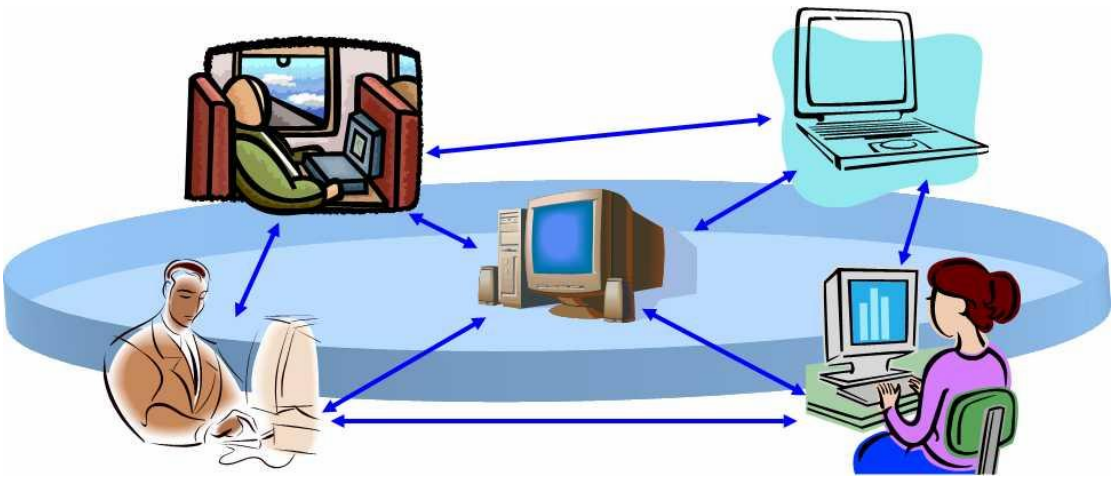
Τεχνολογικό Εκπαιδευτικό Ίδρυμα Πάτρας

Σχολή Διοίκησης & Οικονομίας

Τμήμα Επιχειρηματικού Σχεδιασμού & Πληροφοριακών Συστημάτων

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

PEER-TO-PEER NETWORKS



ΓΕΩΡΓΙΟΣ ΚΑΤΕΦΙΔΗΣ Α.Μ. : 457
ΑΛΕΞΑΝΔΡΟΣ ΤΖΑΝΙΝΗΣ Α.Μ. : 501

Επιβλέπων : ΧΡΗΣΤΟΣ ΓΚΟΥΜΟΠΟΥΛΟΣ

Πάτρα 15/12/2007

ΠΡΟΛΟΓΟΣ

Ένας από τους πολύ γρήγορα αναπτυσσόμενους τομείς της επιστήμης των υπολογιστών είναι η ανάπτυξη των peer to peer (ήP2P) συστημάτων και η έρευνα πάνω στις θεωρητικές αρχές που στηρίζονται . Η παρούσα πτυχιακή εργασία πραγματεύεται τόσο εφαρμοσμένα peer to peer συστήματα ,συγκεντρώνοντας περισσότερο την προσοχή στους αλγόριθμους δρομολόγησης (routing), όσο και την έρευνα που διεξάγεται πάνω στις θεωρητικές αρχές τους .

Το Peer To Peer Network είναι εκείνο το μοντέλο επικοινωνίας και ανταλλαγής πληροφοριών υπολογιστών , που εμφανίστηκε τις τελευταίες δεκαετίες και είναι αποκεντρωμένο , δηλαδή οι κόμβοι επικοινωνούν μεταξύ τους χωρίς την παρουσία ενός κεντρικού server .

Το Napster είναι η εφαρμογή που έδωσε το έναυσμα για την ανάπτυξη τέτοιων συστημάτων . Το συγκεκριμένο σύστημα ανήκει στα μη δομημένα συστήματα peer to peer , έχοντας σαν κεντρική ιδέα ότι ο server « υπάρχει για να διευκολύνει την διασύνδεση δυο υπολογιστών . Ακολουθεί στις αρχές του 2000 το πρωτόκολλο Gnutella στο όποιο απουσιάζει ο κεντρικός server . Επίσης το Gnutella 2 είναι μια πιο πρόσφατη αρχιτεκτονική η οποία βασίζεται στην ιδέα των υπερκόμβων και χρησιμοποιείται ευρέως από πολλές εφαρμογές . Ολοκληρώνοντας την αναφορά μας στα μη δομημένα συστήματα παρουσιάζουμε το Freenet το οποίο είναι ένα αυτό-οργανωμένο σύστημα με χαλαρή δομή και έχει πολλές ομοιότητες με το Gnutella 1 .

Μελετώντας τα δομημένα συστήματα βλέπουμε πως περιλαμβάνουν ένα σημαντικότατο πλεονέκτημα , τους αλγόριθμους δρομολόγησης . Τα πιο δημοφιλή στην κατηγορία αυτή είναι το Chord , το Tapestry , το Pastry , το

CAN , το Kademlia , το Viceroy . Κάνοντας μια περιγραφή της τοπολογίας και των βασικών διαδικασιών λειτουργίας τον παραπάνω συστημάτων καταλήγουμε στην παρουσίαση των επιδόσεων τους στις διάφορες λειτουργίες και σε μια τελική συγκριτική ανασκόπησή τους .

Τέλος γίνεται μια συνοπτική παρουσίαση των μελλοντικών εξελίξεων στον τομέα των Peer To Peer Networks βασιζόμενη σε θεωρητικές προβλέψεις , καθώς επίσης και η αναφορά σε διάφορα συμπεράσματα που προκύπτουν από το σύνολο της εργασίας .

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1

Εισαγωγή στα peer to peer networks

1.1 Λίγα λόγια γενικά για τα δίκτυα

Είναι γεγονός πως τα δίκτυα υπολογιστών αυξάνονται εκρηκτικά τα τελευταία χρόνια Πριν από δυο δεκαετίες , ελάχιστοι είχαν πρόσβαση σε ένα δίκτυο σήμερα η επικοινωνία των υπολογιστών έχει γίνει απαραίτητο μέρος τις υποδομής μας .Η δικτύωση χρησιμοποιείτε σε κάθε δραστηριότητα των επιχειρήσεων , όπως στην διαφήμιση την παραγωγή , την διεκπεραίωση , των σχεδιασμό , την κοστολόγηση , και τη λογιστική. Γι'αυτό , οι περισσότερες εταιρίες έχουν πολλά δίκτυα. Πολλά σχολεία, σε όλες τις βαθμίδες, από τη στοιχειώδη μέχρι τη μεταπτυχιακή εκπαίδευση, χρησιμοποιούν δίκτυα υπολογιστών για να παρέχουν στους διδασκομένους και τους διδάσκοντες άμεση πρόσβαση σε πληροφορίες που υπάρχουν σε ηλεκτρονικές βιβλιοθήκες σε όλο τον κόσμο . Δίκτυα χρησιμοποιούν οι κρατικές, περιφερειακές, και τοπικές δημόσιες υπηρεσίες , καθώς και στρατιωτικοί οργανισμοί . Με μια φράση τα δίκτυα υπολογιστών είναι ''παντού'' . Ειδικότερα ένα δίκτυο υπολογιστών (computer network) αποτελείται από ένα σύνολο υπολογιστών που συνδέονται μεταξύ τους με ειδικές συσκευές και προγράμματα επικοινωνίας με σκοπό την επικοινωνία μεταξύ τους (την ανταλλαγή πληροφοριών ή δεδομένων).

Οι πρώτες δικτυακές τάσεις αναφέρονται στο μοντέλο Client/server(Πελάτης / Εξυπηρετων) Το μοντέλο Client/server είναι ένα μοντέλο λειτουργίας μιας εφαρμογής σύμφωνα με το οποίο διάφορα ανεξάρτητα προγράμματα επικοινωνούν μεταξύ τους, ανταλλάσοντας μηνύματα . Ποιο συγκεκριμένα ένα πρόγραμμα που λέγεται Client αιτείτε

μια υπηρεσία στέλνοντας ένα μήνυμα σε ένα άλλο πρόγραμμα που λέγεται server . Ο server επεξεργάζεται την αίτηση και στέλνει με την σειρά του το αποτέλεσμα στον πελάτη .

Συνοπτικά μπορούμε να πούμε ότι το μοντέλο Client/server είναι ένας διάλογος μεταξύ δυο προγραμμάτων με σκοπό κάποια ανταλλαγή δεδομένων οι αποτελεσμάτων κάποιου υπολογισμού . Η σχέση που αποκαθίσταται κατά την διάρκεια του διαλόγου είναι σχέση ίσος προς ίσον .Όμως δεν σημαίνει ότι τα δυο προγράμματα είναι ακριβός τα ίδια σε ότι αφορά το ρόλο τους , τον εσωτερικό τους κώδικα ή ότι τρέχουν κάτω από το ίδιο λειτουργικό σύστημα . Τέλος μπορούμε να πούμε ότι το μοντέλο Client/server αποτελεί εξέλιξη της οργάνωσης των συστημάτων πληροφορικής. Η εξέλιξη αυτή ξεκίνησε από μια ιεραρχική οργάνωση των συστημάτων , πέρασε από ένα ενδιάμεσο στάδιο κατανεμημένης οργάνωσης για να φτάσει σε μια πλήρη κατανεμημένη οργάνωση στο μοντέλο Client/server.

1.2 Ιστορική αναδρομή

Στο τέλος του προηγούμενου αιώνα μια εφαρμογή έμελλε να αλλάξει το τρόπο με τον οποίο βλέπουν οι χρήστες το διαδίκτυο . Ακόμη υπάρχει στην ανάμνηση μας εκείνο το εξώφυλλο του περιοδικού RAM τον Ιούλιο του 2000 που είχε ως τίτλο « Το γιγαντιαίο κύμα του Mp3» που αναφερότανε στην «επανάσταση>> που είχε επιφέρει το Mp3.Όλα ξεκίνησαν όταν ένας 19χρονος φοιτητής (Shawn Fanning) που καθόταν στο δωμάτιο του στο Northeastern University της Βοστόνης άκουσε τον συγκάτοικο του να παραπονιέται για πεθαμένα links με τραγούδια που ήταν κωδικοποιημένα κατά MPEG-1 Audio Layer 3 . Τότε είχε την ιδέα της ανάπτυξης μιας εφαρμογής που θα επιτρέπει σε διαφόρους χρηστές ανά την υφήλιο να

ανταλλάσσουν αρχεία μεταξύ τους χρησιμοποιώντας το διαδίκτυο. Η εφαρμογή αυτή ήταν το Napster . Αυτή η εφαρμογή αν και τεχνικά δεν είναι peer to peer έδωσε έναυσμα για την ανάπτυξη των peer to peer συστημάτων . Η ιδέα για peer to peer όπως θα δούμε και παρακάτω είναι παλαιότερη , όμως αυτή η εφαρμογή έδρασε επαναστατικά μέσα στα πλαίσια της κοινωνίας δίνοντας έτσι την ώθηση για την ανάπτυξη τέτοιων εφαρμογών . Οι χρήστες του διαδικτύου δεν είναι πια παθητικοί, που το μόνο που μπορούσαν να κάνουν είναι να βλέπουν σελίδες (web pages) του φυλλομετρητή (browser) αλλά μπορούν να ανταλλάσσουν και μεταξύ τους πληροφορίες. Το διαδίκτυο αρχικώς είχε σχεδιαστεί ως ένα peer to peer σύστημα στα τέλη της δεκαετίας του 1960 που είχε ως στόχο την ανταλλαγή υπολογιστικών πόρων στις Η.Π.Α. Οι πρώτοι υπολογιστές στο ARPANET συνδέονταν ως ισότιμοι peers . Οι κύριοι χρήστες ήταν τότε πανεπιστήμια και ερευνητές υπολογιστών. Πολλά peer to peer συστήματα χρησιμοποιούνταν ευρέως που υπάρχουν ακόμη και σήμερα (όπως το Usenet,DNS και FidoNet). Το Usenet που υπάρχει από το 1979 μπορεί να θεωρηθεί και ως «παππούς» των σημερινών συστημάτων Gnutella και Freenet. Είναι ένα σύστημα που δεν χρησιμοποιεί κεντρικό έλεγχο και αντιγράφει αρχεία μεταξύ υπολογιστών . Στο DNS (Domain Name System) ο τρόπος που επικοινωνούν μεταξύ τους οι κόμβοι είναι παρόμοιος με ένα peer to peer σύστημα . Το FidoNet που δημιουργήθηκε το 1984 είναι μια κατανεμημένη εφαρμογή ανταλλαγής μηνυμάτων μεταξύ χρηστών . Το 1994 η δομή του διαδικτύου άλλαξε δραματικά με εκατομμύρια ανθρώπους να συνδέονται σε αυτό. Οι χρήστες τότε δεν ήταν ισότιμοι (Peers) όπως παλιότερα. Ο χρήστης του διαδικτύου περιοριζόταν κυρίως στην περιήγηση των σελίδων (web pages) μέσω κάποιου φυλλομετρητή. Αυτή η χρήση ευνοούσε την ανάπτυξη του πελάτη/εξυπηρετητή (client/server) πρωτοκόλλου που ενσωματώθηκε από τους φυλλομετρητές.

Η βασική αρχή είναι ότι ο client συνδέεται σε έναν server ανταλλάσσοντας μηνύματα , και αφού τελειώσει την «δουλειά» του αποσυνδέεται . Το μοντέλο αυτό ακολουθείται μέχρι σήμερα. Όμως από το έτος 2000 έχουν εμφανιστεί αρκετά συστήματα peer to peer . Όπως αναφέρθηκε το Napster αν και δεν ακολουθεί το μοντέλο peer to peer είναι καταλυτικός παράγοντας για την ανάπτυξη τέτοιων συστημάτων. Το Napster ουσιαστικά «έριξε» την ιδέα στους χρήστες για ανταλλαγή αρχείων και πληροφορίας μεταξύ τους . Έτσι το ακρωνύμιο peer to peer έχει αποκτήσει και ένα μη τεχνικό ορισμό . Αυτός ο δεύτερος «ορισμός» είναι ο ‘‘people-to-people” . Δηλαδή το peer to peer είναι ένα μοντέλο συστήματος που βοηθά τους ανθρώπους-χρήστες του διαδικτύου να μοιράζονται κοινά ενδιαφέροντα . Η αυτοαποκαλούμενη κοινωνική δικτυακή τεχνολογία είναι ένα παράδειγμα αυτού του σκεπτικού. Η επανάσταση λοιπόν ξεκίνησε από το χώρο του διαδικτύου είχε κοινωνιολογικές επιδράσεις και προεκτάσεις. Όμως το Napster που είναι ένα υβριδικό μοντέλο που συνδυάζει χαρακτηριστικά peer to peer αλλά και Client/server , αντιμετώπισε νομικά προβλήματα ερχόμενο αντιμέτωπο με τις δισκογραφικές εταιρίες που ήθελαν το κλείσιμο του θεωρώντας το υπεύθυνο για παράνομη διακίνηση μουσικών αρχείων . Κάτι το οποίο κατέστη εφικτό μετά από αρκετές δίκες . Έτσι ο κεντρικός Server έκλεισε αφαιρώντας την δυνατότητα από τους χρήστες-clients να ανταλλάσσουν αρχεία μεταξύ τους .Αυτήν τη αδυναμία του client/server μοντέλου ήρθαν να καλύψουν τα συστήματα που μελετάμε. Το δίκτυο του Gnutella εμφανίστηκε αμέσως μετά το Napster , και αποτελεί ένα peer to peer δίκτυο. Οι κατασκευαστές του, ήθελαν να φτιάξουν ένα αποκεντρωμένο δίκτυο που δεν θα μπορούσε να κλείσει , θέτοντας εκτός λειτουργίας τον κεντρικό server . Έτσι κατασκευάστηκε ένα δίκτυο όπου χρήστες συνδέονται απευθείας με άλλους χρήστες .

1.3 Τι είναι τελικά το peer to peer ?

Ωραία έως εδώ άλλα τι είναι τελικά το peer to peer ? Από την ανάγνωση των προηγουμένων , θα έχετε βγάλει το συμπέρασμα ότι peer to peer είναι ένα μοντέλο για την επικοινωνία και ανταλλαγή πληροφορίας μεταξύ υπολογιστών εναλλακτικό στο κλασσικό client/server μοντέλο . Πρόκειται δηλαδή για ένα μοντέλο αποκεντρωμένο στο οποίο οι κόμβοι επικοινωνούν άμεσα μεταξύ τους χωρίς την παρουσία ενός κεντρικού Server . Ας αναλύσουμε όμως λίγο περισσότερο στο τι περιέχεται κάτω από αυτήν την έννοια . Μπορούμε να πούμε ότι πρόκειται για ένα πρότυπο επικοινωνιών στο οποίο κάθε συμβαλλόμενο μέρος έχει τις ίδιες δυνατότητες και κάθε συμβαλλόμενο μέρος μπορεί να ξεκινήσει μια σύνοδο επικοινωνίας . Η επικοινωνία αυτή επιτυγχάνεται με την καθιέρωση κάθε επικοινωνιακού κόμβου με τέτοιο τρόπο ώστε να είναι ταυτόχρονα και εξυπηρετητής . Γιαυτο το λόγο οι κόμβοι αυτοί ονομάζονται και Server Client , δηλαδή Servent. Ο Clay Shriky έδωσε τον παρακάτω ορισμό . Peer to peer είναι μια κλάση εφαρμογών που εκμεταλλεύονται τους υπολογιστικούς πόρους που είναι διαθέσιμοι από άκρη σε άκρη σε όλο το Internet . Συστήματα peer to peer δεν συναντάμε μόνο σε εφαρμογές διαμοιρασμού αρχείων αλλά και σε εφαρμογές άμεσης επικοινωνίας (Instant messaging) , εφαρμογές κατανεμημένου υπολογισμού και εφαρμογές groupware. Μια εφαρμογή κατανεμημένου υπολογισμού είναι το SETI . Υπάρχει ένα ραδιοτηλεσκόπιο που ψάχνει να βρει σήματα που προέρχονται από εξωγηινη μορφή ζώης . Ο υπολογιστής του χρήστη επεξεργάζεται τα δεδομένα από το τηλεσκόπιο όταν είναι ανενεργός και όταν συνδέεται με το Internet στέλνει τα αποτελέσματα στο κεντρικό Computer του SETI. Αυτό γίνεται διότι ο όγκος των πληροφοριών που πρέπει να επεξεργαστούν είναι μεγάλος και έτσι χρησιμοποιούνται και οι δυνατότητες των υπολογιστών των χρηστών.

Μια πιο τεχνική περιγραφή από τον Dave Winner της User Land Software Θεωρεί ότι οι εφαρμογές peer to peer έχουν τα παρακάτω χαρακτηριστικά :

1. Η διεπιφάνεια χρήστη δεν εκτελείται μέσω κάποιου φυλλομετρητη ιστοσελίδων .
2. Ο υπολογιστής του χρήστη ενεργεί και σαν πελάτης και σαν εξυπηρετητής .
3. Το όλο σύστημα είναι ευκολοχρηστο και ολοκληρωμένο.
4. Το σύστημα περιλαμβάνει εργαλεία που υποστηρίζει χρήστες που θέλουν να δημιουργήσουν περιεχόμενο η να προσθέσουν λειτουργίες .
5. Το σύστημα παρέχει συνδέσεις με άλλους χρήστες
6. Το σύστημα παρέχει μια νέα υπηρεσία η μια ήδη υπάρχουσα με αποτελεσματικότερο και καινοτόμο τρόπο.
7. Το σύστημα υποστηρίζει «διαδικτυακά» πρωτόκολλα όπως το Soap η την XML-RPX .

1.4 Κατηγοριοποίηση συστημάτων Peer to Peer

Ήρθε η ώρα να βάλουμε μια τάξη στο ομιχλώδες τοπίο των συστημάτων peer to peer προσπαθώντας να τα κατηγοριοποιήσουμε . Η πρώτη απόπειρα κατηγοριοποίησης θα γίνει σύμφωνα με το βαθμό αποκέντρωσης .

Αποκεντρωμένης τοπολογίας όπως ήταν αρχικός το Gnutella 1 και το Freenet. Όλοι οι κόμβοι στο δίκτυο είναι ισότιμοι. Δεν υπάρχει κεντρικός server αλλά κάθε κόμβος είναι sernvent. Αυτά είναι τα «καθαρά» peer to peer

συστήματα. Τα πλεονεκτήματα αυτών των συστημάτων είναι : α)Έμφυτη επεκτασιμότητα, β)Καταμερισμός φορτιού)Ανοχή σε λάθη.

Υβριδικής αποκεντρωμένης τοπολογίας όπως το Napster . Εδώ υπάρχει ένας κεντρικός server που διευκολύνει την αλληλεπίδραση μεταξύ των κόμβων κρατώντας ένα κεντρικό ευρετήριο που περιέχει τα αρχεία που διαμοιράζει κάθε υπολογιστής . Αφού βρεθεί το αρχείο που θα ανταλλαχθεί μεταξύ των 2 client-κόμβων η επικοινωνία τους γίνεται χωρίς τη χρήση του κεντρικού server . Σε αυτά τα συστήματα αν πέσει ο κεντρικός server τότε θα πιέσει και όλο το δίκτυο. Όπως αναφερθεί έτσι συνέβη όταν αποφασίστηκε η αναστολή λειτουργίας του Napster . Τα πλεονεκτήματα αυτών των συστημάτων είναι :

1. Βρίσκουν γρήγορα και αποτελεσματικά την ζητούμενη πληροφορία
2. Τα ψαξίματα καλύπτουν το μεγαλύτερο δυνατό εύρος (Δηλαδή όλους τους υπολογιστές του δικτυού).

Τα μειονεκτήματα τώρα είναι :

1. Ευαίσθητα στην λογοκρισία και στο τεχνικό λάθος .
2. Η πληροφορία που χρησιμοποιείται συχνά γίνεται λιγότερο προσβάσιμη .
3. Εξαιτίας του αριθμού των αιτήσεων που φτάνουν για αυτήν στο κεντρικό server (Slashdot effect).
4. Το κεντρικό ευρετήριο δεν περιέχει τα πιο πρόσφατα δεδομένα διότι ανανεώνεται περιοδικά ,αυτά τα συστήματα από τεχνικής άποψης δεν αποτελούν peer to peer συστήματα αλλά τα αναφέρουμε κυρίως για ιστορικούς λόγους .

Μερικώς αποκεντρωμένης τοπολογίας όπως το Gnutella 2 (εφαρμογές που βασίζονται σε αυτό το δίκτυο είναι το Kazaa,Morpheus,Shareaza,Limewire κ.α.) .Η Βασική ιδέα είναι ίδια όπως και στα καθαρά peer to peer . Η μόνη διαφορά έγκειται στο γεγονός ότι ορισμένοι κόμβοι γίνονται «Υπερκόμβοι» (Supernodes). Αυτοί οι κόμβοι έχουν την επιπλέον ιδιότητα να έχουν τοπικά κεντρικά ευρετήρια για τα αρχεία που μοιράζονται οι «υποκόμβοι» τους . Κριτήριο για την επιλογή των κόμβων που θα γίνουν υπερκόμβοι αποτελούν συνήθως οι υπολογιστικοί πόροι . Είναι σημαντικό να τονίσουμε ότι αν «πέσει» ένας υπερκόμβος τότε δεν πέφτει το δίκτυο . Αυτός ο υπερκόμβος αντικαθίσταται από άλλον. Συνδυάζουν τα πλεονεκτήματα και των 2 άλλων κατηγοριών .Δηλαδή μειώνουν το χρόνο ανακάλυψης σε σχέση με τα καθαρά συστήματα και μειώνουν το φόρτο εργασίας σε κεντρικούς server σε σχέση με τα κεντρικά συστήματα .

Ένας ακόμη τρόπος να τα κατηγοριοποιήσουμε μπορεί να γίνει βάση της δομής της τοπολογίας των δικτύων . Έτσι μπορούμε να τα χωρίσουμε σε δομημένα και μη δομημένα δίκτυα .

Τα μη δομημένα δίκτυα η τοποθεσία των δεδομένων είναι ανεξάρτητη της τοπολογίας .Εφόσον δεν υπάρχει γνώση το που μπορεί να βρίσκεται(σε ποιον κόμβο) η ζητούμενη πληροφορία το ψάξιμο γίνεται τυχαία ρωτώντας διαφόρους κόμβους αν έχουν την ζητούμενη πληροφορία η όχι .Αυτά τα συστήματα διαφέρουν στον τρόπο κατασκευής της τοπολογίας τους αλλά και στον τρόπο που υποβάλλονται οι ερωτήσεις από κόμβο σε κόμβο . Το πλεονέκτημα αυτών των συστημάτων έγκειται στο γεγονός ότι μπορούν εύκολα να ενσωματώσουν μια σημαντική αιφνίδια αύξηση του πληθυσμού των κόμβων .Το μειονέκτημα είναι ότι είναι για να βρεθεί η κατάλληλη

πληροφορία πρέπει να γίνουν πολλές ερωτήσεις προς όλες τις πλευρές του δικτυού.

Τα δομημένα δίκτυα όπως το Chord,Can,Past,Tapestry κ.α. Εμφανιστήκαν κυρίως ως μια προσπάθεια να διευθύνουν τα προβλήματα επεκτασιμότητας που αντιμετωπίζουν τα μη δομημένα δίκτυα. Οι τυχαίοι μέθοδοι ψαξίματος που υιοθετούνται από μη δομημένα δείχνουν να είναι έμφυτα μη επεκτάσιμα, και έτσι προστέθηκαν τα δομημένα συστήματα , στα οποία η τοπολογία του δικτύου ελέγχεται αυστηρά και η πληροφορία (η δείκτες προς την πληροφορία) είναι τοποθετημένη σε καθορισμένες τοποθεσίες. Έτσι σε αυτά τα δίκτυα το ψάξιμο κάποιας συγκεκριμένης πληροφορίας γίνεται υιοθετώντας καθορισμένους αλγόριθμους δρομολόγησης μηνυμάτων .Είναι σύνηθες φαινόμενο αυτά τα συστήματα να παρέχουν μια αντιστοιχία μεταξύ ενός αναγνωριστικού της πληροφορίας (π.χ. File id), σε μορφή ενός κατανεμημένου πίνακα δρομολόγησης (distributed routing table) , έτσι ώστε τα ερωτήματα μπορούν να δρομολογηθούν αποτελεσματικά στον κόμβο με την επιθυμητή πληροφορία.

Το μειονεκτήματα αυτών των συστημάτων είναι ότι είναι δύσκολο να διατηρηθεί η δομή που απαιτείται για δρομολόγηση σε ένα πολύ ευμετάβλητο πληθυσμό κόμβων (κόμβοι εισέρχονται και αποχωρούν συνεχεία από το σύστημα).

Υπάρχουν και δίκτυα με χαλαρή δομή όπως το Freenet που είναι κάτι το ενδιάμεσο. Οι θέσεις που βρίσκεται η πληροφορία επηρεάζεται από κανόνες δρομολόγησης , που δεν είναι όμως σαφώς καθορισμένο, και έτσι δεν επιτυγχάνουν όλα τα ψαξίματα.

1.5 Τα σημαντικότερα ζητήματα γύρω από τα peer to peer networks

1.5.1 Φερεγγυότητας και Ανάπτυξη Εμπιστοσύνης

Τα συστήματα ομότιμων κόμβων αποτελούν ιδανικές πλατφόρμες για την από κοινού χρήση πόρων (αποθηκευτικό χώρο, υπολογισμούς, δεδομένα κλπ.) κάθε υπολογιστή του ιδεατού δικτύου, με στόχο την κοινή ωφέλεια όλων των συμμετεχόντων. Ωστόσο, η συνεισφορά των χρηστών δεν είναι προκαθορισμένη και δεν έχει ορθολογιστική βάση για τους χρήστες όταν δεν απαιτείται από το σύστημα. Οι χρήστες των p2p συστημάτων έχουν την τάση να ενεργούν για το προσωπικό τους όφελος και όταν μπορούν να χρησιμοποιήσουν τους πόρους του συστήματος δωρεάν, δεν έχουν κανένα κίνητρο να ανταποδώσουν συνεισφέροντας τους δικούς τους πόρους. Η μελέτη μετρήσεων στα p2p συστήματα Napster και Gnutella, κατέγραψε ότι πολλοί χρήστες είναι free riders (25% των χρηστών του Gnutella δεν συνεισφέρουν καθόλου αρχεία και 20-40% των χρηστών του Napster συνεισφέρουν πολύ μικρό αριθμό ή καθόλου αρχεία) καθώς και ότι 50% των συνδέσεων διαρκεί λιγότερο από 1 ώρα. Αυτή η συμπεριφορά βλάπτει την οντότητα των p2p δικτύων ως κατανεμημένα συστήματα για την από κοινού χρήση πόρων όταν οι κόμβοι δεν συνεισφέρουν δεν προσθέτουν αξία στο σύστημα και μειώνουν την απόδοση του. Λόγο της εθελοντικής συμμετοχής των κόμβων, οι πόροι του συστήματος είναι ιδιαίτερα ευμετάβλητοι και απρόβλεπτοι, με αποτέλεσμα να μην υπάρχει εγγύηση για την αξιοπιστία του συστήματος. Συνεπώς είναι απαραίτητη η χρήση μηχανισμών κινήτρων που αναγκάζουν τους χρήστες να συνεισφέρουν, και αποκλείουν την μη-ανταποδοτική συμπεριφορά, έτσι ώστε να είναι πιο προβλέψιμη η απόδοση ενός p2p συστήματος.

1.5.2 Διακίνησης παράνομου υλικού

Το πόσο μεγάλη έκταση έχει πάρει το φαινόμενο των δικτύων P2P το αποδεικνύουν έρευνες που βλέπουν καθημερινά το φως της δημοσιότητας. Σύμφωνα με την γερμανική εταιρία ανάλυσης δικτυακής κίνησης Ιπόκ (Ipoque), το ποσοστό της παγκόσμια κίνησης στο ίντερνετ που προέρχεται από δίκτυα P2P κυμαίνεται από 50 ως και 90%, και από αυτό το μεγαλύτερο μέρος είναι οπτικοακουστικό υλικό (βίντεο,ταινίες,μουσική). Η παράνομη μεταφορά περιεχομένου είχε ως άμεση συνέπεια δισκογραφικές και κινηματογραφικές εταιρίες να αντιμετωπίζουν μεγάλες οικονομικές απώλειες, αφού οι καταναλωτές χρησιμοποιούν τα δίκτυα P2P για να «κατεβάσουν» τα τραγούδια και τις ταινίες της αρεσκείας τους, δωρεάν. Η απάντηση των εταιριών αυτών ήρθε δια της δικαστικής οδού, με απαρχή την υπόθεση του Νάπστερ (Napster), το διασημότερο από τα πρώτα προγράμματα P2P. Η εταιρία Έι&Εμ Ρέκορντς οδήγησε το Νάπστερ στο δικαστήριο, με αποτέλεσμα το τελευταίο να σταματήσει την λειτουργία του, τον Οκτώβριο του 2001. Σήμερα οι δικαστικές υποθέσεις που αφορούν προγράμματα ή ιστοσελίδες που προωθούν την παράνομη ανταλλαγή νομικά προστατευόμενου περιεχομένου αποτελούν καθημερινό φαινόμενο.

Τα δίκτυα P2P ωστόσο δεν δημιουργήθηκαν για να φέρουν πλήγμα στην μουσική και την κινηματογραφική βιομηχανία. Αντιθέτως, η φιλοσοφία τους βασίζεται στην ελεύθερη διακίνηση ιδεών, υπηρεσιών και πληροφοριών. Χαρακτηριστικό παράδειγμα αποτελεί το P2P δίκτυο "Μερίδα του Λέοντος" (LionShare) το οποίο κατασκευάστηκε για τον διαμοιρασμό ακαδημαϊκού υλικού στους χρήστες του.

ΚΕΦΑΛΑΙΟ 2

Μη Δομημένα Συστήματα P2P

2.1 Napster

2.1.1 Το Napster και η ιστορία του

Το Napster ανήκει στην κατηγορία των μη δομημένων υβριδικής αποκεντρωμένης τοπολογίας συστημάτων . Στις αρχές του 1999 ο Shawn Fanning ξεκίνησε την υλοποίηση μιας ιδέας, η οποία θα του επέτρεπε να βρει στο Διαδίκτυο τα μουσικά κομμάτια MP3 που ο ίδιος και οι φίλοι του προτιμούσαν να ακούν. Αυτό που δε φαντάστηκε ποτέ, όμως, ο δεκαοχτάχρονος τότε μαθητής ήταν ότι το δημιούργημά του θα άλλαζε τον τρόπο με τον οποίο ακούμε μουσική.

Η βασική ιδέα πίσω από το Napster ήταν η δημιουργία μιας εφαρμογής-πρωτοκόλλου, η οποία θα συνδύαζε τις λειτουργίες μιας μηχανής αναζήτησης (search engine), ενός προγράμματος απευθείας ανταλλαγής αρχείων και ενός IRC client **(** IRC(Internet Relay Chat) είναι ένα σύστημα που επιτρέπει την online επικοινωνία μεταξύ των χρηστών δύο ή περισσότερων υπολογιστών, μέσω του Internet.), προκειμένου να είναι εφικτή η συζήτηση μεταξύ των χρηστών της «κοινότητας» που βρίσκονταν εκείνη τη στιγμή σε ενεργή σύνδεση (online*)*. Ο νεαρός μαθητής δούλεψε για αρκετούς μήνες δουλεύοντας τον κώδικα για την εφαρμογή που, τελικά, εξελίχθηκε στο πρόγραμμα Napster (από το ψευδώνυμο που είχε ο Fanning στο σχολείο). Ο Fanning έκανε upload την πρώτη beta έκδοση του προγράμματος στο

site.download.com, όπου μετά από σύντομο χρονικό διάστημα έγινε ένα από τα πιο «καυτά» downloads. Η συνέχεια ανήκει στην ιστορία.

2.1.2 Τρόπος Λειτουργίας

Η υπηρεσία Napster εισήγαγε μια νέα ιδέα για τον τρόπο διανομής των αρχείων MP3. Αντί τα τραγούδια να βρίσκονται αποθηκευμένα σε έναν κεντρικό υπολογιστή-server, υπάρχουν στους σκληρούς δίσκους των υπολογιστών των χρηστών. Η μέθοδος αυτή ονομάζεται **peer-to-peer ή P2P sharing.** Σε γενικές γραμμές, κάποιος που χρησιμοποιούσε το Napster για να κατεβάσει ένα τραγούδι, συνδεόταν με τον προσωπικό υπολογιστή ενός άλλου χρήστη, ο οποίος μπορεί να βρισκόταν είτε δίπλα του είτε στην άλλη άκρη της Γης.

Πιο αναλυτικά :

- Αυτός που ζητούσε το τραγούδι χρειαζόταν :
    1. Το πρόγραμμα Napster εγκατεστημένο στον υπολογιστή του.
    2. Έναν κατάλογο (directory) στον υπολογιστή του, το οποίο θα είχε δηλωθεί στο Napster ως **shared**, ώστε να μπορούν να έχουν πρόσβαση σε αυτό οι άλλοι χρήστες του Napster.
    3. Μια ενεργό (online) σύνδεση στο Internet.
- Ο προμηθευτής του αρχείου χρειαζόταν :
    1. Το πρόγραμμα Napster εγκατεστημένο στον υπολογιστή του.
    2. Ένα κατάλογο (directory) στον υπολογιστή του, το οποίο θα είχε δηλωθεί στο Napster ως **shared**, ώστε να μπορούν να έχουν πρόσβαση σε αυτό οι άλλοι χρήστες του Napster.
    3. Μια ενεργό (online) σύνδεση στο Internet.

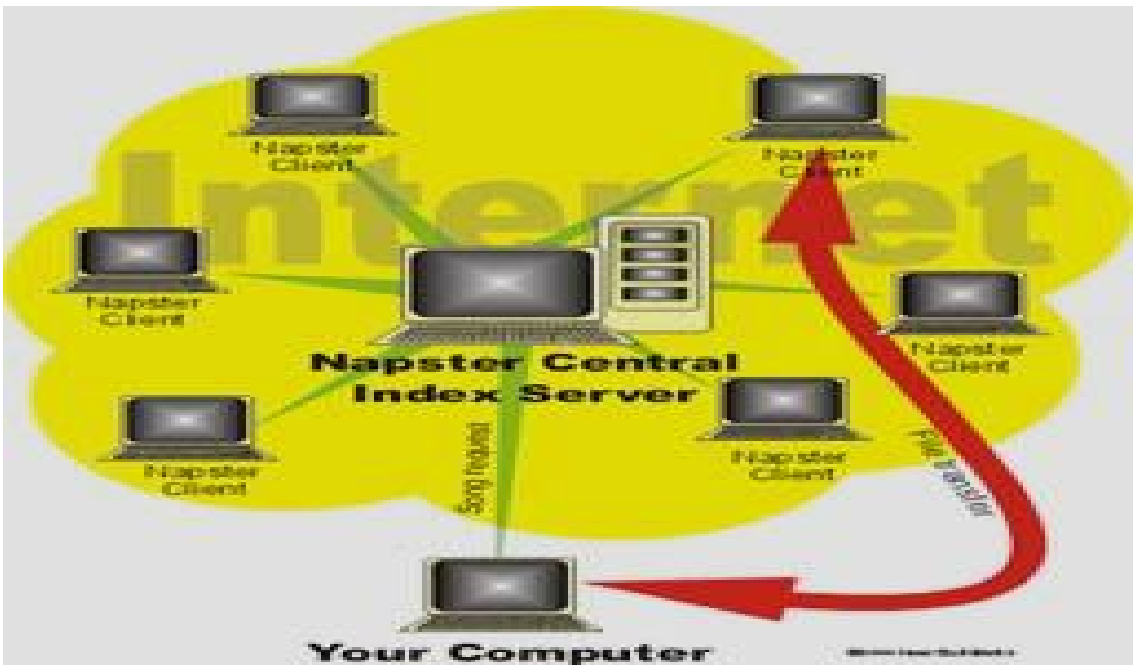
4. Ένα αντίγραφο του αρχείου MP3 που αναζητούσε ο χρήστης που έκανε την αίτηση (request).

Από εκεί και πέρα η διαδικασία που ακολουθούνταν κατά τη χρήση του Napster για την ανταλλαγή ενός αρχείου MP3 ήταν η εξής :

1. Άρχιζε η εκτέλεση του Napster και γινόταν έλεγχος για το αν υπήρχε ή όχι ενεργός σύνδεση στο Διαδίκτυο.
2. Αν όντως υπήρχε, ο υπολογιστής του χρήστη συνδεόταν μέσω του Napster στον κεντρικό server. Σκοπός αυτού του κεντρικού server δεν ήταν παρά η δημιουργία ενός καταλόγου με όλους τους χρήστες του Napster που βρισκόταν εκείνη τη στιγμή online. Ο server αυτός δεν περιλάμβανε στο σκληρό δίσκο του τα μουσικά κομμάτια.
3. Ο χρήστης έγραφε το τραγούδι που επιθυμούσε να κατεβάσει και καθόριζε την επιθυμητή ποιότητα σύνδεσης του προμηθευτή του κομματιού καθώς και οποιαδήποτε άλλη παράμετρο επιθυμούσε.
4. Το Napster εξέταζε τον κεντρικό server, ο οποίος είχε καταγεγραμμένους όλους τους online χρήστες και αναζητούσε μέσα σε αυτούς το κομμάτι καθώς και όλα τα άλλα κριτήρια που είχε θέσει προηγουμένως ο χρήστης.
5. Όταν ο server έβρισκε κάποιο χρήστη που πληρούσε τις προϋποθέσεις, ενημέρωνε το Napster για το πού υπήρχε το αρχείο.
6. Όταν ο server επέστρεφε αποτελέσματα, το Napster δημιουργούσε μια λίστα με τους υπολογιστές που περιείχαν το αρχείο και πληρούσαν τις προϋποθέσεις, στο τμήμα **Search results** του προγράμματος του χρήστη.
7. Ο χρήστης επέλεγε έναν από αυτούς τους υπολογιστές κάνοντας κλικ πάνω σε αυτόν και στη συνέχεια διάλεγε την επιλογή **download** για να κατεβάσει το συγκεκριμένο μουσικό κομμάτι.

8. Το Napster στη συνέχεια προσπαθούσε να πραγματοποιήσει σύνδεση με τον απομακρυσμένο υπολογιστή, ο οποίος περιείχε το συγκεκριμένο αρχείο.
9. Αν η σύνδεση ήταν επιτυχής, ξεκινούσε η αντιγραφή του αρχείου από το δίσκο του απομακρυσμένου υπολογιστή στον τοπικό δίσκο του χρήστη.
10. Όταν τελείωνε η αντιγραφή του αρχείου, το Napster έκλεινε το κανάλι επικοινωνίας με το απομακρυσμένο σύστημα.

Παρακάτω φαίνεται σχηματικά ο τρόπος με τον οποίο επικοινωνούσαν οι χρήστες του Napster μεταξύ τους.



Εικόνα 1 : Τρόπος επικοινωνίας των χρηστών του Napster. Ένας κεντρικός διανομέας και διάφοροι «κλώνοι» του Napster συνδεδεμένοι με αυτόν στο περιβάλλον του Διαδικτύου. Ο κάθε χρήστης συνδέεται σε αυτό το δίκτυο και μπορεί να έχει πρόσβαση και λήψη πληροφοριών από οποιονδήποτε από αυτούς τους υπολογιστές.

Σε γενικές γραμμές η κεντρική ιδέα είναι ότι ο server υπάρχει για να διευκολύνει την διασύνδεση δυο υπολογιστών . Στον κεντρικό αυτο server υπάρχει ένα ευρετήριο με τα μετά – δομημένα (metadata) για κάθε αρχείο . Υπάρχει ένας κατάλογος δηλαδή στον κεντρικό υπολογιστή που δείχνει ποιο αρχείο βρίσκεται που , καθώς και την ημερομηνία που δημιουργήθηκε .Υπάρχει και ένας πίνακας με πληροφορίες για τη σύνδεση του χρήστη (Ip διεύθυνση που συνεχώς ανανεώνεται , την ταχύτητα της σύνδεσης κ.α.). Όταν ένας χρηστής συνδέεται με το Napster τότε φορτώνει από τον υπολογιστή του στο κεντρικό server του Napster των αρχείων που έχει με όλα τα στοιχειά που αναφέραμε πριν . Η λίστα αυτή καταχωρείται στο server . Τότε ο server δείχνει το μήνυμα ότι επιτυχώς παραλήφθηκε η λίστα των αρχείων . Έστω ότι θέλει να βρει ο χρήστης κάποιο αρχείο . Υποβάλλει ένα ερώτημα (query) προς τον server του Napster . Τότε ο server ψάχνει στα μεταδομένα που έχει ποιος έχει το αρχείο και στέλνει στον χρηστή μια λίστα των αρχείων καθώς και των χρηστών που τα κατέχουν. Μετά ο χρηστής επιλεγεί το αρχείο που θέλει να κατεβάσει ειδοποιώντας τον server . Τότε server ανοίγει την σύνδεση ανάμεσα στον χρήστη και σε αυτόν που έχει το αρχείο .

Όπως είπαμε το Napster δεν ήταν καθαρό p2p . Όμως το Napster κατάφερε να περιορίσει τον ρόλο του κεντρικού server , στο να υπάρχουν όλα τα στοιχειά για τα αρχεία και ταυτόχρονα πέτυχε την αποκεντροποίηση του δικτυού γιατί πλέον τα αρχεία που θα κατέβαζε ένας χρήστης θα το έκανε απευθείας από την πηγή και όχι μέσο του κεντρικού υπολογιστή . Επίσης κατάφερε και κάτι άλλο, πρωτοφανές για την περίοδο τότε , αξιοποίησε τον αποθηκευτικό χώρο των εκατομμυρίων Pc , δεν χρειαζόντουσαν να είναι όλα υποθηκευμένα σε ένα server και επίσης ελαχιστοποίησε το κόστος της διαχείρισης τόσων αρχείων .

Το Napster ως ένα μη καθαρό p2p σύστημα έχει μερικά σοβαρά μειονεκτήματα. Πρώτον είναι ευαίσθητο σε αποτυχία κόμβων . Αν καταρρεύσει ο κεντρικός server τότε πέφτει το δίκτυο. Δεύτερον , χρειάζεται μεγάλους υπολογιστικούς πόρους . Π. χ. για το κεντρικό ευρετήριο , όσο μεγαλώνει το δίκτυο θα χρειαζόμαστε και μεγαλύτερο αποθηκευτικό χώρο .

2.2 Gnutella

2.2.1 Gnutella και η ιστορία του

Το δίκτυο της Gnutella ανήκει στην κατηγορία των μη δομημένων αποκεντρωμένης τοπολογίας συστημάτων . Δηλαδή πρόκειται για ένα καθαρό p2p σύστημα . Το Gnutella δημιουργήθηκε τον Μάρτιο του 2000, από τους Justin Frankel και Tom Pepper (υπήρξαν και δημιουργοί του Winamp). Η εσωτερική του δομή αλλάζει και δεν είναι σταθερή . Οι κόμβοι και οι ακμές μπορεί να είναι φυσικά τα ίδια , αλλά το ποιοι κόμβοι και ποιες ακμές συμμετέχουν κάθε φορά στο δίκτυο αλλάζει κάθε δευτερόλεπτο ! Συνδυάζει δηλαδή μια δυναμική εικονική υποδομή πάνω σε στη φυσική υποδομή των δικτύων . Στο Gnutella υπάρχει απουσία κεντρικού Server .

2.2.2 Τρόπος Λειτουργίας (Περιγραφή του πρωτοκόλλου Gnutella)

Το πρωτόκολλο Gnutella είναι ένα ανοιχτό πρωτόκολλο που χρησιμοποιείται κυρίως για διαμοίραση αρχείων. Ο όρος Gnutella επίσης επιλέχθηκε για το ιδεατό δίκτυο από υπολογιστές που έχουν πρόσβαση στο διαδίκτυο και τρέχουν κάποια εφαρμογή που μιλάει το πρωτόκολλο Gnutella. Οι κόμβοι του Gnutella, ονομάζονται servents και εκτελούν λειτουργίες που

σχετίζονται τόσο με εξυπηρετητές (servers) όσο και με πελάτες (clients). Παρέχουν διεπαφές μέσω των οποίων οι χρήστες μπορούν να κάνουν ερωτήσεις και να δουν τα αποτελέσματα, να δεχτούν ερωτήσεις από άλλους servents, να ελέγξουν αν μπορούν να απαντήσουν, και αν ναι, να στείλουν τα αποτελέσματα. Οι ίδιοι κόμβοι είναι υπεύθυνοι για την διαχείριση της κίνησης η οποία υπάρχει και χρησιμοποιείται για την ακεραιότητα του δικτύου.

Για να μπορέσει ένας νέος κόμβος να συνδεθεί στο Gnutella πρέπει πρώτα να συνδεθεί με έναν από ένα πλήθος γνωστών κόμβων οι οποίοι είναι πάντοτε διαθέσιμοι (π.χ router.limewire.com). Έχοντας συνδεθεί στο δίκτυο (έχοντας μία ή περισσότερες ανοιχτές συνδέσεις με κόμβους ήδη συνδεδεμένους στο Gnutella), οι κόμβοι στέλνουν μηνύματα για να αλληλεπιδρούν μεταξύ τους. Τα μηνύματα μπορεί να γίνονται broadcast (π.χ στέλνονται σε όλους τους κόμβους με τους οποίους ο αποστολέας έχει ανοιχτές συνδέσεις) ή απλά να διαδίδονται προς τα πίσω (back-propagated)(π.χ να στέλνονται σε μία συγκεκριμένη σύνδεση στην αντίθετη κατεύθυνση από την διαδρομή που ακολούθησε ένα αρχικό broadcasted μήνυμα). Αρκετά χαρακτηριστικά του πρωτοκόλλου επιτυγχάνουν αυτόν τον μηχανισμό broadcast/back-propagation. Πρώτον, κάθε μήνυμα έχει ένα μοναδικό αριθμό. Δεύτερον, κάθε κόμβος κρατάει στην μνήμη του τα πιο πρόσφατα μηνύματα που δρομολόγησε και χρησιμοποιείται για να αποτρέψει το re-broadcasting και να κάνει εφικτή την υλοποίηση του back-propagation. Τρίτον, τα μηνύματα έχουν ένα πεδίο Time-to-live (TTL) το οποίο υποδηλώνει τον χρόνο ζωής του μηνύματος που προοδευτικά μειώνεται μέχρι να φτάσει το 0 όπου σταματάει η αναμετάδοση του μηνύματος για να μην έχουμε συμφόρηση στο δίκτυο και ένα πεδίο "Hops taken".

Τα μηνύματα που επιτρέπονται στο δίκτυο είναι PING, PONG, QUERY, QUERYHIT και PUSH. Τα μηνύματα PING και PONG χρησιμοποιούνται για την συντήρηση του δικτύου ενώ τα QUERY, QUERYHIT και PUSH για την αναζήτηση αρχείων. Η παραλαβή των αρχείων γίνεται χρησιμοποιώντας το πρωτόκολλο HTTP με απευθείας σύνδεση των δύο κόμβων(αυτού που έχει το αρχείο και αυτού που θέλει να το αποκτήσει).

Ανακεφαλαιώνοντας, για να γίνει μέλος του δικτύου, ένας κόμβος πρέπει να ανοίξει μία ή περισσότερες συνδέσεις με κόμβους οι οποίοι είναι ήδη στο δίκτυο. Στο δυναμικό περιβάλλον όπου το δίκτυο Gnutella λειτουργεί, κόμβοι συχνά συνδέονται και αποσυνδέονται και οι συνδέσεις είναι αναξιόπιστες. Για να συμβιβαστεί με αυτό το περιβάλλον, αφού έχει συνδεθεί με το δίκτυο, ένας κόμβος περιοδικά στέλνει μηνύματα PING στους γείτονες του (γείτονες είναι οι κόμβοι με τους οποίους έχει ανοιχτές συνδέσεις) για να ανακαλύψει και άλλους κόμβους. Χρησιμοποιώντας αυτή την πληροφορία, ένας μη συνδεμένος κόμβος μπορεί σχεδόν πάντα να συνδεθεί με το δίκτυο. Οι κόμβοι αποφασίζουν που θα συνδεθούν στο δίκτυο βασιζόμενοι μόνο σε τοπική πληροφορία και έτσι συνθέτουν ένα δυναμικό δίκτυο από ανεξάρτητες οντότητες. Αυτό το ιδεατό δίκτυο έχει τους servents για κόμβους του και τις ανοιχτές TCP συνδέσεις για συνδέσμους(links).

Το Gnutella παρουσιάζει κι αυτό κάποια μειονεκτήματα αυτά είναι τα εξής :

- Ένα πρόβλημα που εμφανίστηκε, το οποίο είναι αρκετά σημαντικό και συμβάλλει στην αδυναμία κλιμάκωσης του συστήματος είναι οι "ελεύθεροι χρήστες" (free riders). Οι χρήστες αυτοί δεν μοιράζονται αρχεία με αποτέλεσμα πολλές αναζητήσεις (οι οποίες έχουν ένα χρόνο ζωής TTL) να τερματίζονται χωρίς αρκετά αποτελέσματα..

- Ένα άλλο μειονέκτημα είναι ότι η ανάπτυξη λογισμικού για τη χρήση του δικτύου γίνεται από ανεξάρτητους φορείς και πολλές βελτιώσεις αλλά και αλλαγές στο πρωτόκολλο, για την επίλυση τυχόν προβλημάτων, διαφέρουν σε κάθε υλοποίηση, με αποτέλεσμα να μην υπάρχει ουσιαστική επίλυση του προβλήματος [1].

Γενικά όμως το Gnutella έχει αρκετά πλεονεκτήματα, καθώς οι αναζητήσεις επιστρέφουν συχνά αποτελέσματα σε ερωτήσεις των χρηστών. Υπάρχει μεγάλη ανοχή σε σφάλματα, και το σύστημα είναι ευέλικτο σε αλλαγές στην τοπολογία του συστήματος.

2.3 Gnutella2 (πιο πρόσφατη αρχιτεκτονική)

Οι πιο συχνές εφαρμογές που χρησιμοποιούν αυτό το δίκτυο είναι το Kazaa,Morpheus,Limewire,Shareaza κ.α. Είναι ένα μη δομημένο μερικώς αποκεντρωμένης τοπολογίας δίκτυο. Βασίζεται στην ιδέα των Υπερκόμβων (Supernodes) . Ο ρόλος των υπερκόβων είναι εξυπηρετούν ένα μικρότερο υποκομμάτι του δικτύου παρέχοντας ευρετήριο για τα αρχεία των χρηστών που ανήκουν σε αυτό το κομμάτι. Ένας μηχανισμός για δυναμική επιλογή υπερκομβων οργανώνει το δίκτυο , γίνεται SyperPeer και εγκαθιστά συνδέσεις με άλλους SyperPeers ,σχηματίζοντας ένα επίπεδο μη δομημένο δίκτυο από SuperPeers . Επίσης θετή τον αριθμό των πελατών που απαιτούνται για να παραμείνει SuperPeer . Αν λάβει τουλάχιστον τον απαιτούμενο αριθμό συνδέσεων μέσα σε ένα ορισμένο χρόνο παραμείνει SuperPeer .

2.4 Freenet

Το Freenet είναι ένα δίκτυο αποκεντρωμένης τοπολογίας με χαλαρή δομή που συμπεριφέρεται ως ένα αυτό-οργανωμένο σύστημα . Δεν έχει λοιπόν κάποιο κεντρικό server και έχει πολλές ομοιότητες με το Gnutella 1 , έχει και όμως κάποιες διαφορές αν και η τεχνολογία τους είναι παρόμοια . Ένα σύστημα ομότιμων βάσεων για τη δημοσίευση, την αντένσταση και την ανάκτηση αρχείων δεδομένων αντιπροσωπεύετε απτό το δίκτυο Freenet . Ο κύριος στόχος του είναι να παρέχει μια υποδομή που προστατεύει την ανωνυμία των συντακτών και των αναγνωστών των δεδομένων. Έχει σχεδιαστεί με έναν τρόπο που κάνει αδύνατο τον προσδιορισμό της προέλευσης των αρχείων ή του προορισμού των αιτημάτων. Είναι επίσης δύσκολο για έναν κόμβο να καθορίσει τι αποθηκεύει, δεδομένου ότι τα αρχεία κρυπτογραφούνται όταν στέλνονται και αποθηκεύονται μέσω του δικτύου. Έτσι, σύμφωνα με τον συλλογισμό των σχεδιαστών του FreeNet δεν τίθεται θέμα μήνυσης κανενός ακόμη και αν αυτός ή αυτή π.χ αποθηκεύει ή διανέμει παράνομο περιεχόμενο. Εκτός από την άποψη της προστασίας της ανωνυμίας το σύστημα Freenet ενσωματώνει και μια άλλη ενδιαφέρουσα ιδέα: ένα προσαρμοζόμενο σχέδιο δρομολόγησης για αποτελεσματικές αναζητήσεις δρομολόγησης στις φυσικές τοποθεσίες που είναι περισσότερο πιθανό να εμφανιστούν. Προκειμένου να βελτιωθεί η αποδοτικότητα αναζήτησης, το FreeNet διατηρεί πίνακες δρομολόγησης που ενημερώνονται δυναμικά καθώς συμβαίνουν αναζητήσεις και εισαγωγές νέων στοιχείων. Έτσι, ο γράφος του FreeNet εξελίσσεται δυναμικά με το πέρασμα του χρόνου, μέσα από τις ενημερώσεις των πινάκων δρομολόγησης. Για την παραπέρα βελτίωση της απόδοσης της αναζήτησης, το FreeNet χρησιμοποιεί δυναμική

απόκριση σε αναζητήσεις των πιο δημοφιλών αρχείων έτσι ώστε τα αρχεία να μπορούν να μεταφέρονται σε ομότιμους όπου είναι περισσότερο πιθανό να βρεθούν. Κατ' αυτόν τον τρόπο, το FreeNet δεν χρειάζεται ένα κεντρικό εξυπηρετητή όπως ο Napster, και σε σύγκριση με το Gnutella, αποφεύγει την άσκοπη εκπομπή μηνυμάτων.

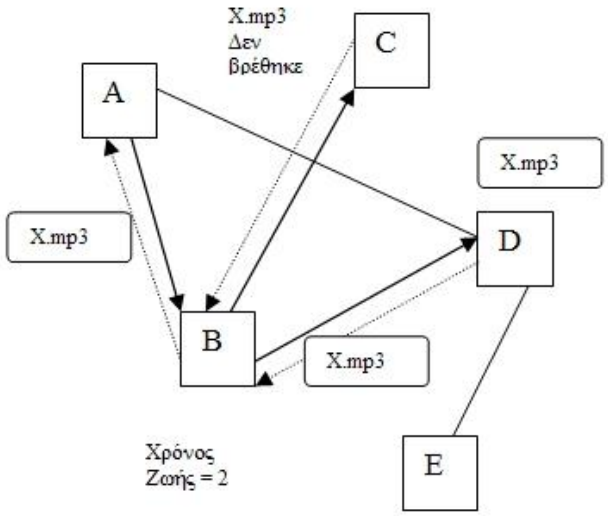
Όταν ένας ομότιμος κόμβος συνδέεται σε ένα δίκτυο FreeNet πρέπει να γνωρίζει κάποιο υπάρχοντα κόμβο στο δίκτυο (όπως και στο Gnutella). Μέσα από την αλληλεπίδραση του με το δίκτυο θα γεμίσει τον πίνακα δρομολόγησής του, ο οποίος είναι αρχικά κενός, και έτσι η δομή του δικτύου του FreeNet θα εξελιχθεί. Οι πίνακες δρομολόγησης στο FreeNet αποθηκεύουν τις διευθύνσεις των γειτονικών ομότιμων κόμβων, τα κλειδιά των στοιχείων δεδομένων που αυτός ο ομότιμος κόμβος αποθηκεύει, και τα αντίστοιχα δεδομένα. Έτσι όταν ένα αίτημα αναζήτησης φθάνει, ο κόμβος αποθηκεύει τα δεδομένα στον πίνακά του και μπορεί αμέσως να απαντήσει στο αίτημα. Διαφορετικά, πρέπει να διαβιβάσει το αίτημα σε έναν άλλο ομότιμο κόμβο. Αυτό γίνεται με την επιλογή εκείνου που έχει το «κοντινότερο» λεξικογραφικά κλειδί. Όταν φθάνει μια απάντηση, ο κόμβος αποθηκεύει την απάντηση στην αποθήκη των δεδομένων του. Εάν η αποθήκη των δεδομένων του είναι ήδη πλήρης, μπορεί να χρειαστεί να διαγράψει άλλες, προηγούμενες καταχωρήσεις. Εδώ η στρατηγική είναι να διαγραφεί η λιγότερο πρόσφατα χρησιμοποιημένη καταχώρηση. Ο ομότιμος μπορεί να αποφασίσει να εκδιώξει τα στοιχεία που αντιστοιχούν σε ένα κλειδί προτού να εκδιώξει τη διεύθυνση του κόμβου, προκειμένου να εξοικονομήσει χώρο στη μνήμη διατηρώντας παράλληλα τις πληροφορίες δρομολόγησης. Δεδομένου ότι οι κόμβοι καθοδηγούν τα αιτήματα αναζήτησης μόνο στον ομότιμο με το πιο κοντινό κλειδί, το FreeNet εφαρμόζει μια στρατηγική αναζήτησης κατά βάθος παρά μια κατά πλάτος στρατηγική όπως το Gnutella.

Για το λόγο αυτό ο χρόνος ζωής των μηνυμάτων είναι ουσιαστικά μεγαλύτερος , χαρακτηριστικά 500. Όπως και στο Gnutella, έτσι και στο FreeNet τα μηνύματα φέρνουν αναγνωριστικά προκειμένου να ανιχνευθούν οι κύκλοι.

Το σχήμα 2.3 επεξηγεί τους μηχανισμούς αναζήτησης ελεύθερου δικτύου και την αναδιοργάνωση του δικτύου. Ο ομότιμος A στέλνει ένα αίτημα αναζήτησης στο B για το αρχείο X.mp3. Δεδομένου ότι ο χρόνος ζωής είναι 2 και ο C δεν έχει τα δεδομένα, το αίτημα αποτυγχάνει. Έτσι ο B προωθεί το αίτημα στον D, όπου και βρίσκεται το αρχείο. Κατά την αποστολή της απάντησης που περιέχει το αρχείο X.mp3, το αρχείο εναποθηκεύεται σε όλους τους ενδιάμεσους κόμβους, δηλ., στον B και τον A. Επιπλέον ο A μαθαίνει για τον νέο κόμβο C και εντοπίζει έτσι μια νέα σύνδεση στο δίκτυο.

Στη συνέχεια, περιγράφουμε το μηχανισμό ενημερώσεων στο FreeNet. Όταν ένα αρχείο εισάγεται στο δίκτυο του FreeNet, υπολογίζεται αρχικά ένα κλειδί για το αρχείο αυτό. Τα κλειδιά παρουσιάζονται σαν ομοιόμορφα προσδιοριστικά πόρων(URLs) τύπου freenet: keytype@data. Στο FreeNet διακρίνουμε πέντε διαφορετικά είδη κλειδιών: κλειδιά υπογεγραμμένα με λέξεις κλειδιά (KSK), κλειδιά επαλήθευσης υπογραφών (SVK),κλειδιά κατακερματισμένου περιεχομένου (CHK). Η πρώτη κατηγορία κλειδιών χρησιμοποιεί ένα σύντομο περιγραφικό κείμενο χρήσης που παρέχεται από το χρήστη προκειμένου να παράγει ένα ζευγάρι δημόσιο/ιδιωτικό κλειδί. Το δημόσιο κλειδί κατακερματίζεται και χρησιμοποιείται ως κλειδί αρχείων, ενώ τα ιδιωτικά κλειδιά χρησιμοποιούνται για να υπογράψουν το αρχείο. Το ίδιο το αρχείο κρυπτογραφείται χρησιμοποιώντας τον καθορισμένο από το χρήστη περιγραφέα(descriptor) σαν κλειδί. Για τον εντοπισμό του αρχείου ο χρήστης πρέπει να ξέρει το περιγραφικό κείμενο. Αυτός ο τρόπος της

κρυπτογράφησης, που υπογράφει τα αρχεία, παρέχει ένα μέτριο επίπεδο ασφάλειας, καθώς είναι ευάλωτος σε επιθέσεις λεξικού. Τα κλειδιά επαλήθευσης υπογραφών επιτρέπουν τη δημιουργία υποχώρων στο FreeNet, πχ ένα ελεγχόμενο σύνολο κλειδιών. Αυτά λειτουργούν όμοια με τα κλειδιά υπογεγραμμένα με λέξεις κλειδιά μόνο που αυτό το ζευγάρι ιδιωτικού/δημόσιου κλειδιού παράγεται τυχαία.



Εικόνα 2 : **Αναζήτηση στο FreeNet**

Οι χρήστες που εμπιστεύονται τον ιδιοκτήτη ενός υποχώρου(subspace), εμπιστεύονται επίσης και τα έγγραφα αυτού του υποχώρου, καθώς η προσθήκη ενός έγγραφου σε αυτόν προϋποθέτει γνώση του ιδιωτικού του κλειδιού. Τα κλειδιά κατακερματισμένου περιεχομένου παράγονται με κατακερματισμό του περιεχομένου του αρχείου. Αποτελούν έτσι ένα τρόπο ελέγχου της αξιοπιστίας (integrity) του αρχείου αυτού. Αφότου δημιουργηθεί ένα κλειδί, ενημερώνονται οι γείτονες του τρέχοντος κόμβου μέσω μηνυμάτων εισαγωγής. Στη συνέχεια, κάθε ένας από αυτούς τους γείτονες κόμβους ελέγχει εάν έχει ήδη καταχωρημένο τοπικά το προτεινόμενο κλειδί. Εάν ναι, επιστρέφει το αποθηκευμένο αρχείο και ο αρχικός κόμβος πρέπει να προτείνει ένα νέο κλειδί. Εάν όχι, τότε δρομολογεί το μήνυμα εισαγωγής στον

επόμενο κόμβο για παραπέρα έλεγχο. Η δρομολόγηση χρησιμοποιεί το ίδιο βασικό κλειδί ομοιότητας με την αναζήτηση. Το μήνυμα εισαγωγής διαβιβάζεται έως ότου ο χρόνος ζωής του μηνύματος μειωθεί σε 0 ή εμφανιστεί μια αποτυχία. Εάν ο χρόνος ζωής του μηνύματος είναι 0 και δεν έχει ανιχνευθεί καμία σύγκρουση, το αρχείο εισάγεται στο μονοπάτι που καθορίζεται από το αρχικό μήνυμα εισαγωγής

2.4.1 Συζήτηση για το FreeNet

Δεδομένου ότι η μέθοδος εναποθήκευσης για τους πίνακες δρομολόγησης έχει σχεδιαστεί με έναν τρόπο που τείνει να ομαδοποιεί τα δεδομένα (κλειδιά) που παρουσιάζουν ομοιότητες στους κόμβους του δικτύου, υποθέτουμε ότι οι κόμβοι αυτοί γίνονται περισσότερο ειδικευμένοι με το πέρασμα του χρόνου. Αυτή η υπόθεση επαληθεύεται και κατά τις προσομοιώσεις του ελεύθερου δικτύου. Περιγράφουμε ένα αντιπροσωπευτικό αποτέλεσμα: Η προσομοίωση έλαβε χώρα σε ένα δίκτυο 1000 πανομοιότυπων κόμβων, όπου κάθε κόμβος αποθηκεύει το ανώτερο 50 στοιχεία δεδομένων και μπορεί να απευθύνεται το ανώτερο σε 200 άλλους κόμβους. Η αρχική τοπολογία του δικτύου ήταν τοπολογία δακτυλίου, όπου κάθε κόμβος συνδέεται με 4 γείτονες με τα πιο κοντινά αναγνωριστικά.

Σε κάθε βήμα του χρόνου της προσομοίωσης εκτελείται τυχαία μια λειτουργία προσθήκης , με έναν χρόνο ζωής 20. Κάθε 100 βήματα, εκτελούνταν από κόμβους - τυχαία επιλεγμένους - 300 ερωτήσεις με χρόνο ζωής TTL = 500. Μπορεί να παρατηρήσει κανείς ότι το μέσο μήκος μονοπατιού των επερωτήσεων κοντά στο μέγιστο TTL 500 μειώνεται αισθητά από τις αρχικές τιμές σε μία τιμή περίπου 6. Αυτή, και άλλες παρόμοιες προσομοιώσεις υποδεικνύουν ότι ο αριθμός των βημάτων κατά την αναζήτηση μεταβάλλεται λογαριθμικά με τον αριθμό των κόμβων στο δίκτυο.

Μια εξήγηση στο γιατί η απόδοση αναζήτησης βελτιώνεται τόσο εντυπωσιακά μπορεί να αναζητηθεί στις ιδιότητες της δομής των γράφων του δικτύου. Οι αναλυτές των δικτύων που δημιούργησαν το FreeNet αποκαλύπτουν ότι το τελευταίο έχει τα χαρακτηριστικά των αποκαλούμενων «γράφων του μικρού κόσμου» (small world graphs). Η έννοια του φαινομένου του «μικρού κόσμου» χρησιμοποιήθηκε για πρώτη φορά από τονStanley Milgram το 1967. Αυτός πραγματοποίησε ένα πείραμα στο οποίο διέταξε τυχαία επιλεγμένους ανθρώπους στη Νεμπράσκα να μεταφέρουν επιστολές σε ένα συγκεκριμένο πρόσωπο-στόχο στη Βοστόνη . Τα πρόσωπα για την προώθηση της επιστολής έπρεπε να γνωρίζονται μεταξύ τους σε επίπεδο ονόματος. Είναι ιδιαίτερα ενδιαφέρον ότι ο μέσος αριθμός βημάτων που απαιτήθηκαν για να φθάσουν οι επιστολές στο δέκτη ήταν μόλις 6.

Η εργασία επεκτάθηκε σε ένα πλαίσιο πληροφορικής από τους Duncan Watts και Steven Strogatz το 1998. Προσδιόρισαν τις κατηγορίες γράφων που εκθέτουν μια ιδιότητα ισχυρής ομαδοποίησης, αλλά περιέχουν μικρά μονοπάτια μεταξύ οποιωνδήποτε κόμβων (small world graphs). Η ισχυρή συγκέντρωση σημαίνει ότι οι κόμβοι που συνδέονται με έναν δεδομένο κόμβο, έχουν επίσης μεγάλη πιθανότητα να συνδεθούν ο ένας με τον άλλον, όπως ακριβώς συμβαίνει για παράδειγμα στις ανθρώπινες σχέσεις. Από την άλλη μεριά όμως τα σύντομα μονοπάτια μία γνωστή ιδιότητα των τυχαίων γράφων, καθώς σύμφωνα με τη θεωρία των γράφων τα μονοπάτια έχουν μήκος λογαριθμικό σε σχέση με τον αριθμό των κόμβων στο γράφο.

Εντούτοις, η υπόθεση ότι τα κοινωνικά δίκτυα είναι τυχαίοι γράφοι είναι αρκετά μη ρεαλιστική. Οι γράφοι μικρού κόσμου έχουν σύντομα μονοπάτια, δεδομένου ότι διασπούν τις λίγες, μεγάλης απόστασης συνδέσεις που υπάρχουν σε μια δομή γράφων που είναι συνήθως τοπικά συγκεντρωμένη. Ο αλγόριθμος του FreeNet επιτυγχάνει τη δημιουργία

τέτοιων γράφων χάρη στην τάση του να ομαδοποιεί τους κόμβους που αποθηκεύουν παρόμοια στοιχεία. Εφόσον, βέβαια η ομαδοποίηση αυτή δεν είναι τέλεια, οι μεγάλης απόστασης συνδέσεις είναι πάντα πιθανό να εμφανιστούν.

2.5 Επισκόπηση μεθόδων αναζήτησης στα Μη δομημένα P2P συστήματα

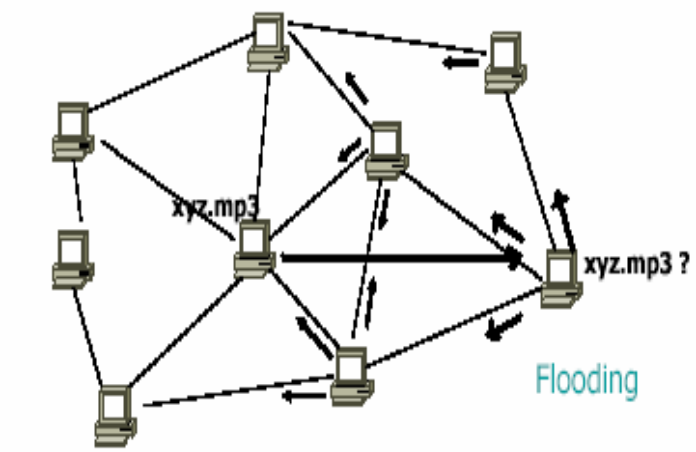
Στα συστήματα αυτά δεν υπάρχει κάποια δομή στην οργάνωση των κόμβων ούτε στον τρόπο με τον οποίο τοποθετούνται τα δεδομένα στο δίκτυο. Η συμπεριφορά ολόκληρου του δικτύου καθορίζεται από τις τοπικές αντιδράσεις. Για να εντοπιστεί ένα δεδομένο, ένας κόμβος ρωτά τους γείτονές του. Η πιο συνηθισμένη μέθοδος αναζήτησης είναι η πλημμύρα, δηλαδή η διαδοχική μετάδοση της ερώτησης σε όλους τους γείτονες μέχρι μια συγκεκριμένη ακτίνα.

Κατά καιρούς έχουν εφαρμοστεί αρκετές μέθοδοι αναζήτησης με σκοπό κυρίως να βελτιώσουν την απόδοση του δικτύου. Οι μέθοδοι αυτές σε καμία περίπτωση δεν υπαγορεύουν τον τρόπο αποθήκευσης των δεδομένων και των αντιγράφων τους στο δίκτυο ούτε επιτρέπεται να τροποποιήσουν την τοπολογία του δικτύου.

Υπάρχουν δύο πιθανές στρατηγικές για αναζήτηση σε ένα μη δομημένο P2P δίκτυο: οι μέθοδοι τυφλής αναζήτησης, όπου η ερώτηση προωθείται σε έναν επαρκή αριθμό κόμβων ώστε να βρεθούν ικανοποιητικά αποτελέσματα, και οι μέθοδοι “ενημερωμένης” αναζήτησης, όπου η μετάδοση της ερώτησης στηρίζεται σε κάποιες πληροφορίες-στατιστικά.[3]

2.5.1 Μέθοδοι τυφλής αναζήτησης

GNUTELLA: Είναι ο κλασικός αλγόριθμος που εφαρμόζεται στο δίκτυο Gnutella και χρησιμοποιεί την πλημμύρα. Ένας κόμβος που επιθυμεί ένα δεδομένο μεταδίδει την ερώτηση στους γείτονές του, αυτοί με τη σειρά τους την προωθούν στους δικούς τους και η διαδικασία αυτή επαναλαμβάνεται μέχρι το δεδομένο αυτό να βρεθεί σε κάποιον κόμβο που ερωτήθηκε ή η ερώτηση να εξαπλωθεί σε μια περιοχή με δεδομένη ακτίνα ή να λήξει ο χρόνος κάποιος ορισμένος χρόνος TTL(Time To Live). Στο Σχήμα περιγράφεται η παραπάνω διαδικασία όπου ο δεξιότερος κόμβος που αναζητά το δεδομένο “xyz.mp3” ρωτά τους γείτονές του και εκείνοι με τη σειρά τους, τους δικούς τους. Παρόλο που ο αλγόριθμος είναι απλός δε βοηθά στη διατήρηση της κλιμάκωσης και επιπλέον επιφέρει μεγάλη επιβάρυνση στο δίκτυο.



Εικόνα 3 : **Πλημμύρα στο Gnutella**

MODIFIED-BFS: Είναι μια παραλλαγή του προηγούμενου αλγορίθμου, όπου κάθε κόμβος επιλέγει τυχαία έναν αριθμό γειτόνων του στους οποίους και προωθεί την ερώτηση. Η μέθοδος αυτή μειώνει το μέσο

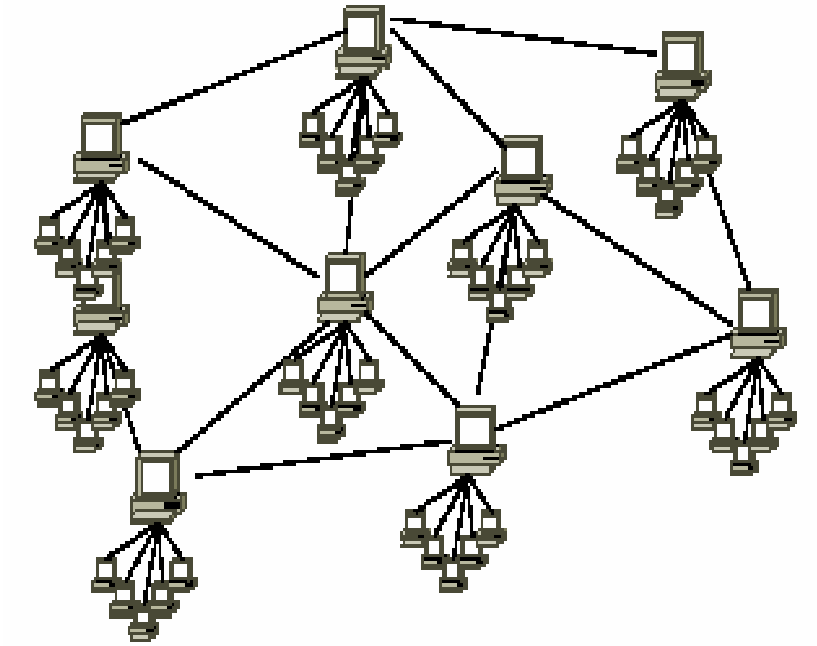
όρο μηνυμάτων-ερωτήσεων άλλα επικοινωνεί πάλι με ένα μεγάλο αριθμό κόμβων.

ITERATIVE DEEPENING: Η αναζήτηση ξεκινά με έναν αλγόριθμο BFS και με κριτήριο τερματισμού μια μικρή τιμή TTL, ενώ ο αλγόριθμος αυτός επαναλαμβάνεται κάθε φορά με μεγαλύτερο βάθος αν η πρώτη εφαρμογή του BFS αποτύχει. Έχει καλά αποτελέσματα αν χρησιμοποιηθεί κάποιο κριτήριο τερματισμού και αν μια "μικρή" πλημμύρα (εξάπλωση της ερώτησης σε μικρή περιοχή) ικανοποιήσει την ερώτηση, διαφορετικά επιφέρει μεγαλύτερη επιβάρυνση από μια απλή πλημμύρα.

RANDOM WALKS: Στο συγκεκριμένο αλγόριθμο ο κόμβος που διατυπώνει μια ερώτηση την προωθεί σε k τυχαία επιλεγμένους γείτονες και στη συνέχεια σε κάθε βήμα καθένας από αυτούς τους γείτονες προωθεί την ερώτηση σε έναν τυχαία επιλεγμένο γείτονά του. Έτσι δημιουργούνται k μονοπάτια προώθησης της ερώτησης τα οποία ονομάζονται "walkers". Υπάρχουν δύο τρόποι τερματισμού ενός μονοπατιού: με βάση κάποιο TTL ή με μια μέθοδο ελέγχου όπου κάθε "walker" περιοδικά ελέγχει (ρωτάει) τον κόμβο που έκανε την ερώτηση αν ισχύει το κριτήριο τερματισμού. Το πιο σημαντικό πλεονέκτημα αυτού του αλγορίθμου είναι ότι μειώνει αισθητά τα μηνύματα που διακινούνται στο δίκτυο. Επίσης προξενεί και κάποιο είδος τοπικής εξισορρόπησης φορτίου.

GNUTELLA2: Αυτός ο αλγόριθμος εφαρμόζεται όταν στο δίκτυο υπάρχουν "super" κόμβοι οι οποίοι συνδέονται μεταξύ τους. Καθένας επίσης από αυτούς συνδέεται με έναν αριθμό από κόμβους "φύλλα". Παράδειγμα ενός τέτοιου δικτύου φαίνεται στο Σχήμα όπου οι μεγάλοι υπολογιστές

αναπαριστούν τους “super” κόμβους και οι μικροί τους κόμβους “φύλλα”. Όταν κάποιος “super” κόμβος δέχεται μια ερώτηση από έναν κόμβο “φύλλο” την προωθεί σε όλους τους υπόλοιπους κόμβους “φύλλα” και στους γειτονικούς του “super” κόμβους, οι οποίοι με τη σειρά τους επεξεργάζονται την ερώτηση τοπικά κι επίσης τη μεταδίδουν στους δικούς τους κόμβους “φύλλα”.



Εικόνα 4 **:** “super” κόμβοι

2.5.2 Μέθοδοι “ενημερωμένης” αναζήτησης

INTELLIGENT-BFS: Είναι μια νέα έκδοση του αλγορίθμου MODIFIED-BFS όπου επιπλέον υπάρχει εκμετάλλευση κάποιων στατιστικών για τη δρομολόγηση της ερώτησης. Κάθε κόμβος κρατά εγγραφές με ζεύγη (πληροφορία ερώτησης, χαρακτηριστικό (ID) γείτονα). Οι πληροφορίες αυτές αφορούν τις πιο πρόσφατες ερωτήσεις που έχουν απαντηθεί μέσω των γειτόνων. Ένας κόμβος που θέλει να προωθήσει μια ερώτηση ακολουθεί τα εξής βήματα: ελέγχει πρώτα με ποιες από τις πιο πρόσφατες ερωτήσεις από αυτές που έχουν περάσει από εκεί ταιριάζει περισσότερο η νέα ερώτηση

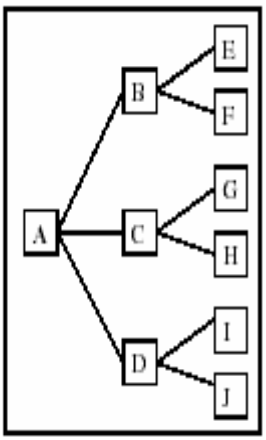
σύμφωνα με κάποιο μέτρο (χρησιμοποιώντας τις πληροφορίες που κρατά) και προωθεί στη συνέχεια την ερώτηση σε εκείνο το σύνολο κόμβων που έδωσε τα πιο ικανοποιητικά αποτελέσματα για τις προηγούμενες ερωτήσεις.

APS: Στο συγκεκριμένο αλγόριθμο, κάθε κόμβος κρατά ένα τοπικό ευρετήριο όπου υπάρχουν εγγραφές κάθε δεδομένου που έχει αναζητήσει ανά γείτονα. Η τιμή της εγγραφής αντιπροσωπεύει τη σχετική πιθανότητα να επιλεγεί ένας συγκεκριμένος γείτονας αυτού του κόμβου σε μια μελλοντική αναζήτηση του ίδιου δεδομένου. Εφαρμόζεται κι εδώ η τεχνική των k ανεξάρτητων “walkers” κι έτσι κάθε ενδιάμεσος κόμβος προωθεί την ερώτηση σε έναν από τους γείτονες με βάση το τοπικό ευρετήριο. Το τελευταίο ενημερώνεται κάθε φορά με αύξηση της πιθανότητας αν ο “walker” επιτύχει και μείωση αντίθετα.

ROUTING INDICES (RIs) : Η βασική ιδέα στην οποία στηρίζεται η μέθοδος αυτή είναι η δρομολόγηση των ερωτήσεων στους γείτονες των κόμβων που είναι πιο πιθανό να έχουν απαντήσεις. Ανήκει στην κατηγορία της κατανεμημένης αναζήτησης με ευρετήριο σε κάθε κόμβο. Τα ευρετήρια αυτά όπως δηλώνει και το όνομά τους δίνουν την κατεύθυνση που πρέπει να ακολουθηθεί για ένα δεδομένο και όχι την πραγματική τοποθεσία. Έτσι το μέγεθός τους δεν αυξάνει πολύ και είναι ανάλογο του αριθμού των γειτόνων και όχι του αριθμού των δεδομένων. Υπάρχουν τρία είδη RI ευρετηρίων: τα “compound” (CRI), τα “hop-count” (HRI) και τα “exponential” (ERI). Ένα RI είναι μια δομή δεδομένων η οποία δεδομένης μιας ερώτησης επιστρέφει μια λίστα με ταξινομημένους γείτονες ανάλογα με κάποιες πληροφορίες που κρατούν ως προς την ερώτηση. Αυτές οι πληροφορίες πρέπει να δηλώνουν τον αριθμό των δεδομένων που μπορούν να βρεθούν μέσω των γειτόνων.

Έτσι κάθε κόμβος περιέχει ένα τοπικό ευρετήριο κι ένα CRI που περιλαμβάνει τον αριθμό των δεδομένων που αποθηκεύονται κατά μήκος κάθε μονοπατιού και τον αριθμό των δεδομένων κάθε κατηγορίας-θέματος. Δεδομένης μιας ερώτησης, ο κόμβος πρέπει να επιλέξει τον "καλύτερο" γείτονα για να την δρομολογήσει. "Καλύτερος" γείτονας θεωρείται αυτός που μεγιστοποιεί την ποσότητα #δεδομένων $\times \prod_i$ (CRI(s_i) / #δεδομένων) με βάση τα στοιχεία του CRI, όπου CRI(s_i) είναι ο αριθμός των δεδομένων μιας συγκεκριμένης κατηγορίας που μπορούν να βρεθούν μέσω ενός γείτονα i, όπως φαίνεται στην εικόνα 5 . Στο συγκεκριμένο παράδειγμα έχουμε το CRI ενός κόμβου A. ο τελευταίος μπορεί να βρει μέσω του γείτονά του B 100 δεδομένα από τα οποία τα 20 αφορούν στην κατηγορία "databases", 0 στην κατηγορία "networks", 10 στην κατηγορία "theory" και 30 στην κατηγορία "languages". Με τον ίδιο τρόπο αναφέρεται και ο αριθμός των δεδομένων κάθε κατηγορίας που μπορούν να βρεθούν μέσω των άλλων γειτόνων του A.

Path	# docs	Documents with topics: DB	N	T	L
B	100	20	0	10	30
C	1000	0	300	0	50
D	200	100	0	100	150

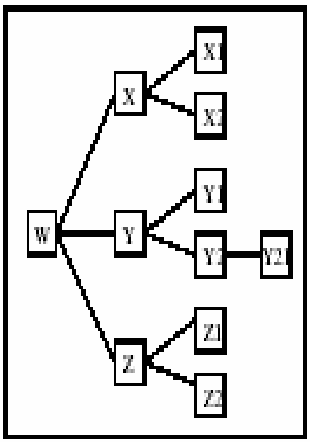


Εικόνα 5: Δείγμα ενός Compound RI για τον κόμβο A

Ένα "hop-count" RI σε σχέση με τα προηγούμενα λαμβάνει υπόψη του και τον αριθμό των "hops", δηλαδή των βημάτων, των αριθμών των μεταβάσεων σε κόμβους, που απαιτούνται για την εύρεση κάποιων δεδομένων. Έτσι αυτά αποθηκεύουν συγκεντρωτικά RIs για κάθε "hop" λαμβάνοντας υπόψη ένα μέγιστο αριθμό από "hops" που ονομάζεται

“horizon”. Όσον αφορά τη δρομολόγηση χρειάζεται κι εδώ ένας εκτιμητής των στοιχείων του HRI ώστε να δρομολογείται η ερώτηση στον “καλύτερο” γείτονα, ακολουθώντας έτσι μια παρόμοια διαδικασία όπως πριν. Στην εικόνα 6 φαίνεται το HRI του κόμβου w με 2 “hops”. Οι πληροφορίες που αποθηκεύονται είναι αυτές που περιγράφτηκαν και σε ένα CRI.

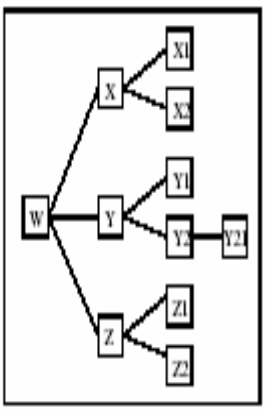
Node	1 Hop #	DB	N	T	L	2 Hops #	DB	N	T	L
X	60	13	2	5	10	20	10	10	4	17
Y	30	0	3	15	12	50	31	0	15	20
Z	5	2	0	3	3	70	10	40	20	50



Εικόνα 6: Δείγμα ενός Hop-count RI για τον κόμβο W

Τέλος ένα ERI αποθηκεύει τα αποτελέσματα ενός εκτιμητή HRI και με βάση αυτά δρομολογεί την ερώτηση σε κάποιον από τους γείτονες. Στην εικόνα 7 υπάρχει το ERI του κόμβου w. Σε αυτό αποθηκεύονται τα αποτελέσματα που προκύπτουν από τον εκτιμητή ενός HRI του συγκεκριμένου κόμβου, δηλαδή κάποιες “ποσότητες” για τις πληροφορίες που αναφέρθηκαν.

Path	#	DB	N	T	L
X	66.67	16.33	5.33	6.33	15.67
Y	46.67	10.33	3.00	20.00	18.67
Z	28.33	5.33	13.33	9.67	19.67



Εικόνα 7 : Δείγμα ενός ERI για τον κόμβο W

2.5.3 Μέθοδοι τοποθέτησης αντιγράφων (replication)

Δεδομένου του συνολικού αποθηκευτικού χώρου στο δίκτυο, το πρόβλημα που προκύπτει είναι πόσα αντίγραφα των δεδομένων μπορούν να δημιουργηθούν και σε ποιες τοποθεσίες ώστε το μέγεθος της αναζήτησης να ελαχιστοποιείται. Υπάρχουν τρεις τύποι “replication” που καθορίζουν και το ποσοστό των αντιγράφων.

Θεωρούμε ότι έχουμε m δεδομένα και n κόμβους. Κάθε δεδομένο αντιγράφεται σε r_i διαφορετικούς κόμβους και ο συνολικός αριθμός των δεδομένων που αποθηκεύονται είναι R τέτοιος ώστε $\sum_{i=1,m} r_i=R$. Επιπλέον, ένα δεδομένο i ζητείται με ρυθμό q_i για το οποίο ισχύει $\sum_{i=1,m} q_i=1$. Μέσο μέγεθος αναζήτησης A όλων των δεδομένων είναι ο μέσος αριθμός των κόμβων που ρωτούνται για κάθε δεδομένο. Το μέσο μέγεθος αναζήτησης για ένα δεδομένο ισούται με $A_i=n/ r_i$. Επομένως $A=\sum_{i=1,m} q_i \times A_i$.

2.5.3.1 Uniform

Δημιουργείται ο ίδιος αριθμός αντιγράφων για κάθε δεδομένο, δηλαδή $r_i=R/m$. Επομένως ισχύει $A_i=n/ r_i=m/\rho$ και $A_{uniform}=\sum_{i=1,m} q_i \times m/\rho= m/\rho$. Αυτό σημαίνει ότι δεν εξαρτάται από την κατανομή των ερωτήσεων, δηλαδή αν τα δεδομένα είναι το ίδιο δημοφιλή ή όχι.

2.5.3.2 Proportional

Ο αριθμός των αντιγράφων που δημιουργούνται για κάθε δεδομένο είναι ανάλογος προς το ρυθμό των ερωτήσεων, δηλαδή $ri=R \times qi$. Επομένως ισχύει

$A_i=n/r_i=n/(R \times q_i)$

$A_{uniform}=\sum_{i=1,m} q_i \times n/(R \times q_i)= m/\rho= A_{uniform}$

Δηλαδή το μέσο μέγεθος είναι πάλι ανεξάρτητο της κατανομής των ερωτήσεων.

2.5.3.3 Square-Root

Ο αριθμός των αντιγράφων για κάθε δεδομένο i είναι ανάλογος προς την τετραγωνική ρίζα του ρυθμού ερωτήσεων q_i, δηλαδή $r_i \propto \sqrt{q_i}$. Επομένως ισχύει

$A=\sum_{i=1,m} q_i \times A_i = n \times \sum_{i=1,m} q_i / r_i$

Αν εφαρμόζεται “uniform replication” όλα τα δεδομένα έχουν το ίδιο μέσο μέγεθος αναζήτησης, ενώ αν εφαρμόζεται “proportional replication” επιτυγχάνεται καλύτερη εξισορρόπηση φορτίου αλλά το μέσο μέγεθος αναζήτησης ποικίλλει έτσι που τα πιο δημοφιλή δεδομένα έχουν μικρότερο μέσο μέγεθος αναζήτησης από τα λιγότερα δημοφιλή δεδομένα.

2.5.3.4 Owner replication

Είναι στρατηγική που εφαρμόζεται για δημιουργία αντιγράφων σε διάφορους κόμβους. Όταν μία αναζήτηση είναι επιτυχής, το δεδομένο αποθηκεύεται μόνο στον κόμβο που έκανε αρχικά την ερώτηση. Αυτή η στρατηγική χρησιμοποιείται συνήθως στο δίκτυο Gnutella.

2.5.3.5 Path replication

Άλλη μία στρατηγική κατά την οποία όταν μία αναζήτηση είναι επιτυχής, το δεδομένο αποθηκεύεται σε όλους τους κόμβους που βρίσκονται

στο μονοπάτι μεταξύ του αρχικού κόμβου και του κόμβου που βρέθηκε το δεδομένο (τελικός).

Αν ένα P2P σύστημα εφαρμόζει αναζήτηση με k-walkers, τότε ο αριθμός των κόμβων μεταξύ του αρχικού και τελικού κόμβου είναι 1/ k των συνολικών κόμβων που έχουν ερωτηθεί. Σε αυτή την περίπτωση η στρατηγική “path replication” καταλήγει στον τύπο “square-root replication”. Προκύπτει όμως το πρόβλημα της αντιγραφής δεδομένων σε κόμβους που είναι τοπολογικά κατά μήκος του ίδιου μονοπατιού.

2.5.3.6 Random replication

Είναι μία στρατηγική κατά την οποία όταν μία αναζήτηση επιτύχει υπολογίζεται ο αριθμός των κόμβων στο μονοπάτι μεταξύ του αρχικού και του τελικού κόμβου, έστω p. Στη συνέχεια επιλέγονται τυχαία p κόμβοι από όλους τους κόμβους που επισκέφτηκαν οι k-walkers ώστε να αποθηκευτούν τα αντίγραφα. Η στρατηγική αυτή είναι πιο δύσκολο να εφαρμοστεί.

2.5.4 Μέθοδοι ενημέρωσης αντιγράφων

Στα περισσότερα P2P συστήματα ισχύει η θεώρηση ότι τα δεδομένα δεν τροποποιούνται. Υπάρχουν όμως και αρκετές εφαρμογές P2P δικτύων στα οποία τα δεδομένα τροποποιούνται με το χρόνο και έτσι είναι απαραίτητο τα αντίγραφά τους να ενημερώνονται. Το πρόβλημα λοιπόν της ενημέρωσης αντιγράφων αφορά σε ένα P2P σύστημα όπου όλοι οι κόμβοι έχουν ίσους ρόλους και η συμπεριφορά των τελευταίων στηρίζεται στη γνώση που έχουν για μια μικρή περιοχή γύρω από αυτούς. Η σύνδεση μεταξύ τους και η τοπολογία γενικότερα του δικτύου δεν λαμβάνεται υπόψη για την αντιμετώπιση του προβλήματος.

Σε ένα τέτοιο σύστημα όλα τα αντίγραφα των δεδομένων πρέπει να ενημερώνονται και να υπάρχει συνέπεια, οι κόμβοι έχουν μικρή πιθανότητα να βρίσκονται σε σύνδεση, οι ενημερώσεις δεν είναι τόσο συχνές σε σχέση με τις ερωτήσεις και η απόδοσή του εξαρτάται από το χρόνο που απαιτείται ώστε να γίνουν οι ενημερώσεις, το ποσοστό των αντιγράφων που ενημερώνονται καθώς επίσης και την επιβάρυνση του δικτύου από τα μηνύματα.

2.5.4.1 Αλγόριθμος “push/pull”

Για την ενημέρωση των αντιγράφων προτείνεται ένας υβριδικός αλγόριθμος που ονομάζεται “push/pull”. Μια θεώρηση που γίνεται είναι ότι τα αντίγραφα σε μια μικρή περιοχή του χώρου δεδομένων “συνδέονται” μεταξύ τους και κάθε αντίγραφο γνωρίζει επίσης ένα υποσύνολο όλων των αντιγράφων. Ο αλγόριθμος αποτελείται από δύο φάσεις. Η φάση “push” αρχικοποιείται στον κόμβο που ξεκινά την ενημέρωση των αντιγράφων και την προωθεί σε ένα υποσύνολο από κατάλληλους κόμβους που γνωρίζει, δηλαδή κόμβους που επηρεάζονται από την ενημέρωση αφού κατέχουν το γνήσιο αντίγραφο του δεδομένου. Οι τελευταίοι με τη σειρά τους προωθούν την ενημέρωση σε άλλους κατάλληλους κόμβους που γνωρίζουν και ούτω καθεξής. Η διαδικασία μοιάζει με τον τρόπο αναζήτησης της πλημμύρας η οποία ολοκληρώνεται σε ένα συγκεκριμένο χρονικό όριο (TTL-Time-To-Live).

Η φάση “pull” αρχικοποιείται από έναν κόμβο ο οποίος χρειάζεται να ενημερώσει το αντίγραφό του για κάποιο λόγο. Ο λόγος αυτός μπορεί να είναι η αποσύνδεση του συγκεκριμένου κόμβου από το δίκτυο για ένα χρονικό διάστημα ή το γεγονός ότι ο κόμβος δεν έχει λάβει ενημερώσεις για κάποιο μεγάλο χρονικό διάστημα ή ότι ο κόμβος αυτός έχει δεχτεί μια αίτηση

για “pull” από έναν άλλο κόμβο και δεν είναι σίγουρος αν έχει το πιο πρόσφατο αντίγραφο του δεδομένου. Οι δύο φάσεις είναι διαδοχικές άλλα μπορεί να επικαλύπτονται σε χρόνο.

2.5.5 Συμπεράσματα

Όπως αναφέρθηκε παραπάνω, οι κόμβοι στο σύστημα συνδέονται και αποσυνδέονται με μια πιθανότητα. Έχει αποδειχτεί με πειραματικές μετρήσεις ότι αν είναι μικρός ο αριθμός των κόμβων που βρίσκονται σε σύνδεση σε σχέση με το συνολικό αριθμό των κόμβων του δικτύου, όταν ξεκινά μια ενημέρωση, τότε η πιθανότητα ο κόμβος στον οποίο θα σταλεί ένα μήνυμα να είναι διαθέσιμος, είναι μικρή. Κατά συνέπεια η ενημέρωση δε θα εξαπλωθεί στο δίκτυο. Ένα άλλο συμπέρασμα που προέκυψε είναι ότι αν κάθε κόμβος προωθεί την ενημέρωση σε έναν περιορισμένο αριθμό από κατάλληλους κόμβους, τότε αυτό είναι αρκετό ώστε η ενημέρωση να γίνει σε όλα τα αντίγραφα. Αν ο αριθμός που αναφέρθηκε πριν είναι μεγάλος, τότε θα διαδοθούν πολλά περιττά και διπλά μηνύματα στο δίκτυο, προκαλώντας έτσι συμφόρηση. Επιπλέον ένας άλλος τρόπος ώστε να αποφευχθούν τα περιττά μηνύματα είναι να μειώνεται η πιθανότητα της προώθησης της ενημέρωσης από κάποιον κόμβο όταν το μήνυμα από τον αρχικό κόμβο είναι αρκετά μακριά από αυτόν. Αυτό προτείνεται γιατί όταν η ενημέρωση έχει εκτελέσει αρκετά βήματα τότε έχουν μείνει λίγοι κόμβοι ώστε να ενημερωθούν και αν συνεχίσουν οι κόμβοι να στέλνουν τον ίδιο αριθμό μηνυμάτων, τότε κάποια θα επαναλαμβάνονται, θα είναι περιττά.

ΚΕΦΑΛΑΙΟ 3

Δομημένα Συστήματα P2P

3.1 Εισαγωγή – Distrubuted Hash Table (DHT)

Σε αυτό το κεφαλαίο θα μελετήσουμε τα δομημένα συστήματα p2p , δηλαδή συστήματα που βασίζονται σε τοπολογία και προσπαθούν να πετύχουν καλυτέρα αποτελέσματα δρομολόγησης . Το σημαντικό πλεονέκτημα των δομημένων συστημάτων είναι οι αλγόριθμοι δρομολόγησης . Το σημαντικό πλεονέκτημα των δομημένων συστημάτων είναι οι αλγόριθμοι δρομολόγησης που περιλαμβάνουν . Έτσι μειώνεται ο αριθμός των βημάτων που απαιτείται για να δρομολογηθεί ένα μήνυμα από τον ένα κόμβο του δικτυού στον άλλο . Τα συστήματα που μελετάμε είναι το Cord , το Taperstry , το Pastry , το Can , το Kademlia και το Viceroy που είναι και τα πιο δημοφιλή στην κατηγορία αυτή .

Τα δομημένα συστήματα ταξινομούνται βάση των αλγορίθμων δρομολόγησης τους σε δυο βασικές κατηγορίες . Υπάρχουν λοιπόν οι ντετερμινιστικοί (deterministic) και οι πιθανοτικοί (randomized) αλγόριθμοι . Ένας ντετερμινιστικός αλγόριθμος είναι ένας αλγόριθμος που συμπεριφέρεται προβλέψιμα . Αν τρέξει σε μια συγκεκριμένη είσοδο , θα παράγει συνεχώς το ίδιο αποτέλεσμα , και η μηχανή θα περνά πάντα από την ίδια ακολουθία καταστάσεων . Ένα απλό μοντέλο για τους ντετερμινιστικούς αλγόριθμους είναι η μαθηματική συνάρτηση . Έτσι όπως μια συνάρτηση παράγει πάντα το ίδιο αποτέλεσμα δεδομένης μιας εισόδου , ακριβώς το ίδιο κάνει και ένας τέτοιος αλγόριθμος . Η διαφορά είναι ότι οι αλγόριθμοι περιγράφουν ακριβώς το ίδιο κάνει και ένας τέτοιος αλγόριθμος . Η διαφορά είναι ότι οι αλγόριθμοι περιγράφουν ακριβώς την ίδια διαδικασία που παίρνεται η έξοδος του

συστήματος από την είσοδο . Ένας πιθανοτικός αλγόριθμος είναι ένας αλγόριθμος είναι ένας αλγόριθμος που του επιτρέπεται να περιστρέψει ένα τυχαίο νόμισμα . Δηλαδή σε κάθε βήμα του αλγορίθμου , το επόμενο βήμα οδηγεί σε διάφορές πιθανές καταστάσεις . Για κάθε επόμενη κατάσταση ορίζεται και μια πιθανότητα .

Ντετερμινιστικοί (deterministic)	Πιθανοτικοί (randomized)
Pastry Tapestry Chord Can Αλγοριθμοι βασισμενοι στα γραφηματα de Bruijin	Viceroy Symphony Randomized-hypercubew Randomized-Chord Συνδυασμός της κατασκευής του Kleinberg με δίκτυα πεταλούδας

Οι αλγόριθμοι που βασίζονται σε DHT είναι βασικά εργαλεία του διαμοιρασμού αρχείων και δεδομένων σε ένα peer to peer σύστημα . Στο κεφάλαιο αυτό παρουσιάζουμε τα πιο γνωστά δομημένα συστήματα εκ των οποίων όλα βασίζονται σε DHT . Ας αναλύσουμε λίγο τον τρόπο με τον οποίο λειτουργούνε αυτά τα συστήματα . Η hash συνάρτηση παίρνει μια συμβολοσειρά από bytes μεταβλητού μήκους και επιστρέφει έναν αριθμό που αντιστοιχεί σε αυτήν την συμβολοσειρά . Οι αλγόριθμοι DHT εφαρμόζουν την hash συνάρτηση σε όλα τα αναγνωριστικά αρχείων/δεδομένων (συνήθως ονομάζονται κλειδιά) και αποθηκεύει τις θέσεις τους στο δίκτυο σε ένα τεράστιο πίνακα hash , ο οποίος είναι κατανεμημένος στους κόμβους του δικτύου . Δηλαδή ένα DHT σύστημα αντιστοιχεί κλειδιά σε κόμβους σε μια δομή peer-to-peer. Για ένα δεδομένο κλειδί οποιοσδήποτε κόμβος του δικτύου μπορεί να χρησιμοποιήσει το DHT για να προσδιορίσει τον κόμβο που έχει το κλειδί .Την λειτουργία των DHT θα τη μελετήσουμε καλυτέρα μέσω των συστημάτων που την υλοποιούν .

3.2 Chord

Το Chord είναι ένα δομημένο αποκεντρωμένης τοπολογίας δίκτυο με δυνατότητα εύκολης μεγένθυσης . Το Chord αντιστοιχεί σε κάθε κόμβο και κάθε δεδομένο ένα κλειδί . Δεδομένα μπορεί να είναι μια διεύθυνση , ένα έγγραφο , ένα αρχείο η οποιαδήποτε συλλογή στοιχειών . Ο εντοπισμός των δεδομένων υλοποιείται σχετίζοντας ένα κλειδί με δεδομένα και αποθηκεύοντας το ζεύγος δεδομένα/κλειδί στο κόμβο που αντιστοιχεί στο κλειδί (π.χ. σε ένα αρχείο , κλειδί μπορεί να είναι το όνομα του και δεδομένο το αρχείο) . Το Chord λύνει ικανοποιητικά τα παρακάτω προβλήματα των peer to peer συστημάτων :

- Ισοκατανομή του φορτιού . Μοιράζει ισοδύναμα τα κλειδιά/δεδομένα στους κόμβους του συστήματος .
- Αποκέντρωση . Όλοι οι κομβόι είναι ισοδύναμοι και δεν απαιτείται κεντρικός server .
- Επεκτασιμότητα . Το κόστος του ψαξίματος στο Chord αυξάνει αναλογικά με τον λογάριθμο του πλήθους των κόμβων και συνεπώς εξυπηρετεί καλά πολύ μεγάλα συστήματα .
- Χρηστικότητα . Αντιμετωπίζει καλά τις συχνές και εκτεταμένες εισόδους και εξόδους κόμβων από το σύστημα .
- Ευέλικτη ονοματολογία . Δημιουργεί ένα μεγάλο και ισοκατανεμημένο πεδίο ορισμού των κλειδιών και έτσι δεν δημιουργεί δεσμεύσεις στην ονοματολογία των δεδομένων .

Οι εφαρμογές επικοινωνούν με το Chord με 2 κυρίως τρόπους . Πρώτα το Chord εξασφαλίζει ένα αλγόριθμο που βρίσκει την IP διεύθυνση του κόμβου που αντιστοιχεί σε ένα κλειδί δηλαδή δίνει την δυνατότητα σε μια

εφαρμογή να βρει το κόμβο που φυλάσσονται κάποια δεδομένα που θέλει μια εφαρμογή , και δεύτερον ενημερώνει κάθε κόμβο για τις αλλαγές της ομάδας των κλειδιών για τα οποία είναι υπεύθυνος , έτσι ώστε να μεταφέρει η εφαρμογή και τα αντίστοιχα δεδομένα στους νέους κόμβους , η να πάρει από τους καταργούμενος κόμβους .

Οι εφαρμογές είναι υπεύθυνες για την χρήση των δυνατοτήτων του Chord για επαληθευσιμότητα , αποθήκευση , αντιγραφή κλπ. Των δεδομένων .

Οι παρακάτω κατηγορίες προγραμμάτων μπορούν να χρησιμοποιήσουν ικανοποιητικά το Chord : Συνεργατικής διανομής αρχείων (Cooperative Mirroring) , Αποθήκευσης διαμοιρασμένου χρόνου (Time – Shared Storage) , διαμοίρασης αρχείων με δείκτες (Distributed Indexes) , Μεγάλης κλίμακας συνδυαστικό ψάξιμο (Large – Scale Combinatorial Search) .

3.2.1 Το πρωτόκολλο του Chord

3.2.2 Πρωτόκολλο αποθήκευσης

Το πρωτόκολλο του Chord υποστηρίζει μια μόνο λειτουργία , δεδομένου ενός κλειδιού , αντιστοιχεί το κλειδί σε έναν κόμβο . Η εφαρμογή που χρησιμοποιεί το Chord μπορεί να αξιοποιήσει την λειτουργία αυτή για να αποθήκευση κόμβο αυτόν δεδομένα που αντιστοιχούν στο κλειδί .

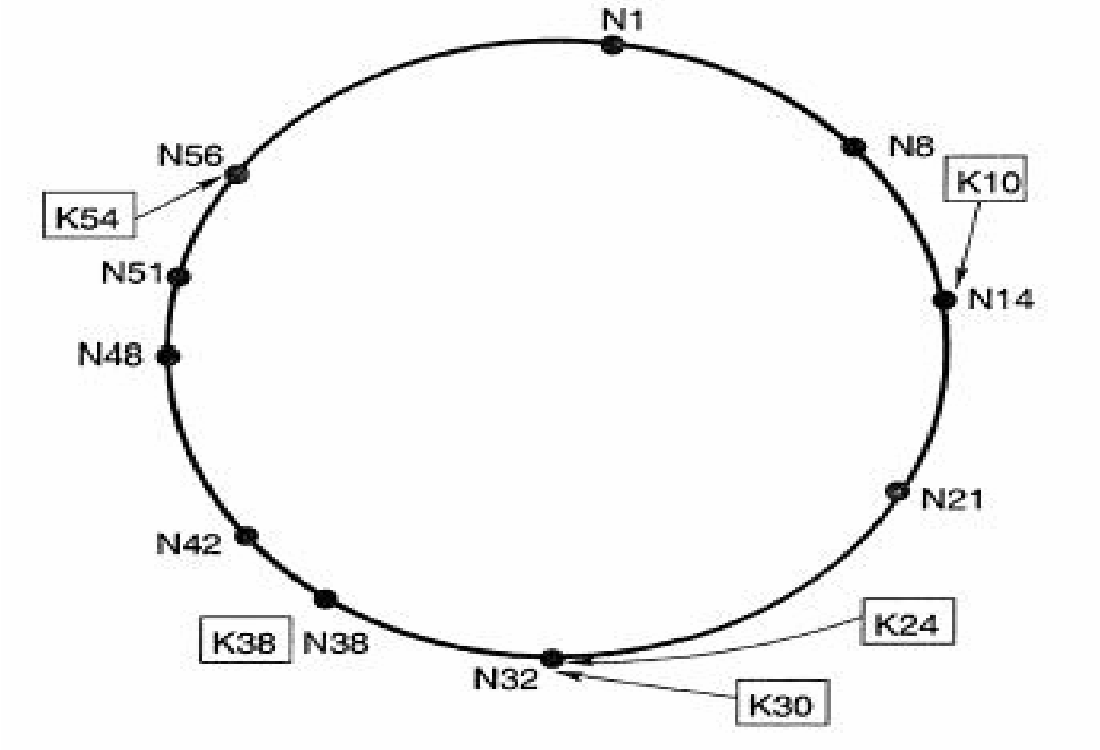
Το Chord χρησιμοποιεί μια σταθερή Hash συνάρτηση , την SHA-1(Secure Hashing Algorithm), για να αντιστοιχίσει κάθε διεύθυνση κόμβου (IP address) και κάθε κλειδί σε ένα m-bit αναγνωριστικό. Το m είναι αρκετό

μεγάλο ώστε να αποτρέπει την πιθανότητα 2 διαφορετικά κλειδιά η κομβόι να έχουν το ίδιο αναγνωριστικό . Έτσι από το IP ενός κόμβου μέσω της SHA-1 προκύπτει το αναγνωριστικό του κόμβου και από το string του κλειδιού (που αντιστοιχεί όπως είπαμε σε ένα πραγματικό δεδομένο η αρχείο) το αναγνωριστικό του κλειδιού .

Το δίκτυο οργανώνεται σε ένα ταξινομημένο κύκλο των αναγνωριστικών (IDs) κόμβων modulo 2^m δηλαδή όταν τα αναγνωριστικά έχουν μέγεθος m bits το δίκτυο έχει μέγεθος 2^m (από 0 εως 2^m -1) .

Ο Βασικός κανόνας είναι η αποθήκευση κάθε κλειδιού στον διάδοχο του κόμβο (successor) , δηλαδή στον ενεργό κάθε στιγμή κόμβο που το αναγνωριστικό του είναι μεγαλύτερο ή ίσο από το αναγνωριστικό του κλειδιού . (π.χ. στον κόμβο με ID 20 αποθηκεύονται τα κλειδιά με αναγνωριστικό 9,12,20). Αν τα αναγνωριστικά των κόμβων παριστάνονται σε ταξινομημένο κύκλο από το 0 έως 2^m - 1 , ο successor (k) δηλαδή ο διάδοχος κόμβος του κλειδιού με ID k , είναι ο πρώτος ενεργός κόμβος που συναντάμε από το k κατά την φορά των δεικτών του ρολογιού .

Παράδειγμα αποθήκευσης 5 κλειδιών σε 10 ενεργούς κόμβους με m=6 δίνει η παρακάτω εικόνα 9.



Εικόνα 9

Όταν μπαίνει ένας καινούργιος κόμβος n στο σύστημα κάποια από τα κλειδιά που αντιστοιχούν στον successor(n) πηγαίνουν στον n ενώ όταν φεύγει τα κλειδιά του πηγαίνουν στον successor(n) . Στο σχήμα μας π.χ. αν μπει στο σύστημα ο κόμβος ID 26 θα πάρει το κλειδί με ID 24 από τον κόμβο με ID 32 .

Αποδεικνύεται το παρακάτω θεώρημα :

Σε ένα δίκτυο με N κόμβους και K κλειδιά με πολύ μεγάλη πιθανότητα :

1. Κάθε κόμβος αποθηκεύει το πολύ (1+ε)K/N κλειδιά .
2. Όταν ο (N+1)-στος κόμβος συνδέεται η εγκαταλείπει το δίκτυο O(K/N) κλειδιά μετατοπίζονται από , σε άλλους κόμβους (και μόνο σε η από τον κόμβο αυτό).

Το ε έχει ανώτατο όριο O (logN) και μπορεί να μειωθεί σε πολύ μικρή τιμή αν κάθε κόμβος δημιουργήσει O (logN) φανταστικούς κόμβους με δικό τους ID.

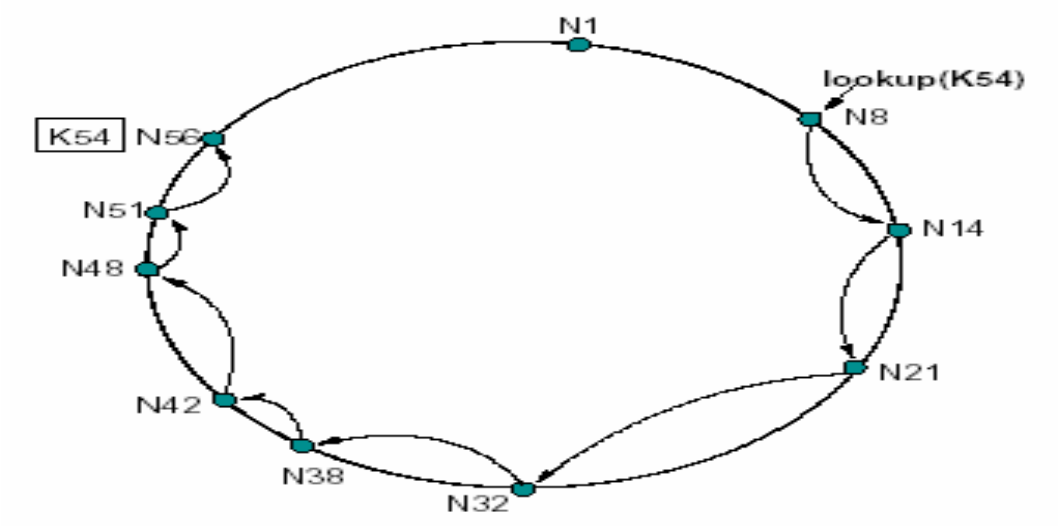
3.2.3 Αλγόριθμοι εύρεσης – Δρομολόγησης

Το ψάξιμο ενός κλειδιού στο δίκτυο μπορεί να γίνει με διαφόρους τρόπους .

Ο πιο απλός αλλά αργός τρόπος είναι το ψάξιμο κατά την φορά του ρολογιού από τον ένα κόμβο στον διάδοχο του , μέχρι να βρεθεί ο κόμβος που περιέχει το κλειδί (διατρέχοντας συνεπώς όλους τους ενδιάμεσους ενεργούς κόμβους). Στην περίπτωση αυτή η μονή πληροφορία που πρέπει να έχει ένας κόμβος είναι , ποιος είναι ο διάδοχος του . Ο ψευδοκώδικας του αλγορίθμου είναι ο ακόλουθος :

```
n.find_successor(id)
  if(id ϵ (n,n.successor))
        return n.successor;
 else
      return syccessor.find_syccessor(id);
```

Στην εικόνα 10 παρακάτω δίνεται ένα παράδειγμα εύρεσης του κλειδιού 54 από τον κόμβο 8.



Εικόνα 10

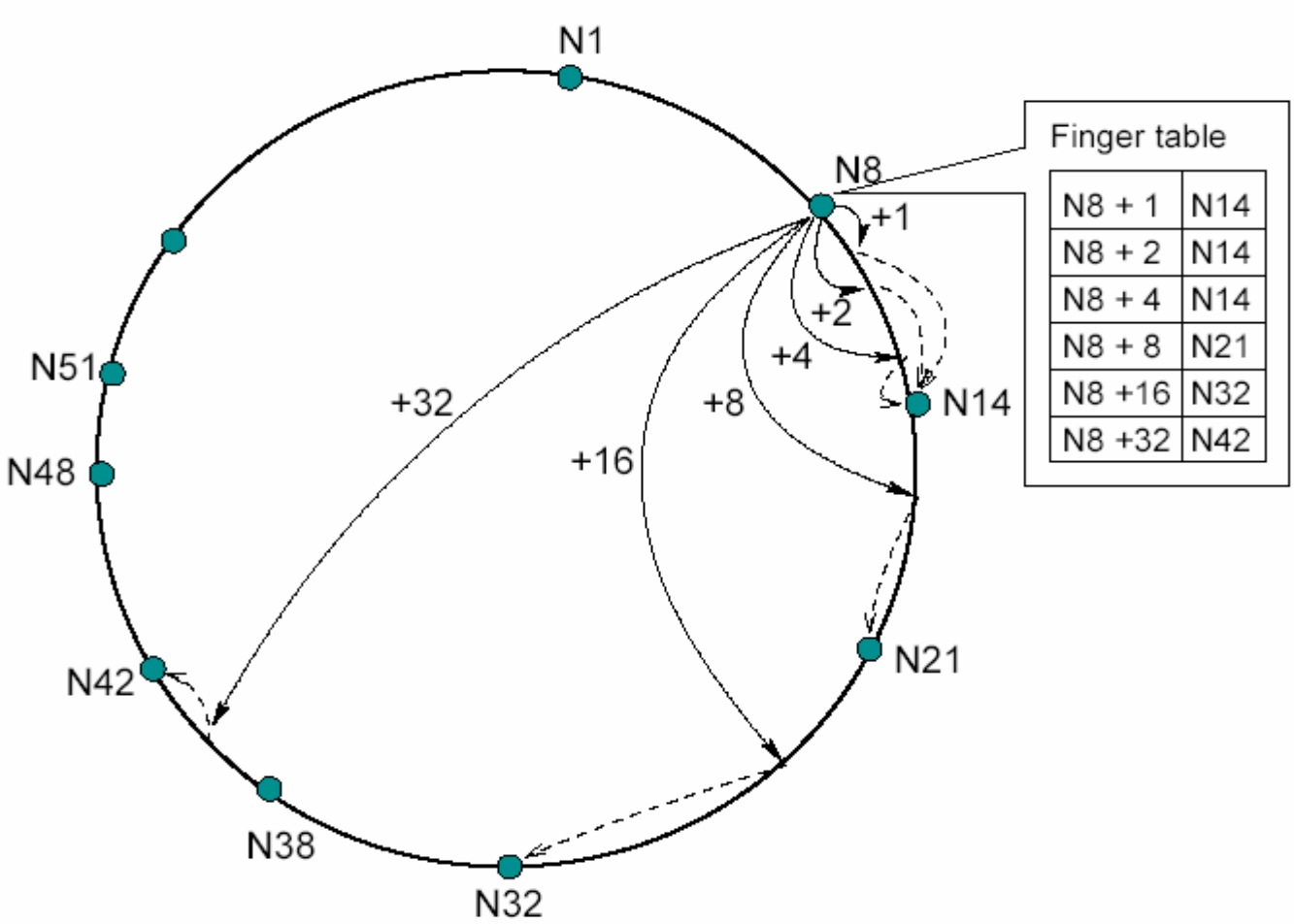
Φαίνεται στην εικόνα μια αναζήτηση του κόμβου 8 (N8) για το κλειδί 54 (K54). Αρχικά, ο N8 εξετάζει αν το K54 βρίσκεται μεταξύ αυτού και του successor του (N14). Δεν το βρίσκει εκεί, άρα η αναζήτηση προχωράει στον successor τού N8, στον N14. Ο N14 εξετάζει αν το K54 βρίσκεται μεταξύ αυτού και του successor του, N21. Δεν το βρίσκει και η αναζήτηση προχωράει στον successor τού N14, στον N21. Η διαδικασία αυτή συνεχίζεται μέχρις ότου βρεθεί ο successor τού κλειδιού K54, που είναι ο κόμβος N56.

Για την επιτάχυνση του ψαξίματος το Chord υποστηρίζει έναν άλλο αλγόριθμο ο οποίος απαιτεί κάθε κόμβος να διατηρεί πρόσθετες πληροφορίες δρομολόγησης , σε έναν πίνακα ο οποίος λέγεται πίνακας δεικτών (finger table). Κάθε κόμβος η διατηρεί ένα πίνακα m δεικτών των διαδόχων του κόμβου με τρόπο ώστε στην i θέση του πίνακα αντιστοιχεί ο διάδοχος s του n + 2i κόμβου (δηλαδή ο πρώτος ενεργός κόμβος σε απόσταση τουλάχιστον 2^{i-1}) ή s= successor $(n+2^{i-1})$,όπου 1<= i <=m . Ονομάζουμε τον κόμβο s τον i οστό δείκτη του κόμβου n και τον δηλώνουμε ως n.finger[i] (ανάλυση του ορού αυτού και των ορών successor και predecessor δες στον παρακάτω πίνακα)

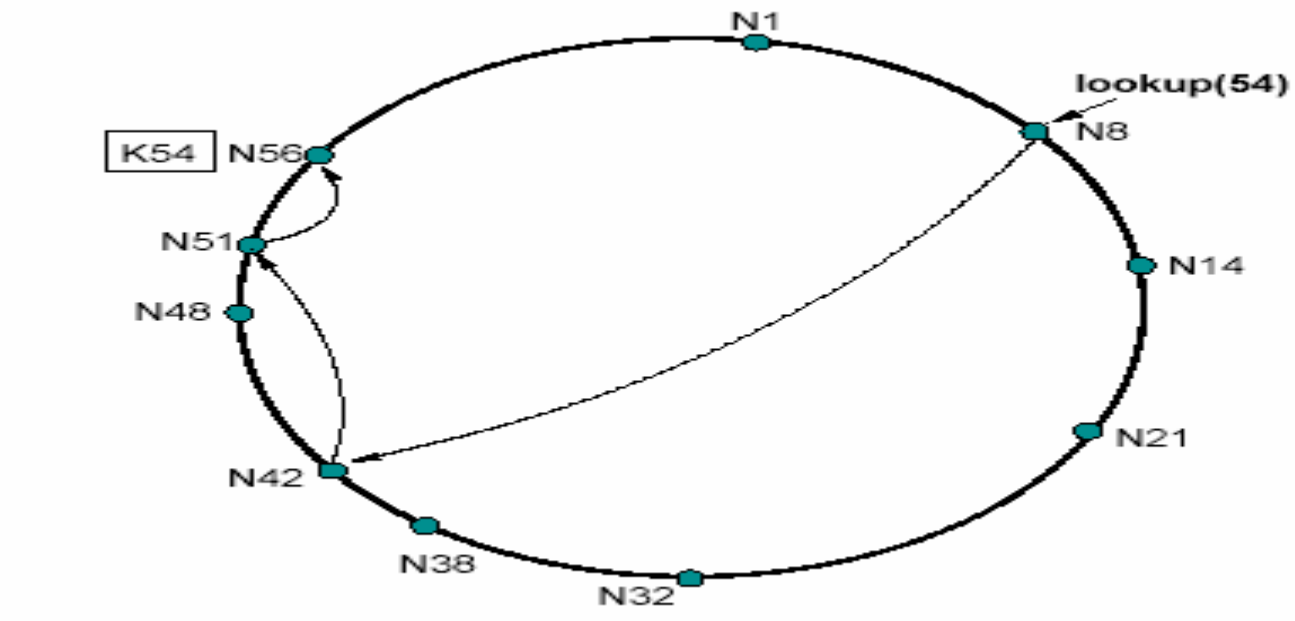
Σημειογραφία	Ορισμός
Finger[k]	Ο πρώτος κόμβος στον κύκλο που διαδέχεται $(n+2^{k-1}) \bmod 2^m$,$1 \leq k \geq m$
successor	Ο επόμενος κόμβος στον κύκλο . finger[1].node
prodecessor	Ο προηγούμενος κόμβος στον κύκλο

Σημειώνουμε ότι ο $1^{ος}$ finger είναι ο successor . Επίσης ότι ο πίνακας αυτός περιέχει και το ID κάθε κόμβου και την IP διεύθυνση του έτσι ώστε να είναι δυνατή επικοινωνία με τον κόμβο αυτό .

Η λογική του τρόπου αυτού , είναι να κατευθύνεται η αναζήτηση όχι στον αμέσως επόμενο ενεργό κόμβο , όπως στον προηγούμενο απλό τρόπο , αλλά στον πλησιέστερα του ζητούμενο κόμβο για τον οποίο διατίθεται πληροφορία . Για όσους περισσότερους κόμβους υπάρχει πληροφορία τόσο μεγαλύτερη είναι η ταχύτητα της εύρεσης . Η συνάρτηση που προσδιορίζει τους κόμβους που περιέχονται στον πίνακα δεικτών (δηλαδή η s= successor $(n+2^{i-1})$) , είναι ικανοποιητική γιατί πρώτον επιταχύνει την αναζήτηση περιέχοντας πληροφορία για μικρό αριθμό προσθέτων κόμβων και δεύτερον γιατί κατανέμει την πληροφορία πιο πυκνά κοντά στον κόμβο που ψάχνει και πιο αραιά μακριά του .



Εικόνα 11



Εικόνα 12

Στις ανωτέρω εικόνες 11 και 12 φαίνεται ο πίνακας δεικτών του κόμβου με ID 8 και ο τρόπος με τον οποίο ο κόμβος αυτός βρίσκει το κλειδί 54 (δηλαδή κάθε φορά , ένας κόμβος ψάχνει στον πίνακα των δεικτών του , τον κόμβο που το αναγνωριστικό του κλειδιού . Όταν το κλειδί βρίσκεται ανάμεσα στο ID του κόμβου και του successor του το κλειδί βρίσκεται στον successor) .

Ο ψευδοκώδικας του αλγορίθμου αυτού είναι ο ακόλουθος :

```
// ρωτά τον κόμβο η να βρει τον διάδοχο του id.
n.find_successor(id)
  if (key ϵ (n,n.successor)
     return n.successor;
else

    n'=closest_preceding_node(id);

return n'.find_successor(id);
//βρες το τοπικό πίνακα για τον προηγούμενο του id

n.closest_preceding_node(id)

 for i=m downto 1

   if(finger[i] ϵ (n,id))

        return finger[i];

return n;
```

Αποδεικνύεται το θεώρημα : Με πολύ μεγάλη πιθανότητα ο αριθμός των κόμβων που πρέπει να ερευνηθούν για να βρεθεί ένα κλειδί σε ένα δίκτυο N κόμβων είναι O(logN). Ακολουθεί η περιγραφή της απόδειξης . Έστω ότι ο κόμβος n στέλνει ένα ερώτημα για τον διάδοχο του k . Έστω p ο κόμβος που προηγείται του κ. Το ερώτημα για τον k τελικά φτάνει στον p

που επιστρέφει τον επόμενο διάδοχο του σε αυτόν που κάνει το ερώτημα . Ακολουθεί η ανάλυση των βημάτων για να φτάσει το ερώτημα στον p. Ο ν στέλνει το ερώτημα του στον κοντινότερο πρόγονο του k που βρίσκει μέσω του πίνακα δεικτών του (finger table). Έστω ότι ο p είναι στο i-οστό διάστημα του κόμβου n . Εφόσον το διάστημα δεν είναι άδειο , ο n θα δείξει σε κάποιον κόμβο f στο διάστημα αυτό . Η απόσταση μεταξύ n και f είναι τουλάχιστον 2^{i-1} . Οι f , p ανήκουν στο i-οστό διάστημα του κόμβου η άρα η απόσταση μεταξύ αυτών είναι το πολύ 2^{i-1} .Οπότε ο f είναι πιο κοντά στον p παρά στον n ή ισοδύναμα η απόσταση από τον f στον p είναι το πολύ η μισή απόσταση από τον n στον p . Αν η απόσταση μεταξύ του κόμβου που χειρίζεται το ερώτημα και του p μειώνεται στο μισό σε κάθε βήμα και αρχικά είναι το πολύ 2m , τότε μέσα σε m βήματα θα έχει γίνει 1 , δηλαδή θα έχουμε φτάσει στο p . Ο αριθμός των βημάτων βασικά είναι O(logN) με μεγάλη πιθανότητα . Μετά από logN βήματα η απόσταση μεταξύ του τρέχοντος κόμβου και του κλειδιού θα μειωθεί στο 2m/N , το οποίο με μεγάλη πιθανότητα είναι περίπου 1 .

3.2.4 Δυναμική συμπεριφορά του Chord

3.2.4.1 Είσοδος κόμβων και σταθεροποίηση του δικτυού

Το Chord υποστηρίζει την είσοδο και την έξοδο κόμβων από το σύστημα με μεγάλη συχνότητα , χωρίς κατάρρευση του συστήματος . Για να το πετύχει αυτό χρησιμοποιεί ένα πρωτόκολλο για την σταθερότητα του συστήματος με κύριο μέλημα την γνώση κάθε κόμβου για το ποιος είναι ο successor του . Επίσης περιοδικά ανανεώνει τους πίνακες δεικτών των κόμβων του , χωρίς όμως αυτό να είναι κρίσιμο για την ευστάθεια του συστήματος . Εάν γίνει η είσοδος κάποιων κόμβων , και ταυτόχρονα πριν ολοκληρωθεί η σταθεροποίηση του συστήματος γίνει κάποια αναζήτηση

υπάρχουν 3 περιπτώσεις . Η συνηθέστερη είναι οι πίνακες δεικτών της περιοχής να είναι σωστοί , οπότε η αναζήτηση ολοκληρώνεται σωστά σε O(logN) βήματα . Η δεύτερη είναι οι πίνακες δεικτών να είναι ανεπίκαιροι , αλλά οι successor να είναι σωστοί . Τότε η αναζήτηση ολοκληρώνεται σωστά αλλά μπορεί να είναι αργότερη . Η τρίτη περίπτωση είναι να μην είναι ούτε οι successor σωστοί οπότε η αναζήτηση μπορεί να αποτύχει . Στην περίπτωση αυτή το πρόγραμμα μπορεί να επαναλάβει την αναζήτηση μετά ένα μικρό χρόνο (γιατί ο χρόνος της διόρθωσης από το Chord είναι μικρός).

Παρακάτω φαίνεται ο ψευδοκώδικας εισαγωγής και σταθεροποίησης όταν εισάγεται ένας κόμβος n καλεί από κάποιον τυχαίο κόμβο n' την συνάρτηση n'.join(n). Η συνάρτηση αυτή βρίσκει τον s= successor(n) , δημιουργεί τον πίνακα δεικτών του s , ενημερώνει τον s ότι ο predecessor του είναι ο n και τον n ότι ο successor του είναι ο s . Σε σύντομα περιοδικά διαστήματα καλείται από κάθε κόμβο και η συνάρτηση stabilize που ελέγχει (και διορθώνει αν χρειάζεται) τον αν ο successor ενός κόμβου θεωρεί ως predecessor τον κόμβο .

```
n.build_fingers (n')
i0  := [log(successor-n)]+1;
for each i≥ i0 index into finger[];

n.join (n')
  predecessor  =  nil;
  s = n'.find_successor (n);
  build_fingers (s);
  successor  = s ;

n.stabilize ()
```

```
x = successor.predecessor;
if( x ϵ (n,successor))
successor  = x; successor.notify(n);

n.notify (n');
if   (predecessor is nil or n' ϵ (predecessor(n)0
predecessor = n';
```

Αποδεικνύεται ότι μετά την είσοδο κάποιων κόμβων , μετά κάποιο χρονικό διάστημα σχηματίζεται ένας κύκλος όπου κάθε κόμβος ξέρει τον επόμενο του από όλους του κόμβους του συστήματος .

3.2.4.2 Επίδραση της εισαγωγής κόμβων στην συμπεριφορά αναζήτησης

Αποδεικνύεται επίσης ότι η μη ανανέωση των πινάκων δεικτών δεν επηρεάζει πολύ την ταχύτητα του ψαξίματος , διότι όπως και ανωτέρω αναφέραμε και προηγουμένως δεν έχει και τόσο μεγάλη σημασία ποιοι ακριβώς κόμβοι περιέχονται στους πίνακες δεικτών , αλλά η ικανότητα τους να μεταφέρουν την ερώτηση μακριά από τον ερωτώντα κόμβο . Έτσι η ταχύτητα αυτή παραμένει και πάλι με μεγάλη πιθανότητα O(logN) .

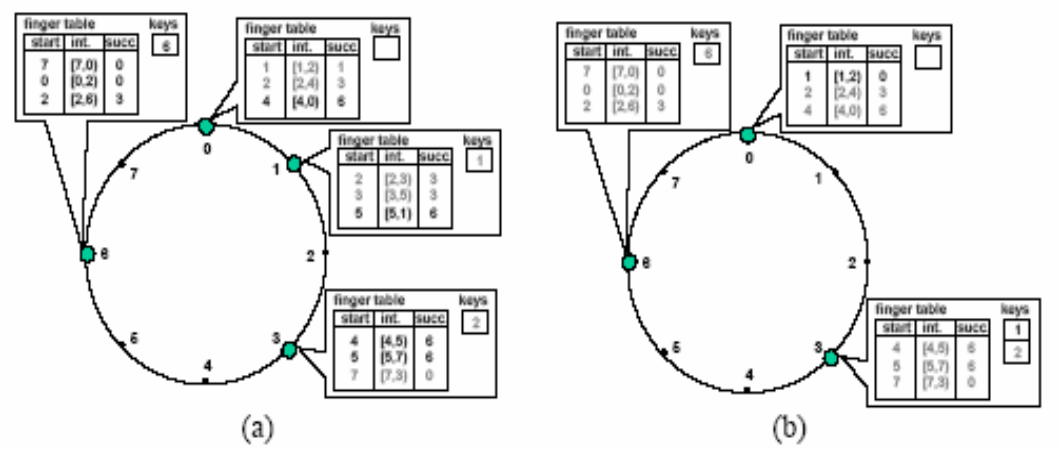
3.2.4.3 Αποχώρηση κόμβων

Η αποχώρηση των κόμβων από το σύστημα μπορεί να γίνει αθέλητα η ηθελημένα . Αν γίνει ηθελημένα ο κόμβος που αποχωρεί μεταφέρει όλα του τα κλειδιά στον successor του και ενημερώνει τον successor και τον predecessor του για την αποχώρηση του .

Για να αντιμετωπιστούν οι αθέλητες αποχωρήσεις που μπορεί να καταρρεύσουν το σύστημα κάθε κόμβος διατηρεί μια λίστα των r successor του . Αν σε μια αναζήτηση ενός successor δεν ανταποκρίνεται , τότε αντικαθίσταται από τον επόμενο του στην λίστα . Μια καλή τιμή του r είναι η 2logN .Κατάρρευση μπορεί να γίνει μόνο αν καταρρεύσουν ταυτόχρονα και οι r κόμβοι .

Αποδεικνύεται το θεώρημα : Αν χρησιμοποιούμε μια λίστα successor μήκους r = O(logN) σε ένα αρχικά ευσταθές σύστημα και κάθε κόμβος απομακρύνεται με πιθανότητα ½ , τότε με μεγάλη πιθανότητα η αναζήτηση του successor ενός κλειδιού βρίσκει τον πλησιέστερο ενεργό κόμβο του κλειδιού . Επίσης τότε ο χρόνος της αναζήτησης είναι O(logN) .

Όταν ένας κόμβος n αποχωρεί εθελοντικά, τότε ενημερώνει τον successor του, ότι τώρα, ο καινούριος predecessor του θα είναι ο predecessor του n (μεταφέρει στον successor και όλα τα κλειδιά που του αντιστοιχούν) και αντίστοιχα, ενημερώνει τον predecessor του ότι ο καινούριος successor του είναι ο παλιός successor του n. Επίσης, ενημερώνονται, αντίστοιχα, για τις αλλαγές αυτές και όλα τα finger tables τα οποία περιείχαν τον n.

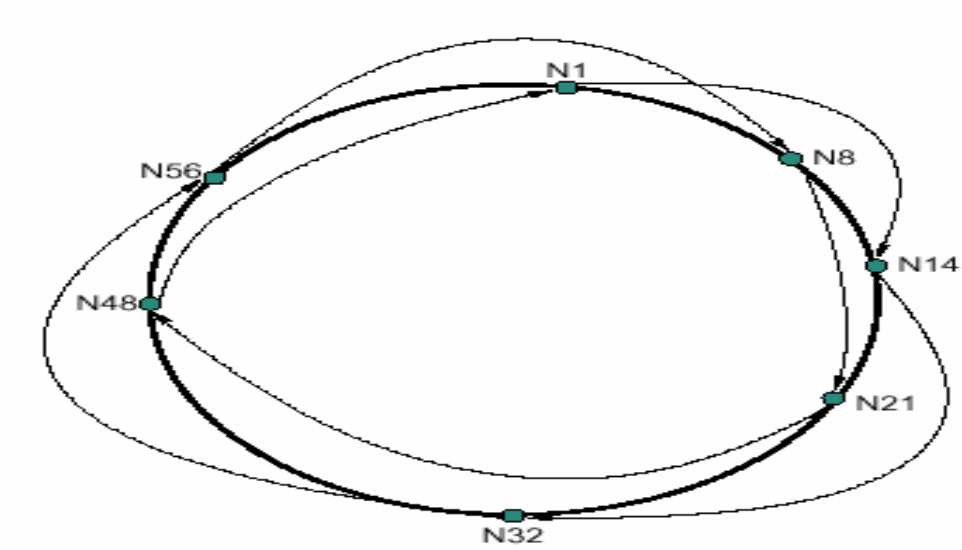


Εικόνα 13 : a) Παράδειγμα εισαγωγής κόμβου. Ο κόμβος 6 εισάγεται στο σύστημα, b) Παράδειγμα αποχώρησης κόμβου. Ο κόμβος 3 αποχωρεί από το σύστημα. Οι αλλαγμένες καταχωρήσεις φαίνονται με μαύρο χρώμα και οι υπόλοιπες με γκρι.

Στην περίπτωση που ένας κόμβος n αποτύχει, πρέπει να βρεθεί ο successor του n προκειμένου να ενημερωθούν τα finger tables στα οποία εμπεριέχεται ο n [2]. Στην εικόνα 6b) φαίνεται ένα παράδειγμα αποχώρησης κόμβου (του κόμβου 3).

3.2.5 Αλγόριθμος Strong Stabilization

Για να αποκτήσει μεγαλύτερη σταθερότητα το σύστημα και να αποφευχθούν φαινόμενα αστάθειας όπως π. χ. όταν το δίκτυο διαμοιράζεται σε 2 υποδίκτυα τότε ακολουθείται ο αλγόριθμος Strong Stabilization .



Εικόνα 14

Λέμε ότι ένα δίκτυο Chord είναι weakly stable αν για όλους τους κόμβους u ισχύει predecessor (successor (u)) = u , strongly stable αν επιπλέον για κάθε κόμβο u δεν υπάρχει κόμβος u ώστε u <u< successor (u) . Αν το δίκτυο όπως στην ανωτέρω εικόνα 14 είναι weakly stable αλλά δεν είναι strongly stable λέγεται loopy stable (π. χ. στην εικόνα 14 για τον κόμβο N14 υπάρχει ο κόμβος N21 όπου 14<21< successor (14) = 32 , άρα είναι loopy stable) .

Ο αλγόριθμος Strong Stabilization αντικαθιστά τον ψευτοκώδικα stabilize που αναφέρθηκε ανωτέρω και είναι ο ακόλουθος .

```
n.join(n' )
  on_cycle = false;
  predecessor = nil;
  s = n' .find_successor(n);
  while (not  s.on_cycle) do
      s := s.find_successor(n');
  successor[0] := s;
successor[1]= s;


n.update_and_notify (i)
    s = successor[i]
    x = s.predecessor;
    if (x ϵ (n, s))
      successor[i] = x;
s.notify(n);
```

```
n.stabilize()
   u = successor[0].find_successor(n);
   on_cycle = (u = n);
   if (successor[0] = successor[1]
                and u ϵ (n, successor[1]))
   successor[1] = u;
 for (i = 0, 1)
  update_and_notify(i);
```

Με τον αλγόριθμο αυτό το δίκτυο που δημιουργείται λέγεται strongly stable και αυτό επιτυγχάνεται σε $O(N^2)$ κύκλους .

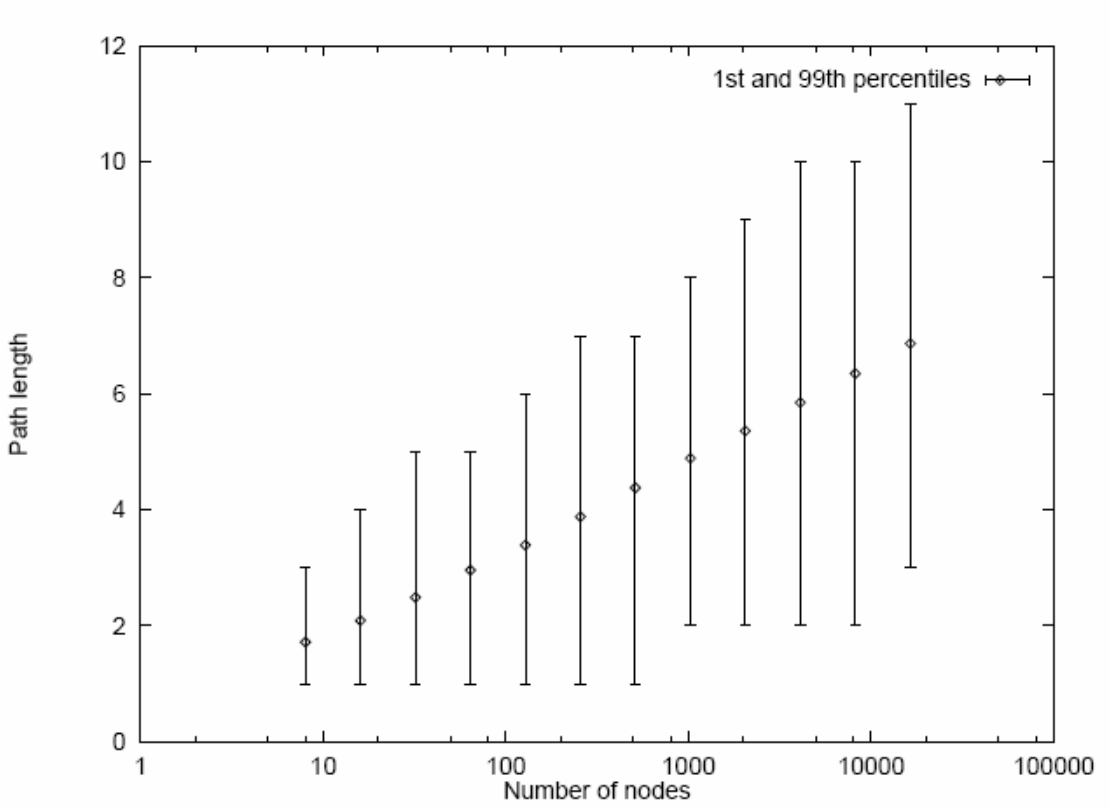
Για να επιταχύνουμε την διαδικασία αυτή κάθε κόμβος κρατάει μια λίστα των ανάστροφων δεικτών (backwards fingers) που περιλαμβάνει τους προς τα πίσω successor του (δηλαδή τους predecessor) όπου ο i δείκτης βρίσκεται σε απόσταση 2^{i-1} και ο αλγόριθμος u.stabilize μετασχηματίζεται ώστε να τους χρησιμοποιεί .

Με τον τρόπο αυτό σε ένα πλήρως χωρισμένο στα 2 δίκτυο ένας κόμβος βρίσκει τον successor του σε $O(\log^2 N)$ βήματα . Επίσης αν ταυτόχρονα όλοι οι κόμβοι κάνουν ένα αυτοψάξιμο και τρέξουν τον ανωτέρω αλγόριθμο σε ένα τέτοιο δίκτυο θα δημιουργηθεί ένα strongly stable δίκτυο σε $O(\log^2 N)$ βήματα . Επίσης αν ταυτόχρονα όλοι οι κομβόι κάνουν ένα αυτοψάξιμο και τρέξουν τον ανωτέρω αλγόριθμο σε ένα τέτοιο δίκτυο θα δημιουργηθεί ένα strongly stable δίκτυο σε $O(\log N)$ βήματα .

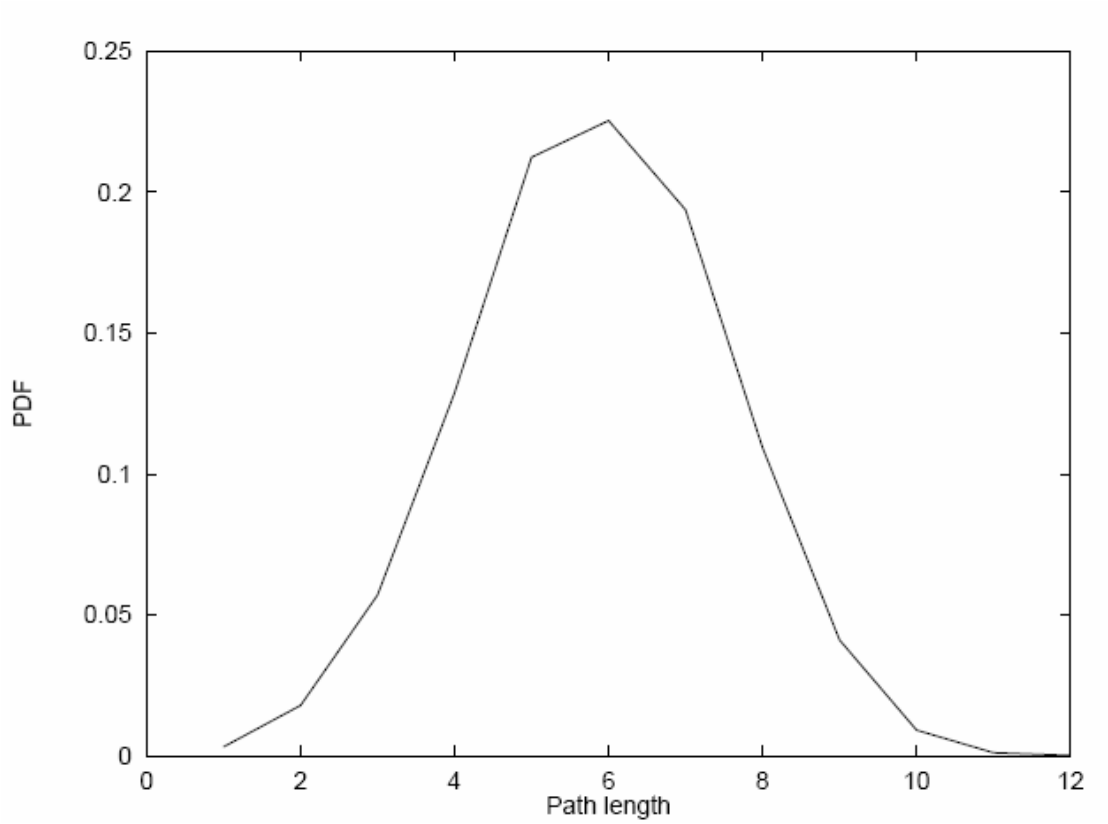
3.2.6 Προσομοίωση και πειραματικά αποτελέσματα

Αποτελέσματα προσομοίωσης με μεγάλο πλήθος κόμβων (100.000) έδειξαν την συμφωνία των θεωρητικών με τα πειραματικά αποτελέσματα.

Οι παρακάτω εικόνες δείχνουν την πλήθος βημάτων σε τυχαίες δρομολογήσεις , ανάλογα με το πλήθος των κόμβων του δικτύου και την PDF (probability density function – συνάρτηση πυκνότητας πιθανότητας) του μήκους του μονοπατιού σε δίκτυο με 2^{12} κόμβους . Η μέση τιμή των βημάτων φαίνεται ότι είναι $½*\log_2 N$.

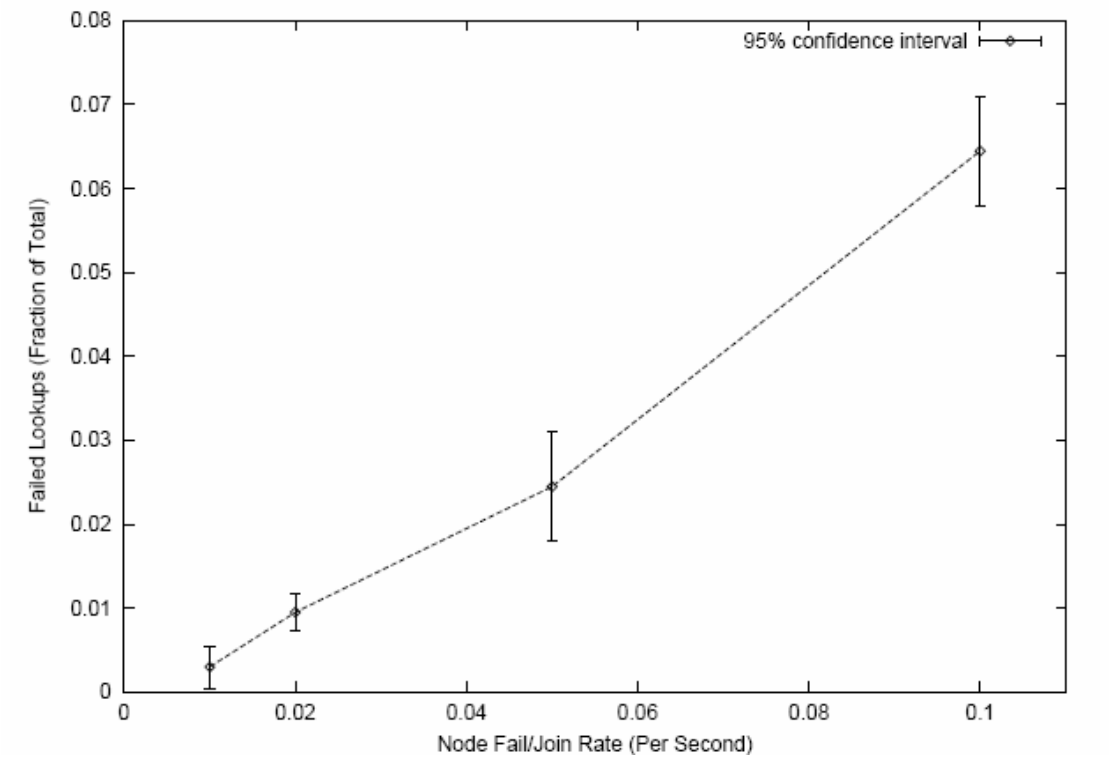


Εικόνα 15



Εικόνα 16

Άλλα πειράματα δείχνουν την καλή συμπεριφορά του Chord στην ταυτόχρονη έξοδο κόμβων από το δίκτυο , όπως φαίνεται και στις παρακάτω εικόνες .



Εικόνα 17

Στην παραπάνω εικόνα σε ένα δίκτυο 500 κόμβων όπου αποχωρεί και εισέρχεται ένας κόμβος με ρυθμό από 0,01 έως 0,1 ανά δευτερόλεπτο , ενώ η σταθεροποίηση (stabilization) γίνεται κάθε 30 sec. Αποδεικνύεται π.χ. ένα ποσοστό αποτυχιών 3% στον ρυθμό εξόδου κόμβων 0,1 δηλ. 3 αποχωρήσεων ανάμεσα σε κάθε σταθεροποίηση το ποσοστό των αποτυχιών στο ψάξιμο μετά την αποχώρηση και την σταθεροποίηση του δικτύου σε σχέση με το ποσοστό των κόμβων που αποχωρούν .

Συμπερασματικά το Chord προσφέρει ένα πλήρως αποκεντρωμένο πρωτόκολλο με εύκολη αποθήκευση και εύρεση των δεδομένων με δυνατότητα μεγάλης αυξομειώσεις του δικτυού και εύκολης επαναφοράς του σε ευσταθή κατάσταση .

3.2.7 Δίκτυο Koorde εξέλιξη του Chord

Ένα δίκτυο που βασίζεται στα γραφήματα de Bruijn και αποτελεί μια εξέλιξη του Chord είναι το Koorde των M. Frans Kaashoek και David R. Karger . Το Koorde είναι ένα DHT σύστημα που κληρονομεί την απλότητα του Chord . Ο αριθμός των βημάτων για κάθε αίτηση αναζήτησης είναι O(logn) αν κάθε κόμβος έχει 2 γείτονες , όπου n είναι ο αριθμός των κόμβων στο DHT . Αν κάθε κόμβος έχει O(logn) γείτονες τότε ο αριθμός βημάτων είναι o(logn/loglogn) .

3.3 Tapestry

3.3.1 Εισαγωγή

Το Tapestry είναι ένα σύστημα που αναπτύχθηκε από τους Ben Zhao , John Kubiatowicz και Antony Joseph στο Berkley University . Παρέχει μια αρχιτεκτονική δρομολόγησης για μια αυτό-οργανωμένη , επεκτάσιμη και ανθεκτική , μεγάλης κλίμακας υποδομή , που μπορεί να δρομολογήσει επιτυχώς αιτήσεις περιεχομένου σε καταστάσεις μεγάλου φόρτου και αποτυχίας κόμβων του δικτύου .

Το Tapestry παρέχει αποκεντρωμένη τοποθέτηση και δρομολόγηση αντικειμένων (DOLR – Decentralized Object Location and Routing) που εστιάζει στην δρομολόγηση των μηνυμάτων στα τερματικά σημεία (όπως οι κόμβοι η τα αντίγραφα αντικειμένων) .

Το Tapestry μαζί με τα Pastry , Chord και Can ανήκουν στην δεύτερη γενιά peer-to-peer πρωτοκόλλων , που εξασφαλίζουν ένα περιορισμένο πλήθος βημάτων στη δρομολόγηση και έχουν την δυνατότητα μεγάλης επεκτασιμότητας και αυτοοργάνωσης . Το Pastry και το Tapestry εκμεταλλεύονται τις πληροφορίες τοπικότητας που φυλάνε ώστε να εξασφαλίσουν την μικρότερη δυνατή απόσταση δρομολόγησης του μηνύματος . Το Tapestry επιτρέπει στις εφαρμογές να τοποθετούν τα αντικείμενα ανάλογα με τις ανάγκες τους σε αντίθεση με ορισμένα συστήματα που θέτουν περιορισμούς στον αριθμό και την θέση των αντικειμένων .

3.3.2 Οι αλγοριθμοι του Tapestry

3.3.2.1 Το API (περιβάλλον) του Tapestry

Για κάθε κόμβο η αντικείμενο δημιουργούνται αναγνωριστικά από ένα μεγάλο πεδίο με ομοιόμορφη κατανομή . (NodeIDs και GUIDs (Globaly Unique Identifiers) αντίστοιχα) . Μπορούν να ορίζονται περισσότεροι από ένας κόμβος στον ίδιο φυσικό αποδέκτη . Σήμερα το Tapestry χρησιμοποιεί ένα πεδίο με αναγνωριστικά των 160 bits , με μια παγκοσμία ορισμένη βάση (π.χ. δεκαεξαδική , η οποία επιτρέπει lds με 40 δεκαεξαδικά ψηφία) . Τα NodelDs και GUIDs δημιουργούνται ομοιόμορφα κατανεμημένα σε αυτό το διάστημα με μια σταθερή hash συνάρτηση , π.χ. την SHA-1 (Secure Hashing Algorithm) .

Επίσης ορίζεται και ένα αναγνωριστικό εφαρμογής (A_{id}) ως παράμετρος κάθε μηνύματος , γεγονός που επιτρέπει σε πολλές εφαρμογές να μοιράζονται το ίδιο δίκτυο (δηλ. κάθε μήνυμα απευθύνεται σε συγκεκριμένη εφαρμογή , έχοντας το συγκεκριμένο αναγνωριστικό ως παράμετρο του) .

Οι συναρτήσεις με τις οποίες επικοινωνεί το Tapestry με τις εφαρμογές και οι οποίες δημιουργούν το περιβάλλον του Tapestry είναι :

- **PUBLISHOBJECT (OG,Aid)** Αναγγέλλει και κάνει προσπελάσιμο στο δίκτυο το αντικείμενο OG .
- **UNPUBLISHOBJECT (OG,Aid)** Αποσύρει το αντικείμενο OG . το δίκτυο .
- **ROUTETOOBJECT (N,Aid,Exact)** Δρομολογεί ένα μήνυμα στην εφαρμογή Aid στον κόμβο N .

- **ROUTETONODE (N,Aid,Exact)** Δρομολογεί ένα μήνυμα στην εφαρμογή Aid στον κόμβο N.

3.3.2.2 Δρομολόγηση και εντοπισμός αντικειμένων

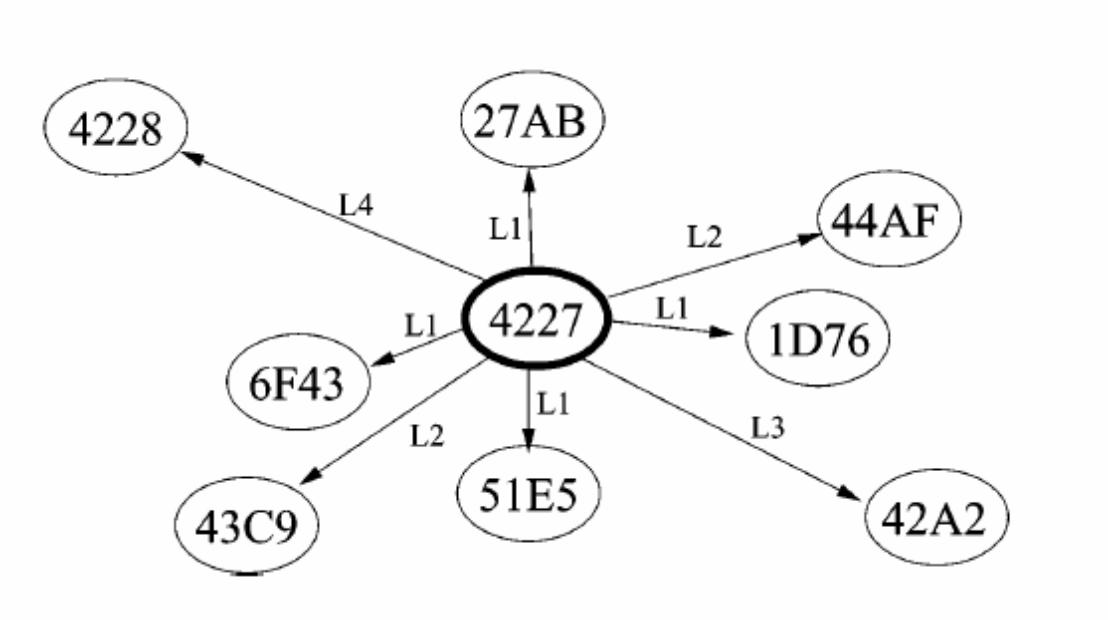
Ο μηχανισμός δρομολόγησης και εντοπισμού αντικειμένων είναι ο ακόλουθος .

Το Tapestry δυναμικά αντιστοιχεί σε κάθε αναγνωριστικό G ένα ενεργό κόμβο N που καλείται ρίζα του αναγνωριστικού G_R (δηλαδή αν ο κόμβος N υπάρχει με Nid=G , ο κόμβος αυτός είναι η ρίζα του αναγνωριστικού , και συνεπώς η δρομολόγηση για την εύρεση του κόμβου που το αναγνωριστικό του είναι το G τελειώνει με την εύρεση του κόμβου G_R) . Σημειώνουμε ότι αν το G είναι αναγνωριστικό ενός αντικειμένου , ο κόμβος N δεν έχει αποθηκευμένο το αντικείμενο που έχει αναγνωριστικό το G αλλά γνωρίζει τον ή τους κόμβους που είναι αποθηκευμένο το αντικείμενο .

Κάθε κόμβος φυλάει ένα πίνακα δρομολόγησης που περιέχει τα αναγνωριστικά και τις IP διευθύνσεις όλων των κόμβων με τους οποίους επικοινωνεί (πίνακας γειτόνων) . Τα αναγνωριστικά αυτά είναι οργανωμένα σε επίπεδα ανάλογα με την απόσταση τους από τον N γεγονός που αποτυπώνεται στα ψηφιά του αναγνωριστικού τους . Τα κοντινότερα είναι εκείνα που διαφέρουν μόνο κατά το τελευταίο ψηφίο από τον N , τα επόμενα εκείνα που διαφέρουν κατά τα 2 τελευταία ψηφιά κλπ. Αν π.χ. το id του N είναι 4227 κοντινότερο είναι το 4225 , μετά το 426A , μετά το 44B9 κλπ. Σε κάθε επίπεδο πρέπει να υπάρχουν στον πίνακα κόμβοι από όλες τις δυνατές τιμές στο ψηφίο που διαφοροποιείται στο επίπεδο αυτό , δηλαδή κόμβοι σε πλήθος ίσο με την βάση του πεδίο των αναγνωριστικών . Π.χ. αν η βάση του πεδίου είναι δεκαεξαδική , στο 4^o επίπεδο του κόμβου 325AE2 πρέπει να

υπάρχουν οι κόμβοι που αρχίζουν από 3250.. , 3251.. , , 3252.. , , 3254.. , κλπ. Μεχρι και τον 325F..

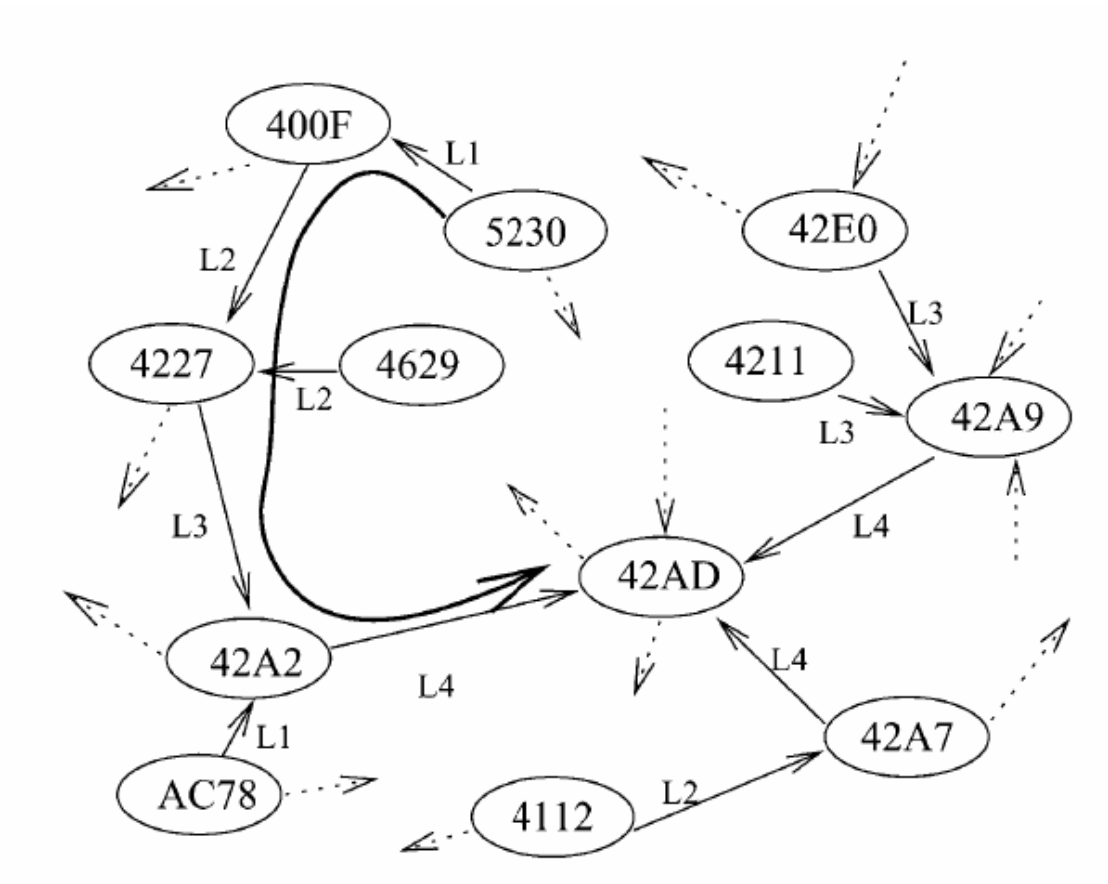
Γενικότερα το περιεχόμενο της i-οστης θέσης στο j επίπεδο είναι ο πλησιέστερος κόμβος που το ID του ξεκινάει με το πρόθεμα (N, j -1)+ i . (π.χ. στην $9^{η}$ θέση του $4^{ου}$ επιπέδου για τον κόμβο 325AE είναι ο πλησιέστερος κόμβος με ID που ξεκινάει από 3259 (το 9 είναι το $9^{ο}$ ψηφίο στη $4^{η}$ θέση του προθέματος) . Η παρακάτω εικόνα εμφανίζει μερικούς από τους γειτονικούς κόμβους του κόμβου 4227 .



Εικονα 18

Έτσι ο τρόπος δρομολόγησης των μηνυμάτων είναι ο εξής : Κάθε κόμβος βρίσκει στον πίνακα των γειτόνων του εκείνο τον κόμβο στο επίπεδο του που έχει αναγνωριστικό πλησιέστερο προς τον προορισμό του μηνύματος . Π.χ. αν κόμβος 5230 στέλνει μήνυμα στον 42AD ψάχνει στο 1^{0} επίπεδο ένας κόμβος που αρχίζει από 4) . Ο 400F βρίσκει στο 2^{0} επίπεδο τον 4227 (αναγκαστικά υπάρχει στο 2^{0} επίπεδο τον 4227 (αναγκαστικά υπάρχει

στο 2^o επίπεδο ένας κόμβος με το 2^o ψηφίο ίσο με 2) κλπ. Ο αριθμός των βημάτων δρομολόγησης είναι $O(\log_\beta N)$, όπου β είναι η βάση των αναγνωριστικών . Στην παρακάτω εικόνα φαίνεται ένα παράδειγμα του μονοπατιού που ακολουθεί ένα μήνυμα από τον κόμβο 5230 στον 42 AD.



Εικονα 19

Ένα δεν μπορεί να βρεθεί ενεργός κόμβος που να ταιριάζει με το ψηφίο που ψάχνει , τότε βρίσκεται εκείνος με το πλησιέστερο . Η διαδικασία αυτή λέγεται αλγόριθμος υποκατάστατου και υποκαθιστά ένα μη ενεργό με παρόμοια ID . Ένας ψευδοκώδικας με την συνάρτηση NEXTHOP που αντιστοιχεί κάθε αναγνωριστικό τον κόμβο ρίζα του G_R φαίνεται παρακάτω .

NEXTHOP(n,G)

```
1       If  n = MAXHOP(R) then
2           return self
3       else
4           d← G_n ; e ← R n,d
5           while e = nil do
6               d ← d + 1 (mod β)
7               e ← Rn,d
8           endwhile
9           If e = self then
10              return NEXTHOP (n + 1 , G )
11          else
12              return e
13          endif
14      Endif
```

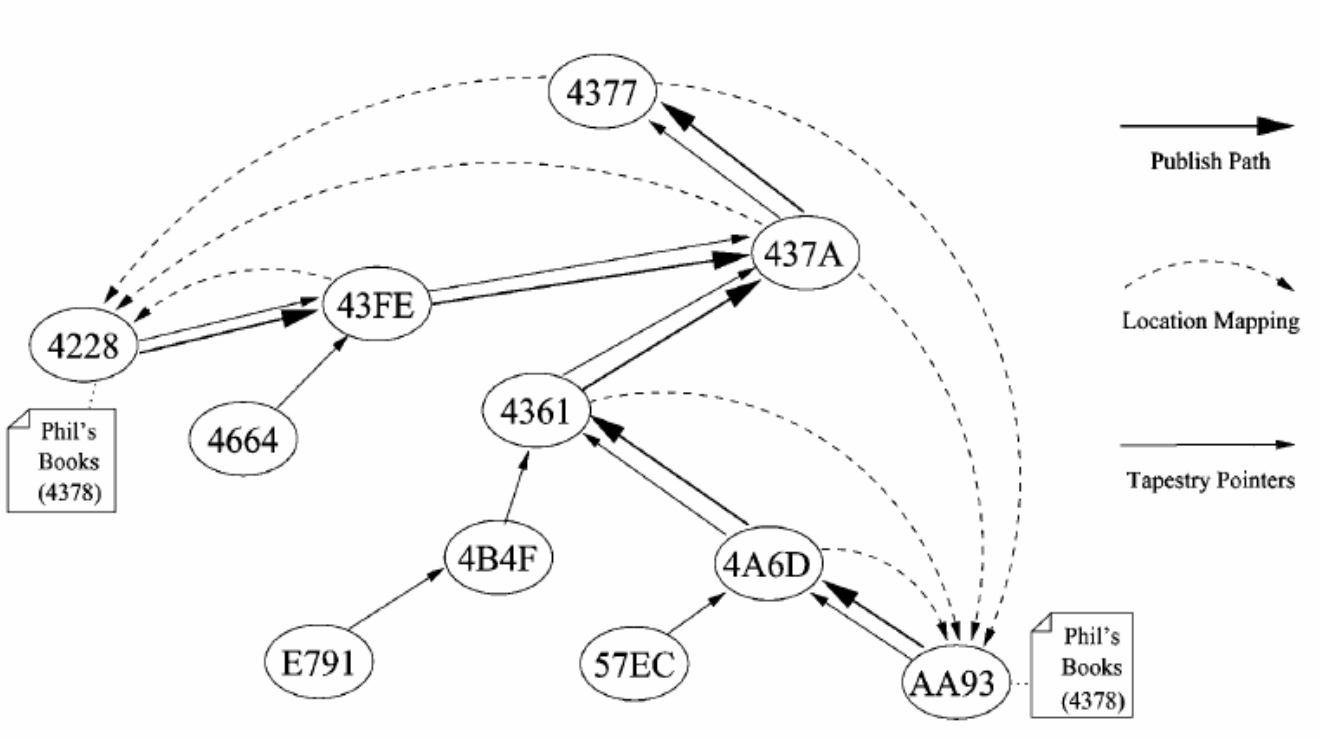
Η διαδικασία αυτή εφαρμόζεται σε ένα κόμβο n και επιστρέφει τον κόμβο του επόμενου βήματος η τον εαυτό του εάν είναι το τελευταίο βήμα .

Για να είναι δυνατή η αποτελεσματική λειτουργία του δικτυού ακόμα και όταν οι κόμβοι καταρρέουν κάθε κόμβος εκτός από τον πίνακα δρομολόγησης με τους κύριους κόμβους που συνδέονται με αυτόν , φυλάει και πρόσθετους συνδεόμενους κόμβους με το ίδιο πρόθεμα . Αν υπάρχουν c κόμβοι , στο n βήμα οι γείτονες κόμβοι που φυλάσσονται στον πίνακα διαφέρουν μόνο κατά το n-οστό ψηφίο τους άρα είναι β το πλήθος (όπου β η βάση του συστήματος των αναγνωριστικών) και συνολικά c *β*$\log_{\beta}$N .

Όπως είπαμε ένα χαρακτηριστικό του Tapestry , σε αντίθεση με το Chord , είναι το γεγονός του ότι ο κόμβος στον οποίον δείχνει το αναγνωριστικό ενός αντικειμένου (ο Gr) , δεν είναι ο κόμβος στον οποίο

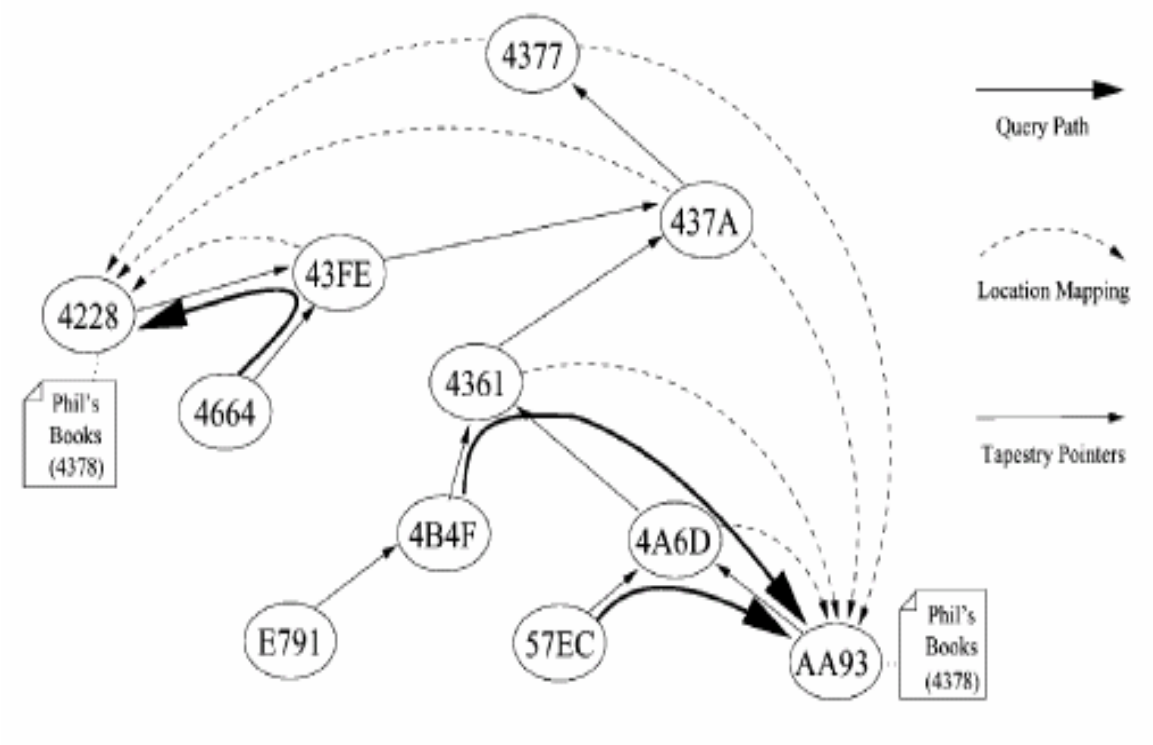
φυλάσσεται το αντικείμενο , αλλά ένας κόμβος ο όποιος ξέρει τον η τους κόμβους στους οποίους αυτό φυλάσσεται . Αυτό δίνει στις εφαρμογές μεγάλη ευελιξία στο να φυλάσσουν όπου θέλουν τα αντικείμενα τους , να δημιουργούν πολλαπλά αντίγραφα κλπ.

Για τον εντοπισμό λοιπόν των αντικειμένων ακολουθείται η διαδικασία της δημοσίευσης – αναγγελίας (Publication) . Ένας server S (ως Server και αργότερα ως Client χαρακτηρίζεται από τους κατασκευαστές μια εφαρμογή που τρέχει σε ένα κόμβο . Δεν πρέπει να υπάρχει σύγχυση με το client-server μοντέλο . Όπως έχουμε πει οι κόμβοι στα p2p είναι servent .) που έχει αποθηκευμένο ένα αντικείμενο O με αναγνωριστικό O_G και κόμβο ρίζας O_R , περιοδικά αναγγέλλει το αντικείμενο στέλνοντας ένα ειδικό μήνυμα προς τον κόμβο της ρίζας του . Κάθε κόμβος που είναι στην διαδρομή από την οποία διέρχεται η αναγγελία αυτή αποθηκεύει την θέση του server . Όταν υπάρχουν αντίγραφα του αντικειμένου σε διαφορετικούς server ,καθένας από αυτούς στέλνει το δικό του μήνυμα . Όταν ένας κόμβος παίρνει μηνύματα από διαφορετικούς server ταξινομημένη ανάλογα με την απόσταση τους από αυτόν . Στην παρακάτω εικόνα φαίνεται αυτή η διαδικασία .



Εικονα20

Αν ένας client ζητήσει το αντικείμενο O , στέλνει μήνυμα προς τον κόμβο ρίζας του O τον O_R . Κάθε κόμβος από τον όποιο περνάει το μήνυμα ελέγχει αν έχει αποθηκευμένα στοιχεία για το O και αν έχει (δηλαδή είναι κάποιος από τους κόμβους που είναι στην διαδρομή της αναγγελίας του server) , ο κόμβος αυτός στρέφει το μήνυμα προς την θέση του πλησιέστερου server που έχει προηγουμένως αποθηκεύσει . Στην παρακάτω εικόνα φαίνεται η διαδικασία αυτή .



Εικονα 21

Στην πιο ακραία περίπτωση το μήνυμα δεν συναντάει τέτοιο κόμβο και φτάνει στον O_R ο οποίος οπωσδήποτε έχει αποθηκεύσει την θέση του ή των server , διότι τα μηνύματα της διαδικασίας του Publication φτάνουν σε αυτόν . Πάντως στην πράξη το μήνυμα συναντάει αρκετά νωρίς έναν ενδιάμεσο κόμβο που το εκτρέπει προς τον πιο κοντινό του server . Με τον τρόπο αυτό το Tapestry εκμεταλλεύεται τις πληροφορίες τοπικότητας που κρατάει ώστε όσο πιο κοντά είναι ένας κόμβος στον server , τόσο πιο γρήγορα φτάνει το μήνυμα .

3.3.2.3 Αλγόριθμοι εισαγωγής και αποχώρησης κόμβων

Το Tapestry επιτρέπει επίσης την δυναμική εισαγωγή και διαγραφή κόμβων .

3.3.2.3.1 Εισαγωγή κόμβων

Ο αλγόριθμος εισαγωγής κόμβων δεν είναι απλός και κάθε εισαγωγή μπορεί να χρειαστεί ένα μη- αμελητέο χρονικό διάστημα . Για την εισαγωγή ενός κόμβου N πρέπει να τηρηθούν οι ακόλουθες απαιτήσεις .

1. Οι κόμβοι που πρέπει , ενημερώνονται για την είσοδο του N
2. Το N γίνεται η ρίζα για μερικά αντικείμενα που το αναγνωριστικό τους είναι κοντά στο N.
3. Δημιουργείται ένας περίπου καλός πίνακας δρομολόγησης για τον N.
4. Οι κόμβοι κοντά στον N , ενημερώνονται ώστε να αλλάξουν τους πίνακες δρομολόγησης τους αν απαιτείται .

3.3.2.3.2 Οικειοθελής αποχώρηση κόμβου

Η οικειοθελής αποχώρηση ενός κόμβου από το σύστημα γίνεται ως εξής :

Ο κόμβος N που αποχωρεί ενημερώνει για την αποχώρηση του όλους τους κόμβους που είναι στην backpointer λίστα του , και του στέλνει επίσης με τους κόμβους που τον αντικαθιστούν σε κάθε επίπεδο . Οι κόμβοι αυτοί αντικαθιστούν σε κάθε επίπεδο . Οι κόμβοι αυτοί αντικαθιστούν τον N με τους αντικαταστάτες του στη λίστα δρομολόγησης τους . Ο N στέλνει επίσης μηνύματα στα αντικείμενα στα οποία είναι ρίζα και τα ενημερώνει για τις νέες ρίζες τους .

3.3.2.3.3 Αθέλητη αποχώρηση κόμβου

Η αθέλητη αποχώρηση ενός κόμβου από το σύστημα αντιμετωπίζεται με την δημιουργία περίσσειας πληροφοριών στους πινάκες δρομολόγησης και

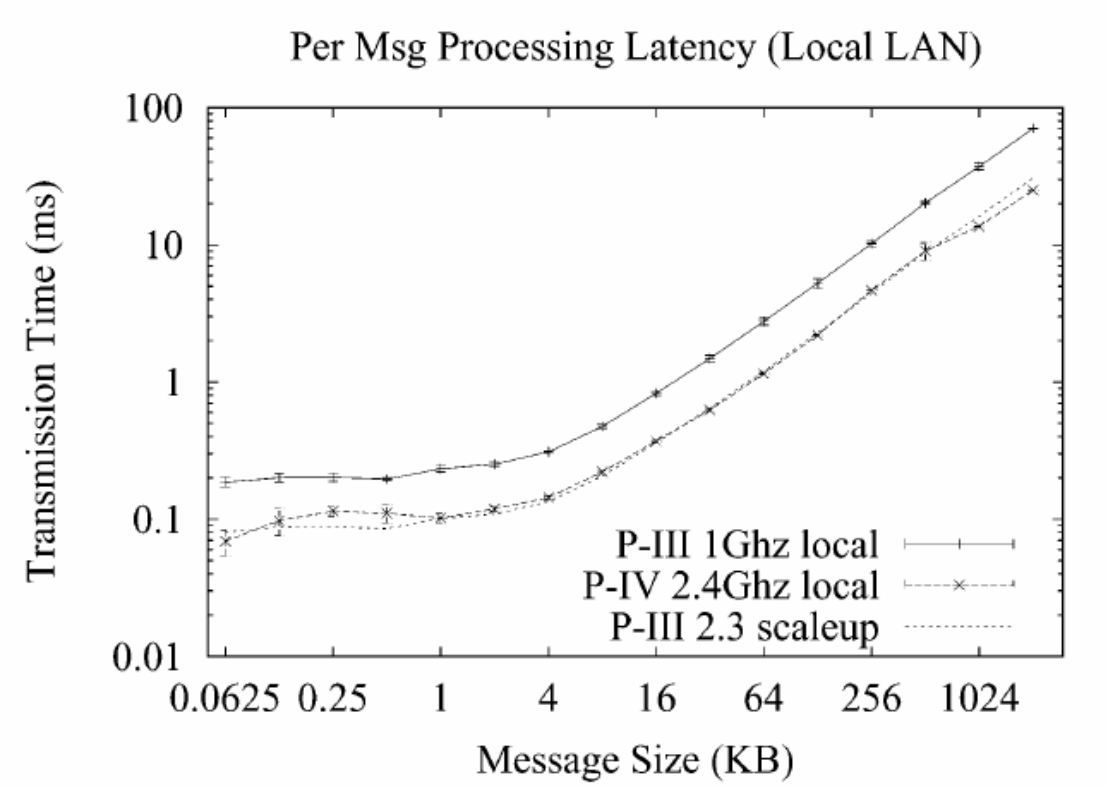
τις αναφορές της θέσης των αντικειμένων . Για τον λόγο αυτό οι κόμβοι στέλνουν περιοδικά μηνύματα στους γειτονικούς τους για να διαπιστώσουν τυχόν αστοχίες . Στην περίπτωση που διαπιστωθεί κάτι τέτοιο επιδιορθώνουν τον πίνακα δρομολόγησης τους και ξεκινούν την ανακατανομή και αντιγραφή των πληροφοριών για την θέση των αντικειμένων .

3.3.3 Υπολογισμοί και πειραματικά αποτελέσματα

Αποτελέσματα προσομοίωσης και πειράματα έδειξαν την συμπεριφορά του δικτύου σε διάφορες περιπτώσεις .

3.3.3.1 Συμπεριφορά σε σταθερό δίκτυο

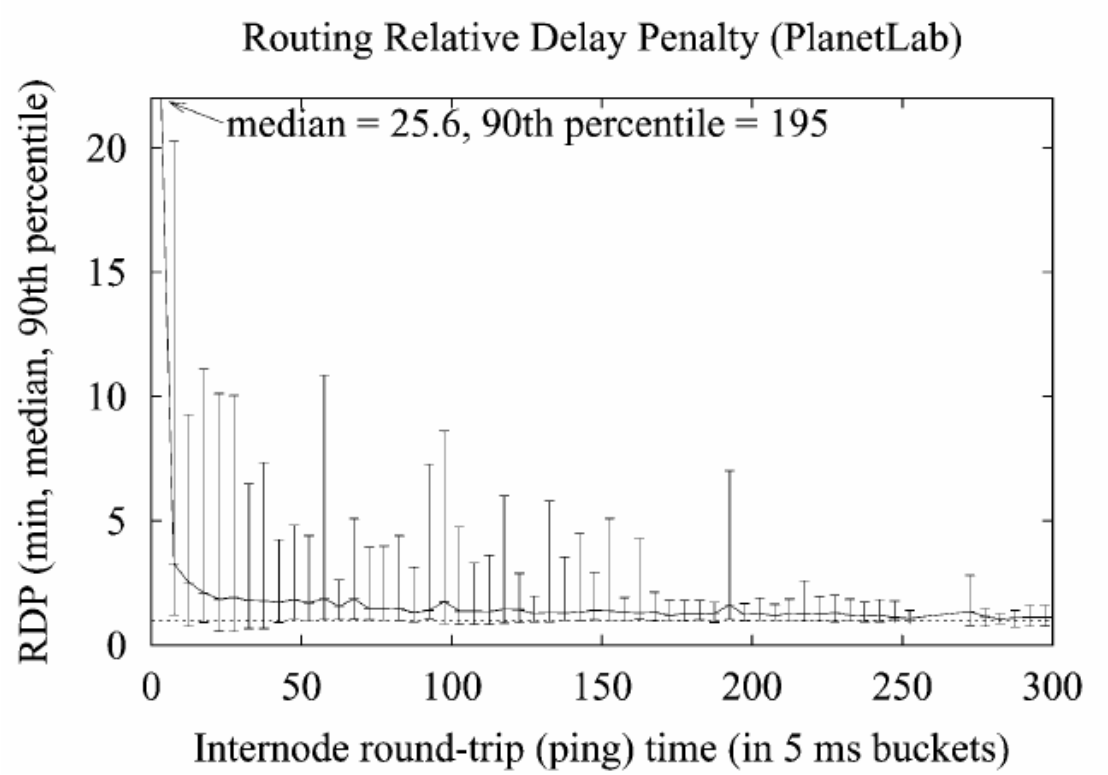
Σε σταθερό δίκτυο μετρήθηκε ο χρόνος μεταφοράς ανάλογα με το μέγεθος του μηνύματος για να προσδιοριστεί ο λανθάνων χρόνος (δηλαδή ο χρόνος που καταναλώνεται από το Tapestry) για την αποστολή μηνυμάτων στο Tapestry , όπως φαίνεται στην παρακάτω εικόνα .



Εικονα 22

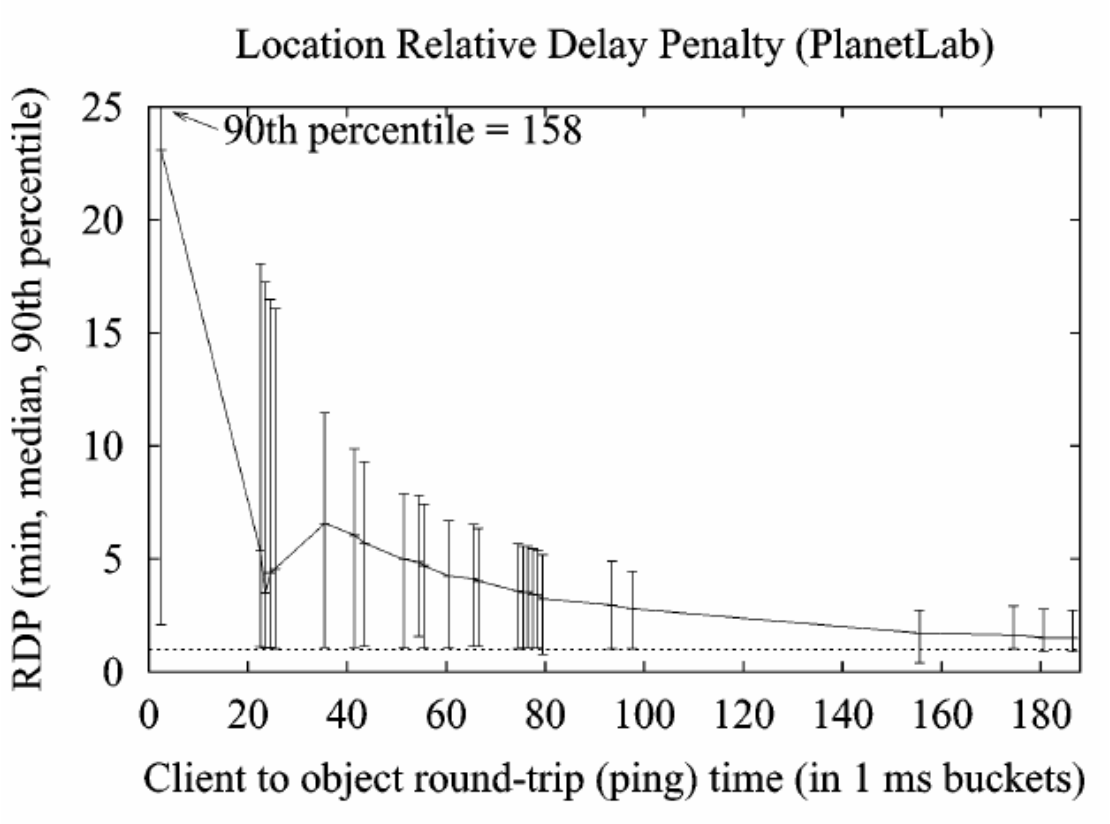
Το σχήμα δείχνει ότι σε μικρό μέγεθος μηνυμάτων ο χρόνος παραμένει σταθερός ανεξάρτητα από το μέγεθος του μηνύματος και εξαρτάται από την ταχύτητα του υπολογιστή. Αυτός ο λανθάνων χρόνος συνεπώς μπορεί να βελτιωθεί στο μέλλον με την αύξηση της ταχύτητας των υπολογιστικών συστημάτων .

Σε σταθερό σύστημα από 400 κόμβους επίσης μετρήθηκε ο RDP δηλαδή ο λόγος του χρόνου της δρομολόγησης σε κόμβους χρησιμοποιώντας το δίκτυο σε σχέση με την συντομότερη IP διεύθυνση . Ο λόγος των διάμεσων τιμών (median) αρχίζει από 3 και τείνει αργά προς το 1 . Αυτό φαίνεται και παρακάτω στην εικόνα 23 .



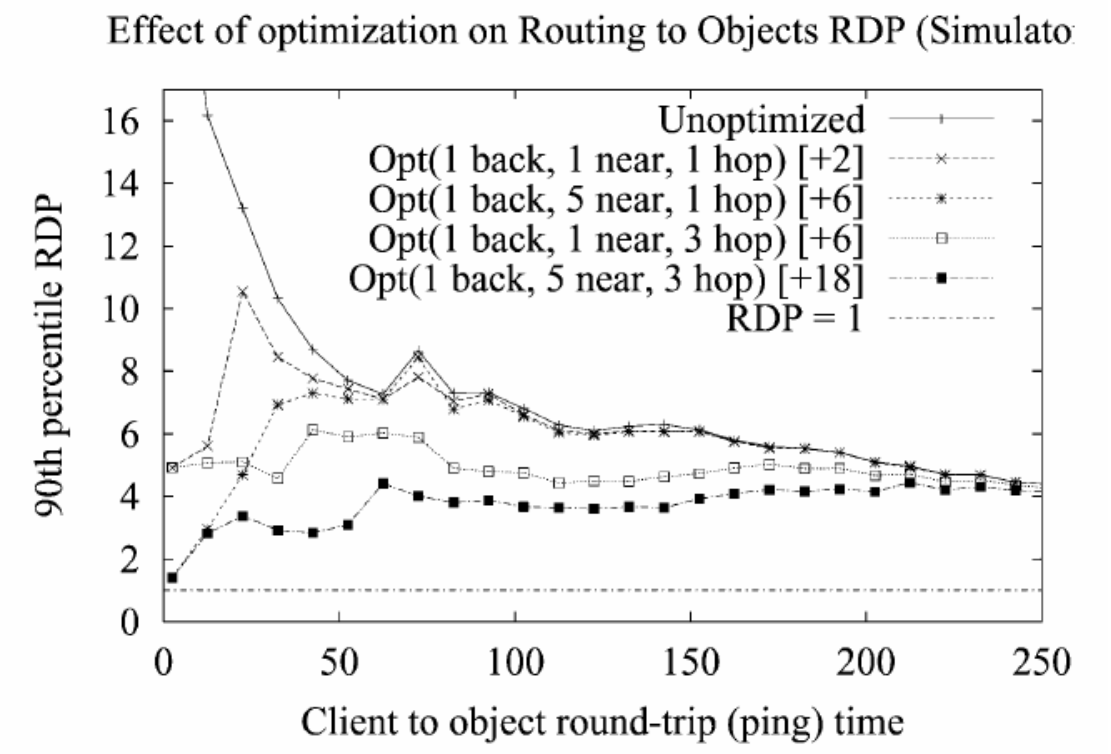
Εικονα 23

Ο ίδιος λόγος για δρομολόγηση σε αντικείμενα φαίνεται στην εικόνα 24 :



Εικόνα 24

Στην παρακάτω εικόνα 25 φαίνεται ο RDP στην δρομολόγηση αντικειμένων χρησιμοποιώντας βελτιστοποιήσεις στο δίκτυο και συγκεκριμένα το πλήθος k των εναλλακτικών κόμβων σε κάθε βήμα των πλησιέστερων γειτόνων στα πρώτα m βήματα κάθε δρομολόγησης .

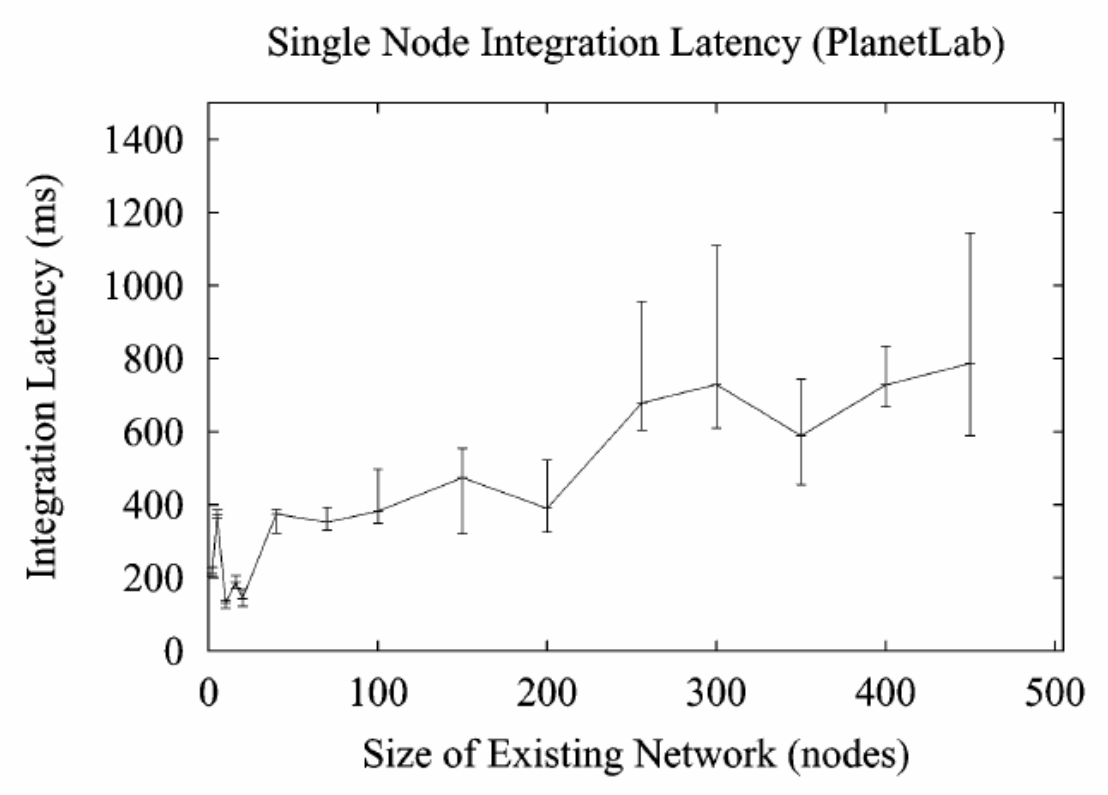


Εικόνα 25

Από την εικόνα αυτή προκύπτει σημαντική μείωση του RDP με μικρή μόνο βελτιστοποίηση .

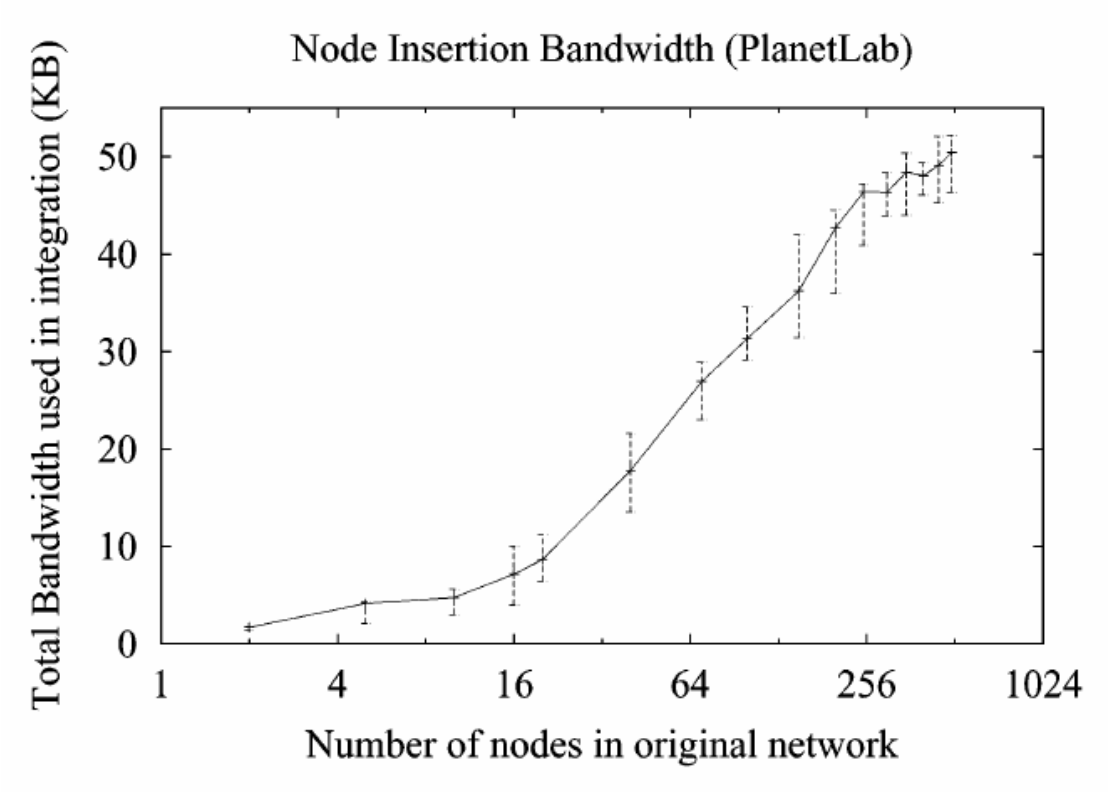
3.3.3.2 Δυναμική συμπεριφορά του δικτύου

Στην παρακάτω εικόνα φαίνεται ο χρόνος που απαιτείται για την εισαγωγή ενός κόμβου (μέχρι να σταθεροποιηθεί το δίκτυο) ανάλογα με το πλήθος των κόμβων .



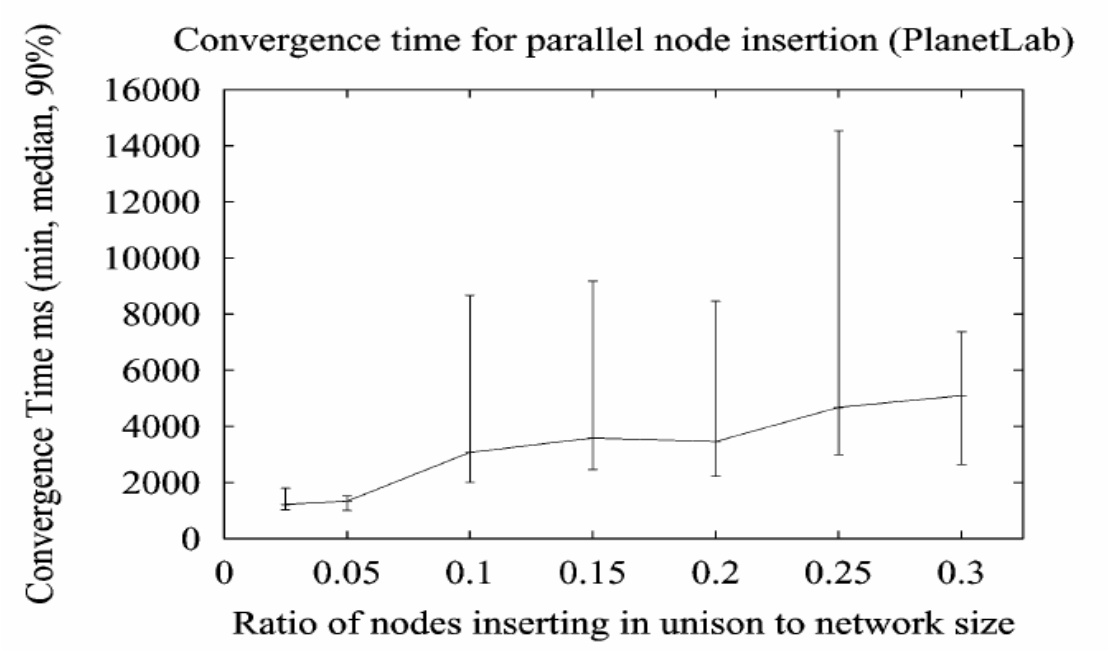
Εικόνα 26

Το μέγεθος των στοιχείων που ανταλλάσσονται μεταξύ των κόμβων από την εισαγωγή ενός κόμβου σε σχέση με το πλήθος του δικτύου αυξάνεται λογαριθμικά και αυτό φαίνεται στην παρακάτω εικόνα.



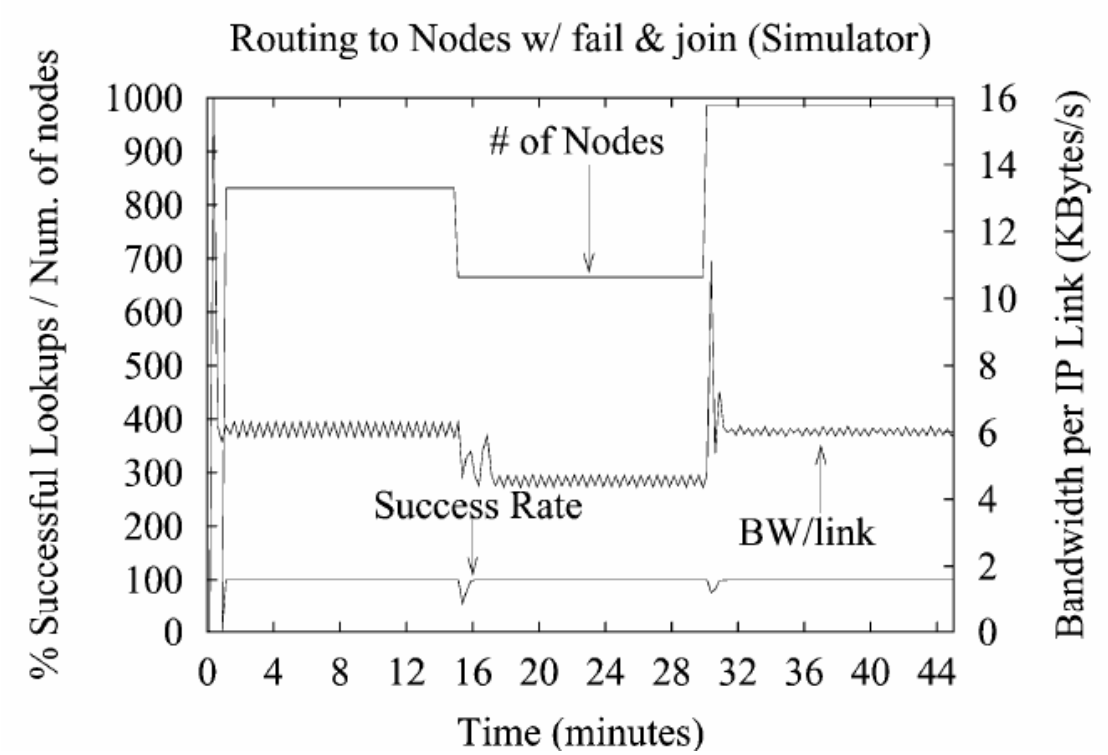
Εικόνα 27

Στο παρακάτω εικόνα 28 φαίνεται ο χρόνος εισαγωγής πολλών κόμβων ταυτόχρονα σε ένα δίκτυο σε σχέση με τον λόγο των εισερχομένων κόμβων προς το σύνολο των κόμβων του δικτύου .

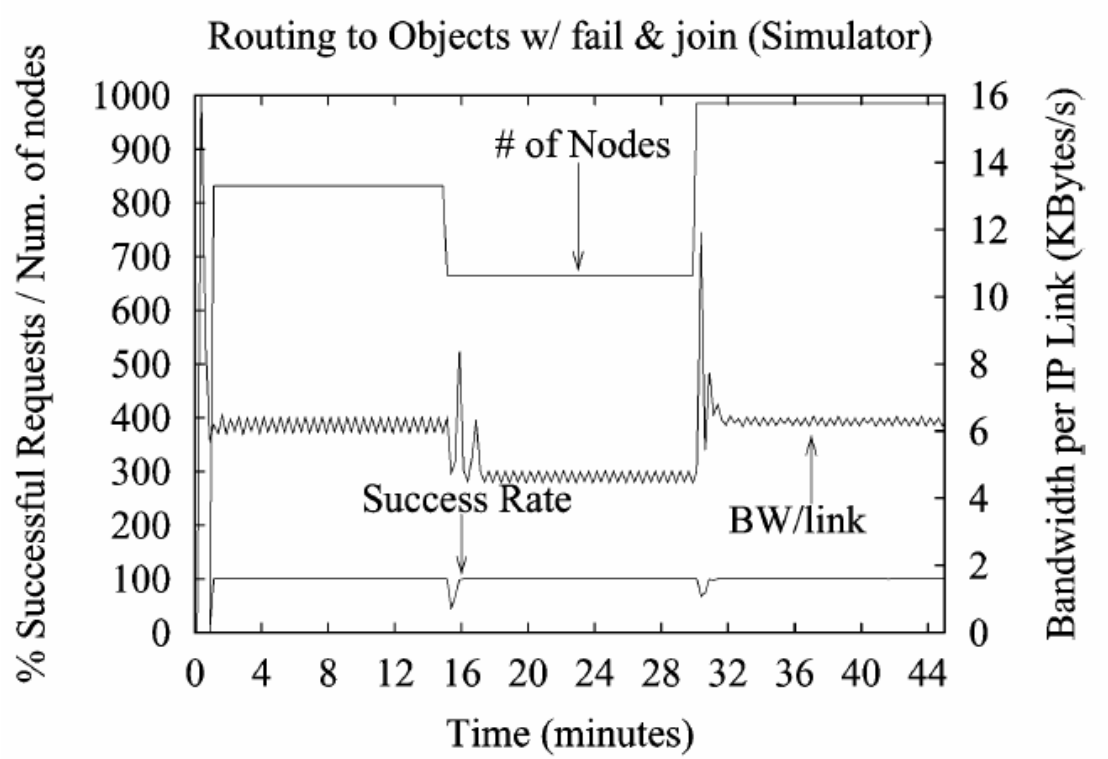


Εικόνα 28

Τέλος μετρήθηκε η συμπεριφορά του δικτύου σε μεγάλη κλίμακα μεγάλες αυξομειώσεις σε δίκτυο 1000 κόμβων . Για τον λόγο αυτό μετρήθηκε το ποσοστό των σωστών δρομολογήσεων των μηνυμάτων καθώς και το μέγεθος των διακινούμενων πληροφοριών όταν κατ' αρχήν μειώθηκε κατά 20 % και μετά 15 λεπτά αυξήθηκε το πλήθος των κόμβων κατά 50 % , όπως φαίνεται στις εικόνες .



Εικόνα 29



Εικόνα 30

Παρατηρείται ότι και στη μείωση και στην αύξηση , μετά πολύ λίγο χρόνο που απαιτείται για την σταθεροποίηση του δικτύου , το ποσοστό των σωστών δρομολογήσεων φτάνει το 100% .

Γενικά το Tapestry είναι ένα αξιόλογο πρωτόκολλο $2^{ης}$ γενιάς που επιτρέπει αυξομειώσεις του δικτύου και εκμεταλλεύεται την τοπικότητα .Παντός έχει σχετικά δύσκολη εισαγωγή κόμβων στο σύστημα και για αυτόν τον λόγο , το Tapestry είναι μια ακατάλληλη αρχιτεκτονική για δίκτυα που αλλάζουν συνεχώς .

3.4 Pastry

3.4.1 Εισαγωγή

Το Pastry αναπτύχθηκε από τους Antony Powstron και Peter Druschel στα πλαίσια project Past της Microsoft . Είναι πλήρως αποκεντρωμένο , ανθεκτικό και αυτό – οργανωμένο και προσαρμόζεται αυτόματα στην άφιξη , αναχώρηση και αποτυχία των κόμβων . Αποτελεί ένα υπόστρωμα για μεγάλης κλίμακας P2P εφαρμογές που παρέχει μηχανισμούς κατανεμημένου εντοπισμού αντικειμένων (object location) και δρομολόγησης (routing) .

Οι εφαρμογές που στηρίζονται στο Pastry χρησιμοποιούν τις ικανότητες εντοπισμού και δρομολόγησης του με διαφόρους τρόπους . Το Past π.χ. δημιουργεί για κάθε αρχείο ένα fileld από το όνομα του αρχείου και του ιδιοκτήτη του μέσω μιας Hash συνάρτησης . Αντίγραφα του αρχείου αποθηκεύει σε k κόμβους του Pastry που έχουν τα πλησιέστερα αριθμητικά nodelds με το field του αρχείου . Αν κάποιος πελάτης (client) θέλει να αναζητήσει το αρχείο στέλνει ένα μήνυμα με κλειδί το fileld του αρχείου . Το Pastry , αν δεν έχουν καταρρεύσει και οι k κόμβοι θα δρομολογήσει το μήνυμα σε έναν από τους κόμβους αυτούς στους οποίους είναι αποθηκευμένο το αρχείο . Επιπλέον λόγω της τοπικότητας που εξασφαλίζει το Pastry το μήνυμα πιθανότατα θα φτάσει σε ένα κόμβο που είναι πλησιέστερα προς τον πελάτη από τους k κόμβους .

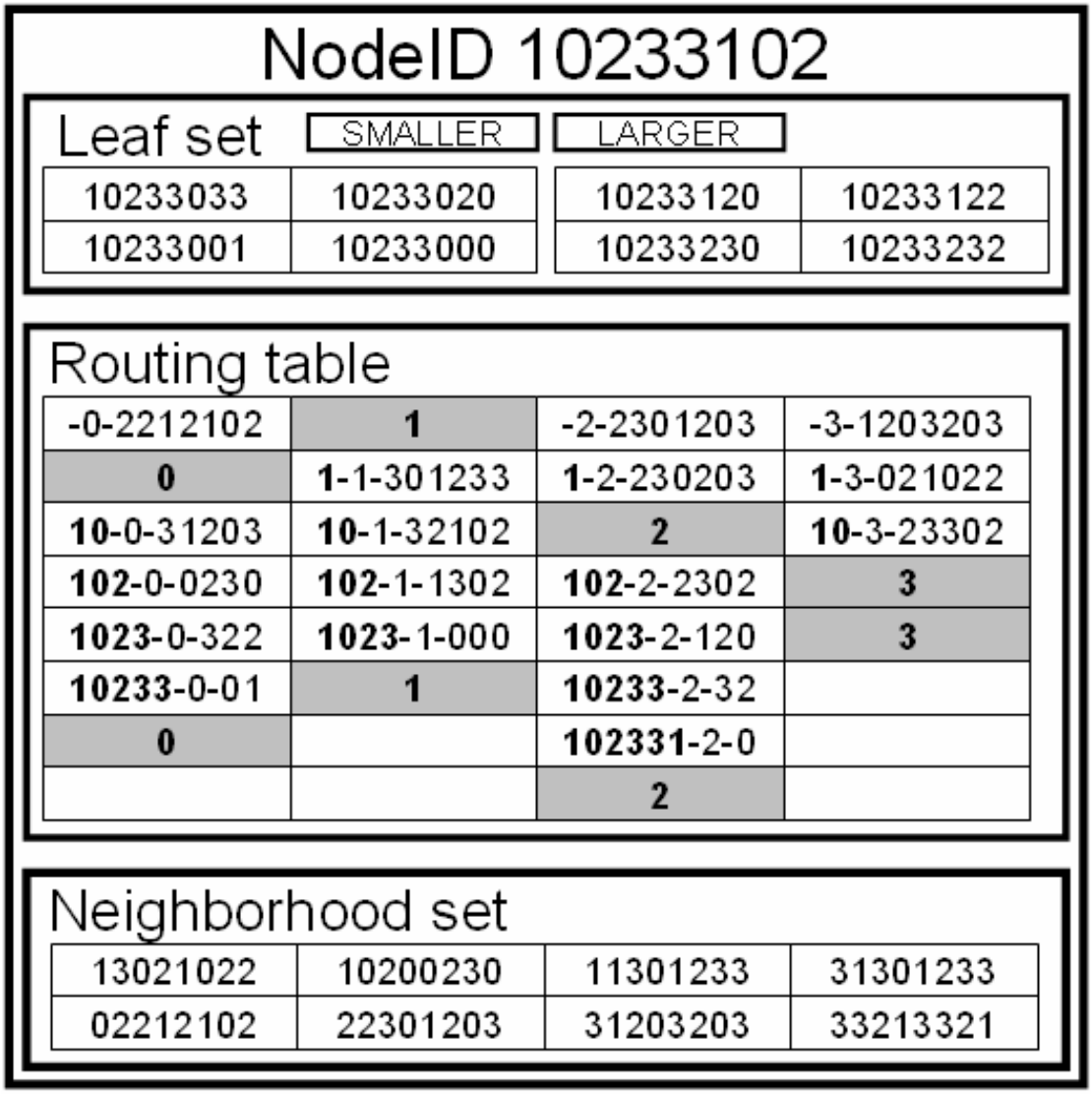
3.4.2 Σχεδιασμός του Pastry

Κάθε κόμβος στο δίκτυο του Pastry έχει ένα ξεχωριστό προσδιοριστή (indentifier) 128 bit το nodelID (δηλαδή σε ένα πεδίο από 0 έως 2^{128} -1) Ο

προσδιοριστής αυτός προσδιορίζεται τυχαία με ομοιόμορφα κατανεμημένο τρόπο όταν ένας κόμβος συνδέεται με το σύστημα . Ο τυχαίος τρόπος (με μια Hash συνάρτηση από την IP address) του προσδιορισμού του nodeID σημαίνει ότι κόμβοι με γειτονικούς nodeIDs διαφέρουν γεωγραφικά , κατά τον ιδιοκτήτη κλπ. Όταν εμφανίζεται ένα μήνυμα και ένα κλειδί (key) , ένας κόμβος του Pastry δρομολογεί αποτελεσματικά το μήνυμα στον κόμβο με nodeID που βρίσκεται πλησιέστερα αριθμητικά στο κλειδί , από τους άλλους ενεργούς κόμβους.

3.4.2.1 Δρομολόγηση και πίνακες στοιχείων των κόμβων

Η δρομολόγηση γίνεται προς τον κόμβο με τον πλησιέστερο nodeID σε σχέση με το κλειδί . Αυτό γίνεται με την οργάνωση των nodeIDs σε ψηφία . Όταν ένα μήνυμα φτάνει σε ένα κόμβο συγκρίνεται το κλειδί με το nodeID των κόμβων που είναι γνωστοί και προωθείται σε εκείνων τον κόμβο που το nodeID του έχει τουλάχιστον n+1 κοινά ψηφιά στο πρόθεμα του με το κλειδί , όπου n είναι το πλήθος των κοινών στοιχείων στο πρόθεμα του nodeID του κόμβου με το κλειδί . Αν π.χ. Στον κόμβο με nodeID 10233102 φθάσει μήνυμα με το κλειδί 10221103 και υπάρχει γνωστός κόμβος με nodeID 10222302 τότε το μήνυμα μπορεί να προωθηθεί εκεί . Αν δεν υπάρχει τέτοιος κόμβος , τότε βρίσκεται ο κόμβος με n κοινά ψηφιά στο πρόθεμα του που είναι ο πλησιέστερος αριθμητικά με το κλειδί . Για να γίνει αυτό κάθε κόμβος φυλάει 3 πίνακες κόμβων , όπως για παράδειγμα στην παρακάτω εικόνα .



Εικόνα 31

Ο routing table αποτελείται από $\log_2{}^b$ N γραμμές και 2^b -1 στήλες και περιέχει τόσο nodeIDs όσο και τις IP διευθύνσεις των κόμβων . Στην n (όπου η $1^η$ γραμμή αρχίζει από το 0) γραμμή περιλαμβάνονται οι κόμβοι που το πρόθεμα του nodeID τους έχει n κοινά ψηφία με το nodeID του κόμβου , ενώ το επόμενο ψηφίο είναι ο αριθμός της στήλης . Π. χ. στην γραμμή 2 και στη στήλη 1 στο παράδειγμα της εικόνας υπάρχει το 10132102 που τα 2 πρώτα

ψηφία είναι κοινά με το nodeID του κόμβου (10) και το 3^{o} είναι το 1 (α/α της στήλης) .

Η λίστα των γειτονικών κόμβων M περιέχει τα nodeIDs και τις IP διευθύνσεις των κόμβων που είναι πλησιέστερα στον τοπικό κόμβο . Δεν χρησιμοποιούνται στην δρομολόγηση των μηνυμάτων αλλά στην διευθέτηση των τοπικών ιδιοτήτων του κόμβου .

Η λίστα των κόμβων φύλλων L είναι οι L / 2 κόμβοι με τα nodeID τους τα πλησιέστερα προς τα κάτω σε σχέση με το nodeID του τοπικού κόμβου και οι L / 2 κόμβοι με τα nodeIDs τους τα πλησιέστερα προς τα πάνω σε σχέση με το nodeID του τοπικού κόμβου . Η λίστα αυτή χρησιμεύει στην δρομολόγηση των μηνυμάτων όταν δεν βρίσκεται κόμβος με τουλάχιστον n+1 κοινά ψηφία στο πρόθεμα του με το κλειδί όπως παραπάνω αναφέρθηκε . Οι τυπικές τιμές για τα M και L είναι 2^{b} ή $2*2^{b}$.

Ένας ψευδοκώδικας της διαδικασίας δρομολόγησης φαίνεται στον παρακάτω πίνακα .

```
(1)  if (L_{-⌊|L|/2⌋} ≤ D ≤ L_{⌊|L|/2⌋}) {
(2)      // D is within range of our leaf set
(3)      forward to L_i, s.th. |D − L_i| is minimal;
(4)  } else {
(5)      // use the routing table
(6)      Let l = shl(D, A);
(7)      if (R_l^{D_l} ≠ null) {
(8)          forward to R_l^{D_l};
(9)      }
(10)     else {
(11)         // rare case
(12)         forward to T ∈ L ∪ R ∪ M, s.th.
(13)             shl(T, D) ≥ l,
(14)             |T − D| < |A − D|
(15)     }
(16) }
```

Τα διάφορα σύμβολα είναι :

Ri,l : Το περιεχόμενο του routing table στην στήλη i και την γραμμή l.

Li : Το περιεχόμενο της λίστας των φύλλων στη θέση i .

Di : Το i ψηφίο του κλειδιού D .

Shl (A,B) : Το μήκος του κοινού προθέματος μεταξύ των A,B

Ο κώδικας αυτός στην αρχή ψάχνει αν το κλειδί βρίσκεται στην περιοχή της λίστας των φύλλων . Αν συμβαίνει αυτό δρομολογεί κατευθείαν το μήνυμα στον κόμβο από τους κόμβους των φύλλων , που το nodeID του είναι το πλησιέστερο αριθμητικά με το κλειδί . Αν όχι ακολουθεί την διαδικασία του routing table που περιγράφηκε ανωτέρω δηλαδή βρίσκει τον κόμβο που το nodeID του τοπικού κόμβου με το κλειδί) . Αν τέτοιος κόμβος δεν υπάρχει ή δεν είναι δυνατόν να συνδεθεί τότε βρίσκεται ο κόμβος με n

κοινά ψηφιά στο πρόθεμα του , που είναι ο πλησιέστερος αριθμητικά με το κλειδί .

Αυτός ο αλγόριθμος εγγυάται πάντα την προώθηση του μηνύματος διότι σε κάθε βήμα ή το μήνυμα σε κόμβο με περισσότερα κοινά ψηφιά με το κλειδί από αυτόν στον οποίο βρίσκεται ή με τα ίδια μεν ψηφία αλλά πλησιέστερο αριθμητικά . Με σωστούς routing table και χωρίς πρόσφατες αστοχίες κόμβων ο αλγόριθμος απαιτεί $\log_2^b$ N βήματα . Σε περίπτωση αστοχίας κόμβων που ταυτόχρονα αστοχούν .

3.4.2.2 Το περιβάλλον (API) του Pastry

Οι συναρτήσεις με τις οποίες επικοινωνεί το Pastry με τις εφαρμογές και οι οποίες δημιουργούν το περιβάλλον του είναι οι παρακάτω .

1. Το Pastry ανταποκρίνεται στις κατωτέρω εντολές από τις εφαρμογές :

    - nodeId=pastryInit(Credentials,Application) Συνδέει στο δίκτυο το κόμβο , κάνει όλες τις απαραίτητες αρχικοποιήσεις και επιστρέφει το nodeId του κόμβου . Η παράμετρος Application είναι ένας χειριστης της εφαρμογής που επιτρέπει στον κόμβο να επικοινωνεί με την εφαρμογή του κάτι συμβαίνει .
    - route(msg,key) Δρομολογεί το μήνυμα στον κόμβο με το πλησιέστερο αριθμητικά nodeId σε σχέση με το κλειδί .

2. Οι εφαρμογές που συνεργάζονται με το Pastry πρέπει να δέχονται από το Pastry τις κατώτερο εντολές :

    - deliver(msg,key) Καλείται από το Pastry όταν λαμβάνεται ένα μήνυμα και το παραδίδει στην εφαρμογή του κόμβου με το πλησιέστερο αριθμητικά nodeld σε σχέση με το κλειδί .
    - forward(msg,key,nextld) (N,Aid,Exact) Ενημερώνει την εφαρμογή ενός κόμβου προτού αποστείλει το μήνυμα στον επόμενο κόμβο .
    - newLeafs(leafSet) Το Pastry ενημερώνει την εφαρμογή ενός κόμβου για οποιαδήποτε αλλαγή στη λίστα των φύλλων του κόμβου .

Αρκετές εφαρμογές έχουν ήδη αναπτυχθεί χρησιμοποιώντας αυτό το απλό περιβάλλον .

3.4.2.3 Είσοδος και έξοδος κόμβων στο σύστημα

Όταν ένας κόμβος εισέρχεται στο σύστημα πρέπει να γνωρίζει έναν κοντινό του κόμβο Α που είναι ήδη στο σύστημα (αυτό πάντα μπορεί να το κάνει) . Παίρνει έναν nodeID έστω Χ (που τον δίνει η εφαρμογή και όχι το Pastry με μια hash συνάρτηση , τυπικά την SHA-1 από την IP διεύθυνση του κόμβου). Ο Χ στέλνει στον Α ένα ειδικό μήνυμα σύνδεσης με κλειδί την τιμή του Χ . Το Pastry όπως κάθε μήνυμα το δρομολογεί προς τον κόμβο με το πλησιέστερο nodeID με το Χ που είναι συνδεδεμένος στο σύστημα . Εφόσον ο Χ δεν έχει ακόμα συνδεθεί στο σύστημα , βρίσκει τον πιο κοντινό του αριθμητικά κόμβο , έστω Z . Στην διαδρομή του μηνύματος ο Α ο Z και όλοι οι ενδιάμεσοι κόμβοι της διαδρομής στέλνουν στον Χ , τους πίνακες της

κατάστασης τους . Ο Χ χρησιμοποιεί τις πληροφορίες αυτές για να αρχικοποιήσει τους πινακες τους . Συγκεκριμένα επειδή ο Α είναι γειτονικός του κόμβος , χρησιμοποιεί την λίστα των γειτονικών κόμβων ως δίκια του λίστα των γειτονικών κόμβων . Επειδή ο Z είναι ο πλησιέστερος αριθμητικά κόμβος στον Χ , χρισημιποει την λίστα των κόμβων των φύλλων , για την δίκια του του λίστα των κόμβων φύλλων . Για την δημιουργία του routing table χρησιμοποιει αρχικά το routing table του Α .

Ας υποθέσουμε για λόγους γενικότητας ότι τα nodeID των Χ και Α δεν έχουν κανένα κοινό ψηφίο . Τότε η 0 γραμμή του Α περιέχει κόμβους που δεν έχουν κανένα κοινό ψηφίο με το Α , άρα κάποιοι από αυτούς δεν έχουν κανένα κοινό στοιχείο με το Χ και μπορούν να χρησιμοποιηθούν για την 0 γραμμή του Χ . Ο πρώτος κόμβος που συναντάται στην δρομολόγηση από τον Α στον Z έχει το $1^{ο}$ ψηφίο του κοινό με του Χ , άρα με μερικά από τα στοιχεία της $1^{ης}$ γραμμής του routing table , μπορεί να γίνει η $1^{η}$ η πρώτη γραμμή του routing table του Χ .Με τον τρόπο αυτό η $2^{η}$ γραμμή του $2^{ου}$ κόμβου είναι η $2^{η}$ γραμμή του Χ και ούτω καθ’ έξης . Κατόπιν ο Χ ενημερώνει όλους τους κόμβους των φύλλων του , τους γειτονικούς και του routing table για την ύπαρξη του , έτσι ώστε και αυτοί να αναπροσαρμόσουν τους δικούς τους πινάκες .

Η αθέλητη αποχώρηση ενός κόμβου από το σύστημα γίνεται γνωστή όταν ένας κόμβος του συστήματος δεν μπορεί να επικοινωνήσει με αυτόν . Οι κόμβοι που τον περιλαμβάνουν στη λίστα των φύλλων τους πρέπει να τον αντικαταστήσουν και γι’ αυτό επικοινωνούν με τον ενεργό κόμβο που ανήκει στη λίστα των φύλλων και που έχει το μεγαλύτερο δείκτη αρίθμησης από την μεριά του κόμβου που χάθηκε . Π.χ. αν χάθηκε ένας κόμβος από την αριστερή πλευρά (των μικρότερων) της λίστας των φύλλων L ,θα ερωτηθεί ο κόμβος

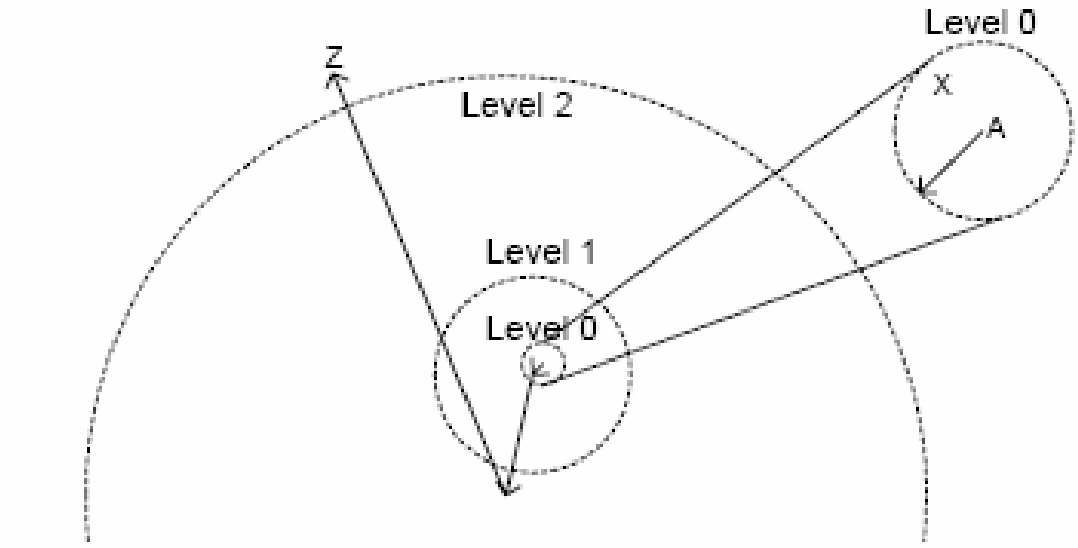
$L_{[l/2]}$ εάν είναι ενεργός , αν όχι ο $L_{[L/2]+1}$ και ούτω καθ' εξής . Από τον κόμβο αυτόν λαμβάνεται η λίστα L' η οποία περιλαμβάνει και κόμβους που δεν υπάρχουν στην L . Από τους κόμβους αυτούς ο καταλληλότερος , εφ' όσον ελεγχθεί ότι είναι ενεργός , αντικαθιστά τον χαμένο κόμβο στην L . Με την διαδικασία αυτή είναι σίγουρη η αντικατάσταση του κόμβου , εκτός αν καταρρεύσουν ταυτόχρονα μέχρι L/2 κόμβοι με γειτονικά nodeID , γεγονός που είναι πολύ σπάνιο .

Αν καταρρεύσει ένας κόμβος που είναι στο routing table ενός αλλού κόμβου , το Pastry συνεχίζει χωρίς σοβαρή απώλεια χρόνου την δρομολόγηση του μηνύματος αλλά ο κόμβος πρέπει να αντικατασταθεί για την ακεραιότητα του πίνακα . Έστω ότι ένας κόμβος πρέπει να αντικαταστήσει τον κόμβο $R_{d,l}$ (όπου $R_{d,l}$ παριστάνει το περιεχόμενο του routing table στην στήλη d και την γραμμή l .). Κατ' αρχήν συνδέεται με ένα άλλο κόμβο στην ίδια σειρά , έστω τον $R_{d,l}$ (όπου i ! = d) και τον ρωτά για τον κόμβο $R_{d,l}$. Επειδή κανένας ενεργός κόμβος στη γραμμή l δεν έχει το κατάλληλο πρόθεμα ο κόμβος συνδέεται με ένα κόμβο στην επόμενη γραμμή τον $R_{i,l+1}$ (όπου i ! = d) . Η λίστα των γειτονικών κόμβων M δεν χρησιμεύει στη δρομολόγηση, είναι όμως σημαντικό να διατηρείται επίκαιρη διότι παίζει ρόλο στην ανταλλαγή πληροφοριών εγγύτητας.

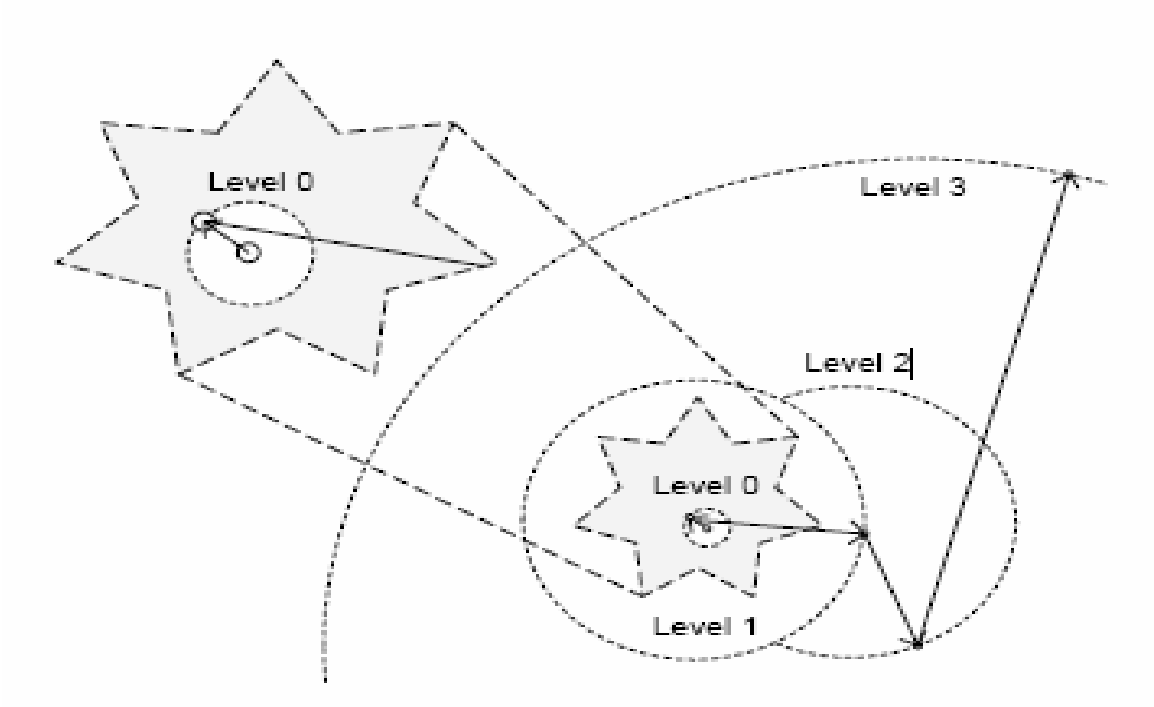
Για τον λόγο αυτό κάθε κόμβος περιοδικά συνδέεται με τους κόμβους που βρίσκονται στην λίστα αυτή για να διαπιστώσει αν είναι ακόμα ενεργοί . Εάν ένας κόμβος δεν απαντά , ο κόμβος επικοινωνεί με τους υπόλοιπους κόμβους της λίστας , ελέγχει την απόσταση του καθενός κόμβου που περιέχεται στη λίστα των γειτονικών κόμβων M' καθενός από αυτούς και ανάλογα συμπληρώνει την δίκια του λίστα M .

3.4.2.4 Τοπικότητα

Μια πολύ ενδιαφέρουσα ιδιότητα του Pastry είναι η τοπικότητα . Αυτό σημαίνει ότι η διαδρομή που επιλεγεί για την δρομολόγηση ενός μηνύματος είναι καλή σε σχέση με ένα μέτρο της εγγύτητας (απόστασης) . Ένα τέτοιο μέτρο μπορεί να οριστεί από τα βήματα της IP address των κόμβων ή την γεωγραφική απόσταση μεταξύ τους . Υποτίθεται ότι οι εφαρμογής που τρέχουν το Pastry διαθέτουν μια συνάρτηση για να υπολογίζει την απόσταση αυτή . Υποτίθεται επίσης ότι η απόσταση αυτή είναι Ευκλείδεια , ότι δηλαδή ισχύει η τριγωνική ανισότητα για τις αποστάσεις μεταξύ των κόμβων του Pastry . Αυτό δεν είναι πάντα σωστό (όπως π. χ. στα βήματα της IP address των κόμβων) , αλλά εδώ θα το θεωρήσουμε δεδομένο .



Εικόνα 32



Εικόνα 33

Με τον τρόπο αυτό επιτυγχάνεται πολύ καλή τοπικότητα στην δρομολόγηση .

Μερικά προγράμματα που χρησιμοποιούν το Pastry όπως το PAST , αντιγράφουν τις πληροφορίες τους σε πολλούς (k) κόμβους που είναι πλησιέστερα σε ένα κλειδί (το PAST φυλάει τα αρχεία έτσι , ώστε να μπορεί να τα χρησιμοποιήσει ακόμα και σε αστοχία αρκετών κόμβων) . Το Pastry δρομολογεί εξ ορισμού το μήνυμα στον πλησιέστερο αριθμητικά ζωντανό κόμβο και εγγυάται ότι το μήνυμα θα φτάσει έστω και αν μονό ένας από τους k κόμβους είναι ζωντανός . Επιπλέον εξ αίτιας της τοπικότητας του Pastry το μήνυμα φτάνει σε ένα κόμβο από τους k που είναι σχετικά κοντά σε σχέση με το μέτρο της εγγύτητας που χρησιμοποιεί . Επειδή το Pastry χρησιμοποιεί τοπικές πληροφορίες κατά την δρομολόγηση και επειδή βασίζεται στην εγγύτητα του προθέματος των nodeIDs με το κλειδί η εγγύτητα αυτή των k κόμβων είναι προσεγγιστική . Στην χειρότερη περίπτωση k/2 -1 από τα

αντίγραφα είναι αποθηκευμένα σε κόμβους που το πρόθεμα τους είναι στο επίπεδο 0 σε σχέση με το κλειδί . Το pastry χρησιμοποιεί μια μέθοδο αυτομάθησης για να λύσει το πρόβλημα αυτό . Με την μέθοδο αυτή το Pastry ανακαλύπτει τον χρόνο που ένα μήνυμα προσεγγίζει την ομάδα των k κόμβων και βρίσκει το πλησιέστερο αντίγραφο . Με την μέθοδο αυτή το Pastry δρομολογεί προς το πλησιέστερο αντίγραφο στο 75% των περιπτώσεων και σε 1 από τα 2 πλησιέστερα στο 91% των περιπτώσεων .

3.4.2.5 Κατάρρευση κόμβων και διαμοιρασμένα δίκτυα

Δυο προβλήματα που αντιμετωπίζει το Pastry είναι τα παρακάτω :

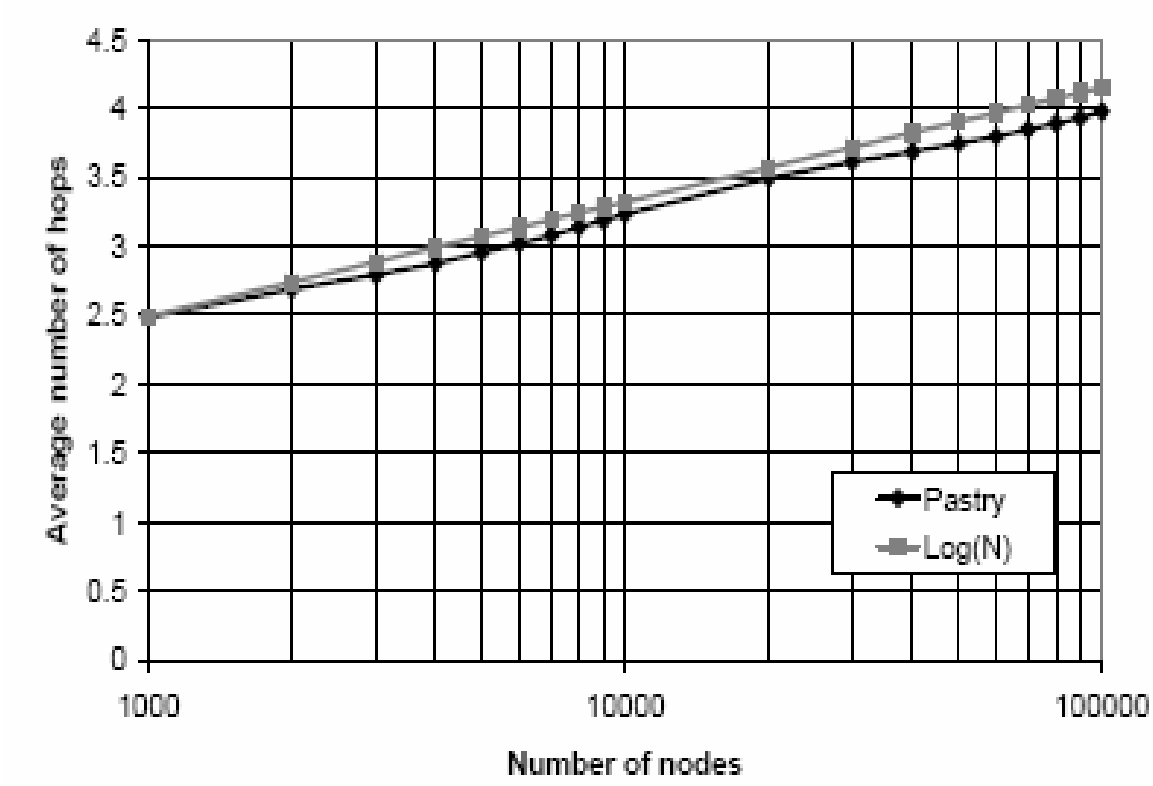
Πρώτα η περίπτωση κόμβων που δεν έχουν καταρρεύσει και ανταποκρίνονται στα μηνύματα αλλά με τρόπο λανθασμένο . Επειδή η δρομολόγηση του Pastry γίνεται με σαφώς καθορισμένο τρόπο , σε περίπτωση τέτοιων κόμβων , επαναλαμβανόμενες ερωτήσεις θα περνάνε από την ίδια διαδρομή και συνεπώς δεν θα βρίσκουν τον κόμβο στόχο . Το πρόβλημα αυτό μπορεί να αντιμετωπιστεί αν στη διαδικασία της δρομολόγησης όταν επιλέγεται ο κόμβος του επόμενου βήματος , αυτό να γίνεται τυχαία και όχι ντετερμινιστικά ανάμεσα στους κόμβους που το nodeID του έχει τουλάχιστον n+1 κοινά ψηφία στο πρόθεμα του με το κλειδί (όπου n είναι το πλήθος των κοινών στοιχείων στο πρόθεμα του nodeID του κόμβου με το κλειδί) .

Ένα άλλο πρόβλημα είναι ανωμαλίες στην IP δρομολόγηση στο Internet που επιτρέπουν σε κάποιους IP κόμβους να είναι προσβάσιμοι από κάποιους αλλά όχι από κάποιους άλλους . Μια λύση είναι το περιοδικό σε τυχαία διαστήματα ψάξιμο μέσω του IP multicast των κόμβων του Pastry για

άλλους στην γειτονιά τους . Το ψάξιμο αυτό γίνεται σε χρόνους που δεν γίνεται κάποια άλλη διεργασία στο δίκτυο .

3.4.3 Πειραματικά αποτελέσματα

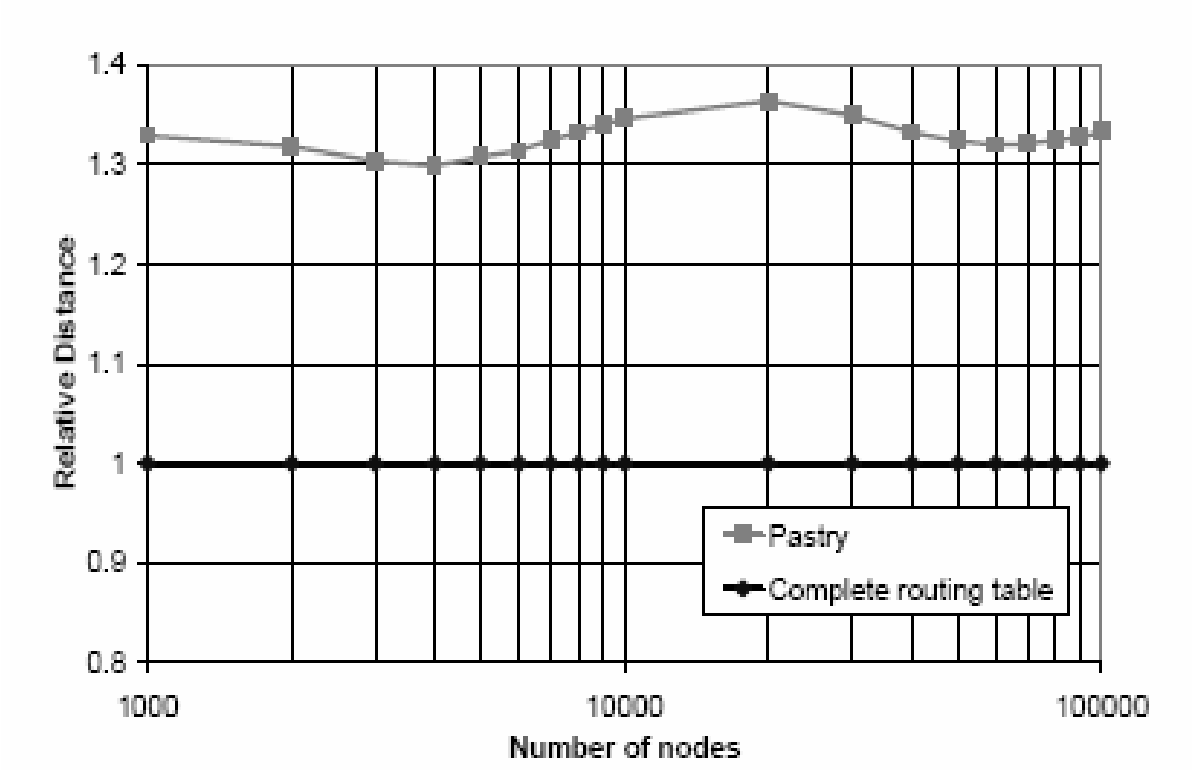
Αποτελέσματα προσομοίωσης με μεγάλο πλήθος κόμβων (100.000) έδειξαν την συμφωνία των θεωρητικών με τα πειραματικά αποτελέσματα . Η παρακάτω εικόνα δείχνει το πλήθος βημάτων σε τυχαίες δρομολογήσεις , ανάλογα με το πλήθος των κόμβων του δικτυού και το συγκρίνει με τα θεωρητικά αποτελέσματα ($\log_2^b N$ βήματα) όπου λήφθηκε b=4 ,L=16 ,M=32 .



Εικόνα 34

Φαίνεται πόσα λίγα βήματα απαιτούνται (περίπου 4) ακόμα και για ένα δίκτυο με 100.000 κόμβους .

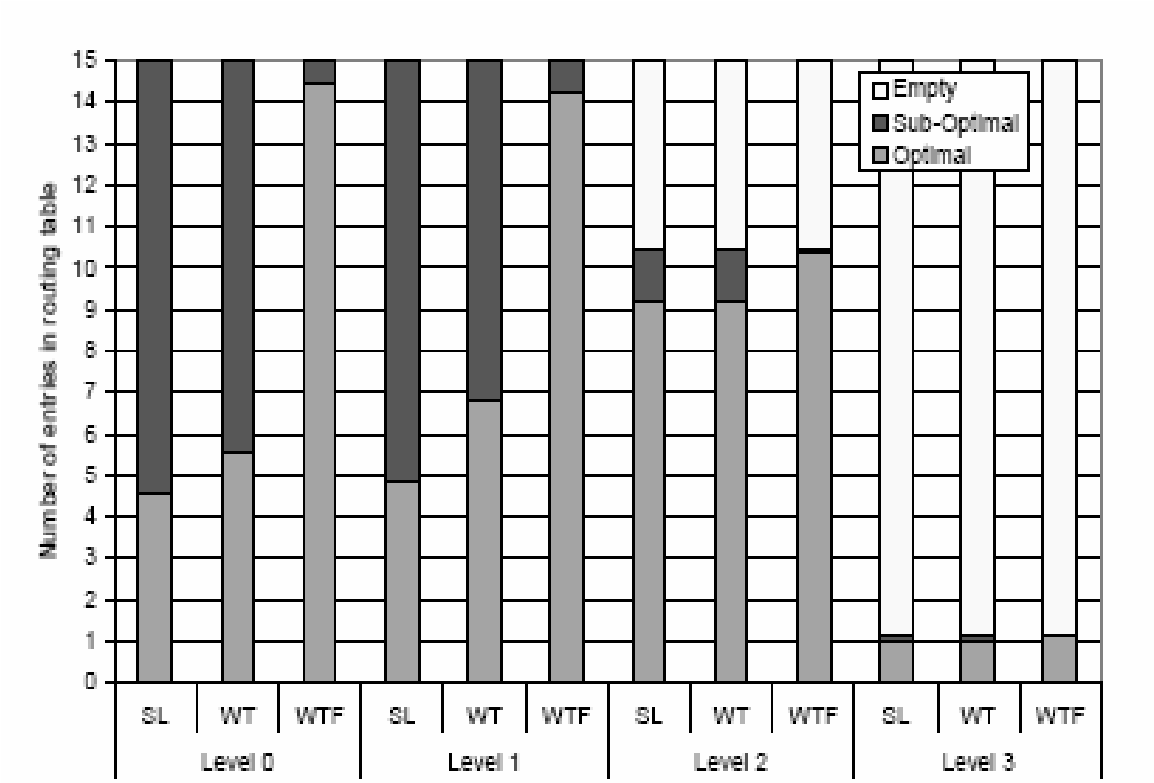
Στην παρακάτω εικόνα φαίνεται η σχέση της απόστασης του μονοπατιού που ακολουθεί ένα μήνυμα στο δίκτυο σε σχέση με την πραγματική άμεση απόσταση των κόμβων πηγής και προορισμού .



Εικόνα 35

Φαίνεται εδώ η καλή τοπικότητα της διαδρομής που εξασφαλίζει το Pastry , δεδομένου ότι η αύξηση είναι της τάξης του 30 με 40 % .

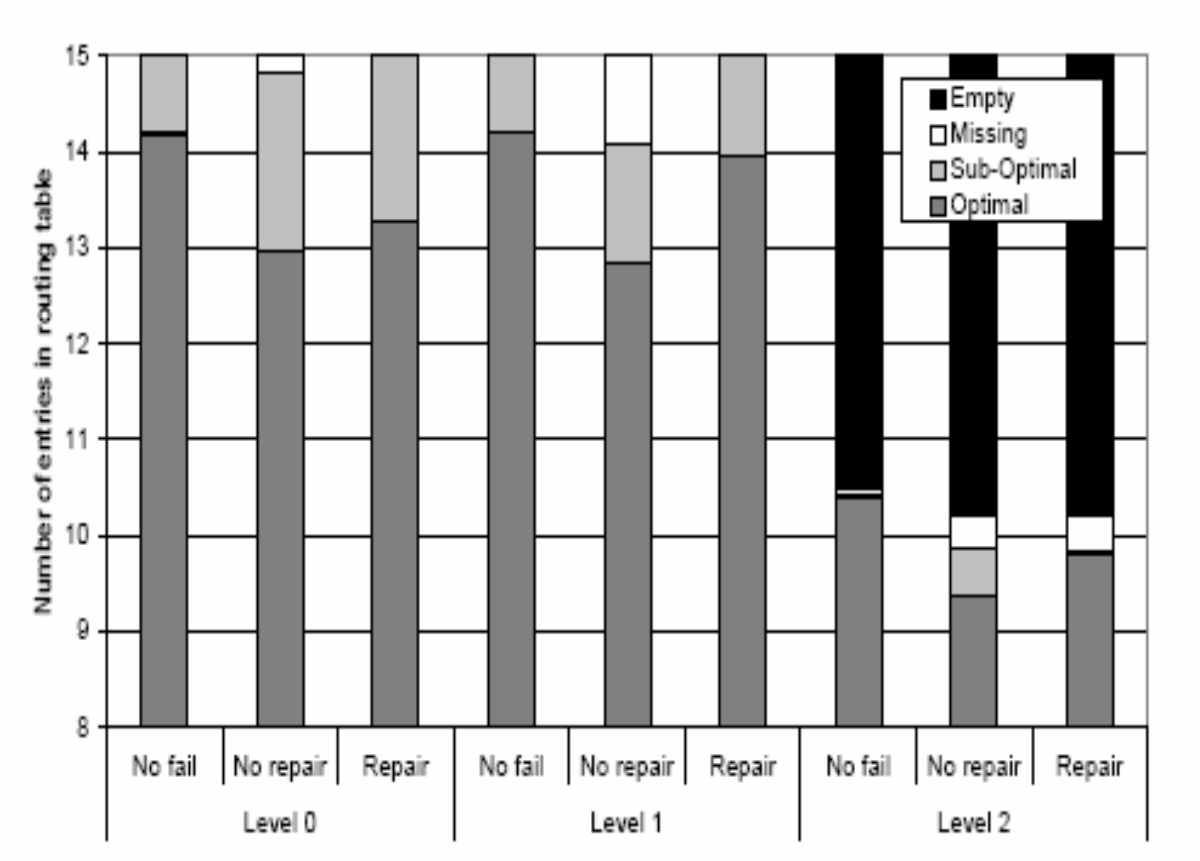
Άλλα πειράματα δείχνουν την καλή συμπεριφορά του Pastry στην είσοδο και την έξοδο κόμβων από το δίκτυο , όπως φαίνεται και στις παρακάτω εικόνες .



Εικόνα 35

Στο πείραμα στην παραπάνω εικόνα εισήχθησαν διαδοχικά 5000 νέοι κόμβοι στο δίκτυο και εξετάστηκαν μετά οι πίνακες δρομολόγησης . Εφαρμόστηκαν 3 μέθοδοι . Στις 2 πρώτες (SL και WT) εφαρμόστηκαν απλοποιημένοι τρόποι για την είσοδο των νέων κόμβων ενώ στην τρίτη (WTF) ο πραγματικός τρόπος που το Pastry εισάγει νέους κόμβους . Με optimal χαρακτηρίζεται ο πίνακας που οι κόμβοι που περιλαμβάνει είναι οι

πλησιέστεροι σε σχέση με το μέτρο εγγύτητας . Φαίνεται η πολύ καλή συμπεριφορά του WTF στην εισαγωγή κόμβων .



Εικόνα 36

Στο πείραμα στην παραπάνω εικόνα σε ένα δίκτυο 5000 κόμβων αποχώρησαν σιωπηρά 500 κόμβοι και μετά από 2 τυχαία εκλεγμένους κόμβους στάλθηκε μήνυμα σε τυχαίο κλειδί 100.000 φορές και εξετάστηκαν μετά οι πινάκες δρομολόγησης . Αυτό έγινε 3 φορές πριν την αποχώρηση των κόμβων , μετά την αποχώρηση άλλα με απενεργοποιημένη την δυνατότητα επιδιόρθωσης των πινάκων και τέλος με ενεργοποιημένη την δυνατότητα αυτή .

3.4.4 Συμπεράσματα

Γενικά το Pastry μαζί με τα Tapestry , Chord και Can ανήκουν στη δεύτερη γενιά peer –to–peer πρωτοκόλλων , που εξασφαλίζουν ένα περιορισμένο πλήθος βημάτων στη δρομολόγηση και δυνατότητα μεγάλης επεκτασιμότητας και αυτοοργάνωσης . Το Pastry και το Tapestry εκμεταλλεύονται τις πληροφορίες τοπικότητας που φυλάνε ώστε να εξασφαλίσουν την μικρότερη δυνατή απόσταση δρομολόγησης του μηνύματος .

3.5 CAN (Content Addressable Network)

3.5.1 Εισαγωγή

Είναι ένα δομημένο «καθαρό» peer to peer σύστημα . Ο Όρος Content Addresable – Network (Can) είναι γενικός και χρησιμοποιείται για την περιγραφή ενός κατανεμημένου συστήματος hash πίνακα . Αυτός είναι ο ορισμός που δίνουν οι Ratnasamy et al. Στην περιγραφή του συστήματος αυτού . Οι Ratnasamy et al. προτείνουν μια συγκεκριμένη σχεδίαση ενός CAN συστήματος . Οι βασικές πράξεις σε ένα CAN είναι η εισαγωγή , η αναζήτηση , και η διαγραφή ζευγαριών κλειδί-δεδομένων . Κάθε κόμβος αποθηκεύει ένα κομμάτι του πίνακα hash το οποίο ονομάζεται «ζώνη» . Ένας κόμβος κρατά πληροφορία για τους κόμβους σε γειτονικές ζώνες . Οι αιτήσεις για τις βασικές πράξεις δρομολογούνται μέσω ενδιάμεσων κόμβων μέχρι να φτάσουν στον κόμβο του οποίου η ζώνη περιέχει το κλειδί .

Το CAN είναι ανθεκτικό στα λάθη καθώς οι κόμβοι μπορούν να δρομολογήσουν ανεξαρτήτως αν έχει πέσει κάποιος κόμβος . Επίσης οι κόμβοι διατηρούν μόνο ένα μικρό κομμάτι της κατάστασης του συστήματος , ανεξάρτητο του αριθμού των κόμβων του συστήματος . Δηλαδή κάθε κόμβος περιέχει ευρετήριο μόνο για μερικά κλειδιά και τους αντίστοιχους κόμβους σε σχέση με κεντρικά ευρετήρια όπου όλη η πληροφορία είναι σε ένα κόμβο .

Το CAN χαρακτηρίζεται ως κλιμακωμένο (scalable) δίκτυο . Έτσι ονομάζεται ένα δίκτυο που συμπεριφέρεται καλά όταν ο αριθμός των κόμβων και / ή των μηνυμάτων αυξάνεται .

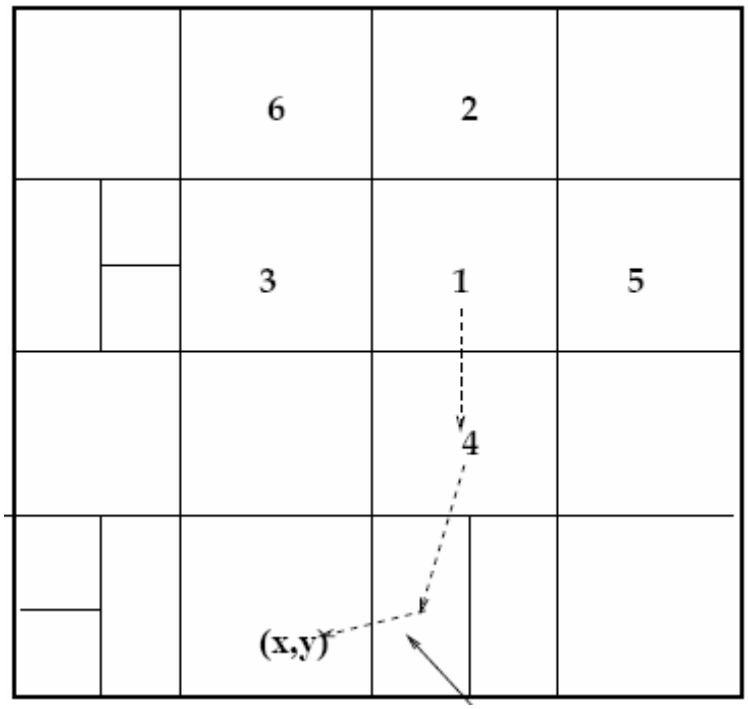
3.5.2 Σχεδιασμός

Το σύστημα χρησιμοποιεί ένα ιδεατό d-διαστάσεων καρτεσιανό χώρο συντεταγμένων (Π. χ. για διαστάσεις είναι το καρτεσιανό επίπεδο) που αντιστοιχεί σε ένα d-torus (ένα πλέγμα δηλαδή που έχει την ιδιαιτερότητα τα άκρα του να ενώνονται) για να αποθηκεύσει το ζεύγος (κλειδί K , τιμή V) .Ο χώρος αυτός τμηματοποιείται δυναμικά έτσι ώστε κάθε κόμβος να έχει μια ζώνη (Αργότερα θα δούμε ότι σε μια ζώνη μπορεί να περιέχονται πολλοί κόμβοι) . Αρχικώς το K αντιστοιχίζεται με ντετερμινιστικό τρόπο σε ένα σημείο P του χώρου . Το ζεύγος (K,V) αποθηκεύεται στο κόμβο που έχει τη ζώνη στην οποία βρίσκεται το P . Για να ανακτηθεί η εγγραφή που συσχετίζεται με το K , οποιοσδήποτε κόμβος μπορεί να εφαρμόσει την ίδια ντετερμινιστική διαδικασία για να αντιστοιχίσει K στο P και μετά να ανακτήσει την αντίστοιχη τιμή V από το P . Αν το P δεν το έχει ο κόμβος που κάνει την αίτηση , η αίτηση πρέπει να δρομολογηθεί από κόμβο σε κόμβο μέχρι να φτάσει το κόμβο , όπου στην ζώνη του βρίσκεται το P.

Οι κόμβοι μαθαίνουν και διατηρούν τις IP διευθύνσεις των κόμβων που διατηρούν κόμβους με ίδιες συντεταγμένες που είναι γειτονικοί στους δικούς τους σε ένα πίνακα δρομολόγησης που επιτρέπει δρομολόγηση μεταξύ τυχαίων σημείων στο χώρο .

3.5.2.1 Δρομολόγηση

Διαισθητικά η δρομολόγηση στο CAN γίνεται ακλουθώντας το μονοπάτι ευθείας γραμμής μέσα στο Καρτεσιανό χώρο από την πηγή στις συντεταγμένες του προορισμού . Κάθε κόμβος κρατά ένα πίνακα δρομολόγησης που έχει πληροφορίες για τους γείτονες του στο χώρο (την διεύθυνση IP και την ζώνη που ανήκει κάθε κόμβος) . Οι γείτονες κάθε κόμβου είναι οι κόμβοι των οποίων οι συντεταγμένες επικαλύπτονται σε d-1 διαστάσεις και στην άλλη διάσταση συνορεύουν . Στην εικόνα παρακάτω ο κόμβος 1 έχει για γείτονες τους 2,3,4,5 . Ένα μήνυμα περιέχει τις συντεταγμένες σε σχέση με τις συντεταγμένες του προορισμού . Στο σχήμα βλέπουμε ένα μονοπάτι δρομολόγησης από τον κόμβο 1 στον (x,y) .



Εικόνα 37

Το μέσο μήκος μονοπατιού δρομολόγησης για χώρο d διαστάσεων χωρισμένο σε n ίσες ζώνες είναι (d/4) ($n^{1/d}$) και κάθε κόμβος έχει 2d γείτονες .

3.5.2.2 Κατασκευή CAN

Οι νέοι κόμβοι που εισέρχονται στο σύστημα τους ανατίθεται ένα κομμάτι του χώρου χωρίζοντας την ζώνη ενός υπάρχοντος κόμβου στο μισό . Τα βήματα που ακολουθούνται είναι τα εξής :

1. Ο νέος κόμβος αναγνωρίζει ένα ήδη υπάρχον κόμβο στο CAN χρησιμοποιώντας κάποιον μηχανισμό αρχικοποίησης (bootstrap mechanism) .
2. Χρησιμοποιώντας το μηχανισμό δρομολογήσεις του CAN , επιλέγει τυχαία ένα σημείο P του χώρου και στέλνει μια αίτηση JOIN στο κόμβο που ανήκει στη ζώνη που περιέχει το P . Η ζώνη θα διαχωριστεί και η μισή θα ανατεθεί στο νέο κόμβο .
3. Ο νέος κόμβος μαθαίνει τις διευθύνσεις IP των γειτόνων του , και οι γείτονες της διαχωρισθείσας ζώνης ειδοποιούνται ώστε η δρομολόγηση να παραλάβει το νέο κόμβο .

Μηχανισμός Αρχικοποίησης : Ένας κόμβος ανακαλύπτει τις IP διευθύνσεις οποιουδήποτε κόμβου του συστήματος . Ο κόμβος αρχικοποίησης κρατά μια λίστα από κόμβους που πιστεύει ότι βρίσκονται εκείνη την ώρα στο σύστημα . Για να μπει στο CAN πρέπει να κοιτάξει στο DNS το domain name για να ανακτήσει την IP του κόμβου αρχικοποίησης . Ο κόμβος αρχικοποίησης παρέχει τις IP διευθύνσεις αρκετών τυχαίων κόμβων .

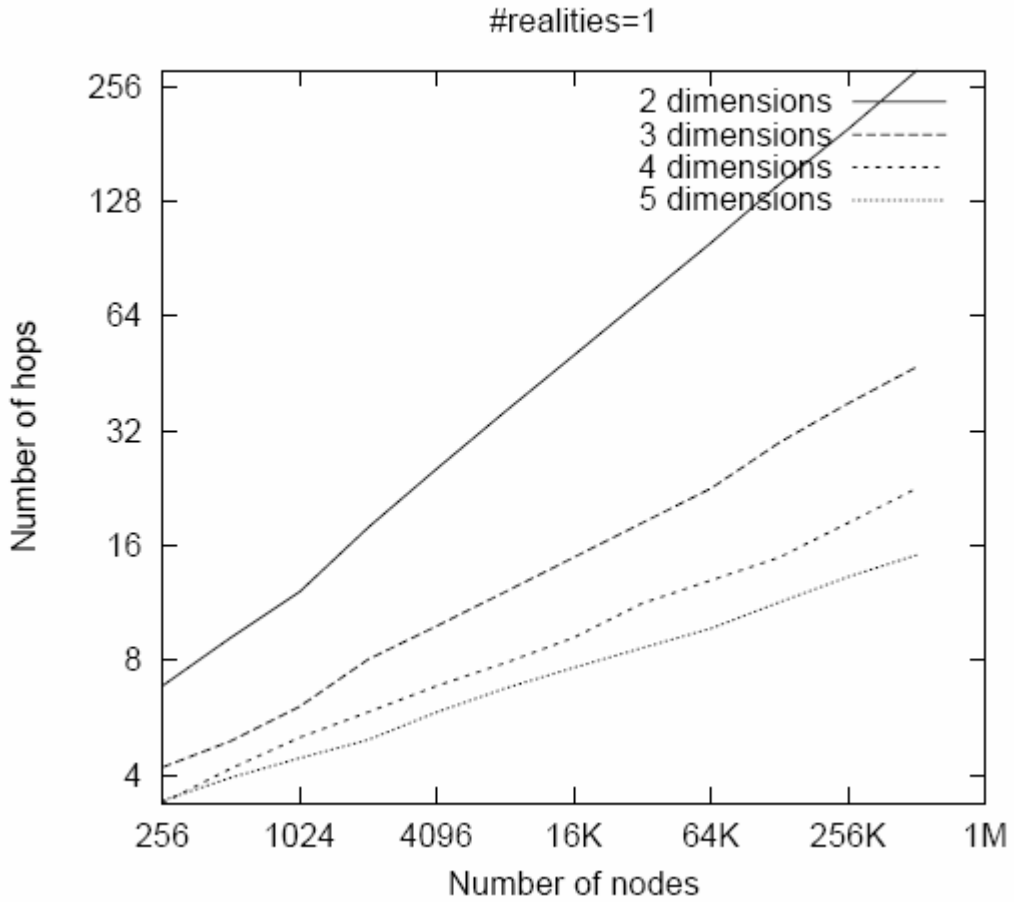
3.5.2.3 Αποχώρηση κόμβου

Όταν οι κόμβοι εγκαταλείπουν το CAN , οι ζώνες που κατείχαν και οι σχετικές εγγραφές του hash πίνακα μετατίθενται σε έναν από τους γείτονες . Σε κανονικές συνθήκες ένας κόμβος στέλνει περιοδικά μηνύματα συγχρονισμού σε καθένα από τους γείτονες του δίνοντας τις συντεταγμένες της ζώνης του , μια λίστα από γείτονες και τις συντεταγμένες τους . Αν υπάρχει μια παρατεινόμενη απουσία ενός μηνύματος συγχρονισμού , ο γειτονικός κόμβος αντιλαμβάνεται ότι υπάρχει αποτυχία , και εισάγει ένα ελεγχόμενο μηχανισμό επίλυσης του προβλήματος (controlled takeover mechanism) . Αν πολλοί από γειτονικούς κόμβους αποτύχουν , ένας επεκτεινόμενος μηχανισμός ψαξίματος σε δακτύλιο (ring search mechanism) εισάγεται από έναν από τους γειτονικούς κόμβους , για να βρει έναν οποιοδήποτε λειτουργικό κόμβο εκτός της αποτυχημένης περιοχής .

3.5.3 Σχεδιαστικές Βελτιώσεις

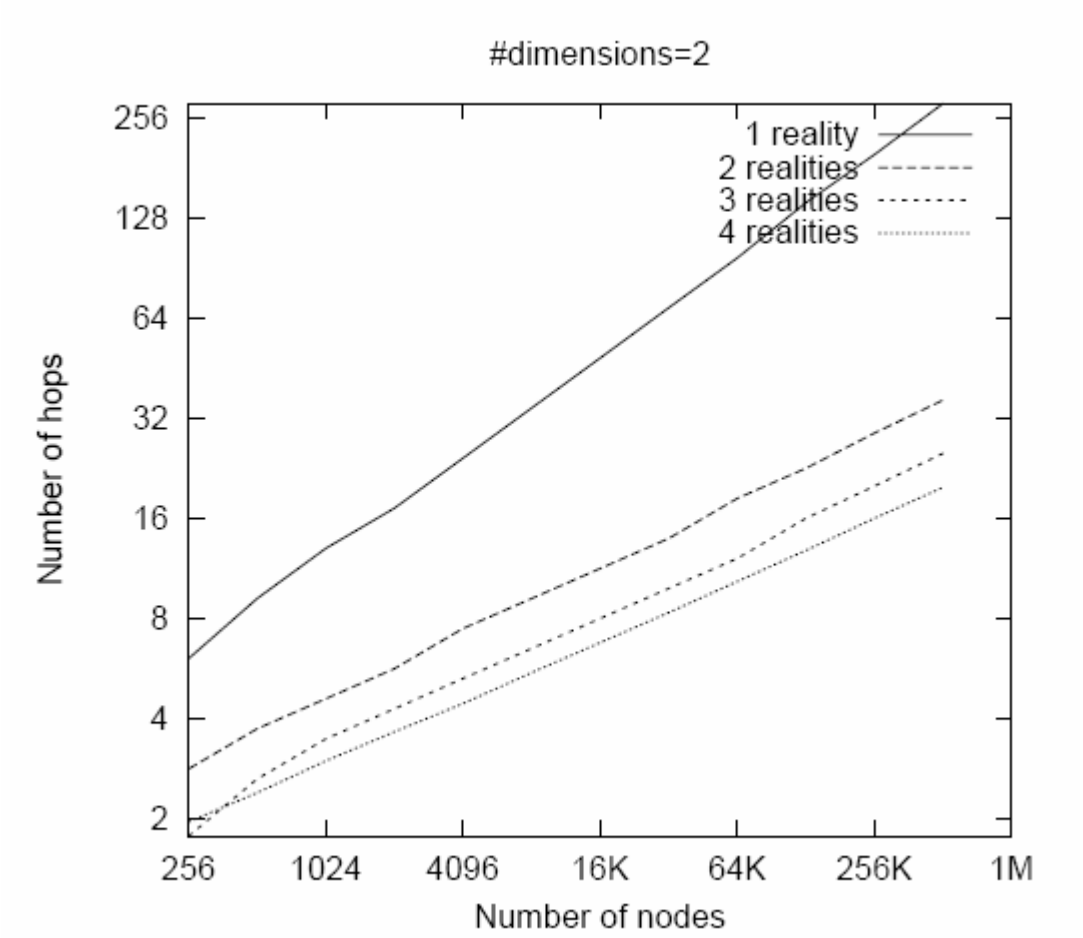
Ο αριθμός των βημάτων που αναφέραμε πιο πάνω είναι για το μονοπάτι στο CAN . Αυτά ονομάζονται βήματα επιπέδου εφαρμογής . Και διαφέρουν από τα βήματα του επιπέδου IP . Η καθυστέρηση κάθε βήματος διαφέρει από τα βήματα του επιπέδου IP . Η καθυστέρηση κάθε βήματος διαφέρει διότι μπορεί δυο κόμβοι να είναι γείτονες στο CAN άλλα να είναι σε εντελώς διαφορετικές γεωγραφικές περιοχές (δηλαδή πολλά βήματα IP) . Οπότε η μέση συνολική καθυστέρηση μιας αναζήτησης είναι το γινόμενο του μέσου αριθμού βημάτων CAN επί την μέση καθυστέρηση κάθε βήματος . Εκτός του βασικού σχεδιασμού υπάρχουν και μερικές βελτιώσεις που περιγράφονται παρακάτω και έχουν ως στόχο την μείωση της συνολικής καθυστέρησης της δρομολόγησης .

- **Χρήση πολύ-διαστατών συντεταγμένων χώρου**



***Εικόνα 38* :** Επίδραση των διαστάσεων στο μήκος του μονοπατιού.

- **Χρήση πολλαπλών χώρων συντεταγμένων (realities)**

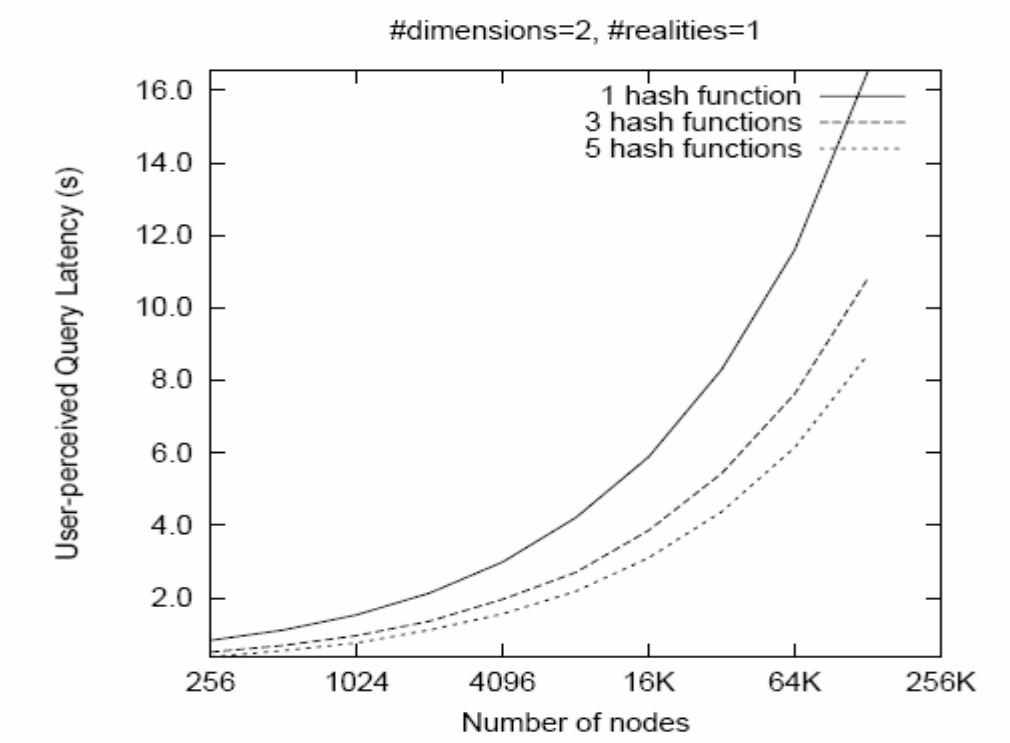


Εικόνα 39 **:** Επιδραση των πραγματικοτήτων στο μήκος του μονοπατιού**.**

- **Καλύτερες μετρικές δρομολόγησης**

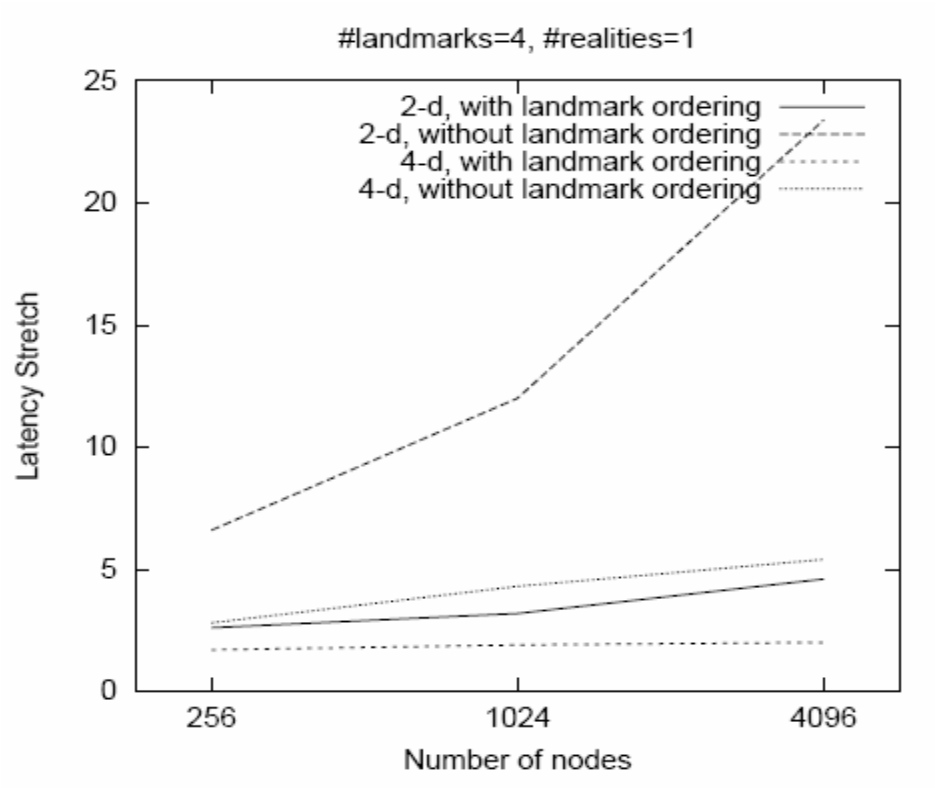
- **Ζώνες συντεταγμένων επιτρέποντας πολλαπλούς κόμβους να μοιραστούν την ίδια ζώνη**

- **Πολλαπλές hash συναρτήσεις**



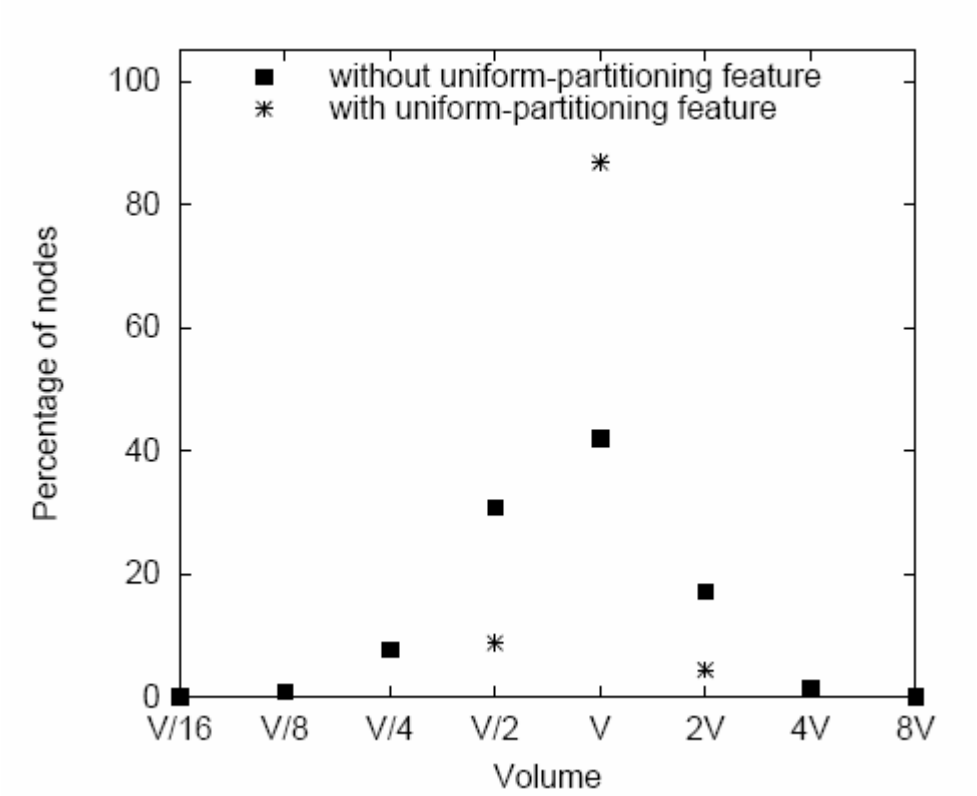
Εικόνα 40 : Μείωση της καθυστέρησης για ερωτήματα με πολλαπλές hash συναρτήσεις .

- **Ευαίσθητη τυπολογικά κατασκευή δικτύου**



Εικόνα 41

- **Περισσότερο ομοιόμορφος διαμερισμός**



Εικόνα 42

- **Τεχνικές διαχείρισης του «hot spot» προβλήματος**

3.5.4 Πειραματικά αποτελέσματα

Για να μετρηθεί η επίδραση όλων των ανώτερων σχεδιαστικών βελτιώσεων οι κατασκευαστές του CAN έτρεξαν πειραματικά σε ένα σύστημα με 2^{18} κόμβους τους 2 παρακάτω αλγόριθμους . Ο πρώτος είναι ένα σύστημα CAN χωρίς τις περισσότερες από τις σχεδιαστικές βελτιώσεις και ο δεύτερος με τις σχεδιαστικές βελτιώσεις . (Εκτός από αυτήν με τα ορόσημα) . Ο αριθμός των διατάσεων στον δεύτερο αλγόριθμο είναι 10 σε σχέση με 2

που είναι στο πρώτο . Στον παρακάτω πίνακα φαίνονται τα αποτελέσματα που περιέχονται στην εργασία .

Μετρική	CAN χωρίς βελτιώσεις	CAN με βελτιώσεις
Μήκος μονοπατιού	198	5
# γειτόνων [19]	4.57	27.1
# peers	0	2.95
IP καθυστέρηση (latency)	115,9 ms	82,4 ms
Καθυστέρηση μονοπατιού CAN	23008 ms	135,29

(Όπου # σημαίνει πληθικός αριθμός)

Όπως βλέπουμε το μεγαλύτερο όφελος από την αύξηση των διαστάσεων είναι η μείωση του μήκους του μονοπατιού από 198 σε 5 βήματα .

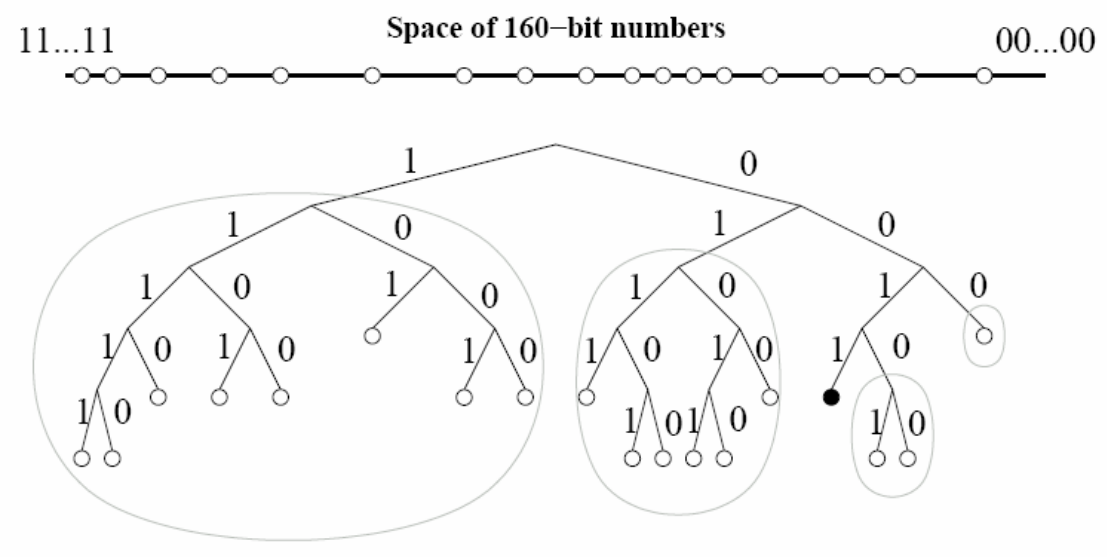
3.6 Kademlia

3.6.1 Εισαγωγή

Η βασική ιδέα του συστήματος είναι ένας DHT πίνακας που έχει καλή απόδοση σε περιβάλλοντα που είναι επιρρεπή σε λάθη . Η δρομολόγηση των ερωτήσεων γίνεται μέσω μιας τοπολογίας που βασίζεται στην μετρική XOR . Η τοπολογία έχει την ιδιότητα ότι κάθε κόμβος «μαθαίνει» από τις ερωτήσεις που δέχεται και δυναμώνει έτσι την διαδικασία λήψης απόφασης για δρομολόγηση . Ο αριθμός βημάτων είναι $\log_2 n$. Αν θεωρήσουμε τα IDs ότι έχουν μέγεθος b bits αντί για 1 bit , τότε αυξάνουμε το μέγεθος του πίνακα δρομολόγησης σε $2^b \log_2{}^b n$ k-buckets . Έτσι μειώνουμε τον αριθμό των βημάτων σε $\log_2{}^b n$.

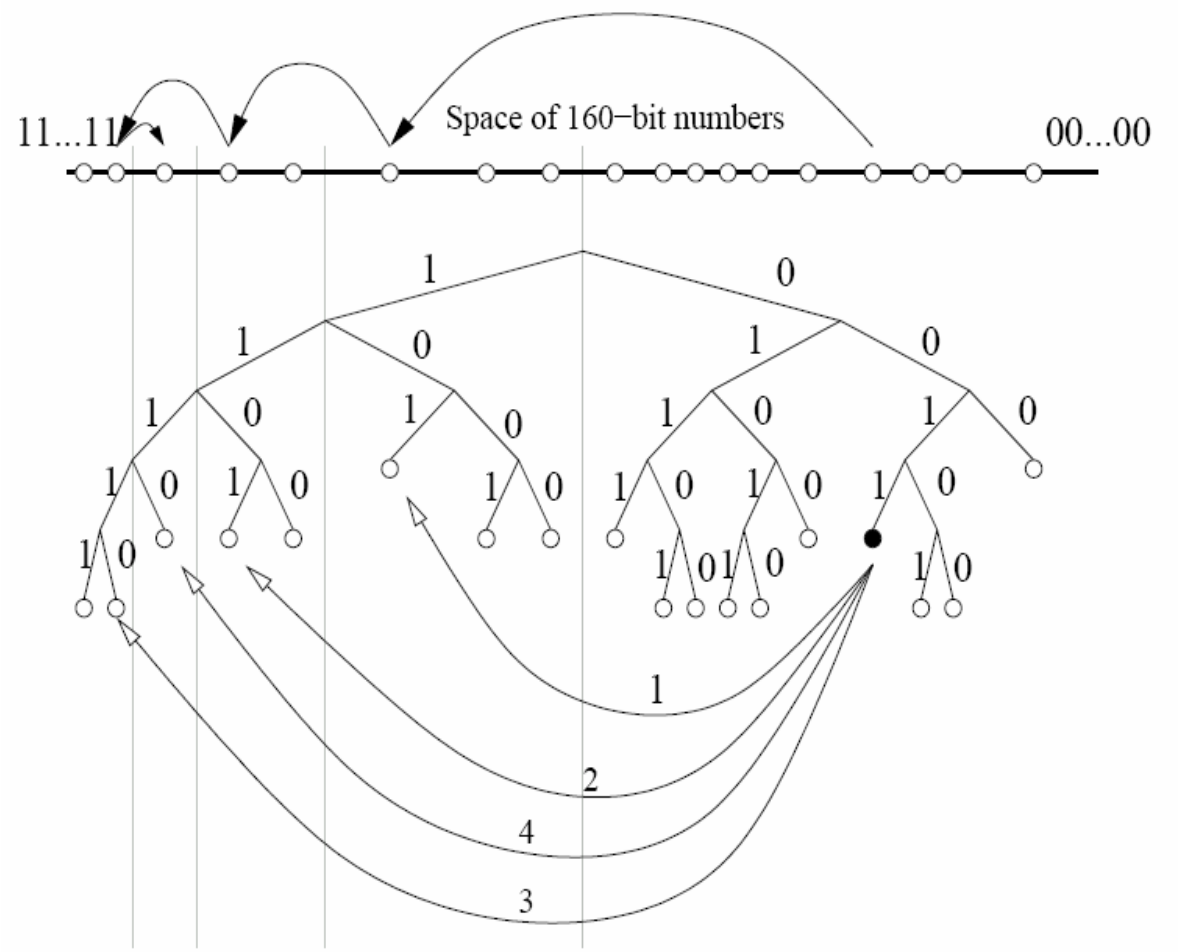
3.6.2 Περιγραφή του συστήματος

Το σύστημα αυτό ακολουθεί την φιλοσοφία των DHTs . Σε κάθε κόμβο ανατίθεται ένα ID των 160-bit . Οι κόμβοι είναι τα φύλλα ενός δυαδικού δέντρου . Η θέση ενός κόμβου ορίζεται από το μικρότερο μοναδικό πρόθεμα του αναγνωριστικού του . Για ένα κόμβο χωρίζουμε το δέντρο σε διαδοχικά υπόδεντρα που δεν περιέχουν το κόμβο . Αρχικά το υποδέντρο σε υψηλότερο επίπεδο περιέχει το μισό δυαδικό δέντρο που δεν περιέχει τον κόμβο . Μετά το επόμενο υποδέντρο περιέχει το μισό υπόδεντρο από το άλλο μισό δυαδικό δέντρο και δεν περιέχει το κόμβο και ούτω καθεξής .



***Εικόνα 43*:** Το δυαδικό δέντρο του Kademlia. Ο μαυρισμένος κόμβος είναι ο 0011…Τα υπόδεντρα σε κύκλο είναι εκείνα όπου ο κόμβος έχει ένα σύνδεσμο .

Σε αυτά τα υπόδεντρα το πρωτόκολλο του Kademlia μας εγγυάται ότι κάθε κόμβος γνωρίζει τουλάχιστον ένα κόμβο . Έτσι οποιοσδήποτε κόμβος μπορεί να βρει που βρίσκεται ένας άλλος κόμβος γνωρίζοντας το ID του . Στο σχήμα φαίνεται ο τρόπος που εντοπίζει ο κόμβος 0011 τον κόμβο 1110 ερωτώντας συνεχόμενα τον «καλύτερο» κόμβο που ξέρει μέχρι η αναζήτηση να συγκλίνει στον κόμβο στόχο .



Εικόνα 44 : Εδώ φαίνονται τα μηνύματα RPC που στέλνει ο 1110 . Πρώτα στέλνει στον 101 . Τα επακόλουθα μηνύματα είναι από κόμβους που επιστρέφονται από προηγούμενα RPCs **.**

3.6.2.1 Μετρική XOR

Κάθε μήνυμα που μεταδίδει ένας κόμβος περιλαμβάνει το ID του κόμβου ώστε να ξέρει ο παραλήπτης την ύπαρξη του κόμβου αποστολέα . Εκτός των κόμβων και τα κλειδιά έχουν IDs μεγέθους 160 bit . Για την ανάθεση των κλειδιών σε κόμβους το Kadmlia βασίζεται στην απόσταση μεταξύ 2 αναγνωριστικών . Η απόσταση αυτή ορίζεται ως το λογικό XOR (αποκλειστικό ή) σε ακέραια μορφή , $d(x,y) = x \oplus y$, όπου x,y τα 2 IDs . Σε ένα

δυαδικό δέντρο των 160-bit IDs , το μέγεθος της απόστασης μεταξύ δυο αναγνωριστικών είναι το ύψος του μικρότερου υποδέντρου που περιέχει και τα δυο . Όταν το δέντρο δεν είναι πλήρες το κοντινότερο φύλλο σε ένα ID x είναι το φύλλο που το ID του που μοιράζεται το μεγαλύτερο κοινό πρόθεμα με το x . Το XOR είναι μονοδιευθυντικό όπως ο κύκλος του Chord που πηγαίνει σύμφωνα με τους κύκλους του ρολογιού . Έτσι εγγυάται ότι όλες οι αναζητήσεις για το ίδιο κλειδί συγκλίνουν κατά μήκος του ίδιου μονοπατιού ανεξαρτήτως του κόμβου από τον οποίο ξεκίνησε . Η τοπολογία του XOR είναι συμμετρική , δηλαδή για κάθε x,y έχουμε d (x,y) = d (y,x) .

3.6.2.3 Πρωτόκολλο Kademlia

Περιλαμβάνει 4 RPC's: το **PING, STORE, FIND_NODE, FIND_VALUE**.

PING : Εξετάζει αν ένας κόμβος είναι online .

STORE : Αποθηκεύει ένα κλειδί και την τιμή του σε ένα κόμβο .

FIND_NODE : Παίρνει ως όρισμα το 160-bit ID.Ο αποδέκτης του RPC επιστρέφει τα πεδία [διεύθυνση IP ,UDP port , ID κόμβου] για τους k κόμβους που γνωρίζει να είναι πιο κοντά στο ID του κόμβου στόχου.

FIND_VALUE : Κάνει την ίδια λειτουργία με το FIND_NODE εκτός και αν ο δείκτης του RPC έχει λάβει ένα STORE RPC για το κλειδί , τότε επιστρέφει την τιμή του κλειδιού .

Σε όλα τα RPC ο δέκτης στέλνει πίσω ένα τυχαίο RPC ID για περισσότερη εγγύηση αποστολής του πρώτου RPC . Η αναζήτηση κόμβου είναι μια σημαντική λειτουργία του Kademlia καθώς έτσι βρίσκονται οι k πιο

κοντινοί κόμβοι ενός κόμβου . Χρησιμοποιείται ένας περιοδικός αλγόριθμος . Αρχικώς επιλέγονται α^{20} κόμβοι από το κοντινότερο k-bucket (αν δεν έχει , τότε επιλέγονται οι κοντινότεροι που γνωρίζει). Μετά στέλνονται παράλληλα ασύγχρονα μηνύματα FIND_NODE στους α κόμβους . Στο περιοδικό βήμα το FIND_NODE στέλνεται σε α κόμβους που έχει μάθει πριν από τα προηγούμενα RPCs και δεν τους έχει ρωτήσει ακόμη . Όταν οι κόμβοι δεν απαντούν τότε αφαιρούνται από την αναζήτηση . Η αναζήτηση τερματίζεται όταν ο κόμβος έχει ερωτήσει και πάρει απάντηση από τους k κοντινότερους κόμβους που έχει δει .

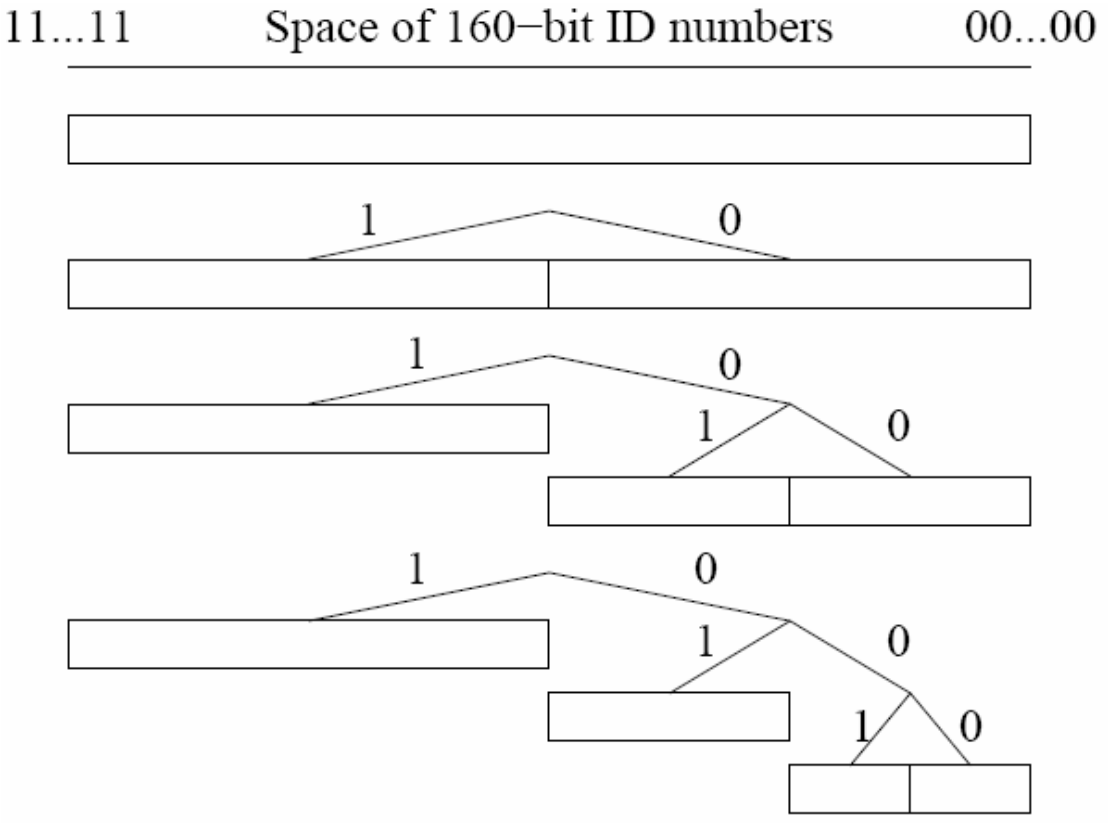
Όπως είπαμε η αναζήτηση είναι σημαντική διότι πολλές λειτουργίες γίνονται δράσης αυτής . Πρώτον για να αποθηκευτεί ένα κλειδί και η τιμή του , ένας κόμβος βρίσκει τους κοντινότερους k κόμβους του κλειδιού και τους στέλνει STORE RPCs . Επίσης η αναδημοσίευση του ζευγαριού κλειδί - τιμή που μας επιτρέπει να έχουμε την πιο πρόσφατη η πληροφορία στο δίκτυο γίνεται βάση της αναζήτησης . Για να βρεθεί ένα ζευγάρι του ίδιου τιμής ένας κόμβος ξεκινά την αναζήτηση για να βρει τους k κόμβους με το πιο κοντινό ID στο κλειδί . Χρησιμοποιείται η FIND_VALUE και μόλις κάποιος κόμβος επιστρέψει την τιμή η διαδικασία σταμάτα. Αν η αναζήτηση είναι επιτυχής ο κόμβος αποθηκεύει το κλειδί και την τιμή στο κοντινότερο κόμβο στο κλειδί που δεν επέστρεψε την τιμή . Αυτό βοήθα στο να γνωρίζουν οι επόμενοι κόμβοι που περίπου βρίσκεται το ζεύγος κλειδί – δεδομένα . Στα επόμενα ψαξίματα είναι πιθανό να πετύχουν αυτούς τους cashed κόμβους .

Η διαδικασία εισαγωγής ενός νέου κόμβου είναι η ακόλουθη .Ένας κόμβος u πρέπει να έχει ένα σύνδεσμο σε ένα κόμβο του δικτύου , έστω w . Ο u εισάγει τον w στο κατάλληλο k-bucket .Μετά ο u κάνει μια αναζήτηση κόμβου στο δικό του ID. Τέλος u ανανεώνει όλα τα k-buckets που είναι πιο μακριά από τον κοντινότερο γείτονα . Όταν λέμε ανανεώνει εννοούμε ότι

φτιάχνει τα δικά του k-buckets αλλά και εισάγεται και στα k-buckets των άλλων κόμβων όπου αυτό είναι αναγκαίο .

3.6.2.4 Πίνακας Δρομολόγησης

Ο πίνακας δρομολόγησης είναι ένα δυαδικό δέντρο όπου τα φύλλα του είναι k-buckets . Κάθε k-bucket περιέχει κόμβους με κοινό πρόθεμα των ID τους . Το πρόθεμα αυτό είναι και η θέση του bucket στο δέντρο . Έχουμε δυναμική ανάθεση στο δέντρο . Αρχικός το δέντρο ενός κόμβου u έχει ένα μοναδικό κόμβο . Μόλις u μάθει ένα νέο σύνδεσμο προσπαθεί να το εισάγει στο κατάλληλο bucket . Αν το bucket είναι ελεύθερο τότε εισάγεται χωρίς πρόβλημα . Αλλιώς αν το bucket περιέχει το ID του u , τότε το bucket χωρίζεται σε δυο νέα bucket και η προσπάθεια εισαγωγής επαναλαμβάνεται . Αν το bucket που δεν περιέχει το u είναι γεμάτο τότε ο νέος σύνδεσμος απορρίπτεται . Έτσι όμως δημιουργούνται μη ισορροπημένα δέντρα (Βλέπε εικόνα) . Μια λύση είναι να διατηρούμε πλήρη πληροφορία. Για το μικρότερο υποδέντρο με τουλάχιστον k κόμβους κοντά στο κόμβο .



Εικόνα 45 : Πίνακας δρομολόγησης για κόμβο με ID το 00..00. Η εξέλιξη του κατά την διάρκεια του χρόνου όπως περιγράφηκε .

3.6.2.5 Αναδημοσίευση κλειδιού

Οι κόμβοι πρέπει να αναδημοσιεύσουν τα κλειδιά ανά περιόδους για να εξασφαλίζεται η εγκυρότητα των ζευγαριών κλειδί – δεδομένα . Υπάρχουν δυο περιπτώσεις στις οποίες η αναζήτηση μπορεί να αποτύχει . Στην πρώτη περίπτωση , κάποιοι από τους κόμβους που έχουν κάποια κλειδιά μπορεί να φύγουν από το σύστημα . Για την αντιμετώπιση αυτού του προβλήματος το Kadmlia αναδημοσιεύσει τα κλειδιά κάθε μια ώρα . Κανονικά κάτι τέτοιο θα απαιτούσε κάθε κόμβος (το πολλοί k κόμβοι) που αποθηκεύει το κλειδί να κάνει μια αναζήτηση κόμβου και να στείλει k-1 STORE RPCs . Όμως το Kademlia ακολουθεί μια καλύτερη διαδικασία για να το βελτιώσει . Μόλις

ένας κόμβος λάβει ένα STORE RPC για ένα δεδομένο ζευγάρι κλειδί – τιμή , υποθέτει ότι το rpc το έχουν ακούσει και οι άλλοι k -1 κοντινότεροι κόμβοι και έτσι ο παραλήπτης δεν θα αναδημοσιεύσει το κλειδί στην επόμενη ώρα . Αυτό εγγυάται ότι αν τα διαστήματα αναδημοσίευσης δεν είναι ακριβώς συγχρονισμένα , μόνο ένας κόμβος θα αναδημοσιεύσει ένα δεδομένο ζεύγος κλειδιού – τιμής ανά ώρα . Στην δεύτερη περίπτωση νέοι κόμβοι μπορούν να εισέλθουν στο δίκτυο με IDs πιο κοντινά σε κάποιο ήδη δημοσιευμένο κλειδί .Οι ήδη υπάρχοντες κόμβοι εκμεταλλευόμενοι την πληροφορία από τα γειτονικά υπόδεντρα τους γνωρίζουν ποια κλειδιά θα αποθηκευτούν στο νέο κόμβο . Έτσι λοιπόν οποιοσδήποτε κόμβος αντιλήφθη την παρουσία νέου κόμβου στέλνει STORE RPC s για να μεταφερθούν τα κλειδιά στο νέο κόμβο . Για να μην στέλνονται όμως πολλά μηνύματα ένας κόμβος στέλνει μόνο αν το ID του είναι το πιο κοντινό στο κλειδί από τους άλλους ήδη υπάρχοντες κόμβους .

3.6.3 Δρομολόγηση

Θα αναφέρουμε στην αρχή κάποιους ορισμούς εννοιών που χρησιμοποιούνται . Για ένα k-bucket που καλύπτει το διάστημα [2i,2i+1) ορίζουμε το i ως index (ευρετήριο) του bucket . Ως depth (βάθος) ενός κόμβου h ορίζουμε το 160-i , όπου i είναι το μικρότερο ευρετήριο ενός μη-άδειου bucket .

Ορίζουμε ως το ύψος του bucket του κόμβου y στον κόμβο x να είναι το ευρετήριο του bucket στο όποιο x θα τοποθετούσε το y – (αριθμητικό μείον) το ευρετήριο του x λιγότερου σημαντικού άδειου bucket . Μετά θεωρείται ότι κάθε k – bucket κάθε κόμβου περιέχει τουλάχιστον ένα σύνδεσμο εάν υπάρχει ένας κόμβος στο κατάλληλο διάστημα . Παίρνοντας αυτό ως υπόθεση δείχνεται ότι η διαδικασία αναζήτησης είναι σωστή σε

λογαριθμικό χρόνο . Έστω ότι ο κοντινότερος κόμβος στο στόχο έχει βάθος h . Αν τα h πιο σημαντικά k-bucket είναι γεμάτα , τότε η αναζήτηση θα βρει ένα κόμβο (του οποίου η απόσταση είναι κατά 1 bit κοντύτερη) σε κάθε βήμα , και έτσι εμφανίζεται ο κόμβος σε h-logk βήματα . Αν ένα από τα k-bucket είναι άδειο τότε μπορεί να είναι η περίπτωση που ο κόμβος (στόχος) ανήκει στο διάστημα του άδειου Bucket . Η αναζήτηση θα προχωρήσει ακριβώς σαν να το bit του κλειδιού που αντιστοιχεί στο άδειο bucket να αντιστραφεί . Έτσι ο αλγόριθμος αναζήτησης πάντα θα επιστρέφει τον κοντινότερο κόμβο σε h-logk βήματα . Εφόσον ο κοντινότερος κόμβος βρεθεί ο αριθμός των βημάτων για να βρεθούν οι υπόλοιποι k-1 κοντινότεροι κόμβοι δεν μπορεί να είναι μεγαλύτερος από το ύψος του bucket του κοντινότερου κόμβου στον κ-οστό κοντινότερο κόμβο , το οποίο είναι απίθανο να είναι περισσότερο από log + c (μια σταθερά) .

Ο αριθμός των βημάτων που απαιτούνται για μια αναζήτηση είναι $\log_2 n$. Όμως αν θεωρήσουμε τα IDs b bits την φορά αντί για 1 bit την φορά , τότε αυξάνουμε το μέγεθος του πίνακα δρομολόγησης σε $2^b \log_2^b n$ k-buckets . Έτσι μπορούμε να μειώσουμε τον αριθμό των βημάτων σε $2^b \log_2^b n$. Όπως είπαμε προηγούμενος ένας κόμβος χωρίζει ένα k bucket όταν αυτό είναι γεμάτο και όταν περιέχει το ID του κόμβου . Η υλοποίηση όμως χωρίζει και buckets που δεν ανήκουν στην περιοχή τιμών ID του κόμβου ως b-1 επίπεδα . Δηλαδή αν b=2 το μισό του ID χώρου που δεν περιέχει τον κόμβο χωρίζεται μια φόρα (σε περιοχές τιμών) , αν b=3 τότε χωρίζεται σε 2 επίπεδα και μέγιστο 4 περιοχές . Άρα γενικά ένας κόμβος χωρίζει ένα γεμάτο bucket αν δεν περιέχεται το ID του και το βάθος d (Το βάθος (depth) είναι το μήκος του προθέματος που έχουν όλοι οι κόμβοι που ανήκουν στο σύνολο τιμών του bucket) του bucket ικανοποιεί το $d \neq 0 (\mod b)$. Η υλοποίηση χρησιμοποιεί b=5 .

3.7 Viceroy – Ένας πιθανοτικός αλγόριθμος

3.7.1 Εισαγωγή

Εκτός των ντετερμινιστικών αλγορίθμων που μελετήσαμε παραπάνω αλγορίθμους , είναι σημαντικό να αναφέρουμε και ένα πιθανοτικό (τυχαιοποιημένο) αλγόριθμο . Οι πιθανοτικοί αλγόριθμοι είναι συνήθως πιο γρήγοροι από τους κλασικούς ντετερμινιστικούς αλγορίθμους και μάλιστα πολλές φορές παρουσιάζουν καλύτερη χρονική πολυπλοκότητα και από τα αποδειγμένα κάτω φράγματα . Αυτό , αν και αρχικά ξενίζει , δεν είναι παράλογο καθότι με τη χρήση τυχαίων επιλογών έχουμε ένα πιο ισχυρό μοντέλο υπολογισμού (Πιο ισχυρό με την έννοια της ταχύτητας επίλυσης κι όχι με την έννοια ότι επιλύει κλάσεις προβλημάτων τα οποία είναι μη αποφασίσιμα στα ντετερμινιστικά μοντέλα υπολογισμού) . Ένα ακόμη πλεονέκτημα των πιθανοτικών αλγορίθμων είναι ότι συχνά είναι πιο εύκολοι στην περιγραφή και την κατανόηση από τους αντίστοιχους ντετερμινιστικούς . Το σύστημα που θα μελετήσουμε είναι το Viceroy . Είναι ένα δυναμικό σύστημα εξομοίωσης της πεταλούδας . Είναι σταθερού βαθμού και λογαριθμικής διαμέτρου δίκτυο με την επιπλέον ιδιότητα ότι η είσοδος ή η αποχώρηση ενός κόμβου από το δίκτυο δεν απαιτεί την ενημέρωση όλων των κόμβων του δικτύου .

Το Viceroy είναι ένα DHT σύστημα που επιτρέπει την ικανοποιητική τοποθέτηση και αναζήτηση αντικειμένων σε μεγάλα και δυναμικά μεταβαλλόμενα δίκτυα . Καλύπτει τις 2 βασικές απαιτήσεις των μεγάλων DHT συστημάτων . Πρώτα την ομοιόμορφη κατανομή των δεδομένων στους ενεργούς κόμβους , πράγμα που το κάνει με παρόμοιο τρόπο με το Chord , και δεύτερο την διατήρηση πληροφοριών δρομολόγησης μεταξύ των servers έτσι που η σύνδεση ή η αποχώρηση των servers από το δίκτυο να μην απαιτεί

την διακίνηση πληροφοριών σε όλο το δίκτυο . Το κόστος τέτοιων συνδέσεων και αποχωρήσεων είναι σταθερό , γεγονός πλεονεκτικό σε σχέση με άλλα δίκτυα .

Ο τρόπος που τοποθετούνται τα ζεύγη κλειδί – τιμή στους ενεργούς servers είναι ο ίδιος με του Chord , δηλαδή όλοι οι κόμβοι αντιστοιχούνται σε ένα αναγνωριστικό πάνω σε ένα μοναδιαίο δακτύλιο [0..10 . Κάθε κόμβος χειρίζεται όλα τα ζεύγη που τα αναγνωριστικά τους είναι ανάμεσα στον κόμβο αυτό και το κατά την φορά του ρολογιού γείτονα του . Χρησιμοποιεί δηλαδή τις έννοιες του successor και predecessor . Για την δρομολόγηση δανείζεται από τις ιδέες των Kleinberg και Barriere et al και κάθε κόμβος συνδέεται εκτός από τους γείτονες του στο δακτυλίδι και με ένα σταθερό αριθμό (5) από απομακρυσμένους συνδέσμους . Με βάση τις ιδέες του Kleinberg αυτοί οι απομακρυσμένοι σύνδεσμοι επιλέγονται τυχαία με μεγαλύτερη προτίμηση στους πλησιέστερους . Για την βελτίωση του δικτύου το Viceroy προσεγγίζει το κλασσικό δίκτυο butterfly . Έτσι το Viceroy είναι μια σύνθεση ανάμεσα στο δίκτυο butterfly και στον δακτύλιο τύπου Chord . Δηλαδή υπάρχει μια τοπολογία δακτυλιδιού , όπου ο κάθε κόμβος έχει ένα αναγνωριστικό στο πεδίο [0..1) και είναι συνδεδεμένος με τον αμέσως μικρότερο του και τον αμέσως μεγαλύτερο του στο πεδίο αυτό (και ο τελευταίος με τον 1^{o}) δηλαδή τους successor και predecessor . Επί πλέον κάθε κόμβος συνδέεται και με 5 απομακρυσμένους κόμβους ως εξής : Κάθε κόμβος επιλέγει τυχαία ένα επίπεδο , έτσι ώστε αν είναι ενεργοί n κόμβοι να δημιουργούνται log n επίπεδα . Ένας κόμβος j στο επίπεδο l συνδέεται με 2 κόμβους στο επίπεδο l + 1 , ο ένας λέγεται « κάτω δεξιά » και είναι σε απόσταση περίπου $1/2^{l}$ και ο άλλος « κάτω αριστερά» και είναι ο πλησιέστερος στον j στο επίπεδο l + 1 . Αν το l είναι μεγαλύτερο από 1 , ο j συνδέεται επίσης με τον πλησιέστερο του κόμβο του επίπεδου l -1 («πάνω») .

Τέλος συνδέονται με τον j άλλοι 2 κόμβοι στο επίπεδο l που είναι ο αμέσως μικρότερος του και ο αμέσως μεγαλύτερος του , δημιουργώντας δηλαδή ένα ακόμα δακτύλιο σε κάθε επίπεδο . Έτσι κάθε κόμβος συνδέεται με 7 συνολικά κόμβους με εξερχόμενες συνδέσεις . Το παρακάτω σχήμα δείχνει ένα τέτοιο ιδανικό δίκτυο , όπου για απλότητα δεν παρουσιάζονται οι πάνω συνδέσεις καθώς και οι σύνδεσμοι δακτυλίου , γενικού και επιπέδου .

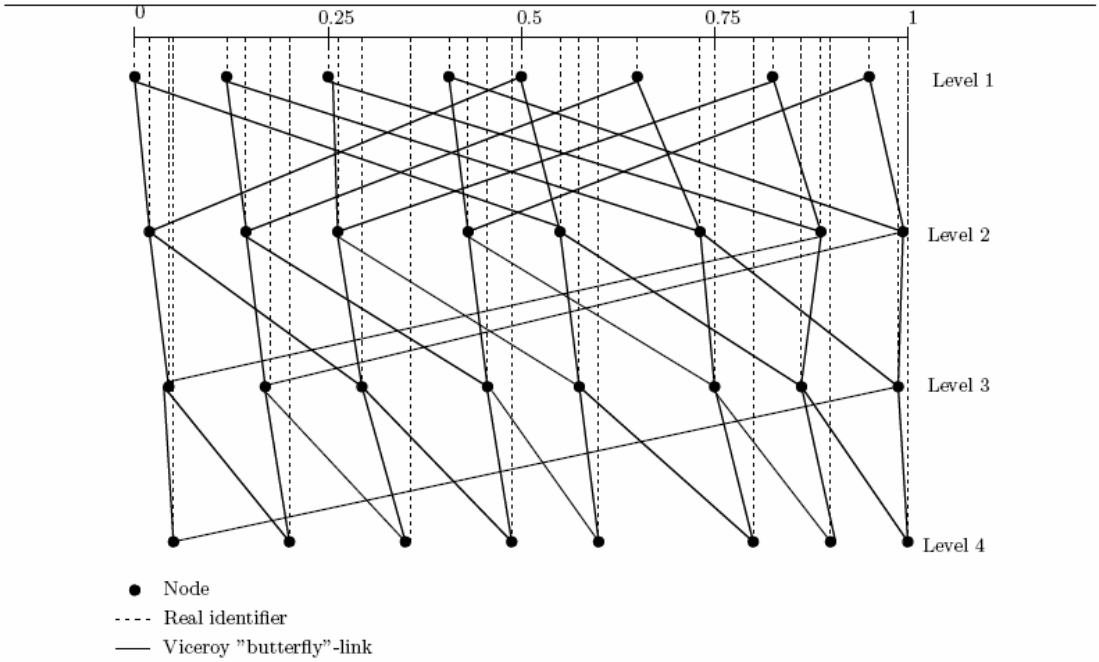


Figure 1: An ideal Viceroy network. Up and ring links are omitted for simplicity.

Εικόνα 46

Η δρομολόγηση γίνεται σε 3 στάδια . Στο 1^{ο} γίνεται ένα ανέβασμα χρησιμοποιώντας την «πάνω» σύνδεση . Μετά κατεβαίνει διαδοχικά σε χαμηλότερο επίπεδο χρησιμοποιώντας την «κάτω δεξιά» ή την «κάτω αριστερά» σύνδεση ανάλογα αν ο στόχος είναι σε απόσταση μεγαλύτερη από $1/2^{l}$ ή όχι . Τέλος (3^{ο} στάδιο) αν βρεθεί κόμβος χωρίς κάτω σύνδεση χρησιμοποιείται το ψάξιμο σε δακτύλιο ή στον γενικό ή σε επίπεδο level . Η δρομολόγηση αυτή εγγυάται O (log n) βήματα . Η είσοδος και η έξοδος ενός server στο σύστημα απαιτεί O (log n) και επηρεάζει την κατάσταση μόνο O (1) server .

3.7.2 Περιγραφή συστήματος

3.7.2.1 Ορισμοί – υποθέσεις

Όπως συνοπτικά περιγράψαμε παραπάνω το σύστημα συνίσταται από μια δυναμικά μεταβαλλόμενη ομάδα από servers οι οποίοι παίρνουν πριν την είσοδο τους στο δίκτυο έναν τυχαίο πραγματικό αναγνωριστικό αριθμό ομοιόμορφα κατανεμημένο στο πεδίο [0..1). Θεωρούμε την αριθμητική που διέπει όλα τα παρακάτω ως modulo 1.

Για κάθε τρεις πραγματικούς αριθμούς x,y,z ορίζουμε ως stretch(x,y) την περιοχή ανάμεσα στα x,y με διεύθυνση κατά την φορά του ρολογιού , των x,y μη συμπεριλαμβανομένων και z ϵ stretch(x,y) όταν το z είναι ανάμεσα στα x,y . Ορίζουμε ως απόσταση d(x,y) την απόσταση από το x στο y κατά την φορά του ρολογιού . Ορίζουμε επίσης ως πυκνότητα q(x,y) , το πλήθος των ενεργών servers στο stretch (x,y) .

Ορίζουμε ως SUCC(x) τον 1^{ο} ενεργό γείτονα του x κατά την φορά του ρολογιού . Όμοια ως PRED(x) τον 1^{ο} ενεργό γείτονα του x κατά την

αντίστροφη φορά του ρολογιού . Σε κάθε server s ορίζουμε ένα πρόσθετο ακέραιο αριθμό το level που συμβολίζεται ως s.level .Ως $NLEVEL_i(x)$ ορίζεται ο πλησιέστερος κατά την φορά του ρολογιού ενεργός γείτονας του x που ανήκει στο επίπεδο i .Αντίστοιχα ως $PRELEVEL_i(x)$ ορίζεται ο πλησιέστερος κατά την αντίστροφη φορά του ρολογιού ενεργός γείτονας του x που ανήκει στο επίπεδο i .Επίσης ορίζονται οι NEXTONLEVEL(x) και PREVONLEVEL(x) ως εξής : NEXTONLEVEL(x)= $NLEVEL_{x.level}(x)$ και PREVONLEVEL(x)= $PLEVEL_{x.level}(x)$. Τέλος ως $q_i(x,y)$ το πλήθος των servers του επιπέδου i που περιλαμβάνονται στο stretch(x,y).

Θεωρούμε ότι ένας server μπορεί να συνδεθεί με κάποιον άλλο αν ξέρει το αναγνωριστικό του .Θεωρούμε επίσης ότι ένας server μπορεί να ενωθεί ή να εγκαταλείψει ηθελημένα το σύστημα , αλλά υποθέτουμε για τους σκοπούς αυτής της ανάλυσης ,ότι ένας server δεν εγκαταλείπει αθέλητα το σύστημα και ότι δεν συμβαίνουν ταυτόχρονα πολλαπλές συνδέσεις και αποχωρήσεις servers . Στην πράξη το σύστημα μπορεί με επιτυχία να αντιμετωπίσει τέτοιες συνθήκες δεδομένου ότι οι servers που για τον λόγο αυτό αλλάζουν την κατάσταση τους με πολύ μεγάλη πιθανότητα δεν επικαλύπτονται .

3.7.2.2 Το δίκτυο του Viceroy

Κάθε server s χαρακτηρίζεται από 2 αριθμούς , το αναγνωριστικό του s.id ή απλά s , που παραμένει σταθερός καθ’ όλη την παραμονή του στο σύστημα και το επίπεδο του (level) l, ένα ακέραιο αριθμό ο οποίος μπορεί να αλλάζει κατά την ανάπτυξη του δικτύου . Όπως παραπάνω αναφέρθηκε το δίκτυο αποτελείται από 3 είδη συνδέσεων . Τον γενικευμένο δακτύλιο που κάθε κόμβος συνδέεται (δηλαδή γνωρίζει τα αναγνωριστικά) με τον successor και τον predecessor του , τον δακτύλιο του level του όπου κάθε κόμβος

συνδέεται με τους 2 πλησιέστερους (πριν και μετά) κόμβους που ανήκουν στο ίδιο level με αυτόν , σχηματίζοντας έτσι έναν δακτύλιο σε κάθε level και τους συνδέσμους τύπου butterfly . Με τους συνδέσμους αυτούς ένας κόμβος s που δεν κόμβος φύλλο , στο επίπεδο l συνδέεται με 2 κόμβους στο επίπεδο l+1 , ο ένας λέγεται «κάτω δεξιά» και είναι ο πλησιέστερος με την φορά του ρολογιού προς τον $s+1/2^l$ (δηλαδή $NLEVEL_{l+1}(s+1/2^l)$) και ο άλλος «κάτω αριστερά» και είναι ο πλησιέστερος με την φορά του ρολογιού στον s στο επίπεδο l+1 (δηλαδή $NLEVEL_{l+1}(s)$). Αν το l είναι μεγαλύτερο από 1 , ο s συνδέεται επίσης με τον πλησιέστερο του με την φορά του ρολογιού κόμβο του επιπέδου l -1 («πάνω»)(δηλαδή $NLEVEL_{l-1}(s)$).

3.7.2.2 Η επιλογή αναγνωριστικού και επιπέδου

Κάθε server s επιλέγει το αναγνωριστικό του ανεξάρτητα και ομοιόμορφα κατανεμημένα στο πεδίο [0..1) . Με τον ίδιο τρόπο θα μπορούσε να επιλεγεί και το επίπεδο του , ομοιόμορφα κατανεμημένα στο πεδίο ανάμεσα στο 1 και το log n , όπου n το πλήθος των ενεργών κόμβων , αν ήξερε το n . Επειδή όμως αυτό θα απαιτούσε την διακίνηση μεγάλου όγκου πληροφοριών στο δίκτυο κάθε φορά που αλλάζει το μέγεθος του δικτύου , υιοθετείται μια προσεγγιστική μέθοδος , όπου υπολογίζεται προσεγγιστικά το $n_o=1/d(s,SUCC(s))$ και μετά l τυχαία και ομοιόμορφα κατανεμημένο στο πεδίο [1.. log n_o] . Η τρόπος υπολογισμού του n_o είναι ευνόητος δεδομένου ότι σε ένα ιδανικό δίκτυο , όπου όλοι οι κόμβοι είναι ομοιόμορφα κατανεμημένοι , η απόσταση ανάμεσα σε δυο γειτονικούς κόμβους είναι 1/n. Με τον τρόπο αυτό ένας server χρειάζεται να αλλάξει το επίπεδο του μόνο όταν αλλάζει ο successor του .

Αποδεικνύονται ότι με την χρήση του παραπάνω αλγορίθμου και με μεγάλη πιθανότητα ισχύουν τα παρακάτω :

- $\log(n/(2\log n)) \leq [\log n_o]\ 3\log n$, Κάθε επίπεδο l που ισχύει $l < \log(n/(2\log n))$ λέγεται ‘συνετό’(‘sane’).
- Για κάθε server s : $q(s,s+(\log n)/n=O(\log n)$
- Για κάθε server s αναμένεται λογαριθμικός αριθμός βημάτων για την εύρεση του επόμενου του σε ένα επίπεδο και λογαριθμικός στο τετράγωνο στη χειρότερη περίπτωση . Δηλαδή αν $i \leq \log(n/(2\log n))$ τότε στη μέση περίπτωση $q(s,NLEVEL_j(s))=O(\log n)$ και στη χειρότερη $q(s, NLEVEL_j(s))=O(\log^2 n)$
- Για κάθε server s: $\min_{j \leq s \log(n/(2\log n))}\{q(s,NLEVEL_j(s))\}=O(\log n)$
- Για κάθε server s και $i \leq \log(n/(2\log n))$ αναμένεται ότι $E[q_i(s,s+(3\log n)/n]=1$ και με πιθανότητα τουλάχιστον ½ ότι $q_i(s,s+(3\log n)/n \geq 1$
- Για κάθε server s και $l,j \leq \log(n/(2\log n))$ αναμένεται ότι $E[q_i(s,NLEVEL_i(s))=O(\log n)$.

3.7.2.3 Η είσοδος και αποχώρηση κόμβων

Όπως έχουμε πει κάθε server s διατηρεί τις εξής πληροφορίες :

- Τον αριθμό s , και s.level
- Τους s.successor και τον s.predecessor (δείκτες συνδέσεων δακτυλίου)
- Τους s.nextonlevel και s.prevonlevel (δείκτες συνδέσεων δακτυλίου στο level)
- Τους s.left s. right και s.up (δείκτες συνδέσεων butterfly)

Με την είσοδο και έξοδο ενός κόμβου στο δίκτυο πρέπει να συμπληρωθούν οι αριθμοί αυτοί στον κόμβο s και να ενημερωθούν οι

επηρεαζόμενοι κόμβοι με τις αλλαγές των δικών τους αριθμών που επηρεάζονται.

Για την είσοδο ενός κόμβου ακολουθείται ο κατώτερο αλγόριθμος:

JOIN

1. Επιλέγεται ένα αναγνωριστικό s σύμφωνα με τον μηχανισμό του προηγούμενου κεφαλαίου .
2. Χρησιμοποιώντας την υπορουτίνα LOOKUP που περιγράφεται παρακάτω βρίσκεται ο SUCC(s) . Ενημερώνονται οι s.successor και s.predecessor του s και οι PRED(s).successor και τον SUCC(s).predecessor των αντιστοίχων κόμβων .
3. Μεταφέρονται από τον s.successor στον s όλα τα ζεύγη κλειδί-δεδομένα που τα κλειδιά τους μεταξύ του s.predecessor και s.
4. Επιλέγεται το s.level σύμφωνα με τον μηχανισμό του προηγούμενου κεφαλαίου. Βρίσκονται (με απλό βηματισμό) τα s'=NEXTONLEVEL(s) και s''=PREVONLEVEL(s) και ενημερώνονται οι αντίστοιχοι αριθμοί ήτοι : s.nextnlevel και s.prevonlevel του s και s''.nextonnlevel και s'.prevonlevel των αντιστοιχών κόμβων .
5. Βρίσκεται (με απλό βηματισμό) το $NLEVEL_{(s:level+1)}(s)$ και αντιστοιχίζεται στο s.left. Βρίσκεται (με το LOOKUP) ο πλησιέστερος κόμβος με την φορά του ρολογιού στον $(s+1/2^i)$ (όπου i=s.level) έστω s' και (με απλό βηματισμό) ο επόμενος κατά την φορά του ρολογιού σε αυτόν , στο επίπεδο s.level δηλαδή ο $NLEVEL_{s:level+1}(s')$ που αντιστοιχίζεται με τον s.up.

Σε περίπτωση αλλαγής του level ενός κόμβου s ακολουθούνται τα 4,5 βήματα του πιο πάνω αλγόριθμου .

Για την έξοδο ενός κόμβου ακολουθείται ο κατωτέρω αλγόριθμος :

LEAVE

Ειδοποιεί όλους τους servers που τον χρησιμοποιούν ως σύνδεση ώστε να τον αντικαταστήσουν (χρησιμοποιώντας το LOOKUP) .Μεταφέρει όλα τα ζεύγη κλειδί-δεδομένα που φυλάει στον successor του.

3.7.2.4 Η απλή αναζήτηση

Στον απλό αυτό αλγόριθμο χρησιμοποιούνται μόνο οι successor και predecessor και οι σύνδεσμοι butterfly .Στον αλγόριθμο αυτό LOOKUP(x,y) σκοπός είναι ξεκινώντας από τον server y να βρεθεί ο πλησιέστερος με την φορά του ρολογιού προς την τιμή x .Γίνεται σε τρεις φάσεις . Στην πρώτη βρίσκεται η ‘ριζα’ δηλαδή ένας server του προηγουμένου επιπέδου χρησιμοποιώντας την up σύνδεση . Στην δεύτερη ξεκινώντας από τη ρίζα σε κάθε επίπεδο βρίσκεται ο εγγύτερος προς τον x συνδεδεμένος server στο επίπεδο αυτό (έστω επίπεδο i) , ο οποίος πάντα βρίσκεται σε απόσταση το πολύ $1/2^{i-1}$ από τον στόχο . Η φάση αυτή σταματάει όταν φτάσει σε ένα κόμβο χωρίς κάτω συνδέσεις ή σε κόμβο που υπερβαίνει την τιμή του x . Στην τρίτη φάση από τον τελευταίο κόμβο βρίσκεται ο στόχος με απλό βηματισμό κατά την φορά του ρολογιού ή με την αντίθετη ανάλογα με την θέση του τελευταίου κόμβου . Παρακάτω περιγράφεται ο αλγόριθμος .

LOOKUP(x,y)

Cur = y

Proceed to root:

If cur.level = 1

goto traverse-tree

Else

Αν υπάρχει το cur.up

Cur=cur.up

Else

Cur=cur.successor

Goto Proceed to root

Traverse-tree :

If $d(cur,x<1/2^{cur.level}$ then

Αν υπάρχει το cur.left then cur=cur.left

If $d(cur,x)\geq 1/2^{cur.level}$ then

Αν υπάρχει το cur.right then cur=cur. Right

Αν δεν υπάρχει ο απαιτούμενος σύνδεσμος ή υπάρχει και είναι Μεγαλύτερος του x then

Goto Traverse-ring (τερματίζει την φάση αυτή και πάει στην επόμενη)

Else

Goto Traverse-tree (επαναλαμβάνει την φάση)

Traverse-tree:

Αν ο cur είναι ο πλησιέστερος κατά την φορά του ρολογιού προς τον x then

Επιστρέφει τον cur και τερματίζει

Else

Cur=cur.successor ή cur.predecessor ανάλογα με το ποιος είναι πλησιέστερα στο x

Goto Traverse-ring (επαναλαμβάνει την φάση)

Αναλύοντας τον παραπάνω αλγόριθμο προκύπτει ότι οι 2 πρώτες φάσεις απαιτούν με μεγάλη πιθανότητα O(log n) βήματα , ενώ η τρίτη στη μεσαία περίπτωση O(log n) βήματα ,ενώ στη χειρότερη περίπτωση με μεγάλη πιθανότητα $O(\log^2 n)$ βήματα και επειδή συνεπώς η φάση αυτή είναι η δυσμενέστερη ,στο σύνολο της η αναζήτηση απαιτεί στη μεσαία περίπτωση O(log n) βήματα , ενώ στη χειρότερη περίπτωση με μεγάλη πιθανότητα $O(\log^2 n)$ βήματα.

Για την μέτρηση της επιβάρυνσης του δικτυού από την αναζήτηση , ορίζεται ως φορτίο για κάθε server η πιθανότητα να εμπλέκεται στην αναζήτηση μεταξύ 2 τυχαίων κόμβων . Αποδεικνύεται πως σε ένα δίκτυο με n servers η εκτιμώμενη τιμή του φορτίου για κάθε server είναι O(log n)/ n) και του μέγιστου φορτίου είναι $O(\log^2 n)/ n)$.

3.7.2.5 Η βελτιωμένη αναζήτηση

Από την προηγούμενη ανάλυση προέκυψε ότι στη τρίτη φάση της αναζήτησης υπάρχουν το πολύ $O(\log^2 n)$ servers ανάμεσα στον τρέχοντα server που έχει φτάσει η αναζήτηση και τον στόχο και επειδή η αναζήτηση

στη φάση αυτή γίνεται με απλό βηματισμό , τα απαιτούμενα βήματα είναι επίσης $O(\log^2 n)$. Μια βελτίωση προκύπτει με την χρήση του δακτυλίου στο level ο οποίος δεν έχει χρησιμοποιηθεί στο απλό lookup . Η τρίτη φάση λοιπόν του αλγόριθμου μετασχηματίζεται ως εξής:

Traverse-ring :

Αν ο cur είναι ο πλησιέστερος κατά την φορά του ρολογιού
προς τον x then
Επιστρέφει τον cur και τερματίζει
Else
If cur.nextonlevel єstretch(cur,x(then cur=
Cur.nextonlevel
Else if cur.prevonlevel єstretch(cur,x) then cur=
Cur.prevonlevel
Else cur=cur.successor ή cur=cur.predecessor αναλογα
Με το ποιος είναι πλησιέστερα στο x
Goto Traverse-ring (επαναλαμβάνει την φάση)

Ο παραπάνω αλγόριθμος απαιτεί με μεγάλη πιθανότητα O(log n) βήματα και συνεπώς και η συνολική αναζήτηση O(log n) βήματα.

3.7.3 Συμπεράσματα

Το Viceroy είναι ένα σύστημα που επιτρέπει την ικανοποιητική τοποθέτηση και αναζήτηση αντικειμένων σε μεγάλα και δυναμικά μεταβαλλόμενα δίκτυα . Δανείζεται στοιχεία από το Chord (όσον αφορά τον τρόπο της κατανομής των πόρων στους κόμβους) , αλλά το βελτιώνει διότι :

a) Ενώ στο Chord κάθε κόμβος κρατάει δεδομένα για log n αριθμό κόμβων με τους οποίους διατηρεί απομακρυσμένη σύνδεση , το Viceroy μόνο για 7 . Αυτό κάνει για μεγάλα δίκτυα πολύ μεγαλύτερο το κόστος εισόδου και αποχώρησης κόμβων στο δίκτυο για το Chord από ότι για το Viceroy .

b) Επίσης ενώ το Chord δεν διαθέτει πολύ ξεκάθαρο σχήμα για την διατήρηση του δικτύου στο χρόνο , το Viceroy αναπτύσσεται σε σαφή τρόπο κατά την είσοδο η έξοδο των κόμβων .

Ο παρακάτω πίνακας δείχνει την συμπεριφορά του Viceroy σε σχέση με άλλα δίκτυα :

Lookup scheme	dilation	congestion	linkage
Chord[20]	$\log n$	$(\log n)/n$	$\log n$
Tapestry[21]	$\log n$	$(\log n)/n$	$\log n$
CAN[18]	$dn^{(1/d)}$	$dn^{(1/d-1)}$	d
Small Worlds [6]	$\log^2 n$	$(\log^2 n)/n$	$O(1)$
Viceroy (ours)	$\log n$	$(\log n)/n$	7

3.8 Λοιπά Πιθανοτικά P2P δίκτυα

Το δίκτυο Sympony βασίζεται στην κατασκευή του Kleinberg και είναι ένα πιθανοτικό P2P δίκτυο .Η διαφορά του σχέση με τον Kleinberg είναι ότι οι κόμβοι είναι τοποθετημένοι σε δακτύλιο αντί για δισδιάστατο πλέγμα και κάθε κόμβος έχει k πολλαπλές απομακρυσμένες συνδέσεις αντί για μια . Το μέσο μήκος του των βημάτων για δρομολόγηση είναι $o(\log^2 n/k)$.

Πρόσφατα 3 ακόμη πιθανοτικά δίκτυα προτάθηκαν που χρησιμοποιούν άπληστη δρομολόγηση και έχουν Θ (logn) απομακρυσμένες συνδέσεις για κάθε κόμβο . Το δίκτυο **Randomized-Chord** είναι μια παραλλαγή του ντετερμινιστικού δικτύου Chord . Το SkipNet(skip-graphs) κατασκευάστηκε βάση των skip-lists. Τα skip-graphs είναι τα μοναδικά δίκτυα δρομολόγησης p2p που υποστηρίζουν prefix-search . Αλλά p2p δίκτυα θεωρούν ότι κόμβοι έχουν αναγνωριστικά που παίρνονται ομοιόμορφα από το διάστημα [0,1) .Το τελευταίο αυτής της κατηγορίας είναι το **Randomized-Hypercube**. Η άπληστη δρομολόγηση γίνεται σε O(log n) βήματα . Ένας πιθανοτικός αλγόριθμος που συνδυάζει ιδέες από το Viceroy και από την κατασκευή του Kleinberg (την οποία θα μελετήσουμε αργότερα) περιγράφεται στην εργασία του Manku , και δρομολογεί σε Θ(logn/logk) βήματα με μεγάλη πιθανότητα (w.h.p.) με k απομακρυσμένους συνδέσμους ανά κόμβο . Οι πιο πρόσφατες μελέτες έχουν καταφέρει την δρομολόγηση σε Θ(logn/loglogn) βήματα με Θ(logn) απομακρυσμένους συνδέσμους ανά κόμβο . Οι τάξεις των δικτύων που επιτυγχάνουν αυτό το φράγμα είναι τα de Bruijn δίκτυα , οι ντετερμινιστικές και οι πιθανοτικές πεταλούδες . Τα ίδια αποτελέσματα επιτυγχάνει και η NoN-Greedy δρομολόγηση .

ΚΕΦΑΛΑΙΟ 4

Αξιολόγηση Συστημάτων

4.1 Αξιολόγηση των Μη δομημένων Συστημάτων P2P

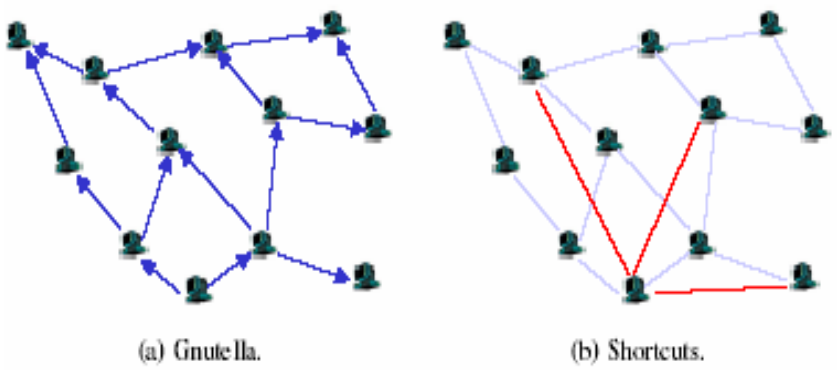
4.1.2 Βελτιώσεις των P2P συστημάτων

Αρκετές προσπάθειες έχουν γίνει κατά καιρούς έτσι ώστε να βελτιωθεί η απόδοση στα ήδη υπάρχοντα δίκτυα. Στις κατηγορίες P2P συστημάτων που παρουσιάστηκαν παραπάνω διαπιστώθηκαν και πλεονεκτήματα και μειονεκτήματα. Οι νέες ιδέες που εφαρμόστηκαν στα ήδη υπάρχοντα δίκτυα προσπαθούν να συνδυάσουν τα πλεονεκτήματα της κάθε κατηγορίας σε νέες εφαρμογές συστημάτων. Η ιδέα για τα δίκτυα αυτά ήταν οι κόμβοι τους να οργανώνονται σε ομάδες με βάση τα ενδιαφέροντά τους ή με την ομοιότητα των δεδομένων που προσφέρουν.

Επιπλέον τα P2P συστήματα που περιγράφτηκαν στα προηγούμενα κεφάλαια υποστήριζαν μόνο απλές αναζητήσεις δεδομένων με βάση το όνομά τους. Πολλές εφαρμογές των P2P συστημάτων στηρίζονται στην υποστήριξη ερωτήσεων πολλών διαστάσεων που αφορούν σε κάποιο εύρος δεδομένων (“multi-dimensional range queries”). Με τον όρο αυτό εννοούμε συζευκτικές ερωτήσεις σε ένα εύρος τιμών δύο ή και περισσότερων γνωρισμάτων μιας σχέσης. Έχουν γίνει βελτιώσεις στα P2P συστήματα ώστε να επιτευχθεί η προηγούμενη ιδιότητα.

4.1.2.1 Οργάνωση των κόμβων με βάση τα ενδιαφέροντα

Interest-based shortcuts : Η ιδέα εφαρμόζεται σε ένα κατανεμημένο P2P δίκτυο και συγκεκριμένα στο Gnutella. Σκοπός είναι η διατήρηση της απλότητας και ευρωστίας του και η βελτίωση της κλιμάκωσης. Η ιδέα των συντομεύσεων παρέχει μια χαλαρή δομή σε υψηλότερο επίπεδο από το Gnutella. Η βασική αρχή στην οποία στηρίζεται η δημιουργία του ανώτερου επιπέδου δικτύου είναι ότι αν ένας κόμβος έχει κάποιο δεδομένο που ενδιαφέρει κάποιον άλλο κόμβο, τότε είναι πιθανό αυτός ο κόμβος να έχει κι άλλα δεδομένα που ενδιαφέρουν τον άλλο κόμβο. Σημαντικά πλεονεκτήματα αυτής της οργάνωσης είναι ότι μπορεί να εφαρμοστεί πάνω σε οποιοδήποτε άλλο σχήμα όπως είναι τα DHTs και δεν επηρεάζει το δίκτυο βάση (εδώ Gnutella) πάρα μόνο την απόδοσή του προς το καλύτερο.



Εικόνα47 : Παράδειγμα ενός δικτύου Gnutella και η εφαρμογή συντομεύσεων σε αυτό.

Πιο συγκεκριμένα στους κόμβους που έχουν κοινά ενδιαφέροντα δημιουργούνται επιπρόσθετες συνδέσεις πάνω από το ήδη υπάρχον P2P. Έτσι κάθε κόμβος διατηρεί λίστα με τις επιπρόσθετες συνδέσεις. Οι συνδέσεις αυτές προκύπτουν εφαρμόζοντας αρχικά μια πλημμύρα στο δίκτυο για ένα συγκεκριμένο δεδομένο. Στη συνέχεια ο κόμβος που εφάρμοσε την πλημμύρα γνωρίζει ποιοι είναι οι κόμβοι που κατέχουν το δεδομένο και επιλέγει τυχαία κάποιους από αυτούς για να τους προσθέσει στη λίστα με τις συντομεύσεις, τις επιπρόσθετες συνδέσεις. Αν κάποιος θελήσει να αναζητήσει κάποιο δεδομένο στο δίκτυο θα προωθήσει πρώτα την ερώτηση στους κόμβους για

τους οποίους έχει επιπρόσθετες συνδέσεις και μόνο στην περίπτωση που δεν βρεθεί σε αυτούς το δεδομένο ακολουθείται ο τρόπος δρομολόγησης του κατώτερου επιπέδου δικτύου (Gnutella). Στην εικόνα 47 (a) αναπαριστάται ένα δίκτυο Gnutella και στο (b) το προηγούμενο δίκτυο με τις επιπρόσθετες συνδέσεις .

4.2 Αξιολόγηση των δομημένων Συστημάτων P2P

Η δεύτερη γενιά των peer-to-peer συστημάτων ,εμφανίστηκε σε μια προσπάθεια να ξεπεραστούν οι περιορισμοί που έθεταν οι πρωτοπόροι όπως το FreeNet και το Gnutella . Τα συστήματα της καινούργιας γενιάς εγγυώνται την απάντηση σε κάθε ερώτημα (την δρομολόγηση κάθε μηνύματος) σε περιορισμένο αριθμό βημάτων , ενώ διατηρούν την επεκτασιμότητα του FreeNet και την δυνατότητα επιδιόρθωσης και αυτοοργάνωσης του δικτύου που προσφέρουν τα δυο προηγούμενα συστήματα .

Στην $2^{η}$ γενιά ανήκουν συστήματα όπως τα Pastry , Tapestry , Chord , Kademlia και Can .

Τα συστήματα αυτά σε κάθε κόμβο αποθηκεύουν συγκεκριμένες πληροφορίες για άλλους κόμβους , με αυστηρά καθορισμένο πρωτόκολλο το καθένα και δρομολογούν τα μηνύματα με ασφάλεια και ταχύτητα στηριζόμενα στις πληροφορίες αυτές . Έχουν την δυνατότητα επίσης για ικανοποιητική ανταπόκριση της δρομολόγησης των μηνυμάτων σε δυναμικά συμβάντα στο δίκτυο (εισαγωγή και έξοδος κόμβων) και για αυτοοργάνωση των πληροφοριών τους και αποκατάσταση της σταθερότητας τους σε περίπτωση τέτοιων γεγονότων .

Το $1^{ο}$ σύστημα εμφανίστηκε σε αυτή την κατηγορία είναι το Chord που στηρίζεται στο DHT (Distrubuted Hash Table) και κύριο χαρακτηριστικό του είναι ο χειρισμός της θέσης αποθήκευσης των αντικειμένων με συγκεκριμένο τρόπο πάνω στον όποιο στηρίζεται η δρομολόγηση των μηνυμάτων και ο εντοπισμός των αντικειμένων . Το γεγονός αυτό δημιουργεί περιορισμούς στις εφαρμογές τόσο όσο αφόρα την θέση όσο και το πλήθος των αντιγράφων των αντικειμένων . Επίσης το Chord (όπως και το CAN) δεν λαμβάνει υπ' όψιν του την φυσική εγγύτητα των κόμβων του μονοπατιού δρομολόγησης (τοπικότητα) και συνεπώς δεν εξασφαλίζει την συντομότερη δυνατή δρομολόγηση . Το πλήθος των βημάτων δρομολόγησης είναι O(logN) (το N είναι το πλήθος των κόμβων του δικτύου).

Το Tapestry στηρίζεται στο DOLR (Decentralized Object Location and Routing) και μεταφέρει τα μηνύματα στους κόμβους με το πλησιέστερο πρόθεμα σε σχέση με το πρόθεμα του κλειδιού του μηνύματος (με παρόμοιο τρόπο με την εργασία του Plaxton) . Εξασφαλίζει εκμετάλλευση της τοπικότητας του δικτύου , γεγονός που επιτρέπει ταχύτερη μετάδοση των μηνυμάτων και λιγότερη φυσική επιβάρυνση του δικτύου . Δεν θέτει περιορισμούς στη θέση και το πλήθος των αντικειμένων που χειρίζονται οι εφαρμογές . Έχει σχετικά δύσκολη διαδικασία αποκατάστασης των πινάκων πληροφοριών των κόμβων του σε περίπτωση εισαγωγής νέων κόμβων και για τον λόγο αυτό δεν ενδείκνυται για γρήγορα μεταβαλλόμενα δίκτυα . Το πλήθος των βημάτων δρομολόγησης είναι $O(\log_{\beta} N)$, όπου β είναι η βάση των αναγνωριστικών .

Το Pastry έχει περίπου την ίδια φιλοσοφία με το Tapestry στον τρόπο δρομολόγησης (μεταφέρει τα μηνύματα στους κόμβους με το πλησιέστερο πρόθεμα σε σχέση με το πρόθεμα του κλειδιού του μηνύματος) αλλά διαθέτει

πιο ευέλικτη οργάνωση των πινάκων πληροφοριών που κρατούν οι κόμβοι του . Αυτό του επιτρέπει καλύτερη ανταπόκριση στα δυναμικά συμβάντα του δικτύου γεγονός που το κάνει κατάλληλο για γρήγορα μεταβαλλόμενα δίκτυα . Εξασφαλίζει επίσης εκμετάλλευση της τοπικότητας του δικτύου , με πολύ χαμηλό συντελεστή αύξησης της διαδρομής σε σχέση με την φυσική απόσταση των κόμβων πηγής και περιορισμού (δες στα πειραματικά δεδομένα όπου ο συντελεστής αυτός υπό ορισμένες συνθήκες γίνεται 1.30).

Το πλήθος των βημάτων δρομολόγησης είναι $O(\log_2^b N)$, όπου b μια παράμετρος διαμόρφωσης με τυπική τιμή 4 . Σε σύγκριση με το CAN όπου το πλήθος αυτό είναι $o(dN^{1/d})$, βλέπουμε ότι στο CAN το πλήθος των βημάτων αυξάνεται γρηγορότερα από ότι στο Pastry με την αύξηση του N.

Η σημαντική διαφορά του Kademlia με τα υπόλοιπα είναι η μετρική XOR που χρησιμοποιεί . Το XOR είναι συμμετρικό επιτρέποντας στους συμμετέχοντες στο Kademlia να δέχονται ερωτήματα αναζήτησης από ακριβώς την ίδια κατανομή κόμβων που περιέχονται στους πίνακες δρομολόγησης τους . Το Chord που δεν έχει αυτήν την ιδιότητα δεν μαθαίνει πληροφορία δρομολόγησης από τα ερωτήματα που δέχονται οι κόμβοι του . Κάθε εγγραφή του πίνακα δεικτών ενός κόμβου στο Chord αποθηκεύει τον ακριβή κόμβο κάποιου διαστήματος στο χώρο των ID .Οποιοσδήποτε κόμβος στο διάστημα αυτό μπορεί να είναι μακριά από τον κόμβο . Το Kademlia εν αντιθέσει μπορεί να στείλει ερώτημα σε οποιοσδήποτε κόμβο μέσα σε ένα διάστημα , επιτρέποντας του να επιλέξει την δρομολόγηση βασισμένο στην καθυστέρηση (latency) ή μπορεί να στείλει παράλληλα ασύγχρονα ερωτήματα σε αρκετούς κατάλληλους κόμβους . Για τον εντοπισμό των κόμβων γύρω από ένα συγκεκριμένο ID . ΤΟ Kademlia χρησιμοποιεί ένα αλγόριθμο δρομολόγησης από την αρχή ως το τέλος σε αντίθεση με άλλα συστήματα που χρησιμοποιούν ένα αλγόριθμο για να φτάσουν κοντά στον

κόμβο και ένα άλλον για τα τελευταία λίγα βήματα . Το Kademlia μοιάζει με την πρώτη φάση του Pastry , στην οποία επιτυχώς βρίσκονται κόμβοι που απέχουν το μισό από το ID του κόμβου στόχου από την μετρική XOR. Το Pastry σε δεύτερη φάση μετατρέπει τις μετρικές απόστασης σε αριθμητική διαφορά μεταξύ ID.

Το Viceroy είναι ένα σύστημα που παρουσιάστηκε ως βελτίωση του Chord . Έτσι ακολουθεί την δομή του διευρυνόμενου δακτυλίου όπως και το Chord από πού παίρνει και τον τρόπο κατανομής των πόρων στους ενεργούς κόμβους , καθώς και την βασική διαδικασία της δρομολόγησης από τον ένα κόμβο στον γείτονα του (δηλαδή χρησιμοποιεί τις έννοιες του successor και predecessor) . Για την ταχύτερη δρομολόγηση κάθε κόμβος συνδέεται εκτός από τους γείτονες του στο δακτυλίδι και με ένα σταθερό αριθμό (5) από απομακρυσμένους συνδέσμους με τρόπο που προσεγγίζει το κλασσικό δίκτυο butterfly . Έτσι το Viceroy είναι μια σύνθεση ανάμεσα στο δίκτυο butterfly και στον δακτύλιο τύπου Chord . Με τον τρόπο αυτό ενώ στο Chord κάθε κόμβος κρατάει δεδομένα για log n αριθμό κόμβων με τους οποίους διατηρεί απομακρυσμένη σύνδεση , το Viceroy μόνο για 7 . Αυτό κάνει για μεγάλα δίκτυα πολύ μεγαλύτερο το κόστος εισόδου και αποχώρησης κόμβων στο δίκτυο για το Chord από ότι για το Viceroy .Ταυτόχρονα διατηρεί τον αριθμό των βημάτων δρομολόγησης του Chord δηλαδή O(logN).

ΚΕΦΑΛΑΙΟ 5

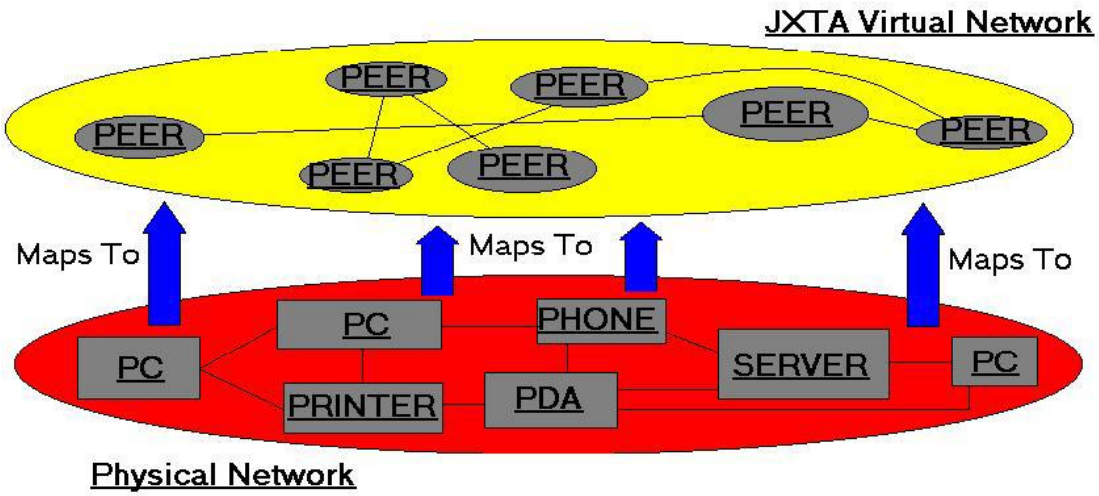
Συμπεράσματα

5.1 Οι εξελίξεις στο τομέα του P2P και το μέλλον

Ερευνητές του Πανεπιστημίου του Wisconsin, που αναπτύσσουν μία τεχνολογία κατανεμημένης επεξεργασίας επονομαζόμενη Condor, εκτιμούν ότι οι περισσότερες επιχειρήσεις αξιοποιούν λιγότερο από το 25% της επεξεργαστικής ισχύος και του χώρου αποθήκευσης που διαθέτουν. Κολοσσιαίες επιχειρήσεις όπως η Intel, η γιγάντια αεροδιαστημική βιομηχανία Boeing, αλλά και η εταιρεία πετρελαίων Amerada Hess, έχουν κάνει κάτι γι' αυτό, υιοθετώντας με επιτυχία συστήματα P2P. Η τελευταία, μέσω του Beowulf Project, έχει ενώσει 200 επιτραπέζιους υπολογιστές της Dell με Ethernet και Linux. Οι συγκεκριμένοι υπολογιστές απασχολούνται στην ερμηνεία πολύπλοκων σεισμικών δεδομένων και έχουν αντικαταστήσει στο έργο αυτό δύο υπερυπολογιστές IBM. Η ίδια εταιρεία έχει αναπτύξει ακόμη δύο σχετικά projects. Στο πρώτο από αυτά κάθε υπολογιστής στο δίκτυο "δανείζεται" κύκλους επεξεργασίας από διπλανά PCs, ενώ το δεύτερο λειτουργεί με τη φιλοσοφία του Napster και έχει ως στόχο την αξιοποίηση του συνολικού κατανεμημένου χώρου επεξεργασίας. Παράλληλα, εταιρείες όπως οι Applied MetaComputing και Groove Networks, αναπτύσσουν προϊόντα και υπηρεσίες αυτού του τύπου. Στις Η.Π.Α. ο κρατικός τομέας κάνει τα πρώτα δειλά βήματα προς το P2P. Τα sites FedStats.gov και FedStats.net επιτρέπουν σε περισσότερους από 70 κρατικούς οργανισμούς, οι οποίοι χρησιμοποιούν 200 στατιστικά προγράμματα, να συνδέονται απευθείας και να ανταλλάσσουν στατιστικά δεδομένα "ταχύτερα, καλύτερα και φθηνότερα" όπως λένε οι υπεύθυνοί τους. Ο οργανισμός DARPA (Defence Advanced Research Projects Agency) έχει ξεκινήσει ένα πειραματικό πρόγραμμα για τη δικτύωση

P2P στρατιωτών στο πεδίο της μάχης. Οι πομποδέκτες των στρατιωτών αναπτύσσονται από την ITT, ενώ το δίκτυο θα βασίζεται στο Linux. Πλεονέκτημα του δικτύου αυτού είναι το γεγονός ότι οι πομποδέκτες θα χρειάζονται μικρότερη ισχύ, με αποτέλεσμα μεγαλύτερη διάρκεια της μπαταρίας και δυσκολότερο εντοπισμό ή παρεμβολές από τον εχθρό. Ανάλογο πρόγραμμα έχει και το αμερικανικό ναυτικό Παρ’ ότι όλα τα ανωτέρω είναι ενθαρρυντικά, παραμένει το γεγονός ότι υπάρχουν αρκετά προβλήματα για τη σχεδίαση ενός απλού προγράμματος στην πλατφόρμα Peer To Peer. Καταρχήν, θα πρέπει να προσδιοριστεί ο σκοπός του προγράμματος ώστε να μπορεί να συνεργαστεί με κατανεμημένο σύστημα. Όπως αναφέρουν οι ειδικοί, πρέπει το πρόγραμμα να είναι "αναίσθητο" στα υποκείμενα επίπεδα (layers). Επιπλέον, μία σοβαρή εφαρμογή θα πρέπει να χρησιμοποιεί κάποιου είδους πιστοποίηση για τους χρήστες που συνδέονται. Γενικότερα, το θέμα της ασφάλειας είναι κάτι που σίγουρα επιδέχεται βελτίωσης στα Peer to Peer προγράμματα που έχουν δημιουργηθεί μέχρι σήμερα. Επίσης, όπως είναι της μόδας τελευταία, για να γίνει πιο προσιτή στο μέσο προγραμματιστή η δημιουργία μίας Peer to Peer εφαρμογής, πρέπει να παρουσιαστεί μια πλατφόρμα με τη μορφή βιβλιοθήκης (library). Με τον τρόπο αυτό η υλοποίηση μιας τέτοιας εφαρμογής θα γίνεται πιο γρήγορα και στην πράξη δεν θα χρειάζεται κάθε προγραμματιστής να "ξαναεφευρίσκει τον τροχό". Όλα αυτά είναι σαφή προβλήματα, τα οποία όμως οδεύουν προς τη λύση τους. Γι’ αυτό και στην Αμερική, η οποία είναι η χώρα της τυποποίησης, γίνονται ήδη προσπάθειες δημιουργίας κάποιων standards για το συγκεκριμένο είδος εφαρμογών. Τον πρώτο λόγο στις προσπάθειες αυτές έχει το “Peer To Peer Working Group” (http://www.peer-to-peerwg.org), μία ομάδα από εταιρείες που προσπαθούν να ωθήσουν την αγορά προς το Peer to Peer computing. Εκτός από την Intel που το ξεκίνησε, σήμερα στο PTPWG έχουν προστεθεί πολλά ακόμα μεγαθήρια της πληροφορικής, όπως η Hewlett-

Packard και η Fujitsu. Την υπόλοιπη ομάδα στελεχώνουν και εταιρείες που έχουν επενδύσει σε αυτή την τεχνολογία και δημιουργούν τέτοιου είδους εφαρμογές. Ήδη διατίθεται από το site του PTPWG μία βιβλιοθήκη για distributed πιστοποίηση χρηστών, με τη βοήθεια της βιβλιοθήκης OpenSSL (SSL=Secure Socket Layer). Η ομάδα έχει ικανοποιητική δραστηριότητα και προσπαθεί να μαζέψει τα απαραίτητα προγραμματιστικά εργαλεία ώστε να γίνεται ευκολότερα στο μέλλον η υλοποίηση μιας Peer to Peer εφαρμογής. Από τη Sun MicroSystems ξεκίνησε το project JXTA, το οποίο είναι ένα έργο ανοικτού κώδικα που ορίζει ένα σετ από πρωτόκολλα για ad-hoc, p2p δίκτυα. Σκοπός είναι να βοηθήσει την ανάπτυξη συστημάτων κι εφαρμογών με χαρακτηριστικό τη διαλειτουργικότητα, δηλαδη εφαρμογές που θα μπορούν να συμπεριλάβουν σαν κόμβους υπολογιστές , PDAs , κινητά κλπ. Για παράδειγμα μία ασύρματη συσκευή που χρησιμοποιεί πρωτόκολλο επικοινωνίας το Bluetooth και ένα PC συνδεδεμένο μέσω TCP/IP θα είναι κόμβοι του ίδιου δικτύου μιας εφαρμογής βασισμένη στο JXTA.



Εικόνα 48 : Η πλατφόρμα του JXTA

Με την βοήθεια ενός μεγάλου και αναπτυσσόμενου αριθμού ειδικών από ακαδημαϊκά ιδρύματα κ επιχειρήσεις, έχει αναπτυχθεί ένα λεπτό επίπεδο

δικτύου που μπορεί να χρησιμοποιηθεί ήδη από εφαρμογές συνεργασίας, ανταλλαγής αρχείων και απομακρυσμένων υπηρεσιών. Τεχνικές αποτελεσματικής και αποδοτικής τοποθέτησης και δρομολόγησης έχουν σχεδιαστεί όπως το TAPESTRY, που παρέχει ανεξάρτητη από τη τοποθεσία δρομολόγηση μηνυμάτων απευθείας στο κοντινότερο αντίγραφο του ζητούμενου αντικειμένου ή υπηρεσίας,, το KADMELIA καθώς και το CHORD. Αν και δεν μπορούμε με απόλυτη σιγουριά να προβλέψουμε ποιο ακριβώς θα είναι, από πλευράς εφαρμογών και αρχιτεκτονικής, το μέλλον του Peer to Peer, μπορούμε με σιγουριά να πούμε ότι αυτό διαγράφεται αρκετά ελπιδοφόρο. Ήδη έχουν συγκροτηθεί αρκετά working groups και ομάδες έρευνας, οι οποίες προσπαθούν να δημιουργήσουν standards και frameworks τα οποία θα κάνουν την ανάπτυξη εφαρμογών πολύ πιο εύκολη. Το πιο ενθαρρυντικό απ' όλα είναι άλλωστε το γεγονός ότι οι περισσότερες από τις ομάδες αυτές δείχνουν διατεθειμένες να μην κρατήσουν την ανάπτυξη των τεχνολογιών υπό ιδιοκτησιακό καθεστώς, κάτι που -συν τοις άλλοις- θα συνεισφέρει στη γενικότερη εξάπλωση και ανάπτυξη του Peer to Peer.

5.2 Συμπεράσματα

Τα συστήματα ομότιμων κόμβων έχουν ξεκινήσει πριν από λίγα χρόνια αλλά έγιναν πολύ δημοφιλή. Τα πρώτα συστήματα ήταν πολύ απλοϊκά, και στη συνέχεια εμφανίστηκαν άλλα συστήματα, με περισσότερο οργανωμένη δομή. Όπως είδαμε, σε κάθε περίπτωση σχεδίασης η αναζήτηση ενός δεδομένου μέσα στο δίκτυο είναι το κύριο θέμα προς επίλυση. Μέσα από την εργασία αυτή είδαμε διάφορες προσεγγίσεις και ιδέες για ένα σύστημα καθαρά κατανεμημένο και με δυνατότητα για γρήγορη και αποδοτική αναζήτηση. Πάντως τα συστήματα ομότιμων κόμβων είναι ένας τομέας στον οποίο γίνεται έρευνα, και είναι λογικό να περιμένει κανείς ότι θα υπάρξουν

νέες ιδέες που ίσως αλλάξουν τον τρόπο με τον οποίο σήμερα αντιλαμβανόμαστε τα συστήματα αυτά

Στα προηγούμενα παρουσιάστηκε μια μεγάλη κατηγορία κατανεμημένων συστημάτων και ευρέως γνωστών, τα P2P δίκτυα τα οποία ομαδοποιούνται στα δομημένα και μη δομημένα δίκτυα. Έγινε μια εκτενής αναφορά πάνω στις δύο αυτές βασικές υποκατηγορίες των P2P συστημάτων καθώς επίσης και σε αρκετές βελτιώσεις των τελευταίων. Από όλα αυτά που αναφέρθηκαν διαπιστώνουμε τα πλεονεκτήματα και τα μειονεκτήματα των κατηγοριών των P2P δικτύων. Ο Πίνακας παρακάτω συγκεντρώνει κάποια γενικά στοιχεία των P2P συστημάτων όπως είναι τα χαρακτηριστικά τους, πλεονεκτήματα που παρουσιάζουν, κάποια θέματα-προκλήσεις που μπορούν να συζητηθούν σχετικά με αυτά και ορισμένες εφαρμογές τους.

Χαρακτηριστικά	Πλεονεκτήματα	Προκλήσεις	Εφαρμογές
Άμεση ανταλλαγή	Αποδοτικότητα	Ασφάλεια	Διαμοιρασμός αρχείων
Client-Server	Διαμοιρασμός δεδομένων	Εξισορρόπηση φορτίου	Κατανεμημένος υπολογισμός
Τα δεδομένα παρέχονται από κόμβους	Μεγάλος όγκος δεδομένων	Αναζήτηση, ενοποίηση δεδομένων	Διαμοιρασμός φωτογραφιών, πολυδιάστατες ερωτήσεις
Αυτονομία	Ανοχή σε σφάλματα	Διαθεσιμότητα δεδομένων, τοποθέτηση αντιγράφων (replication)	Δικτυακά παιχνίδια

Συγκεντρωτικός πίνακας για τα P2P δίκτυα

Όσον αφορά τα δομημένα δίκτυα χαρακτηρίζονται από δύο πολύ σημαντικές ιδιότητες, την κλιμάκωση και την εύρεση “σπάνιων” δεδομένων με αποδοτικό τρόπο. Από την άλλη πλευρά μειονεκτούν ως προς τα μη δομημένα δίκτυα λόγω της μη αποδοτικής υποστήριξης των “partial-match”

ερωτήσεων. Αρνητικό χαρακτηριστικό αποτελεί και η ευαισθησία τους στις αποτυχίες και τις συχνές εισαγωγές ή αποσυνδέσεις των κόμβων για το οποίο ευθύνεται το γεγονός ότι κάθε κόμβος πρέπει να αποθηκεύει ένα συγκεκριμένο κομμάτι του πίνακα κατακερματισμού. Μια άλλη συνέπεια του τελευταίου είναι ότι οι κόμβοι είναι απαραίτητο να αποθηκεύουν δεδομένα για το κοινό συμφέρον κι όχι υποχρεωτικά αυτά για τα οποία ενδιαφέρονται.

Από την άλλη πλευρά τα χαρακτηριστικά που οδηγούν στην επιτυχία της αρχιτεκτονικής των μη δομημένων δικτύων είναι η υποστήριξη των “partial-match” ερωτήσεων και η “χαλαρή” τοπολογία των δικτύων, δηλαδή το γεγονός ότι δεν επηρεάζονται αρκετά από πιθανές καταρρεύσεις-αποτυχίες και συνεχείς εισαγωγές ή αποσυνδέσεις των κόμβων. Σε αντίθεση επίσης με τα δομημένα δίκτυα εδώ δεν υπάρχει η απαίτηση να αποθηκεύουν οι κόμβοι δεδομένα για το κοινό συμφέρον αλλά κάθε κόμβος συμμετέχει προσφέροντας αυτά που επιθυμεί. Μειονέκτημα όμως αποτελεί το γεγονός ότι τα μη δομημένα δίκτυα έχουν μικρή ικανότητα στον εντοπισμό “σπάνιων” δεδομένων και επιπλέον υστερούν ως προς το πιο σημαντικό χαρακτηριστικό που πρέπει να διαθέτουν τα P2P δίκτυα, την κλιμάκωση.

ΕΠΙΛΟΓΟΣ

Πριν ξεκινήσουμε αυτή την πτυχιακή εργασία οι γνώσεις μας γύρω από τα Peer to Peer συστήματα περιορίζονταν γύρω από δικτυακές εφαρμογές που μας βοηθούσαν να κατεβάσουμε κάποιο τραγούδι ή κάποια ταινία από το διαδίκτυο. Τελειώνοντάς την , αντιληφθήκαμε πως μελετήσαμε έναν απ΄ τους πιο γρήγορα αναπτυσσόμενους τομείς της επιστήμης των υπολογιστών ,καθώς επίσης και τις θεωρητικές αρχές πάνω στις οποίες στηρίζονται.

Ξεκινώντας με μια παρουσίαση και μια ιστορική αναδρομή των Peer to Peer συστημάτων ,συνεχίζουμε με την κατηγοριοποίηση αυτών ,σε δομημένα και μη. Στα συστήματα αυτά ,τα οποία και εξετάζονται ένα ξεχωριστά ,περιγράφονται η τοπολογία τους και οι βασικές διαδικασίες λειτουργίας τους. Παρουσιάζονται επίσης οι επιδόσεις τους στις διάφορες λειτουργίες ,αλλά και μια συγκριτική ανασκόπηση τους.

Η γενικότερη εντύπωση που αποκομίσαμε μελετώντας τα Peer to Peer συστήματα είναι πως πρόκειται για έναν ιδιαίτερο και πολύπλοκο τομέα των πληροφοριακών συστημάτων . Δεν ξέρουμε αν καταφέραμε να γίνουμε ‘‘ειδικοί’’ γύρω από την επιστήμη των Peer to Peer , ίσως να μην ήτανε αυτή η επιδίωξή μας , το σίγουρο ότι γνωρίσαμε θεωρητικά κυρίως τον τομέα αυτόν των υπολογιστών που συναντάμε πλέον όλο και περισσότερο πρακτικά στην καθημερινότητά μας .

ΒΙΒΛΙΟΓΡΑΦΙΑ

- Woodcock JoAnne Εισαγωγή στα Δίκτυα Υπολογιστών
- Andrew Tanenbaum Δίκτυα υπολογιστών 3η έκδοση
- Douglas Comer Δίκτυα και Διαδίκτυα Υπολογιστών
- Βασίλης Τσαουσίδης Διαδικτυακά πρωτόκολα
- Karl Aberer, Manfred Hauswirth, An Overview on Peer-to-Peer
- Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma, Steven Lim, A Survey and Comparison of Peer-to-Peer Overlay Network Schemes, March 31, 2004 IEEE.
- Sylvia Ratnasamy, Paul Fransis, Mark Handley, Richard Karp, Scott Shenker, A Scalable Content-Addressable Network, in SIGCOMM'01 August San Diego, California, USA
- Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, Hari Balakrishman, Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications, in SIGCOMM'01 August 2001, San Diego, California, USA
- Dimitrios Tsoumakos, Nick Roussopoulos, A Comparison of Peer-to-Peer Search Methods, in WebDB, June 2003, San Diego, California
- Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In Proceedings of the 2001 ACM SIGCOMM

ΙΣΤΟΣΕΛΙΔΕΣ(Links)

- http://www.napster.com/
- http://gnutella.wego.com/
- http://srhea.net/papers/tapestry_jsac.pdf
- http://en.wikipedia.org/wiki/Peer_to_peer
- http://www.cs.uiowa.edu/~ghosh/Tapestry.ppt
- http://berkeley.intel-research.net/sylvia/cans.pdf
- http://www.cs.rice.edu/CS/Systems/PAST/pastry.pdf
- http://lsirpeople.epfl.ch/hauswirth/papers/WDAS2002.pdf
- http://www.pdos.csail.mit.edu/chord/papers/iptps-evol.pdf
- http://www.daimi.au.dk/~marius/p2p-course/lectures/11/talk.html
- http://www.aeolus.ceid.upatras.gr/sub-projects/deliverables/D611.pdf
- http://pdos.csail.mit.edu/papers/chord:sigcomm01/chord_sigcomm.pdf
- http://pdos.csail.mit.edu/~petar/papers/maymounkov-kademlia-lncs.pdf
- http://www.comp.nus.edu.sg/~ooibc/courses/cs6203/StructuredNetwork
- http://www.cs.berkeley.edu/~satishr/cs273.03/ConsistentHashing.pdf+n.build_fingers