασή μας βελτιώνει σημαντικά την ανεκτικότητα του συστήματος, από 49% για 2 εικονικά δίκτυα έως και 140% για 8, με μόνη πάλι αρνητική επίπτωση την αύξηση του όγκου κίνησης λόγω του αυξημένου αριθμού μηνυμάτων, η οποία είναι γραμμική σε σχέση με τον αριθμό των δικτύων. Τέλος παρουσιάσαμε το συνδυασμό των δύο αυτών επεκτάσεων, ο οποίος επιφέρει τη βέλτιστη επιτευχθείσα απόδοση του αλγορίθμου μας και συνδυάζει τα χαρακτηριστικά, τόσο θετικά όσο και αρνητικά, και των δύο επεκτάσεων. Με βάση όλα τα παραπάνω, συμπεράναμε ότι το σύστημά μας είναι σε θέση να δρομολογεί αναζητήσεις με εκπληκτική ανεκτικότητα για δίκτυα μεγάλου πληθυσμού και υψηλών ποσοστών σφαλμάτων και συνιστούμε τη χρήση 2 ή 4 το πολύ εικονικών δικτύων με ταυτόχρονη χρήση του σχήματος αντιγράφου τοπικής διασποράς και περίοδο επιδιόρθωσης 20T για τον καλύτερο συνδυασμό επιτυχών αναζητήσεων και ελαχιστοποίησης του φόρτου κίνησης.

8.6. Αναφορές

[1] Ghodsi, A., Alima, L.O., Harid, S., “Symmetric Replication for Structured Peer-to-Peer Systems”, The 3rd International Workshop on Databases,

Information Systems and Peer-to-Peer Computing, DBISP2P2005, Trondheim, Norway, August 28-29, 2005.

[2] Clarke, I., Hong, T.W., Miller, S.G., Sandberg, O., Wile, B., "Protecting Free Expression Online with Freenet," IEEE Internet Computing, 6(1), 2002.

9 ΤΟ ΔΙΚΤΥΟ K-UMBRELLA

Στις προηγούμενες ενότητες εξετάσαμε τις ιδιότητες και τη συμπεριφορά του δικτύου Umbrella και όπως παρατηρήσαμε το δίκτυο είναι σε θέση να δρομολογεί αναζητήσεις σε λογαριθμικό αριθμό βημάτων χωρίς ιδιαίτερη επιβάρυνση στον φόρτο κίνησης. Με την προσθήκη μάλιστα των επεκτάσεων και του αλγορίθμου επιδιόρθωσης προστίθεται και η ιδιότητα της υψηλής ανεκτικότητας σε σφάλματα. Στην ενότητα αυτή θα εισάγουμε την έννοια του δικτύου K-Umbrella που αποτελεί επέκταση του Umbrella με αρκετές όμως αλλαγές που προσφέρει επιπλέον δυνατότητες παραμετροποίησης του δικτύου για καλύτερη κατανομή της κίνησης και έλεγχο της συμπεριφοράς του δικτύου σε σχέση με την παραγόμενη κίνηση και τη δρομολόγηση των μηνυμάτων.

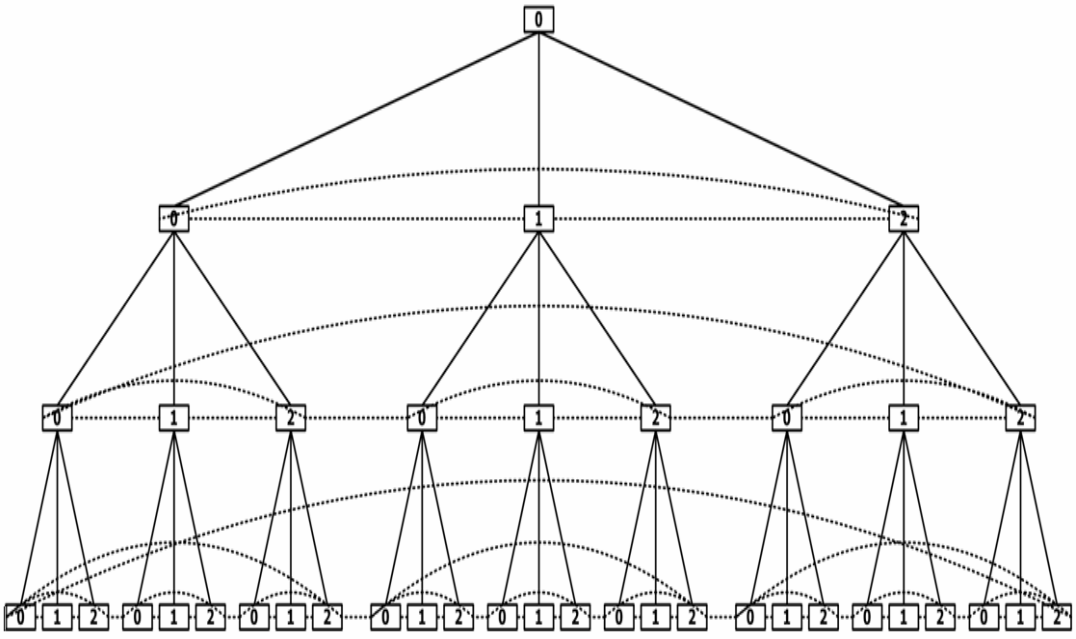
9.1. Επέκταση του Umbrella μέσω του K-Umbrella

Το δίκτυο K-Umbrella αποτελεί επέκταση και ταυτόχρονα παραλλαγή του Umbrella προσφέροντας παραμετροποίηση στον αριθμό των παιδιών που υποστηρίζει κάθε κόμβος και ταυτόχρονα εισάγοντας ένα διαφορετικό πίνακα δρομολόγησης που βασίζεται σε παραμετροποιημένο λογαριθμικό αριθμό γειτόνων. Η διαφοροποίηση αυτή δίνει τη δυνατότητα ορισμού του τρόπου δρομολόγησης ώστε να κατανέμεται καλύτερα η κίνηση αλλά και να διαφοροποιείται ο μέγιστος αριθμός βημάτων για κάθε λειτουργία. Με τον τρόπο αυτό μπορούμε να μεταβάλουμε το δίκτυό μας από τελείως αποκεντρωποιημένο σε σχεδόν ιεραρχική μορφή, ανάλογα αν στο δίκτυο υπάρχουν υπερκόμβοι που είναι σε θέση να υποστηρίξουν μεγαλύτερο όγκο κίνησης ή όλοι οι κόμβοι είναι όμοιοι οπότε έχουμε πλήρως αποκεντρωποιημένο δίκτυο.

9.2. Το δίκτυο K-Umbrella

Το K-Umbrella βασίζεται στα K-δέντρα, τα οποία αποτελούν γενίκευση των δυαδικών δέντρων, όπου κάθε κόμβος έχει το μέγιστο k παιδιά, όπου το k είναι γνωστό ως ο βαθμός διακλάδωσης. Ενώ για τα δυαδικά δέντρα έχουν προταθεί μια σειρά από DHT αλγορίθμους [1], τα K-δέντρα δεν έχουν

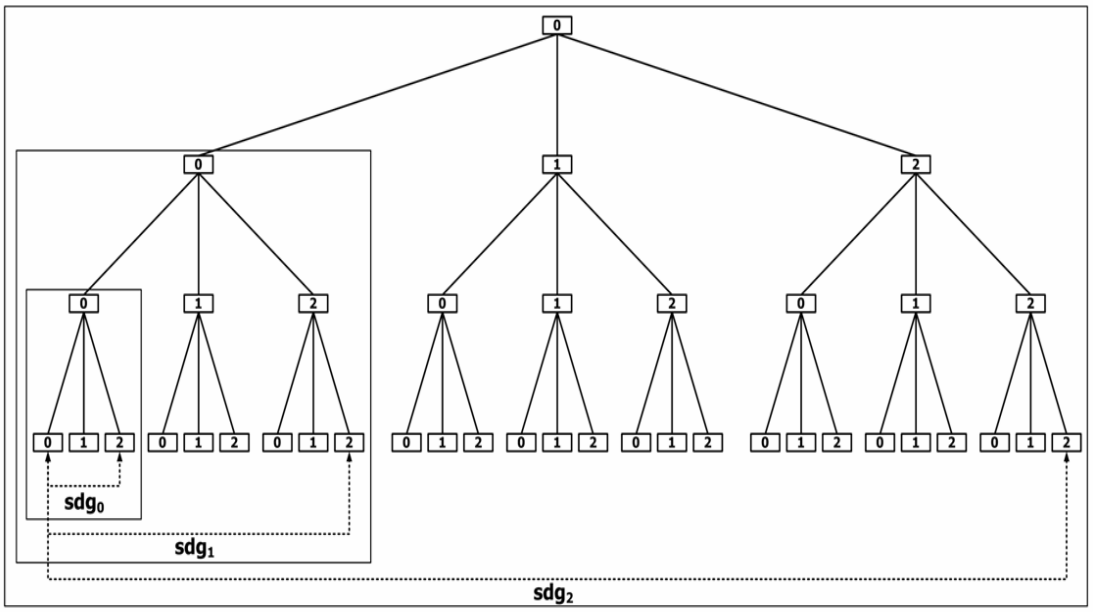
εξερευνηθεί ικανοποιητικά. Το K-Umbrella αποτελεί μια διαφορετική έκδοση του K-δέντρου, καθώς κάθε κόμβος έχει το πολύ k παιδιά αλλά και μια σειρά από επιπλέον συνδέσεις που εγγυώνται πως η δομή του δέντρου μπορεί να κατανεμηθεί σε ένα ομότιμο δίκτυο. Ο βασικός σκοπός της αρχιτεκτονικής K-Umbrella παραμένει ίδιος με αυτόν του Umbrella, δηλαδή να υποστηρίζει την είσοδο κόμβων, τη δημοσίευση περιεχομένων και την επιτυχή αναζήτηση μέσω ενός απλού και δομημένου αλγόριθμου δρομολόγησης ώστε όλες οι λειτουργίες να είναι τόσο αποδοτικές όσο και ανεκτικές σε σφάλματα. Επιπλέον, κάθε κόμβος απαιτείται να διατηρεί μόνο ένα μικρό , παραμετρικά ορισμένο, αριθμό συνδέσεων προς γειτονικούς κόμβους, ώστε το σύστημα να είναι σε θέση να κλιμακώνεται τόσο σε αριθμό κόμβων όσο και περιεχομένων.



Σχήμα 95. Η δομή του δικτύου K-Umbrella

Το K-Umbrella, που παρουσιάζεται στο Σχήμα 95, αποτελεί ένα K-δέντρο όπου κάθε κόμβος n βρίσκεται στο επίπεδο *l* και έχει το πολύ *k* φύλλα (γνωστά ως παιδιά). Ένα παιδί n_k μοιράζεται το ίδιο πρόθεμα $pre_l(n_k)$ με τον πατέρα του n για όλα τα πρώτα *l* ψηφία, οπότε $pre_l(n_k)= pre_l(n)$, και το ψηφίο στη θέση *l* είναι ίσο με *k*. Επιπλέον, κάθε κόμβος διατηρεί μια σειρά επιπλέον συνδέσεων, οι οποίες εγγυώνται την κλιμακοσιμότητα και ανεκτικότητα σε σφάλματα. Με βάση αυτά, κάθε κόμβος n_k διατηρεί δύο σετ δεικτών, ένα σετ προς κάθε μη-κενό γείτονα που βρίσκεται δεξιά και αριστερά του, οι οποίοι δεν είναι απαραίτητο να ανήκουν στο ίδιο υποδέντρο. Επιπλέον, κάθε κόμβος που βρίσκεται στην άκρη ενός υποδέντρου διατηρεί ένα δείκτη προς τον κόμβο που βρίσκεται στην άλλη άκρη στο ίδιο επίπεδο. Οι κόμβοι που βρίσκονται σε μεγαλύτερα επίπεδα υποδέντρων διατηρούν επιπλέον δείκτες άκρων για κάθε επιπλέον επίπεδο. Αποκαλούμαι τους δείκτες αυτούς ακμές

υποδέντρων, $sdg_l(n)$, και τους επεκτείνουμε από τη βάση προς τα πάνω, όπως φαίνεται στο Σχήμα 96.



Σχήμα 96. Παράδειγμα ακμών υποδέντρων

Ένας απλός κανόνας για την εύρεση των ακμών υποδέντρων είναι η οριζόντια κίνηση (αριστερά ή δεξιά) στο δέντρο εωσότου ευρεθεί το τελευταίο παιδί του υποδέντρου. Για παράδειγμα, με βάση το Σχήμα 96, η $sdg_1(n)$ εντοπίζεται με την κίνηση προς τα δεξιά και την εύρεση του τελευταίου παιδιού που ανήκει στο ίδιο υποδέντρο στο επίπεδο 2. Κατά την υλοποίηση του δικτύου μας, καθώς η εύρεση των ακμών υποδέντρων μεγάλων επιπέδων μπορεί να επιφέρει υπερβολική κίνηση μηνυμάτων (k^l+1), περιορίζουμε το επίπεδο των ακμών υποδέντρων που γνωρίζει κάθε κόμβος με ένα άνω όριο m.

Κάθε κόμβος εισέρχεται στο σύστημα μέσω ενός υπάρχον κόμβου, ο οποίος ανακοινώνει τη νέα είσοδο. Όταν η συγκεκριμένη διαδικασία ολοκληρωθεί επιτυχώς, ο νέος κόμβος, έχοντας ανακτήσει και ενημερώσει όλους τους γειτονικούς κόμβους, μπορεί να δημοσιεύσει όλα του τα περιεχόμενα. Η διαδικασία δημοσίευσης είναι παρόμοια με αυτή της εισαγωγής, καθώς όπως και στην περίπτωση του δικτύου Umbrella, το περιεχόμενο χαρακτηρίζεται από μια σειρά κλειδιών, τα οποία αφού κατατεμαχιστούν μπορούν να προωθηθούν με παρόμοιο τρόπο. Όλα τα κλειδιά δημοσιεύονται σε υπάρχον κόμβο με το πιο πλησίον αναγνωριστικό με αυτό του κλειδιού. Ως πιο πλησίον αναγνωριστικό θεωρούμε αυτό που έχει τα περισσότερα ίδια ψηφία, από αριστερά προς τα δεξιά, και το μικρότερο αριθμό σε περίπτωση που δύο κόμβοι έχουν τον ίδιο αριθμό κοινών ψηφίων. Με παρόμοιο τρόπο, η αναζήτηση λειτουργεί μέσω της προώθηση των

αιτημάτων προς τον κόμβο με το πιο πλησίον αναγνωριστικό με αυτό του κλειδιού προς αναζήτηση. Εάν δεν ευρεθεί αυτός ο κόμβος, τότε θεωρούμε πως το περιεχόμενο δεν υπάρχει.

9.3. Κατανομή κλειδιών

Κάθε επίπεδο *l* στην δομή μας μπορεί να υποστηρίξει k^{l+1} κόμβους. Κάθε κόμβος έχει ένα μοναδικό κόμβο πατέρα, ο οποίος βρίσκεται πάντα ένα επίπεδο υψηλότερα, και το πολύ *k* παιδιά σε ένα επίπεδο χαμηλότερο. Έστω *f(x)* η συνάρτηση που επιστρέφει το ψηφίο του αναγνωριστικού που βρίσκεται στη θέση *x*. Τότε, η σχέση που συνδέει τους κόμβους A και B με κοινό πατέρα *p* και κοινό επίπεδο *l* είναι:

$$f_A(x) = f_B(x), \quad 1 \le x \le l+1$$

$$f_A(x) \neq f_B(x), \quad x = l+2$$

Επιπλέον, η σχέση μεταξύ πατέρα και παιδιού απαιτεί πως:

$$f_p(x) = f_A(x) = f_B(x), \quad 1 \le x \le l+1$$

Οι παραπάνω απλοί κανόνες ισχύουν για όλους τους κόμβους που εισέρχονται στο K-Umbrella υπερκείμενο δίκτυο. Και σε αυτό το δίκτυο, χρησιμοποιούμε τη συνάρτηση κατατεμαχισμού SHA-1 για να αναθέσουμε αναγνωριστικά τόσο στους κόμβους όσο και στα περιεχόμενα, παρέχοντας ομοιόμορφη κατανομή στον χώρο των 160-ψηφίων και μη-εθελοντική ανωνυμία [2] του περιεχομένου. Για να μπορέσουμε να εφαρμόσουμε τους αλγορίθμους, που θα περιγράψουμε στην συνέχεια, ορίζουμε μια συνάρτηση σύγκρισης *comp*, που συγκρίνει δυο αναγνωριστικά και υπολογίζει τη διαφορά τους ως ακέραιο. Η συνάρτηση αυτή ορίζεται ως εξής:

$$comp(id_1, id_2) = \sum_{i=1}^{i=160/k} g(i)$$

$$g(i) = \begin{cases} k^{(160/k)-i}, & if \quad f_{id_1}(i) = f_{id_1}(i) \\ 0, & if \quad f_{id_1}(i) \neq f_{id_1}(i) \end{cases}$$

Η χρήση της συνάρτησης κατατεμαχισμού για την κατανομή των αναγνωριστικών στον χώρο των κόμβων και των περιεχομένων επιτρέπει τη δημιουργία ενός εξισορροπημένου K-δέντρου. Η δομή γίνεται ακόμα πιο εξισορροπημένη καθώς ο πληθυσμός των κόμβων αυξάνεται και νέοι κόμβοι καλύπτουν τα κενά διαστήματα (φύλλα). Η χρήση της συνάρτησης κατατεμαχισμού εξισορροπεί επιπλέον την κατανομή των κλειδιών στους κόμβους, όπως έχει αποδειχθεί στο [3] με βάση το παρακάτω θεώρημα:

Θεώρημα 3. Έστω ένα σύνολο κόμβων N και ένα σύνολο κλειδιών R. Τότε με μεγάλη πιθανότητα κάθε κόμβος είναι υπεύθυνος για ένα μέσο όρο κλειδιών R/N, με μέγιστο αριθμό κλειδιών ανά κόμβο (1+a)R/N, όπου a μια παράμετρος με όριο O(logN).

Στην συνέχεια αυτής της εργασίας, όπως και στην περίπτωση του δικτύου Umbrella, δε θα ασχοληθούμε περισσότερο με τα χαρακτηριστικά της συνάρτησης κατατεμαχισμού καθώς αυτά έχουν αναλυθεί και αποδειχθεί λεπτομερέστατα από τον Karger κ.α. [3] και δεν αποτελούν μέρος της έρευνάς μας.

9.4. Πίνακας γειτόνων

Όπως και στα περισσότερα DHT συστήματα, κάθε κόμβος διατηρεί έναν πίνακα δρομολόγησης, με βάση τον οποίον δρομολογούνται τα εισερχόμενα μηνύματα. Κάθε κόμβος είναι υπεύθυνος να διατηρεί ενημερωμένο τον πίνακα γειτόνων του αποστέλλοντας μηνύματα σε όλους τους κόμβους που βρίσκονται σε αυτόν σε τακτά χρονικά διαστήματα. Ο πίνακας δρομολόγησης στο δίκτυό μας αναφέρεται ως πίνακας γειτόνων και παρουσιάζεται στον Πίνακας 13.

Παράμετρος	Επεξήγηση
k	Ο μέγιστος αριθμός παιδιών
M	Το μέγιστο επίπεδο ακμών υποδέντρων
l	Το επίπεδο στο οποίο βρίσκεται
right	Ο μη-κενός κόμβος προς τα δεξιά
left	Ο μη-κενός κόμβος προς τα αριστερά
up	Ο κόμβος πατέρας
K-Umbrella	Όλοι οι κόμβοι παιδιά
sdg_l	Η ακμή υποδέντρου για το επίπεδο *l*

Πίνακας 13. Ο πίνακας γειτόνων του K-Umbrella

Οι παράμετροι στον πίνακα μπορεί να διαχωριστούν σε δύο κατηγορίες, στις σταθερές και στις δυναμικές. Οι σταθερές παράμετροι περιλαμβάνουν τις μεταβλητές που ορίζουν τη δομή του K-δέντρου και πιο συγκεκριμένα το μέγιστο αριθμό παιδιών ανά κόμβο (*k*) και το μέγιστο αριθμό ακμών υποδέντρων (*m*). Οι δυναμικές παράμετροι ορίζουν τη γειτονική τοπολογία (συνδέσεις) κάθε κόμβου και περιλαμβάνουν τους

δείκτες προς τον πατέρα (Up), αριστερό κόμβο (left), δεξιό κόμβο (right), κόμβους παιδιά (K-Umbrella) και κόμβους ακμών υποδέντρων (sdg_i). Τέλος, στις δυναμικές παραμέτρους περιλαμβάνεται και το επίπεδο στο οποίο βρίσκεται ο κόμβος (l), καθώς αυτή η πληροφορία ανακτάται κατά τη διάρκεια του μηχανισμού εισαγωγής.

Τα παραπάνω στοιχεία είναι αρκετά για να διατηρήσουν ομαλή τη δρομολόγηση στην αρχιτεκτονική μας και να κατανείμουν ομοιόμορφα την κίνηση σε όλους τους κόμβους. Κάθε κόμβος είναι υπεύθυνος να τροποποιεί ή να επιδιορθώνει τον πίνακα γειτόνων του καθώς κόμβοι εισέρχονται και εξέρχονται από το δίκτυο ή όταν εντοπιστεί βλάβη επικοινωνίας με έναν κόμβο. Η δομή του πίνακα γειτόνων μας επιτρέπει επίσης να μετακινηθούμε στο K-δέντρο με διαφορετικούς τρόπους, εφαρμόζοντας συνδυασμό κίνησης τόσο κατά τον οριζόντιο όσο και κατά τον κάθετο άξονα. Μπορεί εύκολα να αποδειχθεί πως αν ο αριθμός των επιτρεπόμενων παιδιών τεθεί στο 1, τότε η δομή μας εξομοιώνει έναν καρτεσιανό χώρο 2-αξόνων, όπως στο CAN, ενώ αν τεθεί στο 2 τότε εξομοιώνει ένα δυαδικό δέντρο, όπως στο Kademlia. Πρέπει επίσης να τονίσουμε πως ο πίνακας γειτόνων είναι σταθερού μεγέθους (k+m+3), σε αντίθεση με άλλους DHT αλγόριθμους, και επομένων μπορεί να υποστηρίξει αυξημένη κλιμακοσιμότητα.

Η δομή της αρχιτεκτονικής μας και του πίνακα γειτόνων που έχουμε περιγράψει εγγυώνται πως ένα δημοσιευμένο κλειδί μπορεί να εντοπιστεί μέσω της κατάλληλης αναζήτησης εντός γραμμικών ή λογαριθμικών υπερκείμενων βημάτων σε σχέση με το συνολικό μέγεθος του δικτύου, ανάλογα με τις παραμέτρους που θέσουμε για τη βελτιστοποίηση του πρωτοκόλλου μας. Η παραπάνω πρόταση διατυπώνεται και αποδεικνύεται μέσα από τα παρακάτω δύο θεωρήματα:

Θεώρημα 4. Σε ένα δίκτυο K-Umbrella με N κόμβους και αναγνωριστικά με βάση k τα οποία αποκτώνται μέσω μιας συνάρτησης κατατεμαχισμού, το μέγιστο ύψος της δομής K-δέντρου είναι λογαριθμικού μεγέθους σε σχέση με τον πληθυσμό.

Απόδειξη: Έστω k η βάση των αναγνωριστικών, N ο συνολικός αριθμός των κόμβων και l ένα τυχαίο επίπεδο στην K-Umbrella δομή. Τότε με βάση το πρωτόκολλο του K-Umbrella, σε κάθε επίπεδο μπορούν να ανήκουν το πολύ k^l κόμβοι, με k^0=1. Επομένως, εάν s είναι ο απαιτούμενος αριθμός επιπέδων για τον παραπάνω πληθυσμό, τότε με βάση το Θεώρημα 1, έχουμε την παρακάτω σχέση.

$$N = \sum_{z=0}^{s-1} k^z = \frac{k^0 - k^{s-1+1}}{1-k} = \frac{1-k^s}{1-k} \Leftrightarrow s = \log_k\left[N(k-1)+1\right]$$

Επομένως το μέγιστο ύψος m της δομής μας είναι $O(\log_k N)$. ▪

Θεώρημα 5. Μια επιτυχημένη αναζήτηση σε ένα δίκτυο K-Umbrella απαιτεί, με μεγάλη πιθανότητα, είτε γραμμικό είτε λογαριθμικό αριθμό βημάτων σε σχέση με το συνολικό μέγεθος του δικτύου, ανάλογα με τις παραμέτρους του πρωτοκόλλου.

Απόδειξη: Έστω πως ένας κόμβος p ο οποίος ανήκει στο επίπεδο l_p αναζητεί έναν συγκεκριμένο κλειδί z το οποίο βρίσκεται μέσα στο ίδιο δίκτυο σε έναν άλλο κόμβο f στο επίπεδο l_f.. Εάν s είναι ο αριθμός των επιπέδων του δικτύου, N ο πληθυσμός των κόμβων, k ο μέγιστος αριθμός παιδιών και H ο απαιτούμενος αριθμός βημάτων, τότε μπορούμε να ισχυριστούμε πως το χειρότερο σενάριο απαιτεί του δύο κόμβους να βρίσκονται στο επίπεδο s και με μέγιστη απόσταση μεταξύ τους (δηλαδή ο κόμβος p είναι ένα παιδί επιπέδου s του πρώτου παιδιού στο επίπεδο 1 και ούτως καθεξής και ο κόμβος f είναι ένα παιδί m επιπέδου του k παιδιού στο επίπεδο 1 και ούτως καθεξής). Με βάση το πρωτόκολλό μας μπορούμε να κάνουμε μια αναζήτηση με βάση δύο διαφορετικές προσεγγίσεις.

Η πρώτη προσέγγιση αφορά την οριζόντια μετακίνηση στην K-ary δομή χωρίς τη χρήση των ακμών υποδέντρου. Αυτό θα απαιτούσε ένα γραμμικό αριθμό βημάτων, όπως διατυπώνεται παρακάτω:

$$H = k^s \Rightarrow O(k^s) \Rightarrow O(k^{\log_k[N(k-1)+1]}) \Rightarrow$$
$$O(N(k-1)+1) \Rightarrow O(Nk) \Rightarrow O(N)$$

Η δεύτερη προσέγγιση αφορά την κάθετη μετακίνηση στην τοπολογία της δομής μας, ανεβαίνοντας και κατεβαίνοντας το K-ary δέντρο προτού μετακινηθούμε οριζόντια. Εάν u είναι ο αριθμός των κάθετων μετακινήσεων, τότε κάθε τέτοια κίνηση μειώνει τον απαιτούμενο αριθμό βημάτων κατά ένα παράγοντα k, με την προσθήκη βέβαια 2 βημάτων, για την άνοδο και την κάθοδο πάλι στο προηγούμενο επίπεδο. Επομένως, μια επιτυχής αναζήτηση απαιτεί λογαριθμικό αριθμό βημάτων, όπως διατυπώνεται παρακάτω:

$$H = k^{s-u} + 2u \overset{u=s}{\Rightarrow} O(k^{s-s} + 2s) \Rightarrow$$
$$O(1 + 2\{\log_k[N(k-1)+1]\}) \Rightarrow$$
$$O(\log_k[N(k-1)+1]) \Rightarrow O(\log_k N(k-1)) \Rightarrow$$
$$O(\log_k Nk) \Rightarrow O(\log_k N)$$

Από τα παραπάνω μπορούμε να συμπεράνουμε ότι ο αριθμός των απαιτούμενων βημάτων μπορεί να μεταβληθεί από λογαριθμικό σε γραμμικό σε σχέση με το μέγεθος του δικτύου ανάλογα με το είδος της κίνησης που θα εφαρμόσουμε. ▪

Ο λογαριθμικός βαθμός βημάτων απαιτεί την κάθετη κίνηση στην τοπολογία του δικτύου μας και επομένως επιφέρει επιπλέον κίνηση στους κόμβους που βρίσκονται στην κορυφή της δομής μας σε σχέση με αυτούς που βρίσκονται σε χαμηλότερα επίπεδα, δημιουργώντας ανομοιόμορφη

κατανομή κίνησης. Αντίθετα, ο γραμμικός βαθμός βημάτων δεν είναι βέβαια αποδεκτός καθώς το δίκτυο μπορεί να έχει εκατομμύρια κόμβους. Παρόλα αυτά, όπως θα αποδειχθεί στην επόμενη ενότητα μέσω των εξομοιώσεών μας, ο συνδυασμός αυτών των δύο κινήσεων και η χρήση των ακμών υποδέντρων επιτρέπει τη βελτιστοποίηση στο δίκτυό μας μεταξύ της κίνησης συντήρησης και του αριθμού των βημάτων. Το χαρακτηριστικό της παραμετροποίησης του πρωτοκόλλου μας επιτρέπει να αλλάζουμε τη συμπεριφορά του ανάλογα με τις απαιτήσεις της εφαρμογής που βασίζεται σε αυτό.

9.5. Βασικοί αλγόριθμοι του δικτύου K-Umbrella

Στην ενότητα αυτή θα παρουσιάσουμε τους βασικούς αλγορίθμους δρομολόγησης που αναπτύξαμε για την αρχιτεκτονική μας, οι οποίοι περιλαμβάνουν την κατασκευή του δικτύου, την εισαγωγή ενός κόμβου, τη δημοσίευση ενός περιεχομένου, την αναζήτηση, την αποχώρηση από το δίκτυο και ένα μηχανισμό επιδιόρθωσης.

Κατά τη δημιουργία του υπερκείμενου δικτύου, ο πρώτος κόμβος που εισέρχεται δημιουργεί το νέο δίκτυο τοποθετώντας τον εαυτό του στο πρώτο επίπεδο. Καθώς νέοι κόμβοι εισέρχονται, τοποθετούνται σύμφωνα με το αναγνωριστικό τους, όπως έχει περιγραφεί στην προηγούμενη ενότητα. Ένας κόμβος χρειάζεται απλά να επικοινωνήσει με έναν υπάρχον κόμβο του συστήματος για να εισέλθει (στην διατριβή αυτή δε θα ασχοληθούμε με τέτοιους μηχανισμούς καθώς υπάρχουν ήδη αρκετές λύσεις στην βιβλιογραφία, όπως αυτούς που δώσαμε για τη δομή Umbrella). Μονάχα ο πρώτος κόμβος εισέρχεται αυτόματα ανεξάρτητα από το αναγνωριστικό του. Όλοι οι υπόλοιποι κόμβοι τοποθετούνται στο σύστημα με βάση τον αλγόριθμο εισαγωγής.

```
insert ⟨new_node⟩
if (new_node.first_lvl_digits = current_node.first_lvl_digits) then
    num = new_node.get_digit_lvl + 1
    if (current_node.has_leaf(num)) then
        current_node.leaf(num).insert(new_node)
    else
        current_node.set_leaf(new_node,num)
        new_node.get_and_inform_neighbours(current_node)
else
    dL = calculate_horizontal_distance(new_node, current_node.left)
    dR = calculate_horizontal_distance(new_node, current_node.right)
    dU = calculate_horizontal_distance(new_node, current_node.up)
    dH = min(dL,dR)
    if (dH < dU+c) then
        if (dL < dR) then
            current_node.left.insert(new_node)
        else
            current_node.right.insert(new_node)
    else
        current_node.up.insert(new_node)
```

Σχήμα 97. Ο αλγόριθμος εισαγωγής

Ο αλγόριθμος εισαγωγής φαίνεται στο Σχήμα 97 και βασίζεται στον απλό κανόνα ταυτοποίησης όλων των *l* ψηφίων σε κάθε επίπεδο *l*. Εάν αυτή η προϋπόθεση δεν ικανοποιείται, τότε ο μηχανισμός εισαγωγής προωθεί το νέο κόμβο είτε οριζόντια είτε κάθετα (με προτίμηση στην οριζόντια κίνηση για καλύτερη κατανομή κίνησης μεταξύ όλων των επιπέδων). Όταν ο κόμβος που ικανοποιεί την προϋπόθεση ευρεθεί, τότε ο μηχανισμός συνεχίζει να προωθεί την αίτηση για εισαγωγή κατακόρυφα στο ίδιο υποδέντρο ανάλογα με τα αντίστοιχα ψηφία του αναγνωριστικού του νέου κόμβου, εωσότου ευρεθεί ένα κενό φύλλο (παιδί) για να τοποθετηθεί. Αφού ευρεθεί η κατάλληλη θέση, ο νέος κόμβος ενημερώνεται για τους γείτονές του και αντιστρόφως. Επιπλέον, ο κόμβος πατέρα τοποθετεί στον νέο κόμβο όσα κλειδιά κατέχει αλλά είναι πιο κοντά (περισσότερο όμοια) με το αναγνωριστικό του νέου κόμβου.

Στον παραπάνω μηχανισμό απαιτείται μια συνάρτηση υπολογισμού της οριζόντιας απόστασης μεταξύ δύο κόμβων. Αυτή η συνάρτηση υπολογίζει τον απαιτούμενο αριθμό βημάτων που χρειάζονται για να ευρεθεί ο κόμβος

προορισμός, με βάση την οριζόντια προώθηση στο ίδιο επίπεδο, είτε μέσω των δεξιών και αριστερών γειτόνων είτε μέσω των ακμών υποδέντρων. Ο αλγόριθμος υπολογισμού αυτής της απόστασης για ένα πλήρως ανεπτυγμένο K-δέντρο δίνεται στο Σχήμα 98, όπου υπολογίζεται η απόσταση καθώς η κίνηση γίνεται προς τα δεξιά. Ένας παρόμοιος αλγόριθμος χρησιμοποιείται για την περίπτωση μετακίνησης προς τα αριστερά (λόγω ομοιότητας παραλείπεται).

Κατόπιν αυτού του υπολογισμού, όλοι οι μηχανισμοί στο μοντέλο μας επιλέγουν εάν θα προωθήσουν οριζόντια ή κατακόρυφα με βάση μια μεταβλητή *c*, η οποία στην υλοποίηση μας είναι πάντα ίση με τον αριθμό των μέγιστων ακμών υποδέντρων *m*. Πρέπει να επισημάνουμε πως κατά τον υπολογισμό της οριζόντιας απόστασης, πολλαπλασιάζουμε το αποτέλεσμα με έναν παράγοντα *h*, τον οποίο αποκαλούμε οριζόντια διόρθωση. Ο παράγοντας αυτός εξομοιώνει την επίδραση ενός μη-πλήρους K-δέντρου, καθώς στον αλγόριθμό μας υποθέτουμε ότι το δέντρο είναι πλήρως ανεπτυγμένο. Ο παράγοντας αυτός έχει τεθεί στο 0.91, το οποίο έχει ευρεθεί από το μέσο όρο μιας σειράς εξομοιώσεων.

```
calculate_horizontal_distance_right (new_node)
key = new_node.id  nd = current_node.id
while ( not same_up_to(nd , key , level-1) ) do steps++
  if (nd[level-1]⊂(0,k-1)) then nd[level-1]++ else
    if (nd[level-1]==k-1) then nd[level-1]=0  level = level-2
      while (level ≥ 0 && nd[level]==k-1) do
        nd[level-1]=0  level-  if (nd[level-1]≠k-1) then nd[level-1]++
    else
      if (nd[level-2] ≠0) then
        nd[level-1]=1  dist_A =calculate_distance(nd,key,level-1)  nd[level-1]=k-1  dist_B = calculate_distance(nd,key,level-1)
        if (dist_A<dist_B) then nd[level-1]=1
      else
        nd[level-1]=1  nd[level-2]=0  dist_A =calculate_distance(nd,key,level-1)  nd[level-1]=k-1
        dist_B = calculate_distance(nd,key,level-1)  if (dist_A<dist_B) then nd[level-1]=1
        nd[level-1]=k-1  nd[level-2]=k-1  dist_C = calculate_distance(nd,key,level-1)
        if (dist_A<dist_B) then
          if (dist_A<dist_C) then nd[level-1]=1  nd[level-2]=0
        else if (dist_B<dist_C) then nd[level-2]=0
        level=level-2  dupl=nd
        while (nd[level]==0) do for (1≤i≤current_node.level-level)
          dupl[level-i]=k-1  dist_A =calculate_distance(dupl,key,level-1)  dist_B =calculate_distance(nd,key,level-1)
          if (dist_A<dist_B) then nd=dupl
          if ( (level>0) && (m-(current_node.level-level)>0) ) then level— else break
```

Σχήμα 98. Η συνάρτηση οριζόντιας απόστασης

Ο αλγόριθμος δημοσίευσης είναι παρόμοιος με αυτόν της εισαγωγής και περιλαμβάνει την προώθηση προς κατώτερα επίπεδα εάν όλα τα *l* ψηφία είναι ίδια, όπου *l* είναι το επίπεδο στο οποίο ανήκει ο κόμβος. Εάν όχι, τότε το μήνυμα δημοσίευσης προωθείται είτε οριζόντια είτε προς τα άνω. Όταν ικανοποιηθεί αυτή η συνθήκη, το κλειδί προωθείτε προς κατώτερα επίπεδα, εωσότου δεν είναι δυνατή η εύρεση κατάλληλου παιδιού για περεταίρω προώθηση, οπότε δημοσιεύεται στον παρόν κόμβο. Ο κώδικας για το μηχανισμό δημοσίευση δίνεται στο Σχήμα 99.

```
publish ⟨key⟩
if (node.first_lvl_digits = key.first_lvl_digits) then
        num = new_node.get_digit_lvl + 1
        if (node. has_leaf⟨num⟩) then node.leaf⟨num⟩.publish⟨key⟩
        else node.publish_here⟨key⟩
else
        d_L = calculate_horizontal_distance_left⟨key⟩
        d_R = calculate_horizontal_distance_right⟨key⟩
        d_U = calculate_horizontal_distance_up⟨key⟩
        d_H = min⟨d_L,d_R⟩
        if (d_H < d_U +c) then
                if (d_L < d_R) then node.left.publish⟨key⟩
                else node.right.publish⟨key⟩
        else node.up.publish⟨key⟩
```

Σχήμα 99. Ο αλγόριθμος δημοσίευσης

Ο μηχανισμός αναζήτησης είναι σχεδόν ίδιος με αυτόν της δημοσίευσης και περιλαμβάνει την προώθηση των μηνυμάτων αναζήτησης είτε οριζόντια είτε κάθετα εωσότου ευρεθεί το κατάλληλο κλειδί. Ο αλγόριθμος δίνεται αναλυτικά στο Σχήμα 100.

```
lookup ⟨key⟩
if (node.search_local_keys_for⟨key⟩ = true ) then
|       return doc_containing_key
else
        if (node.first_lvl_digits = key.first_lvl_digits) then
        |       num = key.get_digit_lvl + 1
        |       if (node. has_leaf⟨num⟩) then node.leaf⟨num⟩.lookup⟨key⟩
        |       else return false
        else
        |       d_L = calculate_horizontal_distance_left⟨key⟩
        |       d_R = calculate_horizontal_distance_right⟨key⟩
        |       d_U = calculate_horizontal_distance_up⟨key⟩
        |       d_H = min⟨d_L, d_R⟩
        |       if (d_H < d_U + c) then
        |       |       if (d_L < d_R) then node.left.lookup⟨key⟩
        |       |       else node.right.lookup⟨key⟩
        |       else node.up.lookup⟨key⟩
```

Σχήμα 100. Ο αλγόριθμος αναζήτησης

Ο τελευταίος μηχανισμός που υποστηρίζει το πρωτόκολλό μας είναι αυτός της εθελούσιας εξόδου. Όταν ένας κόμβος εκτελεί εθελούσια έξοδο τότε ακολουθούνται τα παρακάτω βήματα:

Βήμα 1: Εάν ο κόμβος δεν έχει παιδιά τότε όλα του τα κλειδιά προωθούνται στον πατέρα του και ενημερώνει όλους τους γείτονές του για την αποχώρηση

Βήμα 2: Εάν έχει κάποιο παιδί, τότε επιλέγει τυχαία ένα από αυτά και αντιγράφει όλα τα κλειδιά του και τις πληροφορίες γειτόνων σε αυτό προτού αποχωρήσει. Το επιλεγμένο παιδί ανεβαίνει ένα επίπεδο και αντικαθιστά τον κόμβο που αποχωρεί

Βήμα 3: Ο νέος κόμβος ενημερώνεται για όλους τους γείτονές του και ανανεώνει όλους του κόμβους ακμών υποδέντρων

Βήμα 4: Η διαδικασία συνεχίζεται από το Βήμα 2 εωσότου φτάσει σε κάποιο κόμβο χωρίς παιδιά

Οι αλγόριθμοι που παρουσιάσαμε έως αυτό το σημείο είναι σε θέση να διατηρούν το σύστημα σε ομαλή λειτουργία κάτω από φυσιολογικές συνθήκες, όπως αποδείχτηκε από μια σειρά εξομοιώσεων που κάναμε, οι οποίες θα παρουσιαστούν στην επόμενη ενότητα. Το σύστημα όμως υπόκειται και σε αποχωρήσεις κόμβων, είτε εθελούσιες είτε λόγω προβλημάτων, όπως διακοπές στο δίκτυο ή διακοπή λειτουργίας των κόμβων. Οι εθελούσιες αποχωρήσεις διαχειρίζονται μέσω του αλγόριθμου αποχώρησης και ως εκ τούτου θα επικεντρωθούμε στις απότομες αποχωρήσεις κόμβων, τις οποίες

αποκαλούμε όπως και στην περίπτωση του δικτύου Umbrella «σφάλματα κόμβων».

Η ύπαρξη των ακμών υποδέντρων και η δυνατότητα μετακίνησης τόσο οριζόντια όσο και κάθετα επιτρέπει στο σύστημά μας να παρακάμπτει σφάλματα κόμβων σε σημαντικό βαθμό. Για να αντιμετωπίσουμε το πρόβλημα των σφαλμάτων κόμβων ακόμα περισσότερο, σχεδιάσαμε ένα μηχανισμό επιδιόρθωσης (παρόμοιο με αυτόν στο δίκτυο Umbrella), ο οποίος καλείτε κάθε φορά που εντοπιστεί ένα τέτοιο σφάλμα. Ο αλγόριθμος χρησιμοποιεί το μηχανισμό εθελούσιας αποχώρησης για να επιδιορθώσει το σφάλμα σε έναν κόμβο παιδί. Μπορεί να αποδειχθεί (όπως δείξαμε στο δίκτυο Umbrella) πως όλες οι περιπτώσεις σφαλμάτων μπορούν να μετασχηματιστούν σε σφάλμα παιδιού μέσω της χρήσης των συνδέσεων στον πίνακα γειτόνων και την προώθηση του μηνύματος επιδιόρθωσης σε υψηλότερα ή χαμηλότερα επίπεδα εωσότου εντοπιστεί ο κόμβος πατέρα του κόμβου που παρουσιάζει το σφάλμα. Όταν εντοπιστεί ο κατάλληλος κόμβος και ειδοποιηθεί για το σφάλμα του παιδιού του, ο μηχανισμός επιδιόρθωσης καλείτε και επιδιορθώνει το σφάλμα αντικαθιστώντας τον κόμβο που σφάλει με ένα από τα παιδιά του ή απλά με τη διαγραφή του εάν δεν υπάρχουν παιδιά. Ο αλγόριθμος περιγράφεται στο Σχήμα 101. Κάθε κόμβος είναι υπεύθυνος να ελέγχει τον πίνακα γειτόνων του περιοδικά μέσω της αποστολής μηνυμάτων (ping) σε όλες τις καταχωρήσεις στον πίνακα και την εκτέλεση του μηχανισμού επιδιόρθωσης όποτε εντοπιστεί σφάλμα.

```
repair ⟨old_kid⟩
if (node.has_other_children) then
    new_kid = node.get_other_child⟨old_kid⟩
    node.contact_neighbors_of_change⟨new_kid, old_kid⟩
    new_kid.get_new_neighbors⟨⟩
else
    node.contact_neighbors_of_change⟨old_kid⟩
```

Σχήμα 101. Ο αλγόριθμος επιδιόρθωσης

9.6. Σύνοψη - Συμπεράσματα

Στο κεφάλαιο αυτό εισάγαμε την έννοια του δικτύου K-Umbrella που αποτελεί επέκταση του Umbrella με αρκετές όμως αλλαγές που προσφέρει επιπλέον δυνατότητες παραμετροποίησης του δικτύου για καλύτερη κατανομή της κίνησης και έλεγχο της συμπεριφοράς του δικτύου σε σχέση με

την παραγόμενη κίνηση και τη δρομολόγηση των μηνυμάτων. Αρχικά περιγράψαμε τη δομή του K-δένδρου που χρησιμοποιήσαμε καθώς και την έννοια των ακμών υποδένδρων. Εν συνεχεία περιγράψαμε τον τρόπο κατανομής των κλειδιών και τον πίνακα γειτόνων, προτού αναλύσουμε θεωρητικά την πολυπλοκότητα και αναμενόμενη απόδοση του συστήματος. Μέσα από τη θεωρητική ανάλυση, αποδείξαμε ότι το δίκτυό μας μπορεί να δρομολογήσει σε γραμμικό έως λογαριθμικό αριθμό βημάτων με βάση τις παραμέτρους που ορίζουμε, δηλαδή την επιλογή των αριθμών των παιδιών, των ακμών υποδέντρων και την επιλογή οριζόντιας ή κάθετης κίνηση στην δενδρική δομή. Τέλος παραθέσαμε όλους τους βασικούς αλγορίθμους του δικτύου (εισαγωγή, δημοσίευση, αναζήτηση και έξοδο) καθώς και τον ψευδοκώδικα για κάθε έναν από αυτούς.

9.7. Αναφορές

[1] Alima, L., El-Ansary, S., Brand, P., Haridi, S., "DKS(N,k,f): a family of low communication, scalable and fault-tolerant infrastructures for P2P applications", 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid, p 344-350, 2003.

[2] Milojicic, D., Kalogeraki, V., Lukose, R., Nagaraja, K., Pruyne, J., Richard, B., Rollins, S., Xu, Z., "Peer-to-Peer Computing", Report nr HPL-2002-57R1, 2002.

[3] Karger, D., Lehman, E., Leighton, F., Levine, M., Lewin, D., Panigrahy, R., "Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web", 29th Annual ACM Symposium on Theory of Computing, El Paso, TX, p 654-663, 1997.

10 ΑΠΟΤΕΛΕΣΜΑΤΑ ΤΟΥ K-UMBRELLA

Στην ενότητα αυτή θα εξετάσουμε τη συμπεριφορά του δικτύου K-Umbrella με βάση τις κεντρικές παραμέτρους του δικτύου, μέσω μιας σειράς εξομοιώσεων. Κατά τη διάρκεια των εξομοιώσεων θέσαμε διαφορετικές τιμές για το μέγιστο αριθμό παιδιών *k* και για το μέγιστο επιτρεπόμενο αριθμό ακμών υποδέντρων *m*. Για να κατανοήσουμε καλύτερα τη συμπεριφορά του δικτύου διαχωρίσαμε τις εξομοιώσεις σε δύο βασικά σετ. Στο πρώτο σετ μεταβάλλαμε τον αριθμό των παιδιών και των ακμών υποδέντρων και αναλύσαμε την επίδρασή τους στην απόδοση του δικτύου. Κατά τη διάρκεια του δεύτερου σετ εξομοιώσαμε το δίκτυό μας με βάση την κατανομή περιεχομένου, εισαγωγής και διάρκειας των συνδέσεων με βάση την ανάλυση των Klemm κ.α. [1], με στόχο να επιβεβαιώσουμε πως το δίκτυό μας προωθεί τους σταθερούς χρήστες.

10.1. Πρώτη σειρά εξομοιώσεων

Στην πρώτη σειρά πειραμάτων εξετάσαμε την επίδραση των δύο μεταβλητών στην αρχιτεκτονική μας, του μέγιστου αριθμού παιδιών k και του αριθμού των επιτρεπόμενων υποδέντρων m για κάθε κόμβο. Αρχικά εξετάσαμε την επίδραση κάθε μιας μεταβλητής στον αριθμό των συνολικών μηνυμάτων ανά κόμβο κατά τη διάρκεια ενός πλήρους πειραματικού κύκλου ζωής, με μεταβλητό αριθμό κόμβων. Στην συνέχεια αυτής της διατριβής, όποτε αναφερόμαστε σε πειραματικό κύκλο ζωής για ένα συγκεκριμένο πληθυσμό N, θα αναφερόμαστε σε ένα πείραμα με τα χαρακτηριστικά που δίνονται στον Πίνακας 14. Επιπλέον, όλες οι τιμές που παρουσιάζονται αναφέρονται σε μέσους όρους από την εκτέλεση κάθε πειράματος 10 φορές, ώστε να μειώσουμε τα τυχαία σφάλματα.

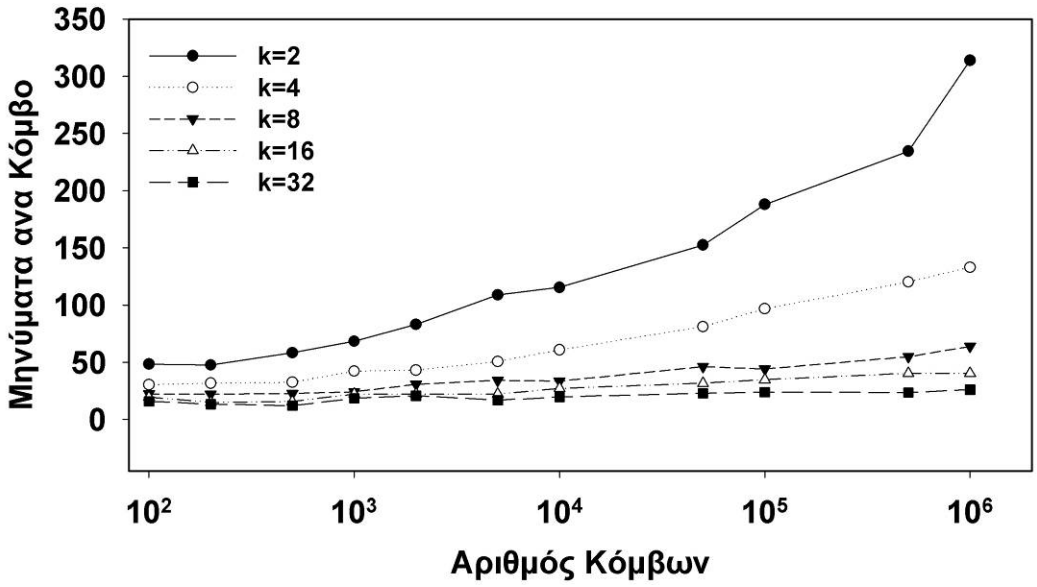
Στο Σχήμα 102 σχεδιάσαμε το συνολικό αριθμό μηνυμάτων ανά κόμβο, για όλες τις βασικές λειτουργίες, σε σχέση με τον αριθμό των κόμβων, που κυμαίνεται από 1.000 έως 100.000. Στο Σχήμα 102 (α) μεταβάλαμε τον αριθμό των επιτρεπόμενων παιδιών από 2 έως 32 για το μέσο όρο όλων των διαφορετικών συνδυασμών της παραμέτρου m. Στο Σχήμα 102 (β)

σχεδιάσαμε το αντίστροφο, μεταβάλλοντας την παράμετρο m από 1 έως 64 και παίρνοντας το μέσο όρο για την παράμετρο k. Πρέπει να επισημάνουμε πως στα σχήματα που μεταβάλλεται ο πληθυσμός, ο άξονας είναι σε λογαριθμική κλίμακα.

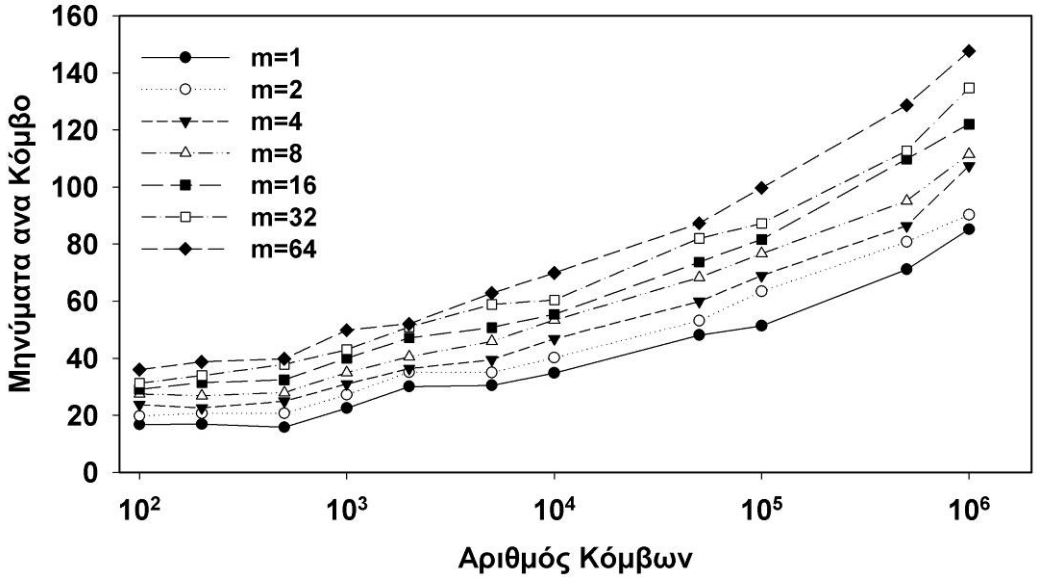
Λειτουργία	**Πληθυσμός / Κατανομή**	
Αριθμός κόμβων που εισέρχονται	N	
Αριθμός κόμβων που δημοσιεύουν	N	
Αριθμός εγγράφων ανά κόμβο	70% του N	Ομοιόμορφα(0 , N∗0,0001)
	20% του N	Ομοιόμορφα (0 , N∗0,001)
	7% του N	Ομοιόμορφα (0 , N∗0,002)
	3% του N	Ομοιόμορφα (0 , N∗0,01)
Αριθμός κλειδιών ανά έγγραφο	Ομοιόμορφα (1 , 10)	
Αριθμός αναζητήσεων ανά κόμβο	LogNormal Κατανομή	σ = 1,306 μ = 0,520

Πίνακας 14. Πειραματικός κύκλος ζωής για πληθυσμό N κόμβων

Από αυτά τα δύο σχήματα μπορούμε να παρατηρήσουμε πως ο μέσος αριθμός μηνυμάτων ανά κόμβο μειώνεται καθώς ο μέγιστος αριθμός παιδιών αυξάνει και ο αριθμός των ακμών υποδέντρων μειώνεται. Και οι δύο αυτές τάσεις ακολουθούν λογαριθμική τάξη. Η συμπεριφορά αυτή αποδεικνύει την αρχική μας δήλωση πως ο συνδυασμός οριζόντιας και κάθετης κίνησης μαζί με τη χρήση των ακμών υποδέντρων επιτρέπει στο πρωτόκολλό μας να δρομολογεί σε $O(\log_k N)$ υπερκείμενα βήματα.



(α)

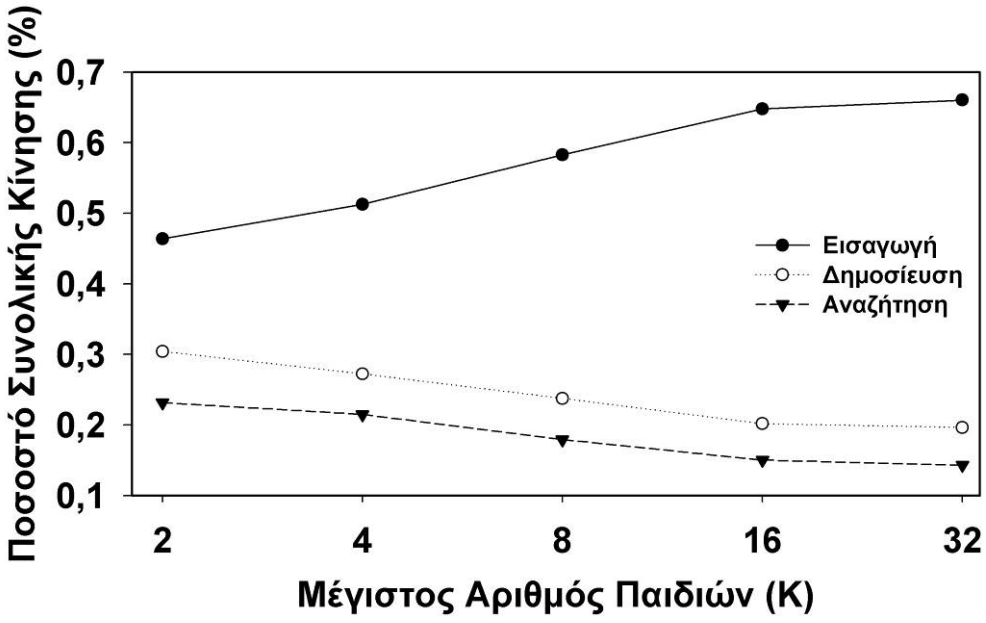


(β)

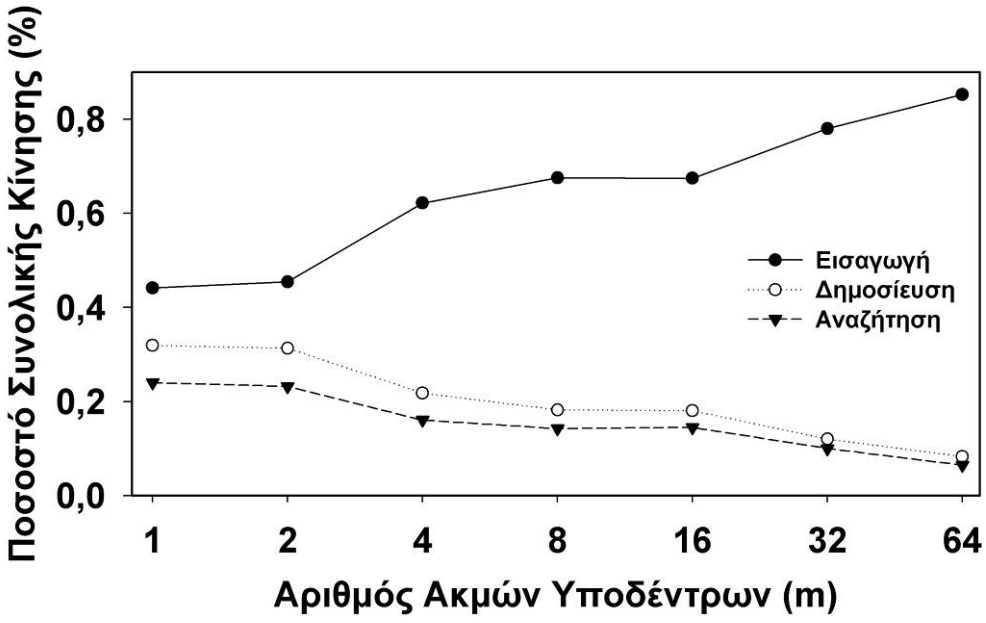
Σχήμα 102. Μηνύματα ανά κόμβο για διαφορετικούς πληθυσμούς ως προς (α) το μέγιστο αριθμό παιδιών και (β) τον αριθμό των ακμών υποδέντρων

Για να αναλύσουμε καλύτερα τα παραπάνω αποτελέσματα, διαχωρίσαμε τα ποσοστά της παραγόμενης κίνησης ανά λειτουργία. Όπως φαίνεται στο Σχήμα 103 (α), καθώς ο αριθμός των επιτρεπόμενων παιδιών αυξάνει, το μεγαλύτερο ποσοστό κίνησης δημιουργείται από το μηχανισμό εισαγωγής, ενώ ο μηχανισμός δημοσίευσης και κυρίως ο μηχανισμό αναζήτησης είναι υπεύθυνοι για όλο και μικρότερο ποσοστό της κίνησης. Η ίδια τάση παρατηρείται και για τον αριθμό των ακμών υποδέντρων, όπως φαίνεται στο Σχήμα 103 (β). Αυτή η συμπεριφορά του πρωτοκόλλου θεωρείται

επιθυμητή καθώς κάθε κόμβος καλεί τους μηχανισμούς εισαγωγής και δημοσίευσης μόνο μια φορά, ενώ ο μηχανισμός αναζήτησης μπορεί να χρησιμοποιηθεί πολλαπλές φορές κατά τη διάρκεια συνόδου ενός κόμβου.



(α)

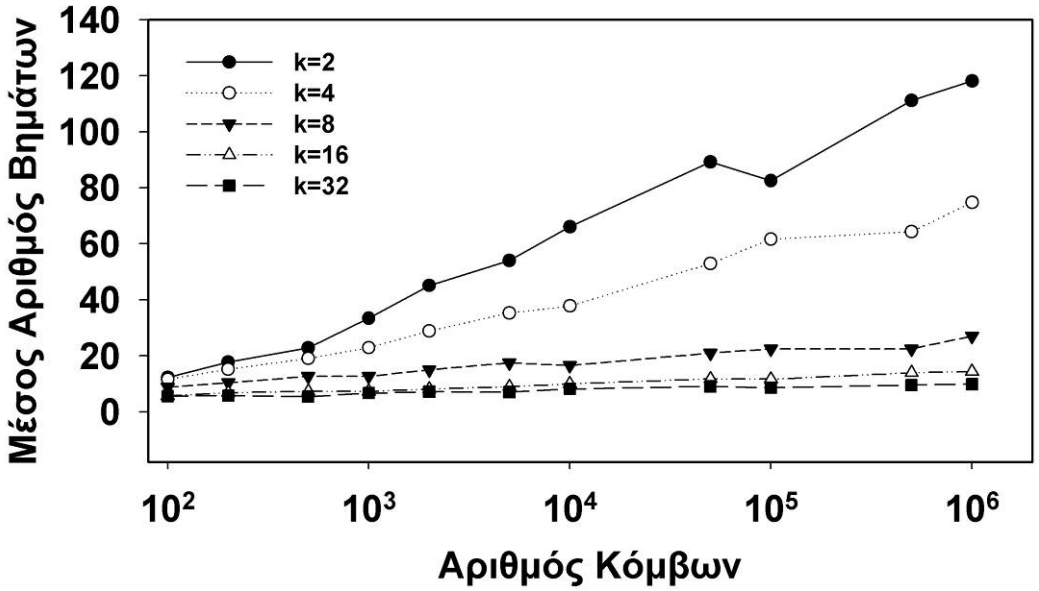


(β)

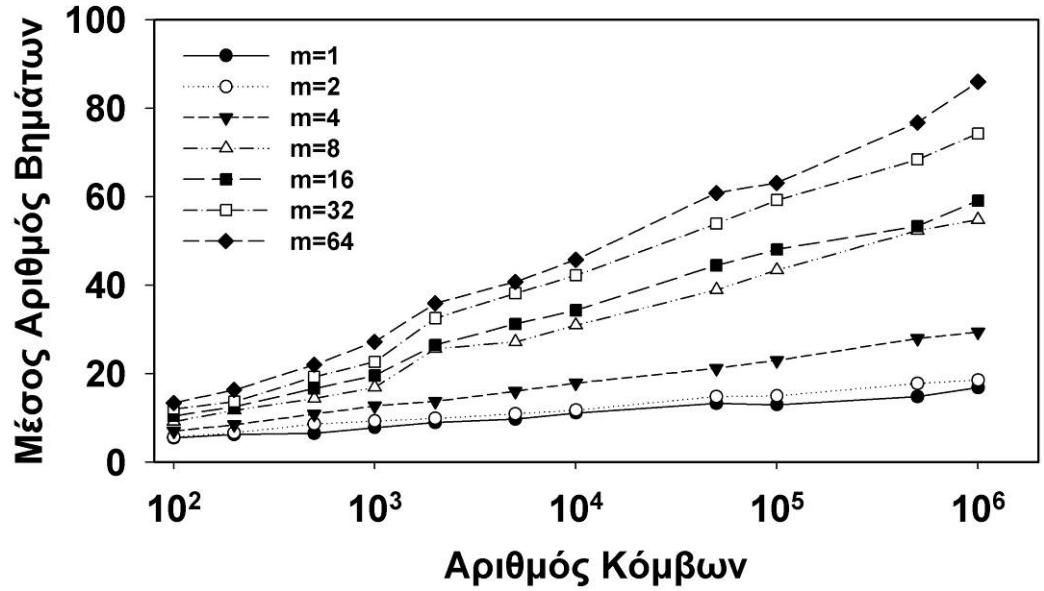
Σχήμα 103. Ποσοστό συνολικής κίνησης ανά λειτουργία για διαφορετικό αριθμό (α) παιδιών και (β) ακμών υποδέντρων

Στα επόμενα σχήματα παρουσιάζουμε το μέσο αριθμό βημάτων για όλες τις λειτουργίες (εισαγωγή, δημοσίευση, αναζήτηση) για διαφορετικούς πληθυσμούς κόμβων και παρατηρούμε την επίδραση των δύο παραμέτρων του K-Umbrella. Στο πρώτο σχήμα, Σχήμα 104 (α), παρουσιάζουμε το μέσο

αριθμό βημάτων για διαφορετικό αριθμό παιδιών. Όπως φαίνεται, καθώς ο αριθμός των παιδιών αυξάνει ο μέσος αριθμός βημάτων μειώνεται λογαριθμικά, όπως ακριβώς αναμενόταν με βάση τη θεωρητική μας ανάλυση, καθώς ο αναμενόμενος αριθμός βημάτων είναι $O(\log_k N)$. Στο Σχήμα 104 (β) παρατηρούμε την επίδραση των ακμών υποδέντρων m στον μέσο αριθμό βημάτων. Μια αύξηση στον αριθμό υποδέντρων m αυξάνει λογαριθμικά το μέσο αριθμό των βημάτων. Αυτή η συμπεριφορά οφείλεται κυρίως στην αρχική μας επιλογή να θέσουμε τον αριθμό των ακμών υποδέντρων m ίσο με την παράμετρο c, με βάση την οποία ο αλγόριθμος επιλέγει αν θα δρομολογήσει κάθετα ή οριζόντια. Επομένως, μια αύξηση στην παράμετρο m μεταφράζεται σε περισσότερα οριζόντια βήματα σε σχέση με κάθετα. Αυτό έχει ως αποτέλεσμα την αύξηση μεν των βημάτων αλλά και την αποκεντροποίηση παράλληλα της κίνησης από τους κόμβους των υψηλότερων επιπέδων προς τους κόμβους χαμηλότερων, προσφέροντας πιο ομοιόμορφη κατανομή.



(α)

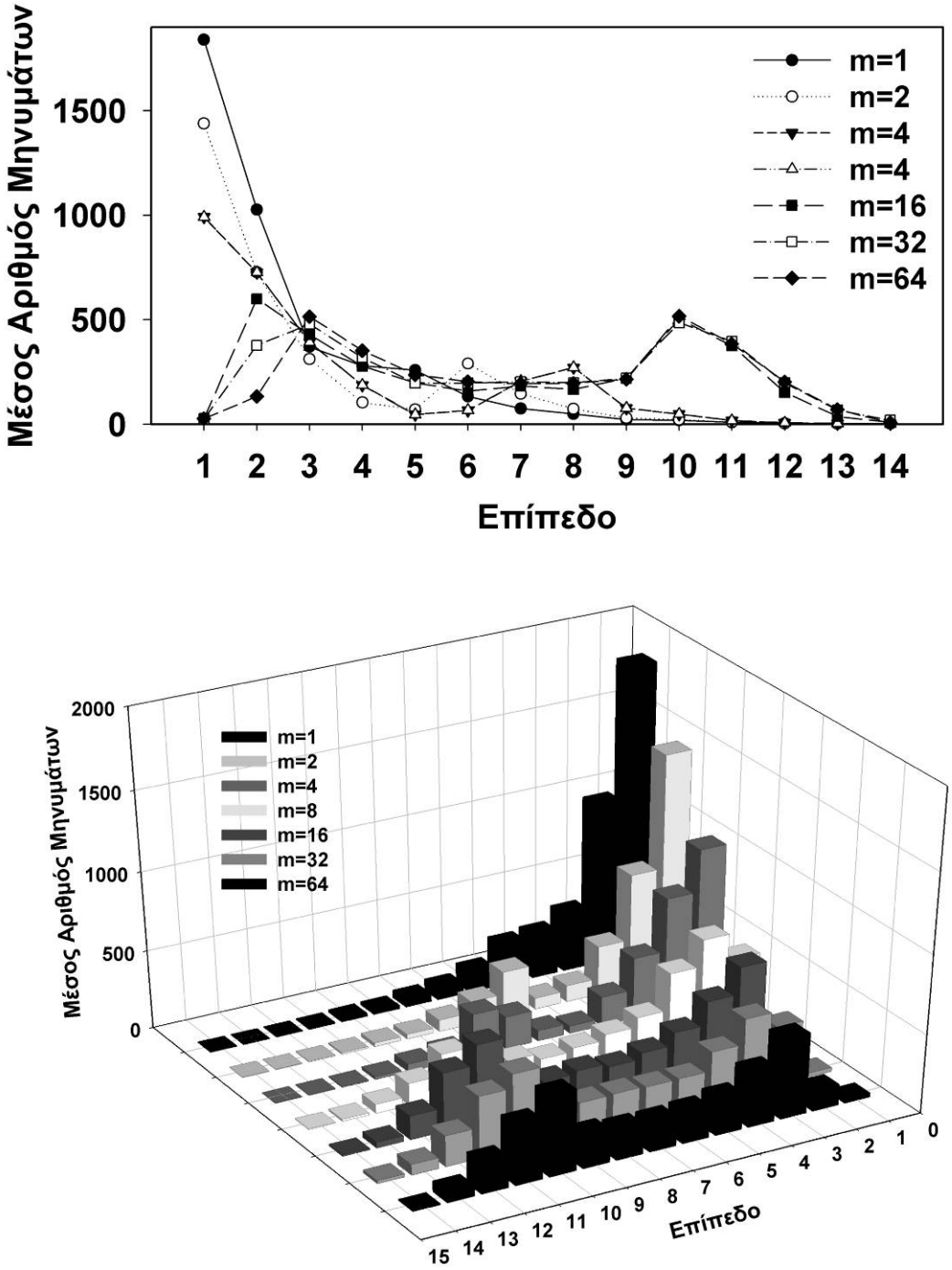


(β)

Σχήμα 104. Μέσος αριθμός βημάτων για όλες τις λειτουργίες για διαφορετικό αριθμό (α) παιδιών και (β) ακμών υποδέντρων

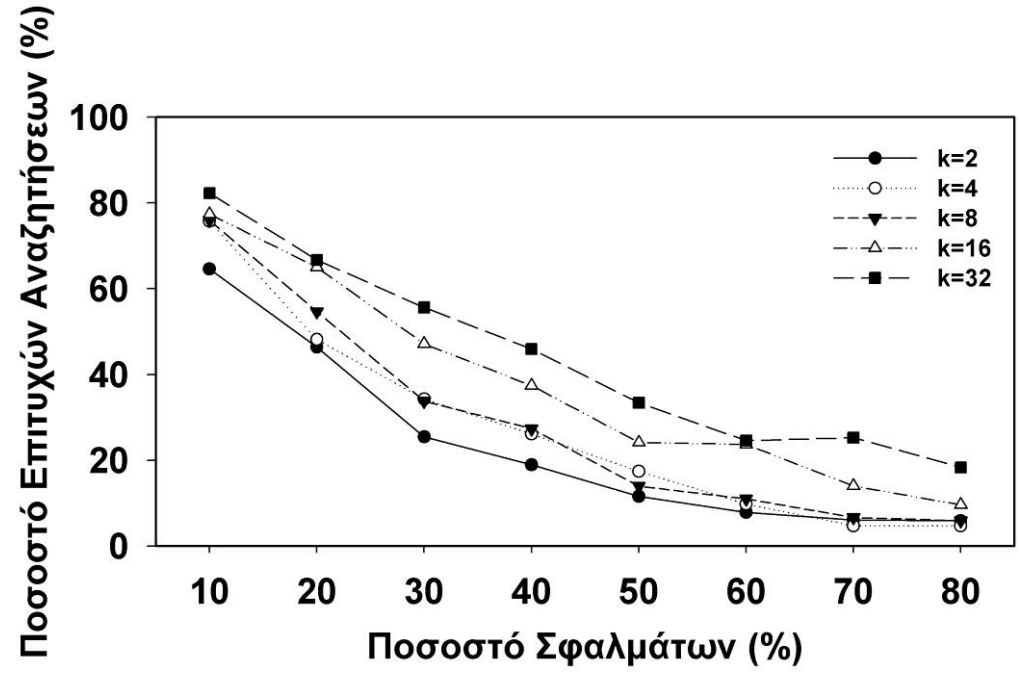
Στο Σχήμα 105 αναλύσαμε την επίδραση του αριθμού επιτρεπόμενων ακμών υποδέντρων στην κατανομή των μηνυμάτων ανά επίπεδο. Όπως είδαμε στο προηγούμενο σχήμα, η παράμετρος αυτή αυξάνει μεν το μέσο αριθμό βημάτων αλλά επιτρέπει μια πιο ομοιόμορφη κατανομή της κίνησης σε όλους τους κόμβους. Στο σχήμα παρουσιάζουμε την κατανομή των μηνυμάτων ανά επίπεδο για πείραμα 2.000 κόμβων και για διαφορετικούς αριθμούς ακμών υποδέντρων. Όπως παρατηρούμε, καθώς ο αριθμός των

ακμών υποδέντρων αυξάνει, η κατανομή μετατοπίζεται από τα υψηλά επίπεδα σε όλο το φάσμα του δικτύου μας, παρουσιάζοντας μια σχεδόν ομοιόμορφη κατανομή. Παρατηρούμε επίσης πως ένας αριθμός ακμών υποδέντρων ίσο με 8 είναι ικανός να παρέχει ικανοποιητική αποκεντροποίηση χωρίς ωστόσο να επιβαρύνει υπερβολικά τη συνολική κίνηση (με βάση το Σχήμα 104 (β)).

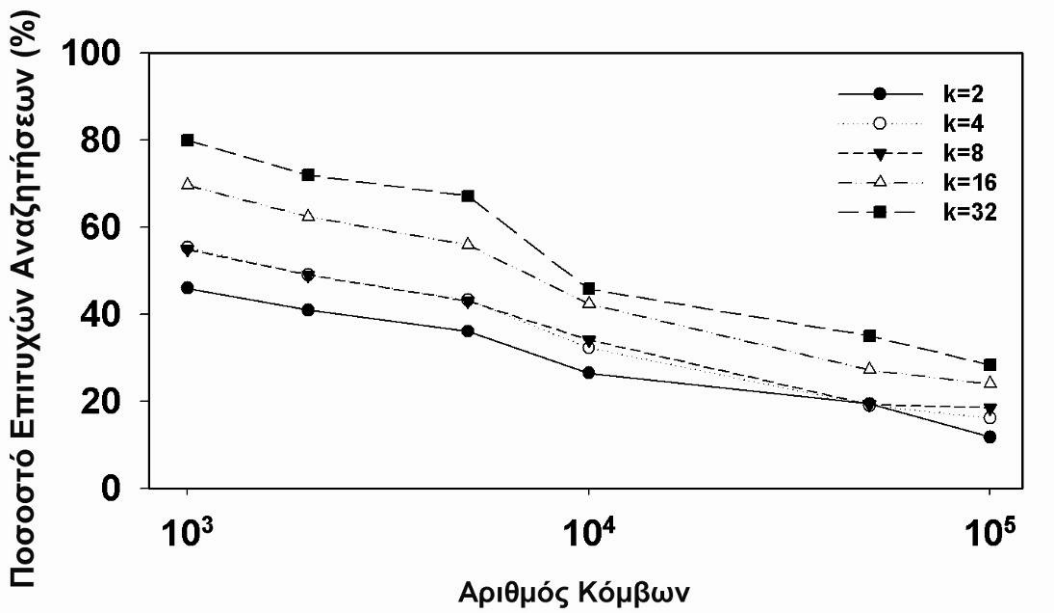


Σχήμα 105. Κατανομή μηνυμάτων για διαφορετικό αριθμό επιτρεπόμενων ακμών υποδέντρων

Εν συνεχεία, εξετάσαμε την επίδραση των δύο παραμέτρων στα ποσοστά επιτυχίας αναζητήσεων καθώς οι κόμβοι στο σύστημα αρχίζουν να παρουσιάζουν σφάλματα. Σε όλα τα προηγούμενα πειράματα, δεν εξομοιώσαμε σφάλματα κόμβων και για το λόγο αυτό όλες οι αναζητήσεις ήταν επιτυχείς (και για αυτό δεν εξετάσαμε αυτό το κριτήριο). Σε αυτήν τη σειρά πειραμάτων προκαλέσαμε σταδιακά σφάλματα από το 10% του πληθυσμού έως και το 80%. Για λόγους παρουσίασης, σε όλα τα επόμενα σχήματα αυτής της ενότητας, παρουσιάζουμε τις μέσες τιμές όλων των παραμέτρων εκτός αυτών που εξετάζουμε. Όπως φαίνεται στο Σχήμα 106 (α), το ποσοστό επιτυχίας ανεβαίνει καθώς αυξάνει ο αριθμός των επιτρεπόμενων παιδιών. Επίσης, από το Σχήμα 106 (β) βλέπουμε πως καθώς ο πληθυσμός αυξάνει τα ποσοστά επιτυχίας μειώνεται μεν αλλά πάντα τα δίκτυα με μεγαλύτερο αριθμό επιτρεπόμενων παιδιών παρουσιάζουν υψηλότερα ποσοστά επιτυχίας. Πρέπει σε αυτό το σημείο να επισημάνουμε ότι σε αυτήν τη σειρά των εξομοιώσεων δεν χρησιμοποιήθηκε ο μηχανισμός επιδιόρθωσης ή οι επεκτάσεις του πρωτοκόλλου (εικονικά δίκτυα και σχήματα αντιγράφων) και για αυτόν το λόγο η απόδοση που παρουσιάζει το πρωτόκολλο θεωρείται παραπάνω από ικανοποιητική (καθώς όπως είδαμε και κατά την εξέταση του πρωτοκόλλου Umbrella, τα ποσοστά αυτά βελτιώνονται σημαντικά με τη χρήση των τριών αυτών μηχανισμών).



(α)

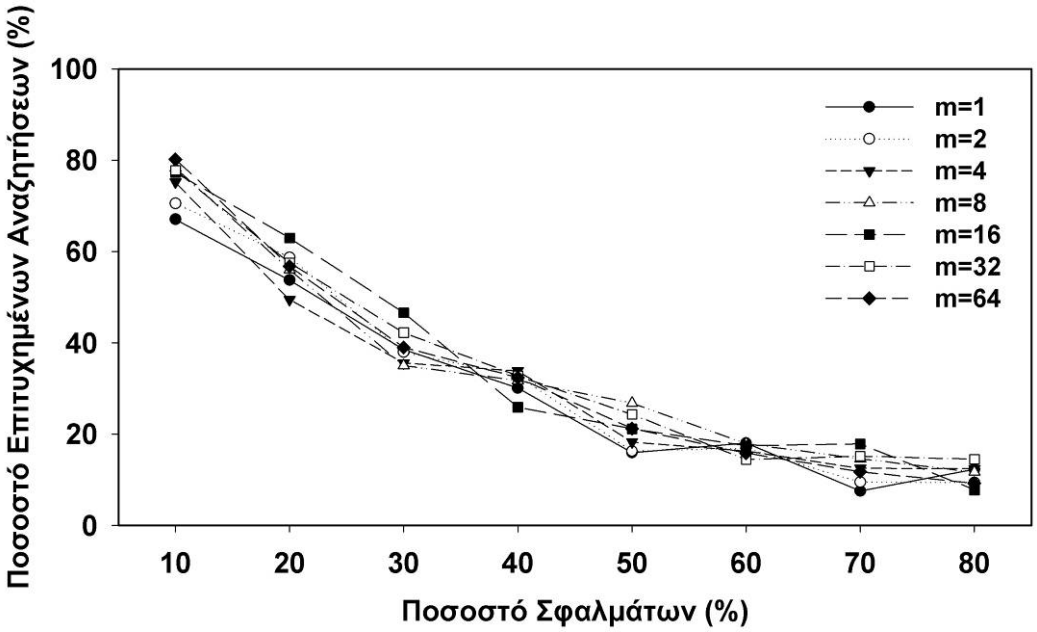


(β)

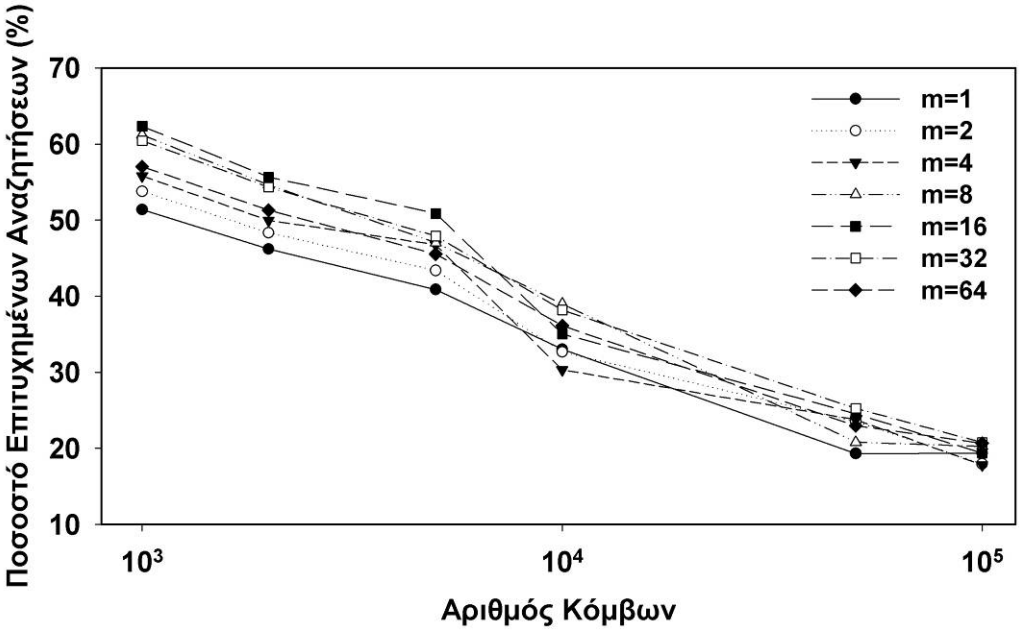
Σχήμα 106. Ποσοστό επιτυχημένων αναζητήσεων για διαφορετικό αριθμό παιδιών ως προς (α) το ποσοστό σφαλμάτων και (β) τον πληθυσμό του δικτύου

Στο Σχήμα 107 παρατηρούμε την επίδραση της δεύτερης παραμέτρου, αυτής του αριθμού ακμών υποδέντρων, στα ποσοστά επιτυχών αναζητήσεων. Όπως παρατηρούμε από το Σχήμα 107 (α) και Σχήμα 107 (β) τα ποσοστά επιτυχίας είναι κατά ένα μικρό ποσοστό βελτιωμένα για υψηλότερο αριθμό ακμών υποδέντρων τόσο για μεταβλητό αριθμό κόμβων που παρουσιάζουν σφάλμα όσο και για διαφορετικούς αριθμούς πληθυσμών. Παρόλα αυτά, η

διαφορά αυτή είναι πάρα πολύ μικρή και όχι πάντα σταθερή και επομένως μπορούμε να συμπεράνουμε ότι τα ποσοστά επιτυχίας δεν επηρεάζονται ιδιαίτερα από τον αριθμό ακμών υποδέντρων.



(α)



(β)

Σχήμα 107. Ποσοστό επιτυχημένων αναζητήσεων για διαφορετικό αριθμό επιτρεπόμενων ακμών υποδέντρων ως προς (α) το ποσοστό σφαλμάτων και (β) τον πληθυσμό του δικτύου

Από το σύνολο των παραπάνω εξομοιώσεις παρατηρήσαμε πως καθώς ο αριθμός των ακμών υποδέντρων αυξάνει επέρχεται αύξηση τόσο του αριθμού μηνυμάτων ανά κόμβο όσο και του μέσου αριθμού βημάτων αλλά

και καλύτερη κατανομή της κίνησης σε όλα τα επίπεδα του δικτύου. Επομένως, η βέλτιστη λύση βρίσκεται σε ένα μεσαίο αριθμό ακμών υποδέντρων μεταξύ 4 και 8. Όσον αφορά τη δεύτερη παράμετρο, παρατηρήσαμε ότι καθώς αυξάνει ο αριθμός των επιτρεπόμενων παιδιών τόσο ο αριθμός των μηνυμάτων ανά κόμβο όσο και ο μέσος αριθμός μηνυμάτων μειώνονται. Σε σχέση με τη συμπεριφορά του δικτύου σε συνθήκες σφαλμάτων κόμβων, παρατηρήσαμε ότι οι υψηλές τιμές για επιτρεπόμενο αριθμό παιδιών βελτιώνει την ανεκτικότητα του συστήματος ενώ ο αριθμός των ακμών υποδέντρων δεν επηρεάζει ιδιαίτερα την απόδοση. Επομένως είναι προφανές ότι το σύστημά μας λειτουργεί καλύτερα όταν ο αριθμός των παιδιών αυξάνει. Επιπλέον, η ύπαρξη των δύο αυτών παραμέτρων επιτρέπει στο πρωτόκολλό μας να παραμετροποιείται κατάλληλα ώστε να εξυπηρετεί διαφορετικές δικτυακές απαιτήσεις, ανάλογα με τις υπερκείμενες εφαρμογές.

10.2. Δεύτερη σειρά εξομοιώσεων

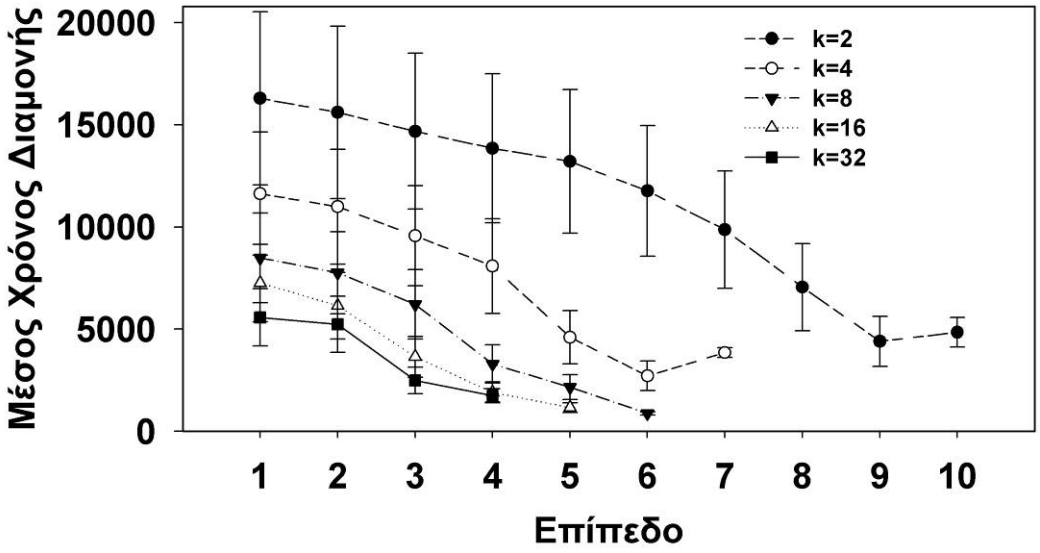
Στην δεύτερη σειρά εξομοιώσαμε το δίκτυό μας χρησιμοποιώντας την κατανομή για περιεχόμενο, αφήξεις και διάρκεια των ομότιμων συνδέσεων με βάση την ανάλυση των Klemm κ.α. [1] ώστε να επιβεβαιώσουμε ότι το πρωτόκολλό μας προωθεί τους χρήστες που παρουσιάζουν σταθερή συμπεριφορά. Στον Πίνακας 15 παρουσιάζουμε τις ακριβείς παραμέτρους της διάταξής μας σύμφωνα με τα αποτελέσματα των Klemm κ.α. για τις χρονικές περιόδους αιχμής για Βόρεια Αμερική, που είναι η πιο απαιτητική περίοδος όσον αφορά την κίνηση από όλες τι διατάξεις που παρουσιάζονται. Κατά τη διάρκεια αυτής της εξομοίωσης μεταβάλλαμε τον πληθυσμό από 100 έως 2.000 κόμβους, από τους οποίους μόνο ένα ποσοστό αυτών βρίσκεται ταυτόχρονα στο δίκτυο, με βάση τους ρυθμούς αφίξεων και αποχωρήσεων που δίνονται στον πίνακα. Λόγω της επιπλέον επεξεργασίας αλλά και περιορισμών στη μνήμη που επιφέρει ο υπολογισμός της κατανομής και των παραμέτρων για τις αφήξεις και τις συνόδους, διατηρήσαμε τον αριθμό των κόμβων χαμηλό. Αυτό όμως δεν περιορίζει την αξιοπιστία των αποτελεσμάτων μας καθώς, όπως θα δούμε, είναι αρκετά ενδεικτικά.

Λειτουργία			Πληθυσμός / Κατανομή
Εισερχόμενοι κόμβοι			N = 100,200,500,1000,2000
Ενεργοί κόμβοι (%)			Ομοιόμορφα (15%,20%)
Ανενεργοί κόμβοι Διάρκεια σύνδεσης		Σώμα	1-2 λεπτά (75%) Lognormal(σ = 2.502,μ = 2.108)
		Ουρά	> 2 λεπτά (25%) Lognormal(σ = 2.749,μ = 6.397)
Αναζητήσεις ενεργών κόμβων	# αναζητήσεων		Lognormal(σ = 1.360,μ = -0.0673)
	Χρόνος έως την πρώτη αναζήτηση	<3 αναζητήσεις	Σώμα 0-45 δευτερ. Weibull(α = 1.477, λ = 0.005252)
			Ουρά > 45 δευτερ Lognormal(σ = 2.905,μ = 5.091)
		3 αναζητήσεις	Σώμα 0-45 δευτερ Weibull(α = 1.261,λ = 0.01081)
			Ουρά > 45 δευτερ Lognormal(σ = 2.045,μ = 6.303)
		>3 αναζητήσεις	Σώμα 0-45 δευτερ Weibull(α = 0.9821,λ = 0.02662)
			Ουρά > 45 δευτερ Lognormal(σ = 2.359,μ = 6.301)
	Χρόνος μεταξύ των αφίξεων	Σώμα	<103 δευτερόλεπτα Lognormal(σ = 1.625 ,μ = 3.353)
		Ουρά	>103 δευτερόλεπτα Pareto(α = 0.9041,β = 103)
	Χρόνος μετά την τελευταία αναζήτηση	1 αναζήτηση	Lognormal(σ = 2.361,μ = 4.879)
		2-7 αναζητήσεις	Lognormal(σ = 2.259,μ = 5.686)
		>7 αναζητήσεις	Lognormal(σ = 2.145,μ = 6.107)

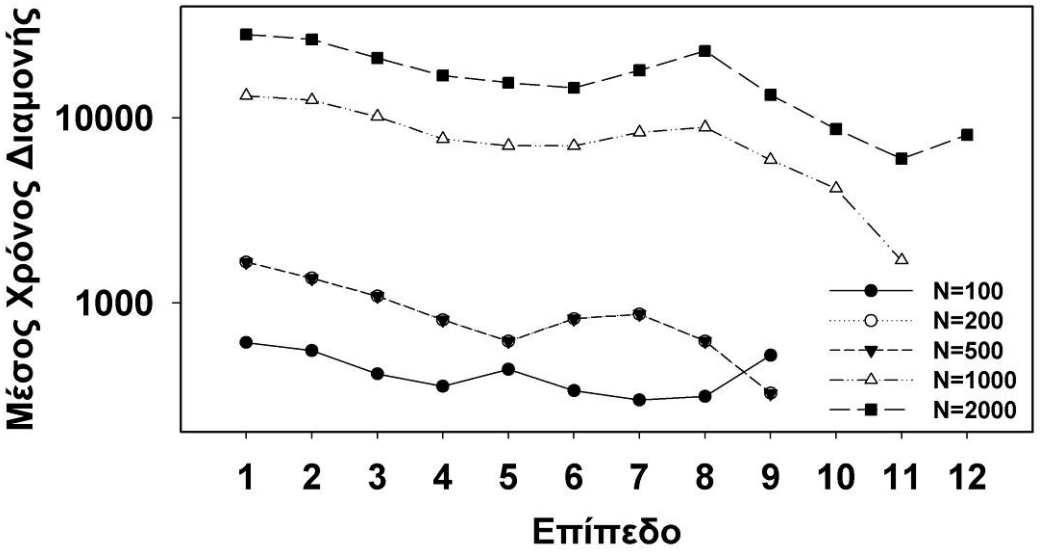
Πίνακας 15. Πειραματική διάταξη για την κατανομή των συνδέσεων

Αφού εξομοιώσαμε το δίκτυό μας, εξάγαμε πληροφορίες σχετικά με το μέσο χρόνο σύνδεσης κάθε κόμβου σε κάθε επίπεδο της K-Umbrella δομής για διαφορετικούς πληθυσμούς κόμβων και διαφορετικό αριθμό παιδιών. Όπως παρατηρούμε στο Σχήμα 108 (α), για όλους τους συνδυασμούς παιδιών, οι

κόμβοι που βρίσκονται σε υψηλότερα επίπεδα της αρχιτεκτονικής μας είναι αυτοί που παρουσιάζουν μεγαλύτερο χρόνο διαμονής στο δίκτυο. Στο σχήμα παρουσιάζουμε και τη διακύμανση με βάση τους διαφορετικούς πληθυσμούς. Στο επόμενο σχήμα, Σχήμα 108 (β), παρουσιάζουμε το μέσο χρόνο διαμονής για διαφορετικούς πληθυσμούς κόμβων (ο χρόνος σε αυτό το σχήμα δίνεται σε λογαριθμική κλίμακα). Και σε αυτήν την περίπτωση είναι προφανές ότι οι κόμβοι που βρίσκονται σε υψηλότερα επίπεδα της τοπολογίας μας μετά την ολοκλήρωση του πλήρους πειραματικού κύκλου είναι αυτοί που παρουσιάζουν το μεγαλύτερο χρόνο διαμονής. Αυτό επιβεβαιώνει την αρχική μας διατύπωση ότι το πρωτόκολλό μας προωθεί του κόμβους που παρουσιάζουν μεγαλύτερη σταθερότητα (παραμένουν συνδεδεμένοι περισσότερη ώρα). Αυτό το χαρακτηριστικό του δικτύου μας μπορεί να αξιοποιηθεί από εφαρμογές που ευνοούν τους σταθερούς χρήστες, όπως εφαρμογές ροοθήκευσης [2],[3],[4] ή ιεραρχικά συσταδοποιημένες τοπολογίες [5],[6].



(α)



(β)

Σχήμα 108. Μέσος χρόνος διαμονής ανά επίπεδο για διαφορετικό (α) αριθμό επιτρεπόμενων παιδιών και (β) πληθυσμό δικτύου

10.3. Σύνοψη - Συμπεράσματα

Στην ενότητα αυτή εξετάσαμε τη συμπεριφορά του δικτύου K-Umbrella με βάση τις κεντρικές παραμέτρους μέσω μιας σειράς εξομοιώσεων. Κατά τη διάρκεια των εξομοιώσεων θέσαμε διαφορετικές τιμές για το μέγιστο αριθμό παιδιών k και για το μέγιστο επιτρεπόμενο αριθμό ακμών υποδέντρων m. Για να κατανοήσουμε καλύτερα τη συμπεριφορά του δικτύου διαχωρίσαμε τις

εξομοιώσεις σε δύο βασικά σετ. Στο πρώτο σετ μεταβάλλαμε τον αριθμό των παιδιών και των ακμών υποδέντρων και αναλύσαμε την επίδρασή τους στην απόδοση του δικτύου. Από το σύνολο των εξομοιώσεις παρατηρήσαμε πως καθώς ο αριθμός των ακμών υποδέντρων αυξάνει επέρχεται αύξηση τόσο του αριθμού μηνυμάτων ανά κόμβο όσο και του μέσου αριθμού βημάτων αλλά ταυτόχρονα καλύτερη κατανομή της κίνησης σε όλα τα επίπεδα του δικτύου. Επομένως, συμπεράναμε ότι η βέλτιστη λύση βρίσκεται σε ένα μεσαίο αριθμό ακμών υποδέντρων μεταξύ 4 και 8. Επίσης παρατηρήσαμε ότι καθώς αυξάνει ο αριθμός των επιτρεπόμενων παιδιών τόσο ο αριθμός των μηνυμάτων ανά κόμβο όσο και ο μέσος αριθμός μηνυμάτων μειώνονται. Κατά τη διάρκεια του δεύτερου σετ εξομοιώσαμε το δίκτυό μας με βάση την κατανομή περιεχομένου, εισαγωγής και διάρκειας των συνδέσεων με βάση την ανάλυση των Klemm κ.α. [1] και επιβεβαιώσαμε πως το δίκτυό μας προωθεί τους σταθερούς χρήστες, καθώς για όλους τους συνδυασμούς παιδιών, οι κόμβοι που βρίσκονται σε υψηλότερα επίπεδα της αρχιτεκτονικής μας είναι αυτοί που παρουσιάζουν το μεγαλύτερο χρόνο διαμονής στο δίκτυο.

10.4. Αναφορές

[1] Klemm, A., Lindemann, C., Vernon, M., Waldhorst, O., "Characterizing the Query Behavior in Peer-to-Peer File Sharing Systems", ACM Internet Measurement Conference (IMC), Taormina, Italy, p 55-67, 2004.

[2] Hefeeda, M., Habib, A., Xu, D., Bhargava, B., Botev, B., "Collectcast: A peer-to-peer service for media streaming", ACM/Springer Multimedia Systems, 2003.

[3] Tran, D., Hua, K., Do, T., "A Peer-to-Peer Architecture for Media Streaming", IEEE Journal on Selected Areas in Communications 22(1): 121-133, 2004.

[4] Liao, X., Jin, H., Liu, Y., Ni, L., Deng, D., "AnySee: Peer-to-Peer Live Streaming", Barcelona, Spain, April, 2006.

[5] Hsiao, R., Wang, S., "Jelly: A Dynamic Hierarchical P2P Overlay Network with Load Balance and Locality", IEEE ICDCS 2004, p 534–540, 2004.

[6] Chung, T., Chang, Y., Liu, C., Chen, Y., "Architecture and Implementation of Cluster-based Peer-to-Peer Topology and Its Application in Search", Journal of Internet Technology (JIT), 7(1):23–34, 2006.

11 ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΑΝΟΙΧΤΑ ΕΡΕΥΝΗΤΙΚΑ ΘΕΜΑΤΑ

Στο κεφάλαιο αυτό θα αναφέρουμε μια σειρά από συμπεράσματα για την αρχιτεκτονική μας, καθώς και για τις επεκτάσεις και αλλαγές που παρουσιάσαμε και θα δώσουμε κατευθύνσεις για μελλοντικά βήματα πάνω σε ανοιχτά ερευνητικά θέματα.

11.1. Συμπεράσματα

Στην διατριβή αυτή προσπαθήσαμε να δώσουμε μια συγκεκριμένη λύση στο πρόβλημα ανταλλαγής περιεχομένου σε υπερκείμενα δίκτυα. Μέσα από μια αναλυτική βιβλιογραφική αναφορά εντοπίσαμε τα προβλήματα και τα μειονεκτήματα των υπαρχόντων συστημάτων αλλά και τα θετικά σημεία, τα οποία θέλαμε να υποστηρίξουμε και στην δική μας αρχιτεκτονική. Αυτή η έρευνα μας οδήγησε στον σχεδιασμό του δικτύου μας και στην επιλογή χρήσης μιας δενδρικής δομής για την τοποθέτηση των χρηστών. Με τη χρήση K-δέντρων, είτε στην αρχική μορφή του συστήματος Umbrella, όπου ο αριθμός των επιτρεπόμενων παιδιών ήταν σταθερός, είτε στην επέκταση της αρχιτεκτονικής με το σύστημα K-Umbrella, όπου είχαμε ένα παραμετροποιημένο K-δέντρο, καταφέραμε να οργανώσουμε απλά και αποδοτικά τους χρήστες στο υπερκείμενο δίκτυό μας.

Μέσα από θεωρητική ανάλυση αλλά και μια σειρά εξομοιώσεων, αποδείξαμε αρχικά τη σωστή και ομαλή λειτουργία της αρχιτεκτονικής Umbrella. Οι αλγόριθμοι εισαγωγής, δημοσίευσης, αναζήτησης και διαγραφής που παρουσιάσαμε ήταν σε θέση να οργανώσουν απόλυτα τους κόμβους και να διατηρούν το σύστημα διαρκώς οργανωμένο, ανεξάρτητα με τον αριθμό των κόμβων ή των περιεχομένων.

Είναι επίσης σημαντικό να επισημάνουμε την καλή απόδοση του συστήματος, το οποίο είναι σε θέση να διατηρεί τη συνολική κίνηση σε πολύ χαμηλό επίπεδο καθώς απαιτείται ελάχιστη ανταλλαγή μηνυμάτων και διατήρηση ενός σταθερού και περιορισμένου αριθμού γειτόνων από κάθε κόμβο. Όπως φάνηκε από τα αποτελέσματα, τα περισσότερα μηνύματα τόσο

αναζήτησης όσο και δημοσίευσης ή εισαγωγής απαιτούν λίγα μόνο βήματα (κάτω των 10).

Όσον αφορά την ανεκτικότητα του συστήματος σε περιπτώσεις αποτυχίας κόμβων, αυτή κρίθηκε αρχικά ικανοποιητική αφού έως ένα αρκετά υψηλό ποσοστό απότομων αποχωρήσεων (έως και 22%) το σύστημα διατηρεί αρκετά υψηλά ποσοστά επιτυχών αναζητήσεων (άνω του 80%). Ήταν βέβαια απαραίτητο να γίνουν κάποιες επιπλέον βελτιώσεις, ώστε αυτό το ποσοστό να αυξηθεί και να μπορέσει το σύστημα να διατηρεί απόλυτη οργάνωση ακόμα και σε υψηλά επίπεδα αποτυχίας. Για τον λόγο αυτό παρουσιάσαμε στην συνέχεια μια σειρά βελτιώσεων και επιπρόσθετων λειτουργιών, όπως ο αλγόριθμος επιδιόρθωσης, τα σχήματα αντιγράφων και τα εικονικά δίκτυα.

Με τη χρήση του αλγορίθμου επιδιόρθωσης επιτύχαμε να αλλάξουμε τον τρόπο που μειώνονται οι επιτυχείς αναζητήσεις στο δίκτυό μας όταν παρουσιάζονται μαζικά σφάλματα (churn) και από λογαριθμική μείωση πλέον το σύστημά μας παρουσίασε γραμμική μείωση ως προς το ποσοστό σφαλμάτων. Αυτό σημαίνει ότι ακόμα και όταν οι μισοί κόμβοι στο δίκτυό μας αποτύγχαναν, οι μισές αναζητήσεις ήταν ακόμα επιτυχείς. Επίσης, ο αλγόριθμος επιδιόρθωσης βελτίωσε σημαντικά την απόδοση του συστήματός μας ανεξάρτητα από το ποσοστό σφαλμάτων του δικτύου αλλά και από τον πληθυσμό.

Όσον αφορά τις επεκτάσεις, αρχικά παρουσιάσαμε μια σειρά από σχήματα αντιγράφων, τα οποία επιτρέπουν τη δημοσίευση των κλειδιών σε παραπάνω από έναν κόμβο. Μέσα από μια σειρά εξομοιώσεων παρουσιάσαμε πως η χρήση των σχημάτων αντιγράφων βελτιώνει την ανεκτικότητα του συστήματος έως και ποσοστό που αγγίζει το 400%. Μάλιστα, το ποσοστό αυτό αυξάνεται όσο αυξάνεται το ποσοστό σφαλμάτων.

Εν συνεχεία παρουσιάσαμε την επέκταση των εικονικών δικτύων, όπου κάθε κόμβος μπορεί να συμμετέχει σε μια σειρά από δίκτυα με διαφορετικό αναγνωριστικό. Και σε αυτήν την περίπτωση, μέσα από μια σειρά εξομοιώσεων παρουσιάσαμε την επίδραση της επέκτασης στο δίκτυό μας. Παρατηρήσαμε πως όσο αυξάνει ο αριθμός των εικονικών δικτύων τόσο αυξάνει και η ανεκτικότητα του συστήματος σε σφάλματα, ανεξάρτητα από τον πληθυσμό του δικτύου. Παρατηρήσαμε επίσης ότι η αύξηση αυτή μεγαλώνει όσο αυξάνει το ποσοστό των σφαλμάτων, γεγονός που αποδεικνύει πως η χρήση των εικονικών δικτύων επιτρέπει στο σύστημά μας να λειτουργεί καλύτερα σε περιβάλλοντα πολλαπλών σφαλμάτων και απότομων αποχωρήσεων, αύξηση που φτάνει έως και το 250% για την περίπτωση των 2 εικονικών δικτύων, το 460% για την περίπτωση των 4 εικονικών δικτύων και το 722% για την περίπτωση των 8.

Εν συνεχεία παρουσιάσαμε ένα πιο γενικό πλαίσιο για το πρωτόκολλό μας με την εισαγωγή του δικτύου K-Umbrella, το οποίο βασίζεται σε ένα παραμετροποιημένο δίκτυο K-δέντρου, το οποίο δίνει τη δυνατότητα για καλύτερη κατανομή του φόρτου κίνησης σε όλα τα επίπεδα της δομής. Και σε αυτήν την περίπτωση, αρχικά παρουσιάσαμε τη δομή μας αναλυτικά και εν συνεχεία αναλύσαμε τη συμπεριφορά του δικτύου μέσα από μια σειρά εξομοιώσεων.

Από το σύνολο των εξομοιώσεων παρατηρήσαμε πως καθώς ο αριθμός των ακμών υποδέντρων αυξάνει επέρχεται αύξηση τόσο του αριθμού μηνυμάτων ανά κόμβο όσο και του μέσου αριθμού βημάτων αλλά ταυτόχρονα καλύτερη κατανομή της κίνησης σε όλα τα επίπεδα του δικτύου. Επομένως, η βέλτιστη λύση βρίσκεται σε ένα μεσαίο αριθμό ακμών υποδέντρων μεταξύ 4 και 8. Όσον αφορά τη δεύτερη παράμετρο του δικτύου μας, παρατηρήσαμε ότι καθώς αυξάνει ο αριθμός των επιτρεπόμενων παιδιών τόσο ο αριθμός των μηνυμάτων ανά κόμβο όσο και ο μέσος αριθμός μηνυμάτων μειώνονται. Σε σχέση με τη συμπεριφορά του δικτύου σε συνθήκες σφαλμάτων κόμβων, παρατηρήσαμε ότι οι υψηλές τιμές για επιτρεπόμενο αριθμό παιδιών βελτιώνει την ανεκτικότητα του συστήματος ενώ ο αριθμός των ακμών υποδέντρων δεν επηρεάζει ιδιαίτερα την απόδοση. Επομένως είναι προφανές ότι το σύστημά μας λειτουργεί καλύτερα όταν ο αριθμός των παιδιών αυξάνει. Επιπλέον, η ύπαρξη των δύο αυτών παραμέτρων επιτρέπει στο πρωτόκολλό μας να παραμετροποιείται κατάλληλα ώστε να εξυπηρετεί διαφορετικές δικτυακές απαιτήσεις, ανάλογα με τις υπερκείμενες εφαρμογές. Παρατηρήσαμε τέλος, πως το πρωτόκολλό μας προωθεί του κόμβους που παρουσιάζουν μεγαλύτερη σταθερότητα (παραμένουν συνδεδεμένοι περισσότερη ώρα).

11.2. Ανοιχτά ερευνητικά θέματα

Αν και κατά τη διάρκεια αυτής της ερευνητικής εργασίας επιτύχαμε να δώσουμε λύσεις σε αρκετά προβλήματα, πιστεύουμε ότι υπάρχουν ακόμη μια σειρά από ανοιχτά ερευνητικά θέματα, που παρουσιάζουν αρκετό ενδιαφέρον. Κάποιες από τις ιδέες και θέματα που θα θέλαμε να μελετήσουμε μετά το τέλος αυτής της διατριβής ώστε να βελτιώσουμε την αρχιτεκτονική μας και το σύστημα γενικότερα είναι:

- Δημιουργία μηχανισμού συμπίεσης του δικτύου καθώς οι κόμβοι εγκαταλείπουν και πολλές θέσεις παραμένουν κενές. Η αναδιοργάνωση αυτή θα επιφέρει και μείωση των βημάτων που απαιτούνται για επιτυχή αναζήτηση, δημοσίευση ή εισαγωγή.

- Βελτιστοποίηση της γνώσης των γειτονικών κόμβων που θα οδηγήσει στην μείωση των μηνυμάτων ή / και στην αύξηση της ανεκτικότητας σε απότομες εξόδους.
- Διερεύνηση της χρήσης του δικτύου για τη δημιουργία ομάδων για ρευμάτωση περιεχομένου.
- Περαιτέρω πειράματα, εξομοιώσεις και ανάλυση αποτελεσμάτων ώστε να βελτιώσουμε την απόδοση του συστήματος.
- Προσπάθεια υλοποίησης της αρχιτεκτονικής και δοκιμαστικής λειτουργίας της τόσο σε ερευνητικές/πειραματικές υποδομές όπως το PlanetLab [1] όσο και υπό μορφή ανοιχτού κώδικα (open source) για ελεύθερη χρήση μέσω του διαδικτύου.

Εκτός όμως από τα παραπάνω θέματα, τα οποία επεκτείνουν και βελτιώνουν τη δομή και την απόδοση της αρχιτεκτονικής μας, μέσα από αυτήν τη διατριβή εντοπίσαμε και μια σειρά από ερευνητικά θέματα στον χώρο των ομότιμων δικτύων που πιστεύουμε ότι παρουσιάζουν αρκετό ενδιαφέρον και θα μπορούσαν να αποτελούν μελλοντικές ερευνητικές δραστηριότητες. Μερικά από αυτά τα θέματα είναι:

- Υποστήριξη περίπλοκων αναζητήσεων πάνω από αρχιτεκτονικές ομότιμων δικτύων
- Δημιουργία κοινών κριτηρίων για τη σύγκριση της απόδοσης και της ανεκτικότητας των ομότιμων αρχιτεκτονικών
- Δημιουργία πρωτοκόλλων για τη διασύνδεση των διαφορετικών ομότιμων αρχιτεκτονικών
- Υποστήριξη και βελτιστοποίηση των ομότιμων αρχιτεκτονικών για εφαρμογές ρευμάτωσης πολυμεσικού περιεχομένου

Το μεγαλύτερο ενδιαφέρον από τα παραπάνω θέματα παρουσιάζει η δυνατότητα εκτέλεσης περίπλοκων αναζητήσεων πάνω από το σύστημά μας. Με την έννοια περίπλοκη αναζήτηση αναφερόμαστε σε εννοιολογικές αναζητήσεις, που επιτρέπουν την εκτέλεση εκφράσεων boolean καθώς και την αναζήτηση με βάση οντολογιών.

Αυτές οι αναζητήσεις υποστηρίζονταν αρχικά από συστήματα που βασίζονταν σε εννοιολογικούς δείκτες αλλά τα τελευταία χρόνια υπάρχει τάση στην έρευνα χρήσης DHT αλγορίθμων έναντι εννοιολογικών. Για τη δυνατότητα αυτή χρησιμοποιούνται συνήθως RDF (Resource Description Framework), τα οποία περιγράφουν τα μεταδεδομένα για κάθε περιεχόμενο και με βάση των οποίων δίνεται η δυνατότητα εκτέλεσης περίπλοκων αναζητήσεων. Συνήθως γίνεται χρήση τριπλέτων RDF για την απεικόνιση των μεταδεδομένων σε συνδυασμό με τη δημιουργία δικτύων DHT πολλαπλών επιπέδων και χρήση γλωσσών ερωτήσεων (query languages) συναφή με SQL.

Μερικά από τα πιο γνωστά συστήματα που χρησιμοποιούν DHT αλγορίθμους για την εκτέλεση/υποστήριξη RDF ερωτήσεων είναι το RDFPeers [2] και το GridVine [3] καθώς και η ερευνητική εργασία των Battre κ.α. [4].

Με βάση τα παραπάνω, πιστεύουμε πως θα είχε μεγάλο ερευνητικό ενδιαφέρον η χρησιμοποίηση του δικτύου K-Umbrella σε συνδυασμό με την επέκταση των εικονικών δικτύων για τη δημιουργία μιας σειράς υπερκείμενων δικτύων που θα υποστήριζαν RDF ερωτήσεις. Το βασικό δίκτυο (επίπεδο) θα επέτρεπε αναζήτηση πρώτου επιπέδου με βάση ομαδοποίησης οντολογίας, ενώ το δεύτερο επίπεδο θα επέτρεπε αναζήτηση σε επίπεδο δεδομένων. Το δίκτυο Unistore [5] ήδη χρησιμοποιεί αυτήν τη λογική με τη χρήση δυαδικού δέντρου και πιστεύουμε ότι η επέκταση σε K-δέντρο θα έχει τα πλεονεκτήματα που ήδη επιδείξαμε μέσα από αυτήν τη διατριβή.

11.3. Αναφορές

[1] Chun, B., Culler, D., Roscoe, T., Bavier, A., Peterson, L., Wawrzoniak, M., Bowman, M., "PlanetLab: An Overlay Testbed for Broad-Coverage Services," SIGCOMM Comp. Comm. Rev., 33(3), 2003

[2] Cai, M., Frank, M. R., Yan, B., MacGregor, R. M., "A Subscribable Peer-to-Peer RDF Repository for Distributed Metadata Management," Journal of Web Semantics: Science, Services and Agents on the World Wide Web, 2(2):109–130, December, 2004.

[3] Aberer, K., Cudre-Mauroux, P., Hauswirth, M., Pelt., T.V., "GridVine: Building Internet-Scale Semantic Overlay Networks," in Proceedings of the Thirteenth International World Wide Web Conference (WWW2004), New York, May, 2004.

[4] Battre, D., Hoing, A., Heine, F., Kao, O., "On Triple Dissemination, Forward-Chaining, and Load Balancing inDHT based RDF stores," in Fourth International Workshop on Databases, Information Systems and Peer-to-Peer Computing in scope of 32st International Conference on Very Large Data Bases (VLDB 2006), Seoul, Korea, September, 2006.

[5] Karnstedt, M., Sattler, K-U., Richtarsky, M., Müller, J., Hauswirth, M., Schmidt, R., John, R., "UniStore: Querying a DHT-based Universal Storage," in Proceedings of the 23nd International Conference on Data Engineering, ICDE 2007, Istanbul, Turkey, April, 2007.